UNIVERSITY OF OKLAHOMA GRADUATE COLLEGE

INVESTIGATION OF DIMENSIONALITY REDUCTION TECHNIQUES IN THE SURROGATE MODEL DEVELOPMENT FOR SPATIOTEMPORAL STORM SURGE PREDICTIONS

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By SUJATA SAHU

Norman, Oklahoma

December, 2024

INVESTIGATION OF DIMENSIONALITY REDUCTION TECHNIQUES IN THE SURROGATE MODEL DEVELOPMENT FOR SPATIOTEMPORAL STORM SURGE PREDICTIONS

A THESIS APPROVED FOR THE
GALLOGLY COLLEGE OF ENGINEERING

BY THE COMMITTEE CONSISTING OF

INVESTIGATION OF DIMENSIONALITY REDUCTION TECHNIQUES IN THE SURROGATE MODEL DEVELOPMENT FOR SPATIOTEMPORAL STORM SURGE PREDICTIONS

Abstract

by

Sujata Sahu

Coastal communities face unique challenges due to their increased vulnerability and exposure to natural hazards, with humans and their assets being at a close proximity to the ocean and to a variety of severe weather phenomena such as tropical cyclones. Storm surge, which is the abnormal rise in seawater level due to storm winds, can lead to catastrophic flooding, erosion and infrastructure damage, putting lives and properties at risk. With the frequency of extreme weather phenomena like tropical cyclones increasing due to climate change the need to be able to predict their impact in coastal areas is increasing. Accurate predictions of storm surge, as a tropical cyclone is approaching, are crucial for informing evacuation planning efforts, allowing communities to have enough lead time to prepare and respond effectively. Through robust predictions which provides timely, location-specific information, such vulnerable regions resilience can be improved by saving lives and reducing property damage, thereby aiding recovery efforts.

In this effort to obtain reliable predictions, advances in computational and numerical models, have enabled the development of physics-based models, that can predict storm surge based on complex fluid dynamics equations and atmospheric conditions. These

models are the closest approximation possible to the physical phenomena that contribute to the experienced surge, providing valuable and detailed insights for large regions of interest. Unfortunately, due to their inherent complexity, they are computationally intensive, requiring large memory and number of processors, with the time for each analysis necessitating a couple of days in a state-of-the-art supercomputing facility. These limitations prevent the use of such accurate and detailed models during real-time risk assessment efforts, where a specific tropical cyclone has been formed and is hours away from landfall. Especially these settings, it is of crucial importance to be able to produce time-series storm surge estimates of high spatial resolution fast and accurately so that preparedness and recovery efforts can be guided appropriately and utilized efficiently, maximizing their impact to the extent possible.

In an effort to overcome these limitations, surrogate models have emerged as a versatile alternative, that offer a fairly simple, tractable, mathematical approximation of a computationally expensive model, allowing the production of fast and accurate surge predictions, provided that they are properly calibrated on available datasets for the region of interest. Such efficiency without the significant loss of any accuracy makes them a promising alternative in real-time emergency planning and response when resources and time to respond are limited. This requires the development of surrogate models that will be able to handle large geospatial regions providing detailed predictions in the most computationally efficient way to allow for enough lead time for the evacuation managers to develop their plans, exploring at the same time a large number of potential storm-evolving scenarios.

Pursuing model simplicity and computational efficiency, past efforts have explored the development of surrogate models of different mathematical complexity, that leverage the underlying spatial correlation that the grid locations have within the domain of interest. This allowed for linear and nonlinear dimensionality reduction techniques like Principal Component Analysis (PCA) and AutoEncoder (AE) architectures to generate latent space representations, that can be used as the reduced size input during the surrogate model development and deployment phases. After predictions are made in that reduced, latent space, the actual surge predictions are compiled through an inverse mapping. So far, such investigations have not thoroughly examined nonlinear dimensionality reduction techniques for large geospatial domains where time-series predictions are warranted, especially for locations that are onshore where highly nonlinear surge behavior may be recorded during different tropical cyclone events.

This thesis advances these efforts by performing a detailed investigation on the development of a nonlinear inverse mapping (utilizing the versatile family of autoencoders) designed to capture non-linear relationships for onshore nodes tackling challenges related to providing time-series storm surge predictions over a large number of regions. The latent space will be used as input in a Gaussian Process surrogate model to provide storm surge time-series predictions on a storm suite that the models have not been trained upon. This development seeks to support high-resolution forecasting, making it suitable for real-time emergency response as well as for long-term planning and infrastructure design for any region of interest, given the existence of a well-calibrated region-specific surrogate model.

CONTENTS

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Professor Aikaterini Kyprioti, for her invaluable guidance throughout my thesis journey. Her strength and encouragement were instrumental throughout my research. Her thoughtful and constructive ideas made me think outside the box, while her patience and wisdom were very important in the completion of this thesis. I would also like to extend my sincere thanks to the Data Science and Analytics department for allowing me to transition into thesis work. Special thanks to Professor Nicolson for his continuous support. I am grateful to my entire committee for their unwavering assistance and guidance.

Finally, I want to express heartfelt thanks to my family and friends for their unwavering support and for providing invaluable ideas throughout my thesis journey. Your encouragement made this journey possible.

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1:

INTRODUCTION

1.1 Motivation

The exposure of coastal communities to natural hazards due to urbanization trends and climate change patterns is constantly increasing as more people are moving towards large cities or hazard prone areas such as coastlines [1, 2] and as more extreme events are being recoded with unprecedented frequencies. As a result of this, socioeconomic risks have increased significantly, affecting communities locally, and challenging national readiness and ability to respond to ensure the wellbeing and smooth recovery of those directly affected, restoring the affected infrastructure and other critical facilities [3]. Owing though, to the interdependencies that have been developed along the years at national and international levels, an extreme event, rarely has a local impact, but rather it causes larger disruptions that affect supply chains and disproportionally challenge underrepresented and underserved communities, that typically have limited resources to recover from such disasters in a fast and efficient way [4]. Coastal communities across the world and the

United States are facing increasing challenges, related not only to natural disasters, but other climate change driven phenomena such as sea level rise [5, 6], erosion [7], and saline water infiltration [8], necessitating immediate action plans for boosting community resilience to effectively protect lives, livelihoods and infrastructure. In order to achieve this, communities should aim to generate long- and short- term resilience plans when it comes to natural hazards and the quantification of their impact on communities and their assets [9]. For long term planning, hazards of different nature and magnitudes should be assessed under a changing climate, to effectively inform the design of future, hazard-resistant infrastructure [9-11]. For short-term planning, efforts should be placed on predicting the evolution of ongoing hazards, trying to best plan and organize the response and recovery efforts [9, 11]. For hazards like tropical cyclones, that typically evolve for at least a couple of days before they affect a certain region, the ongoing challenge is to provide high accuracy projections with enough lead time for the evacuation managers and state officials to develop plans and guidelines for the public to follow [12].

Providing more lead time to landfall and a more precise forecast of the magnitude and extent of the impact of tropical cyclones, would allow communities to deploy resources, and create customized evacuation plans with high regional resolution, enabling targeted evacuations, preventing crowding and heavy traffic on evacuation routes [12, 13], enhancing security for susceptible groups such as individuals with restricted mobility or limited resources to evacuate at the last minute (for example citizens that do not owe a car or are institutionalized/hospitalized) [14] and navigating cascading events like the impact of COVID-19 capacity restrictions on finding shelter [15]. A step further would be to generate adaptive evacuation routes overlayed with traffic monitoring tools, that are updated in real-time, providing flexible solutions for evacuation routes as the storm gets closer to landfall (optimizing the route based on traffic and time to reach a safe destination rather than shortest distance to safety).

The National Oceanic and Atmospheric Administration (NOAA) and other meteorological agencies forecasts are of critical importance to the management of evacuations in coastal regions susceptible to severe weather events. Such agencies typically forecast hurricane tracks through advisories that contain information related to the expected landfall region, storm intensity at the time of landfall and the actual hour of landfall, given the history of the storm thus far and the recorded storm characteristics of that moment. These advisories they issue, are updated every six hours with new ones that contain further information as the phenomenon evolves, making increasingly more accurate estimates as the storm approaches landfall. For such forecasts, they typically use a variety of predictive models (statistical, physical, etc.) to ensure that modeling errors are not impacting the estimates, and different resolutions and modeling assumptions are further neutralized. In order to provide timely estimates though, these operational models typically have low spatial resolution typically over a coarse grid of limited locations [12, 16]. Such methods create difficulties for highly localized decision-making (resource allocation, community-level evacuation orders, etc.), even though they offer helpful insights into broad evacuation plans and large-scale emergency management. Small-scale differences in storm impacts such as localized storm surges flooding and wind intensity are not captured by such relatively low-resolution models [12, 17]. This prevents emergency managers from working with location-specific information and providing instructions that could result in more effective, timely, and focused evacuations. This lack of detail can result in over-evacuation in some places and under-preparation in others, adding to the challenge of allocating resources and have the public's trust on such decisions, making it difficult to adequately protect all populations that are at risk.

Despite ongoing advancements in high-resolution numerical modeling that can capture the physical phenomenon of storm propagation and other physical processes directly associated with that, the computational demands of operationalizing these models on large scales with hard time constraints are posing a significant challenge. This means that although they can accurately predict outcomes, they are unable to handle a large number of hypothetical scenarios that are essential for quantifying risk for evacuation planning in coastal areas within the given time frame [12, 17].

To balance accuracy, computational efficiency (or cost in terms of execution time and memory requirements) and the imposed by the phenomenon time constraints, one needs to integrate in computational models, such as surrogate models, that are able to leverage datasets from high-fidelity expensive simulations, to generate predictive models that will be providing storm surge estimates projected at a very high resolution across an affected region, with as much accuracy as possible. The utility of such tools for guiding decisions of emergency response managers is widely recognized as essential for Disaster Risk Reduction (DRR) efforts [18-22]. Surrogate models show promise in closing this existing gap of integrating high accuracy physics-based models in real-time hazard quantification settings.

Surrogate models have shown great abilities in substituting computational routines such as high-fidelity numerical models, in a faithful way, decreasing the associated computational burden that the original model would have. They are able to successfully map complex processes by using databases produced by high fidelity models to train and calibrate their parameters [23-25]. Many such data-driven models are available, with the most popular ones being: Gaussian Process (GP) Regression or Kriging [25, 26], Support Vector Machines (SVM) [27], Radial Basis Functions (RBF) [28] and many more. Every surrogate model is unique in its way and has the power to work best depending on the problem description, complexity, potential constraints, as well as the dimensionality of the data, with all of them showing promising results in giving accurate estimates for uncertainty quantification (UQ) settings. In this investigation Gaussian Process Regression will be leveraged, that is able to provide storm surge predictions, quantifying simultaneously the uncertainty about the estimate itself, paving the way for more comprehensive risk estimations [26, 29-31]. Along with quantifiable uncertainty bounds, GPs have been successful in handling multi-input problems and make predictions using straightforward matrix operations in a very little computational time [31]. This GP feature is important in risk assessment settings, especially in Monte Carlo-based estimations where multiple predictions are needed to better quantify risk and uncertainty. The challenge in establishing a reliable surrogate model, as in every other data-driven technique, is associated with the robust training/calibration of the model. Once the model is setup and the training is complete, the model is able to provide predictions very fast [32, 33]. The research in this thesis focuses on working towards real-time predictions of time-series predictions using a GP surrogate model.

1.2 Overview of Storm Surge Surrogate Modelling

Past decades of progress in numerical modeling and computer sciences have enabled the development of high-fidelity simulation models that are able to offer detailed approximations of the hydrodynamic processes and their interactions [34, 35]. These models, often called as "high-fidelity" due to their ability to capture reliably physical processes, although capable of supporting high-accuracy forecasting, have a significant computational burden requiring thousands of CPU hours per simulation [36]. This poses a significant restriction on their implementation to tasks such as emergency response management and storm risk assessment, where a significant number of runs is needed. During evolving landfalling storm events, high-accuracy models can be deployed to execute only a small number of hypothetical storm paths, given the time restrictions. High-accuracy models cannot support the exploration of thousand-run ensembles to quantify the impact of forecasting errors in the predicted track or the production of fast prediction updates once new storm track information becomes available through a new advisory [37].

In order to overcome these computational constraints a number of predictive techniques, including interpolation methods and surrogate models have been investigated including the implementation of response functions, selection of database storms based on similarity indices, artificial neural networks or kriging [31, 38-41]. These data-driven approaches are able to provide quick estimates for the quantities of interest, such as the expected storm surge, while maintaining similar accuracy to the numerical model that produced the original database of high-fidelity synthetic storm simulations. Thus, such methods, are able to significantly improve computational efficiency which makes them ideal for applications that involve emergency management [42, 43] and thorough probabilistic assessments of coastal risk [44, 45]. Additional support in the efficient and robust development for these data-driven methods, comes from the development and maintenance of high-fidelity databases for regional flood and coastal hazard studies [46-49].

In these data-driven input/output approximations, the typical input for the model is associated to the features that describe uniquely each storm track, for example the parameters of the storm wind-field model, while the output of interest may vary depending on the investigation: for example storm surge (either maximum value for a location or a time-history surge evolution), wind field velocities or the wave heights for various geographic locations known also as nodes may be considered. Coastal offshore, onshore, and inland areas are covered by grid nodes which may correspond directly to the computational nodes of the underlying numerical simulation model (the one that was used to get the initial database and the respective surge estimates) or to some few, representative locations called “save points” (SP). Depending on the region, these geographic locations of interest can include several areas in the model domain. It should be noted that surrogate models are quite versatile when it comes to their development, since any high-fidelity simulation model can be used for the numerical simulation of the storm database, with any desired parameterization of the synthetic storm database and any output/response quantity of interest. As for most data driven tools that are currently popular in Machine Learning (ML), there is no requirement for any explicit information regarding the mapping details of the input/output relationship, but rather the inherent variability between each input/output pair needs to be mapped out so that the proper ML model is used for the task at hand. One popular emulator, is Kriging, which has been used extensively in a number of studies [31, 50, 51] as the surrogate modeling technique of choice. Kriging provides precise storm surge and wave forecasts (both peak and time-series [50, 51] formats have been handled efficiently), even when trained upon databases of different sizes [31, 50, 51], and for regions of interest of different scales [50]. Kriging has also been combined with dimensionality reduction techniques, primarily principal component analysis (PCA) [52], to enhance computational efficiency and allow the simultaneous prediction of storm surge for thousands of locations within the domain of interest. It has been also used in the broader flood risk estimation literature to address the joint rainfall-runoff and storm surge response of tropical cyclones [53] as well as for the risk assessment of estuaries [54].

An important unexplored topic in the studies mentioned above is related to the efficient development of metamodels for time-series storm surge predictions for a large number of near-shore nodes within the domain of interest. This topic will be the main topic of this thesis, with some existing literature review and current research gaps presented below.

1.3 Storm Surge Time-series Predictions over Large Geospatial Regions

To effectively support flood preparedness, emergency response managers must be able to predict how inundation levels will rise and recede with the region of interest to be able to guide timely people and first responders. This extends to the ability to forecast storm surge time-series within a window that extends several days before and after a storm to capture the temporal evolution of surge [41]. This turns the problem into a spatio-temporal one, that involves hundreds of time steps and multiple spatially distributed points (locations), constraints that are greatly increasing the computational demands in terms of memory and time necessary to run numerical models with hypothetical scenarios.

So far, a limited number of studies has dealt with time-series forecasting which is frequently simplified as a task by using a limited number of locations, called save points (SPs) or even targeted locations (typically less than 10 within a region), with the majority of the studies focusing on peak surge prediction excluding the time dependence [50, 55]. Methods that have been used in the past to provide such predictions in a surrogate modeling setting, include Kriging with multiple outputs for each time instance (parallel formulation), separable Gaussian Processes, Kriging with time evolution as part of the input, and different nature of artificial neural networks (ANNs). Kriging in particular, has demonstrated its computational efficiency and scalability across large geospatial domains, indicating its potential for accurate time-series storm surge predictions [56].

So far, the high dimensionality of the output, has been addressed by combining Principal Component Analysis (PCA) [31] with the surrogate model formulation. which enables effective predictions over large geospatial domains. Some other efforts are deploying ANNs and combinations of surrogate models for a very small number of locations (couple of hundreds) [57, 58], aiming to capture the nonlinear behavior of the phenomenon for certain locations. The investigation presented in this thesis is motivated by recently published literature [59], that develop surrogate models trained on database of synthetic storms (Coastal Hazards System–North Atlantic Coastal Comprehensive Study (CHS-NACCS) [60]) to accurately simulate time-series storm surge. Gaussian Process (GP) techniques (including a separable formulation) or Deep ANNs are used to develop a predictive surrogate model for substituting the need of an expensive numerical model. To tackle the computational complexity stemming from time series data, linear and nonlinear dimensionality reduction techniques are employed for enhanced model accuracy and efficiency.

The most popular thus far linear dimensionality reduction technique for reducing the dimensionality of the output space has proven to be PCA [50, 59], enabling the development of more efficient surrogate models that are trained explicitly on each latent dimension. The original study by Jia, Taflanidis [50] was the first one that aimed to develop a predictive model for surge time series, utilizing PCA to reduce the dimensionality of the output. In that investigation, only a limited number of locations was selected and successfully predicted. Later on efforts, managed to extend such predictions, using again the combination of PCA and Kriging, to larger datasets counting thousands of locations [59]. PCA [61] was successfully implemented in those aforementioned cases, leveraging the spatial and temporal underlying correlations in the high-dimensional data by capturing significant variance in the data utilizing fewer principal components. This allowed for the training of multiple surrogate models explicitly on each component achieving significantly high accuracy despite the problem complexity. Though PCA successfully reduced the data, as a linear dimensionality technique, it was unable to capture the cross-storm nonlinear behavior for many locations within the domain. As a continuation of this work, focusing on the non-linearity of the surge across different storm scenarios becomes essential for predicting more accurately locations that might be experiencing such behavior, and thus advanced techniques like autoencoders have emerged as techniques that are able to better capture these complex patterns.

Nonlinear dimensionality reduction techniques like AutoEncoders (AE) have been used in storm surge time-series predictions to a lesser extent [55, 62, 63] and for a relatively small number of locations. Previous work has considered only a small number of locations of interest (a couple of hundreds) for predicting storm surge time-series, where a Deep AutoEncoder (DAE) was combined with Deep Neural Network (DNN) to successfully predict surge. The latent representation obtained from DAE was utilized as the output quantity of interest in the DNN with the input comprising of the storm parameters to predict the experienced surge [62]. The DNN then predicts the latent space representation and the decoder reconstructs that back to generate the predictions of whole area. Further the work on this topic, was extended to Convolutional AutoEncoders (CAEs) with only a limited number of spatial locations as the target (around 600) [63]. The data from dimensionality reduction was further used with Hierarchical Deep Neural Network (HDNN) for storm surge predictions [63] demonstrating extremely good accuracy. CAE was deployed to reduce the dimensionality of the data, and the latent dimensions obtained from CAE were further used along with storm parameters to develop the HDNN. The HDNN was able to map the storm parameters to the lower dimensional representation, enabling the sequential prediction across different time scales. The area of interest in this case was selected to include locations in New York and New Jersey, totaling around 600 points of interest [63]. Results demonstrate really good agreement for both time-series for specific storms and locations, as well as spatially (for certain time steps and storms). Comparison though are never established for the entire domain, time steps and across storms to provide an estimate of the total predictive accuracy of the framework.

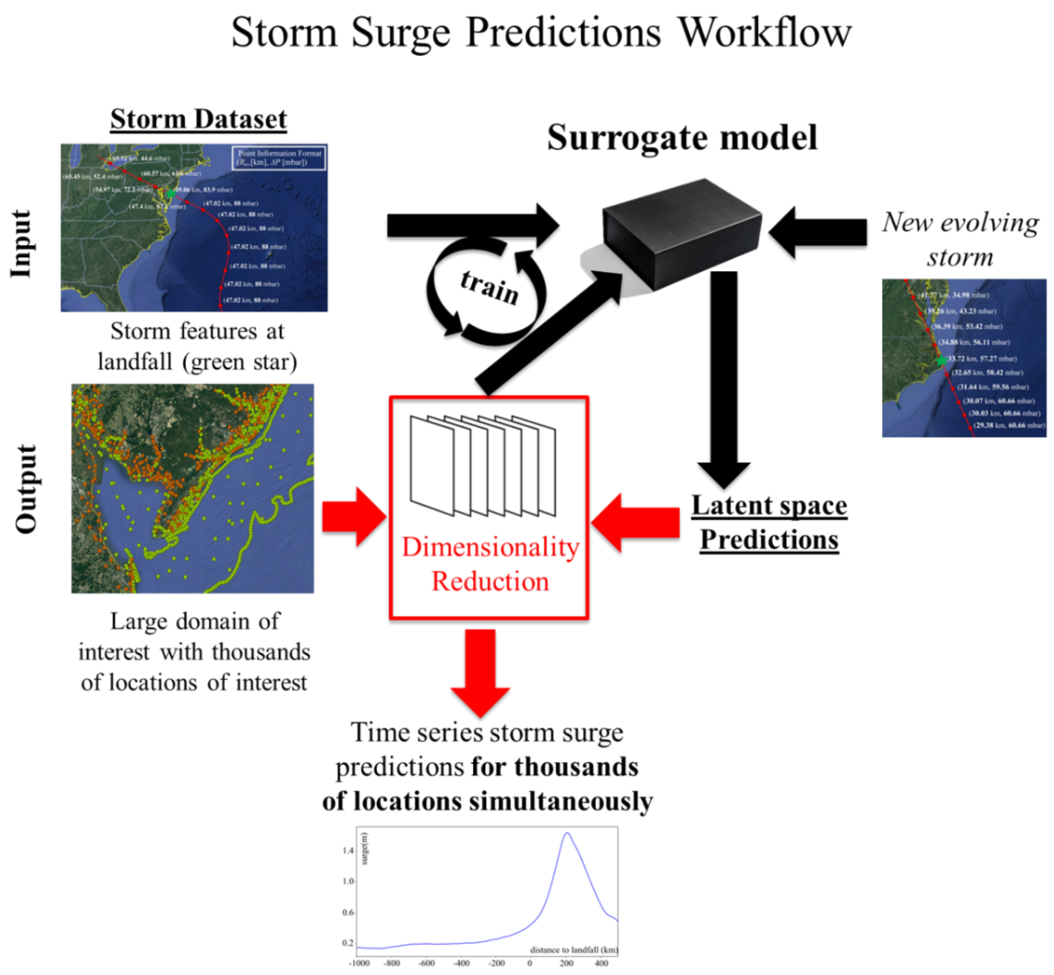

Storm Surge Predictions Workflow
Storm Dataset
Input
Storm features at landfall (green star)
Output
Large domain of interest with thousands of locations of interest
Surrogate model
train
New evolving storm
Dimensionality Reduction
Latent space Predictions
Time series storm surge predictions for thousands of locations simultaneously
surge(m)
distance to landfall (km)

Figure 1-1: The overall surrogate modeling framework along with the focus of the thesis highlighted.

This thesis aims to provide a surrogate model framework that will leverage nonlinear dimensionality reduction techniques to provide time-series storm surge predictions over large spatial domain, containing a couple of thousands of locations of interest. The goal is to develop a data-driven model, that will be able to predict time-series storm surge for certain locations and time steps across different storm scenarios. The storm evolution in each time step, for a certain location, across all storms, it tends to experience nonlinear characteristics, that are related both to the nonlinearities of the phenomenon itself (especially for larger category storms), but also to the local geomorphology of the location (whether surge can accumulate or drain if there is a grade in the region, the pre-existence of standing water, etc.). To achieve higher efficiency in the predictions and minimize the time needed for computations the surrogate model (GP-based) will be trained on the latent data as its output, while the input will contain the storm description as described in detail in Chapter 2. The overall workflow of the storm surge prediction and the contribution of this thesis is provided in Figure 1-1.

1.4 Research Objectives

Overall, the proposed research presented here seeks to explore nonlinear dimensionality reduction (NLDR) techniques within a GP surrogate modeling formulation for predicting fast and accurate storm surge time-series, with the below mentioned main research objectives. All the objectives focus on the implementation of nonlinear dimensionality reduction techniques and combine them with pre-existing surrogate modeling formulations to consistently compare efficiency and accuracy benefits with current practices. The research objectives are to:

1) Investigate a variety of nonlinear dimensionality techniques and select the ones that better fit the problem and the available data. Address challenges related to interpretability, invertibility, and overfitting. This objective should yield several candidate methods/ML techniques that will be explored further in the objectives below.
2) Implement such Dimensionality Reduction (DR) techniques in a certain programming environment and transfer the surrogate model development to the

same environment to facilitate and streamline computations to achieve computational efficiency.

3) Explore different formulations and architectures for the selected and implemented DR techniques, here an in-depth understanding and investigation of each technique is warranted in order to select the most appropriate one for the problem at hand. In this analysis, the techniques will be evaluated by aiming to balance among availability of information, model overfitting, and over interpretation of trends or results, ensuring simultaneously that predictions are made over large geospatial domains for at least a couple of thousands of nodes. Here a rigorous comparison should be established across problematic locations (that experience high nonlinearities in the storm surge) between linear (LDR) and nonlinear (NLDR) techniques, but also among nonlinear techniques and architectures of certain methods to identify the one that can better capture the surge behavior without overfitting in the dataset.
4) Validate the DR techniques' accuracy when combined with a surrogate model to offer time-series predictions and identify the one that could achieve a higher accuracy for onshore locations with high nonlinearities. This last step is necessary to understand whether a nonlinear dimensionality reduction technique is disproportionally affecting the surrogate model accuracy across all locations (even the ones that PCA was facilitating an accurate prediction). Some comparative investigations with the already established model of GP and PCA will be performed over the same test dataset to facilitate consistent comparisons and enhance conclusions.

The remainder of this dissertation is organized as follows. In the next chapter (Chapter 2) the storm surge estimation problem is briefly reviewed, followed by a brief description of the surrogate model formulation (Gaussian Processes) that will be used throughout the investigation. A brief description of the storm database that will be used to inform the metamodel developments is also offered in the same chapter. The advances of this investigation are not database-specific, but they will have a broader implementation on any synthetic storm surge time-series database that will become available. As part of Objective 1, Chapter 3 discusses the investigation on the available nonlinear DR techniques, offering a comprehensive literature review on the state-of-the-art methodologies. Chapter 4 offers a brief description of the programming skills developed throughout the project implementation and the practical challenges that were tackled, aligning with Objective 2, while Chapter 5 presents the exploration and development of the different DR techniques (Objective 3). Chapter 6 addresses the final objective (Objective 4), by using the latent representations in the surrogate model developments, to consistently compare the predictive performance of the different selected DR techniques. Finally, Chapter 7 offers the conclusions and the additional commentary with respect to the work presented in this Master’s thesis, along with the plans for future work.

CHAPTER 2:
DATABASE DESCRIPTION AND FUNDAMENTAL FORMULATION OF GAUSSIAN PROCESS SURROGATE MODELING

This chapter offers a brief description of the database that will be leveraged here to train the surrogate model to provide time-series storm surge predictions, identifying the challenges that motivated this research, especially for onshore locations, along with a brief storm surrogate modeling formulation description. The surrogate model input selection (storm parametrization), the normalization of the surge output and the validation metrics for the surrogate model implementation will be also described here. A Gaussian Process surrogate model will be used here, following past work on storm surge time series for accommodating large geospatial regions [64]. This chapter supports the advancements that are made in this investigation for reducing effectively the dimensionality of the spatio-temporal output, utilizing the same type surrogate model (appropriately tuned for the available input/output), to establish a consistent comparison between current methods and the ones explored here.

2.1 High-fidelity Numerical Models for Storm Surge Estimation

To quantify storm surge, high-fidelity numerical hydrodynamic models such as the Advanced Circulation model [35], that utilize a parametric description of the storm wind-field model have been developed. The typical input to these models is uniquely describing the storm from its formation, as it travels through the ocean to reach landfall, to even after the storm starts dissipating as it moves further inland. For that description the time evolution of the storm track (latitude and longitude of storm center), intensity (pressure difference between storm center and ambient temperature) and size (radius of maximum winds) are key modeling input parameters.

With the evolving storm characteristics described for each storm scenario, a parametric description of the storm wind-field can be obtained to drive the numerical simulation for the estimation of the storm surge [35, 65]. In some cases, additional parameters to the aforementioned parametric description might be necessary, like the Holland B number [66]. It should be stressed here that although mathematical relationships relating such parameters to the track/size/intensity characteristics exist, variability in the regression estimates might generate further input parameters for the wind-field description. The vector composed of the storm characteristics used as input by the storm surge numerical model will be denoted herein as $\mathbf{x}(t)$ where t indicates the time evolution of the input. With the primary storm characteristics available, secondary ones like the storm heading and translational speed can be derived. Such secondary characteristics are frequently used to characterize the storms or define the main characteristics in the context of the Joint Probability Method (JPM), which allows for the identification of synthetic storm scenarios. The output of interest for this investigation coming from the high-fidelity numerical model is the predicted surge at the different computational grid nodes, that represent different geographical locations. Denoting surge by ζ the functional output relationship for the output is expressed as $\zeta(t, \mathbf{s}, |\{x(\tau), \tau[0, \tau]\})$, where **s** is the vector with spatial information (latitude and longitude) for each grid node and "|" denotes the dependency relationship of the quantities that lie on the left of the symbol to quantities that lie on the right. For this case, this indicates the storm surge from the beginning of the storm until time instance τ given the storm input characteristics for the same time window. This means that for a given storm input the surge is provided at different discrete times and locations (nodes), and quite frequently the output is further simplified by obtaining the peak surge instead of the time-series.

Studies that aim to describe and quantify storm-related coastal hazards [46-48, 67, 68] typically, perform a large number of synthetic storm simulations, offering storm databases that can be leveraged to support the development of surrogate models for storm surge predictions. The ensemble of synthetic storms is carefully selected based on the historical storm data [69, 70], and/or local climatology [71] of the region under investigation. The most widely used approach for the storm ensemble selection is the Joint-Probability Method (JPM) [72, 73], widely treated as the standard practice for the determination of hurricane surge probabilities of occurrence [74]. Though different variants of JPM exist, the formulation follows common principles.

Each storm from the ensemble is characterized by unique features taken with respect to a reference point, corresponding to the storm landfall for landfilling storms. These features typically include: the track location, the heading, the forward speed, the radius of maximum winds (size), and the central pressure (intensity). Based on these features and regional historical data the time evolution of the storm input $\mathbf{x}$ (t) can be identified for each of the synthetic storms in the storm ensemble. This storm time-evolution with respect to the storm characteristics may vary from study to study with some databases having really small and gradual pre-landfall characteristics while post-landfall changes in size and intensity typically tend to be large and abrupt. These changes, tend to be consistent with basic tropical cyclone physics (with the exception of rapidly intensifying storms), which relies on the fact that a storm will weaken and widen when it moves inland where the warm, moist air it feeds upon is cut off. In addition to this, the synthetic storm ensemble tracks are based on a typically small (in the range of five to ten) master tracks or MTs. For each of these MTs similar along track characteristics are shared, but each MT corresponds to a different landfall location. This typically allows for a more organized exploration of the landfalling location impact, for the same storm characteristics. This distinction of storms to MTs is a key feature in most of JPM variants.

2.2 Database Formulation for Surrogate Model Development

With regional coastal hazard studies providing high-fidelity hydrodynamic simulation databases, that include storm surge estimates, an input/output relationship can be established for each synthetic storm simulation. This description of the storm input $\mathbf{x}(t)$ is uniquely connected to the surge output $\zeta(t,\mathbf{s})$ for each time instance across all the locations within the domain of interest for each of the storms within the database (typically each MT). Such databases include information for other quantities of interest such as wave height, current velocities, but focus here will be placed explicitly on the time evolution of the storm surge.

Moving to the database output, this is typically provided in for specific geographic locations only, corresponding to either all the computational nodes of the numerical model or to some selective save points (SPs). In the former case, the locations follow the discretization of the high-fidelity model, with a very dense grid, leading to thousands or even above a million locations within the domain of interest, while in the latter, a smaller number of locations are defined to cover the area of interest at a lower resolution. The distribution of nodes and SPs in either case is not expected to follow any canonical pattern. Notation $_i$ is used herein to distinguish the output for location *i*, with ζ_i denoting, for example, the peak surge for node or SP *i*. Here the grid points will be directly used, so no additional commentary will be offered on the SP side and how these locations are identified to downscale the spatial density of the grid.

2.3 Parametrization of Storm Database and Surrogate Model Mathematical Description

In order to train any surrogate model for time-series storm surge prediction, an appropriate parameterization of the synthetic storm database is warranted. This parameterization needs to capture all the important storm features that uniquely identify each storm, striving to remain simple enough to avoid over-parametrization, that can cause a loss in computational accuracy. The goal for this input parametrization is to express the temporal variability of the storm characteristics along the track path with a limited number of parameters, corresponding to some specific instance, typically the time of landfall. This parametrization includes features related to the storm's path (track) and other characteristics such as storm intensity/size/speed. For the latter, the central pressure deficit ΔP, forward speed v_f, radius of maximum winds R_{mw} and *Holland B* parameter can be utilized, while for the former the latitude s_{lat} and longitude s_{long}, as well as the heading direction (angle) ξ, can be used. With these characteristics identified, a replacement of $\mathbf{x}(t)$ with some n_x dimensional vector $\mathbf{x}$ (that is independent of time) is achieved, serving as the input to the surrogate model. The definition of $\mathbf{x}$ requires the selection of some reference time instance t_{ref} along the storm track where the storm characteristics at that specific time instance will be taken at the representative ones for the storm. A more robust alternative would be to consider the variations around landfall by averaging the storm characteristics over a time window around that instance, using ultimately those averaged quantities to inform input $\mathbf{x}$. Here, the characteristics at landfall will be directly used to inform the surrogate model development, which is the common practice for surrogate model developments [41, 45, 75]. This is identical to the implementation adopted for the JPM storm parameterization, corresponding to storm features directly provided in most coastal hazard studies. The geographic location points when each landfalling storm crosses the shoreline, and the storm characteristics at those points are used to describe the temporal evolution for the entire storm. To avoid any inconsistencies stemming from the local coastline geomorphology (bays, river deltas, etc.) a simplified, linearized boundary of the coast might be selected as the landfalling location. This approach can facilitate a more consistent storm parametrization, correlating effectively storms with similar tracks within the database that have similar landfalling locations. This leads to the n_x dimensional input vector $\mathbf{x}$ for each storm, consisting typically of the reference point latitude x_{lat} and longitude x_{long}, the heading direction ξ, the pressure deficit ΔP, the radius of maximum winds R_{mw} and the forward speed v_f, leading to:

$$\mathbf{x} = \begin{bmatrix} x_{lat} \\ x_{long} \\ \xi \\ \Delta P \\ R_{mw} \\ v_f \end{bmatrix} \tag{0.1}$$

Depending on the study, the *Holland B* number might need to be additionally considered as an input parameter, if the value utilized when creating the database cannot be immediately derived from the rest of the above parameters, for the database that will be used in this thesis, *Holland B* is not considered an extra input parameter.

The storm parametrization leads to a database description with n total storm scenarios of $\{\mathbf{x}^h; h = 1,\ldots, n\}$ with the superscript $.^h$ distinguishing between different storm and each individual component of $\mathbf{x}$ denoted as x_i.

The database of high-fidelity storm simulations provides predictions of the surge for n total storms across a number of n_s different locations. An imputation for the database will be needed for some of the onshore/nearshore locations that have remained dry for some of the storms within the available database as detailed in [64]. The surrogate model developed to provide the surge predictions, has as input the vector $\mathbf{x} \in \Re^{n_x}$ obtained through the storm parameterization and as output the surge vector at specific time-intervals, i.e. for every 10 to 15 mins. For typical databases like the one used herein, these series describe the evolution of the surge response, as the an independent variable, the distance *dtr* along the track is selected based on recommendations and comparisons that were made in [59]. Most of the surrogate model formulations that have been developed so far for predicting the temporal variability of storm surge have used time as the independent variable [41, 50, 76, 77], but here the use of distance will be investigated as a much more versatile option.

Nevertheless, the time or distance to landfall are directly related through the translational storm speed at each time instance, allowing an easy transition from one independent variable to the other.

Each storm simulation is typically recorded in the numerical model, based on the proximity of that storm to the coast, with the length of the surge data-series being generally for each storm different. This discrepancies in the lengths of timeseries, do not allow for the establishment of a consistent reference for independent variable values among the different storms. For the surrogate model development, the synchronization of the series-data is required as well as the employment of an interpolation scheme to produce data that corresponds to the same n_t instances. With the variation of the series data being relatively smooth, a linear interpolation was employed. For the synchronization of the series, the landfall was used as the point with 0 distance, with negative values spanning before and positive values spanning after. For the GP development, it should be noted that the data series do not need to be equally spaced along the evolution of the storm. In fact, more closely-spaced values around landfall should be selected, since the stronger – and potentially faster evolving – the storm is, it is expected to have a much faster change in the surge that is bringing along, and that potentially will also impact a much larger geographic domain.

With $\zeta(n_t, \mathbf{s}, \mathbf{x})$ denoting the storm surge for a location that corresponds to coordinates $\mathbf{s}$, with the storm causing it described through the parametrization $\mathbf{x}$, and the series instance characterized by variable n_t, different subscripts are established to characterize the different elements of vectors or matrices and superscripts to distinguish elements of a set. For example, x_i denotes the i-th component of the vector $\mathbf{x}$, whereas x^h

describes the *h*-th storm parametrization within the database. The synthetic storm database ultimately provides a three-dimensional observation matrix $\mathbf{Z} \in \Re^{n_t \times n_s \times n}$ whose Z_{ijh} elements correspond to the *i*-th series instance (n_t^i value), *j*-th node ($\mathbf{s}^j$ coordinates) and *h*-th storm ($\mathbf{x}^h$ storm parametrization). In addition to this, the database is also providing the input matrices for all the quantities in the series, meaning that distance to landfall is given by $\mathbf{D} = \left[d^1, ..., d^{n_t} \right] \in \mathbb{R}^{n_t \times 1}$, node locations are given by $\mathbf{S} = \left[\mathbf{s}^1, ..., \mathbf{s}^{n_s} \right] \in \mathbb{R}^{n_s \times 2}$ and storm characteristics as $\mathbf{X} = \left[\mathbf{x}^1, ..., \mathbf{x}^n \right] \in \mathbb{R}^{n \times 6}$. Bold capital letters will be utilized for all the available data, even though some like **D** correspond to vectorized quantities. The surrogate model's objective is to establish a GP based approximation for the experienced storm surge $\zeta(n_t, \mathbf{s}, \mathbf{x})$ utilizing the above input matrices. Emphasis of the emulation in the application examined here is on the prediction of the storm surge for new storms (that do not belong in **X** dataset), but for the same nodes and series values prescribed by **S** and **D**, respectively.

Here the GP model will be established for the latent representation that will be produced through the AE implementation. This means that, although the input will corresponding to matrix **X**, the output will be a matrix of latent dimensions $\mathbf{L}(\mathbf{X}) = \left[\mathbf{l}(\mathbf{x})^1, ..., \mathbf{l}(\mathbf{x})^n \right] \in \mathbb{R}^{n \times n_l}$. In this GP approach, matrices **D** and **S** are not utilized, since the responses for different nodes and series instances are treated as different outputs, with no effort to establish a parameterized relationship between them (through use of **D** and **S**). Instead, the GP formulation will infer potential correlation among these responses from a pure data-driven perspective. Due to the small number of latent variables ($n_l < 200$) the establishment of the GP is fairly simple without any concerns regarding invertibility or handling of the data. One surrogate model is developed here, with the calibration stage being of complexity $O\left(n_l^3\right)$.

Gaussian Process (GP) or kriging approximates the true response as one realization of a Gaussian process (GP) along with an underlying linear regression term [25]. The fundamental building blocks of the GP process are the n_b dimensional basis vector, **f**(**x**), and the correlation function $R(x_l,x_m|\boldsymbol{\theta})$ that for a zero mean stationary stochastic Gaussian process is denoted as G(**x**), with **θ** representing the vector of hyper-parameters. The basis vector is responsible for the established regression model, that is expressed as a combination of **f**(**x**) through a coefficient vector **β**, with the addition of the GP model, leading to the representation of for the *j*-th element of **l**:

$$l_j(\mathbf{x}) = \mathbf{f}(\mathbf{x})^T \boldsymbol{\beta}_j + G_j(\mathbf{x}) \tag{2}$$

For the correlation function, here the power exponential will be utilized that is recognized as one of the most versatile ones [33]. The function is given by:

$$R(\mathbf{x}^l, \mathbf{x}^m \mid \boldsymbol{\theta}) = \exp[-\sum_{j=1}^{n_x} \theta_j \mid \mathbf{x}_j^l - \mathbf{x}_j^m \mid^{\theta_{n_x+1}}] \; ; \; \boldsymbol{\theta} = [\theta_1 \; \cdots \; \theta_{n_x+1}] \tag{3}$$

For **f**(**x**), a common choice is to use some low-order polynomial [33], though again specialized selections can be established. For example, for the linear polynomial selection, **f**(**x**) corresponds to:

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} 1 & x_1 & x_2 & \ldots & x_{n_x} \end{bmatrix}^T \tag{4}$$

whereas if a complete quadratic is included for components x_1 and x_2, **f**(**x**) transforms to:

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} 1 & x_1 & \ldots & x_{n_x} & x_1^2 & x_1 x_2 & x_2^2 \end{bmatrix}^T \tag{5}$$

The regression facilitated through the above is aiming to de-trend the data, while the GP targets to fit the residuals through the selected correlation function. The characteristics of the correlation function are responsible for the induced smoothness of the surrogate model approximations (in terms of both continuity and differentiability), assuming that the regression function (which here is a polynomial) is sufficiently smooth. For the set of n observations with the input matrix $\mathbf{X}$ as identified above, a $n \times n_b$ basis matrix $\mathbf{F}(\mathbf{X}) = \left[\mathbf{f}(\mathbf{x}^1),...,\mathbf{f}(\mathbf{x}^n)\right]^T$ and the $n \times n$ correlation matrix $\mathbf{R}(\mathbf{X})$ with the km-element defined as $R(\mathbf{x}^k, \mathbf{x}^m \,|\, \boldsymbol{\theta})$ with k, m = 1, …, n. In order to improve the numerical stability of the correlation matrix, or even its accuracy when fitting noisy data [78]an extra term, called nugget is included in the formulation, leading to:

$$\underline{\mathbf{R}}(\mathbf{X}) = \mathbf{R}(\mathbf{X}) + \delta \mathbf{I}_n \tag{6}$$

where δ is the nugget value and $\mathbf{I}_n$ is the identity matrix of dimension n. For every new input $\mathbf{x}$, a n-dimensional correlation vector $\mathbf{r}(\mathbf{x}|\mathbf{X}) = [\mathrm{R}(\mathbf{x},\mathbf{x}^1|\mathbf{s}) \ldots \mathrm{R}(\mathbf{x},\mathbf{x}^n|\mathbf{s})]^T$ between the new input and each of the elements of $\mathbf{X}$ is established. The GP prediction (given in the form of a row vector), corresponding to the predictive mean, is given by [25]:

$$\begin{aligned}\tilde{\mathbf{l}}(\mathbf{x} \,|\, \mathbf{X}) &= \mathbf{f}(\mathbf{x})^T \boldsymbol{\beta}^*(\mathbf{X}) + \mathbf{r}(\mathbf{x} \,|\, \mathbf{X})^T \underline{\mathbf{R}}(\mathbf{X})^{-1}(\mathbf{L}(\mathbf{X}) - \mathbf{F}(\mathbf{X})\boldsymbol{\beta}^*) \\ &= \mathbf{f}(\mathbf{x})^T \boldsymbol{\beta}^*(\mathbf{X}) + \mathbf{r}(\mathbf{x} \,|\, \mathbf{X})^T \boldsymbol{\alpha}^*(\mathbf{X})\end{aligned} \tag{7}$$

where $\boldsymbol{\beta}^*(\mathbf{X}) \in \mathbb{R}^{n_b \times n_l}$ corresponds to the weighted least squares solution:

$$\boldsymbol{\beta}^*(\mathbf{X}) = (\mathbf{F}(\mathbf{X})^T \underline{\mathbf{R}}(\mathbf{X})^{-1} \mathbf{F}(\mathbf{X}))^{-1} \mathbf{F}(\mathbf{X})^T \underline{\mathbf{R}}(\mathbf{X})^{-1} \mathbf{L}(\mathbf{X}) \tag{8}$$

whereas $\boldsymbol{\alpha}^*(\mathbf{X}) \in \mathbb{R}^{n \times n_l}$ has been defined as:

$$\begin{aligned}\boldsymbol{\alpha}^*(\mathbf{X}) &= \underline{\mathbf{R}}(\mathbf{X})^{-1}(\mathbf{L}(\mathbf{X}) - \mathbf{F}(\mathbf{X})\boldsymbol{\beta}^*) \\ &= \underline{\mathbf{R}}(\mathbf{X})^{-1}(1 - \mathbf{F}(\mathbf{X})(\mathbf{F}(\mathbf{X})^T \underline{\mathbf{R}}(\mathbf{X})^{-1} \mathbf{F}(\mathbf{X}))^{-1} \mathbf{F}(\mathbf{X})^T \underline{\mathbf{R}}\mathbf{X})^{-1})\mathbf{L}(\mathbf{X})\end{aligned} \tag{9}$$

The prediction of $\tilde{\mathbf{l}}$ is what is also known as the best linear unbiased predictor (BLUP)[25]. When δ=0 then the model collapses to an exact interpolation for the existing dataset, while with the inclusion of the nugget, this interpolation is transformed to a regression. The matrix inversion of the $n \times n$ correlation $\underline{\mathbf{R}}(\mathbf{X})$ is the most numerically demanding task in the kriging formulation, with a computational complexity of order $O(n^3)$. The introduction of the nugget δ term is intended to address any such ill-conditioning challenges related to this inversion. Here only the predictive mean will be utilized as described above, and for the training of the surrogate model, which pertains to the selection of the hyper-parameters [$\boldsymbol{\theta}$, δ]. The regression coefficients $\boldsymbol{\beta}(\mathbf{X})$, although they could be considered as additional hyper-parameters, they are typically identified through the Maximum Likelihood Estimation (MLE) process. This leaves for calibration only the [$\boldsymbol{\theta}$, δ] vector, where two different approaches can be adopted such as MLE or leave-one-out-cross (LOOCV). For the former, the optimization is described using as objective function the negative log-likelihood of the data [79]:

$$[\boldsymbol{\theta} \quad \delta]^* = \underset{[\boldsymbol{\theta} \quad \delta]}{\arg\min} \left(\ln\left|\underline{\mathbf{R}}(\mathbf{X})\right| + n \ln \sum_{j=1}^{n_y} \tilde{\sigma}_j(\mathbf{X}) \right) \tag{10}$$

where |.| stands here for the determinant of a matrix, and $(\tilde{\sigma}_j(\mathbf{X}))^2$ is the process variance. The second alternative the hyper-parameter optimization, is to use cross-validation techniques (CV) [80], utilizing the surrogate model's predictive capabilities in the selection of the hyper-parameters. The common practice in this case is to use Leave-One-Out Cross-Validation (LOOCV) leading to the optimization:

$$[\boldsymbol{\theta} \quad \delta]^* = \underset{[\boldsymbol{\theta} \quad \delta]}{\arg\min} \frac{1}{n_y} \sum_{j=1}^{n_y} \gamma_j \left[\frac{1}{n} \sum_{h=1}^{n} \left(l_j(\mathbf{x}^h) - \tilde{l}_j(\mathbf{x}^h \mid \mathbf{X}_{-h}) \right)^2 \right] \tag{11}$$

where γ_j is the weight for the *j-th* component of response **l**, $\mathbf{X}_{-h}$ denotes the entire database excluding the $\mathbf{x}^h$ training point, and $\tilde{l}_j(\mathbf{x}^h \mid \mathbf{X}_{-h})$ is the prediction made by the surrogate for l_j for the $\mathbf{x}^h$ training point, established by using the entire database excluding that training point. In the LOOCV implementation each of the observations from the database is sequentially removed, then the remaining training points are used to predict the output for it and the error between the predicted and real responses is evaluated. Analytic expressions are facilitating the computation of $\tilde{l}_j(\mathbf{x}^h \mid \mathbf{X}_{-h})$ without the need to explicitly compute the surrogate model that corresponds to the reduced $\mathbf{X}_{-h}$ database [81, 82].

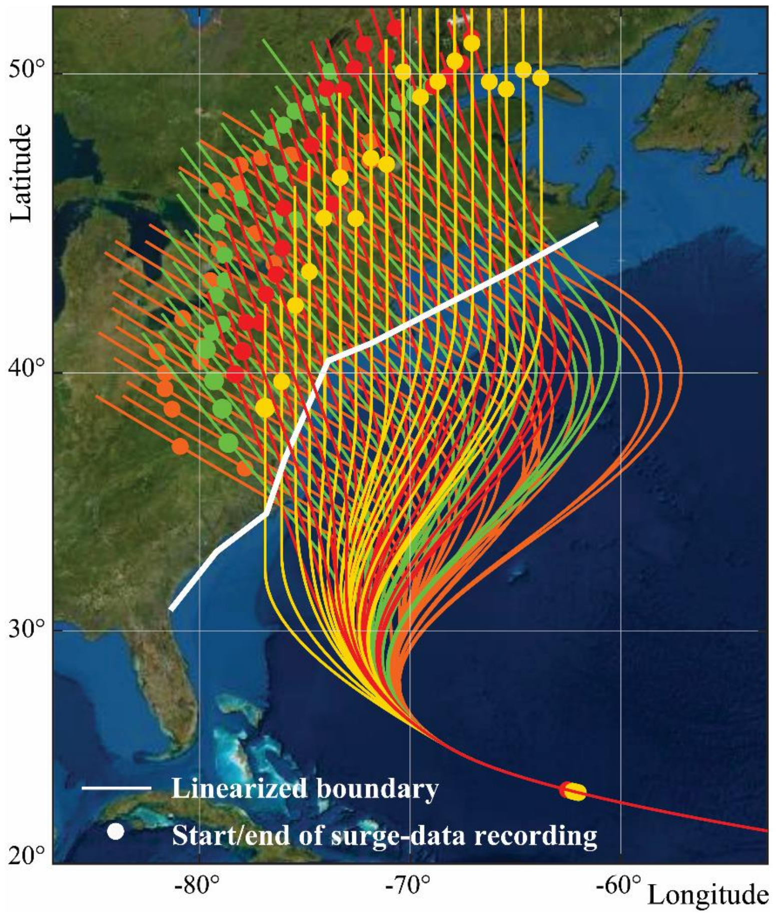


Figure 2-1: NACCS MTs and the time series evolution that has been synchronized for all storms, along with the linearized boundary for the definition of landfall.

After predictions are made by the surrogate model in the latent space, the trained Decoder part of the DAE is employed to project the latent space back to the original storm surge space, in which the final predictions should be provided for interested parties (like evacuation managers, state officials and the public).

2.4 Overview of the North Atlantic Comprehensive Coastal Study (NACCS) Database

The NACCS data is produced by the U.S. Army Corps of Engineers (USACE) as part of their Coastal Hazards Systems (CHS) program [83, 84] The original dataset consists of a total of 1031 validated synthetic storms separated into two clusters, 595 landfalling storms and 436 bypassing storms. The landfalling scenarios will be the only ones that will be used in this thesis, with 89 MTs in total as shown in Figure 2-1. Storm surge was simulated by coupling a high-fidelity hydrodynamic model composed of STWAVE (Steady-State Spectral Wave model) [85] and ADCIRC [35], with the Planetary Boundary Layer (PBL) model providing the wind and pressure feedback to the original model. The original computational domain for generating the synthetic storms consists of 3.1 million computational nodes and 6.2 million elements, including parts of the western North Atlantic, the Gulf of Mexico, and the western extent of the Caribbean Sea. A steric water level adjustment s_t of 0.1089 m was used in order to account for the elevation of the water surface above the geoid due to thermal expansion and seasonal variation in salinity. Further numerical details for the synthetic storm simulations can be found in [86].

The joint probability method with optimal sampling (JPM-OS) was used to generate the storm suite, considering the historical characteristics of regional storms. Each master track (MT) is described by vectors containing information regarding the landfall location x_{land}, the heading direction ξ at that location, the central pressure deficit ΔP, the forward speed v_f, and the radius of maximum winds R_{mw}. A simplified parametrization is established as detailed earlier, at the time that the storm has its maximum intensity [this intensity starts to reduce close to landfall [86]. The utilized input parameters (ξ, ΔP, v_f, and R_{mw}) are identical to the ones within JPM-OS framework and their distribution may be found also in [75].

Table 2-1: NACCS database for landfalling storms arranged per MT.

MT ID	Storm parameter ranges in [ ]			Storm Heading ξ (°)	Number of landfalling locations (different MTs)	Number of storms per MT
	ΔP *(mbar)*	R_{mw} *(km)*	v_f *(m/s)*			
1	[28 88]	[25.2 154.8]	[3.3 19.7]	-60	27	142
2	[28 88]	[25.8 153.6]	[3.3 19.7]	-40	31	145
3	[38 98]	[25.9 166.7]	[3.3 20.0]	-20	33	123
4	[28 88]	[25.2 170.6]	[3.3 24.4]	0	23	185

The combination of latitude (x_{lat}) and longitude (x_{long}) along with the heading direction (ξ), dictate the MT, and uniquely determine each track within each MT. The remaining parameters that dictate the storm strength (ΔP), size (R_{mw}) and speed (v_f), further distinguish storms that might correspond to the same MT. For the development of this database, the strength and size characteristics have been retained relatively constant prior to landfall, with the forward speed maintained as constant across the entire storm track [48, 49]. Table 2-1 presents an overview of the storm characteristics, with the reported values for each parameter corresponding to the ones at the peak of intensity of each storm. For the characterization of the landfall locations, a simplified linearized boundary of the coastline has been used. Some MTs for the storm that will be used here (landfalling storms), are presented in Figure 2-1 along with the linearized boundary for the characterization of the landfall locations along the coastline.

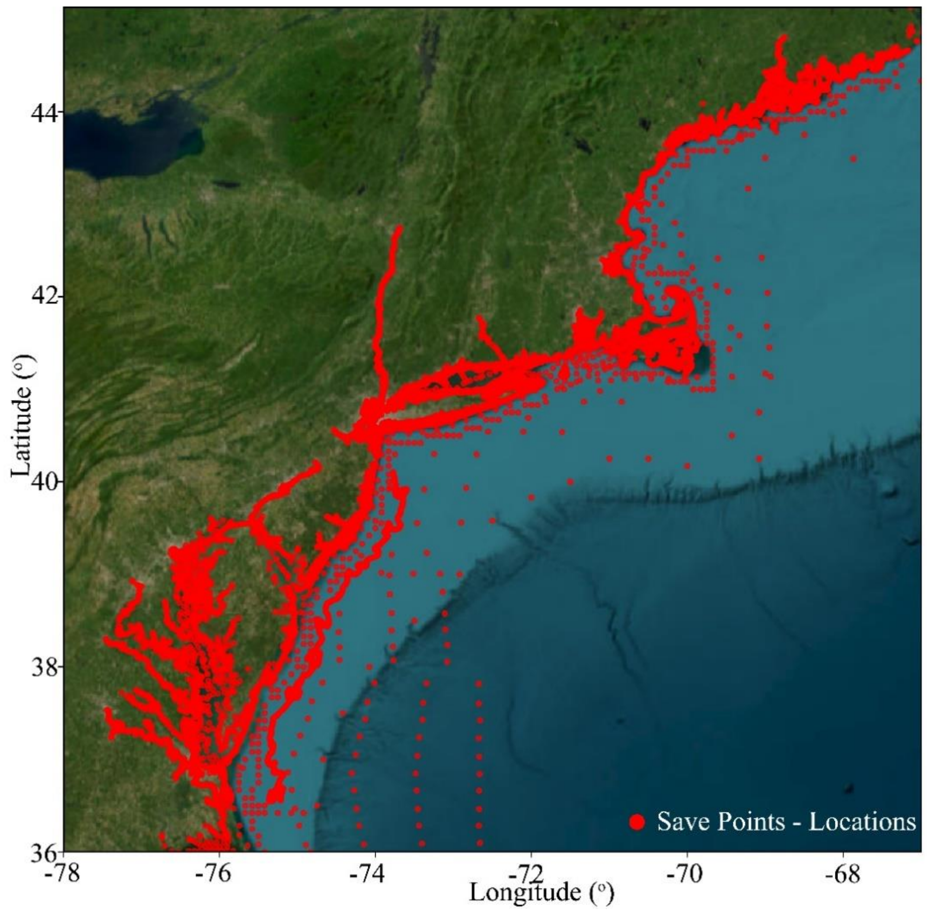


Figure 2-2: The region of interest for the NACCS database. Locations are plotted herein that are wet across all storm scenarios.

Despite the great versatility of the NACCS database, that includes more than half a million locations directly from the ADCIRC grid, only 18,977 save points (SPs) are available when it comes to time-series records. For these points/locations of interest, the entire time-series is recorded for each storm time history evolution. For accommodating near-shore and onshore SPs that either remain or become dry for some time instances for some storms in the database, a data imputation process is necessary to allow for the completion of information in order to facilitate a consistent surrogate model development.

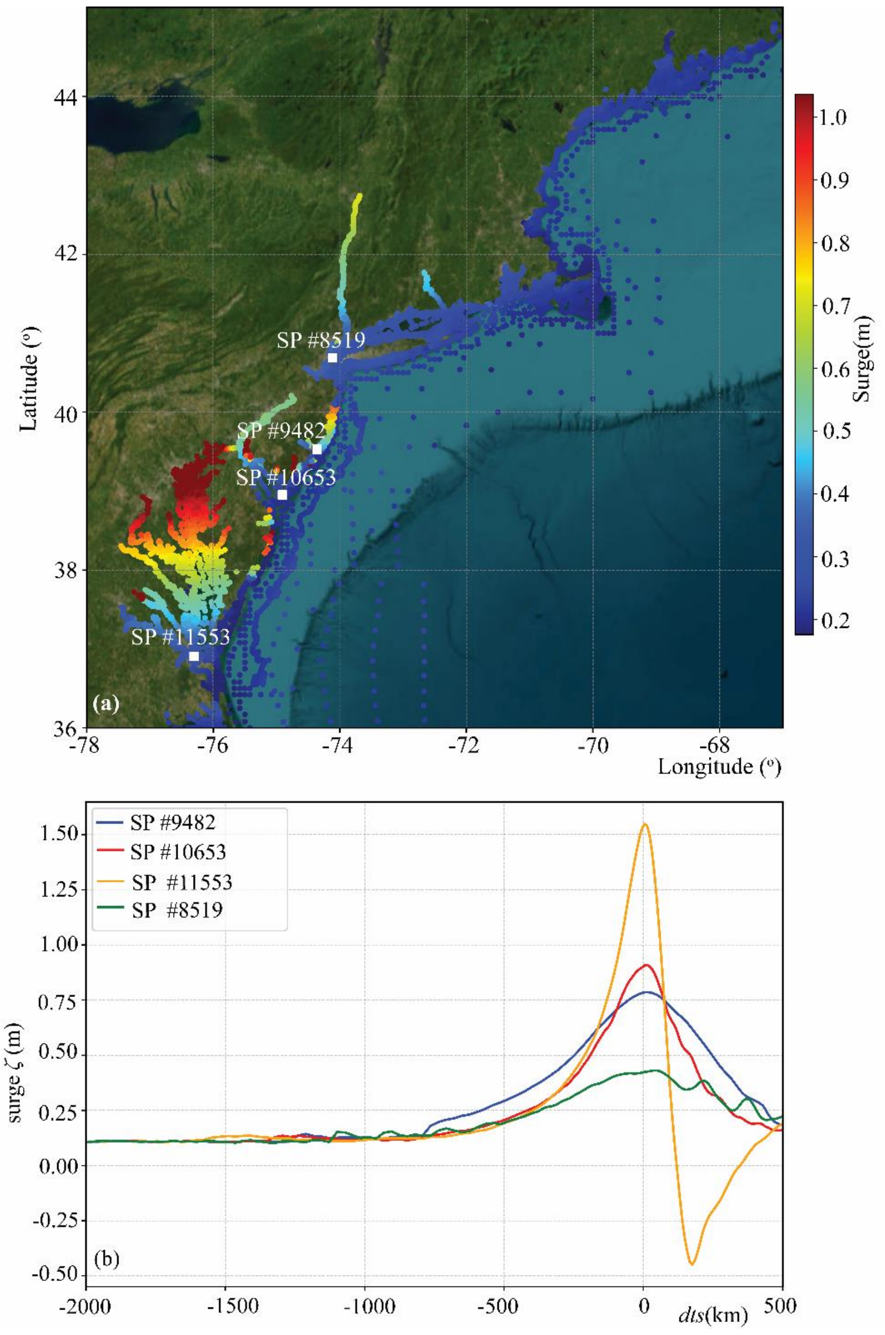


Figure 2-3: NACCS storm scenario: (a) surge response across the region of interest for a specific time instance; (b) time-series response for four different locations (SPs) within the domain of interest.

Due to the inconsistencies though, in the process of imputing storm surge that some points might experience, and the possibility of such imputed data impacting the metamodel accuracy, an additional screening of the timeseries is performed to eliminate unnatural or even entirely problematic storm surge records, which leads to keeping only 12,603 SPs that have remained inundated across all the storms (for all time instances) that are also plotted in Figure 2-2.

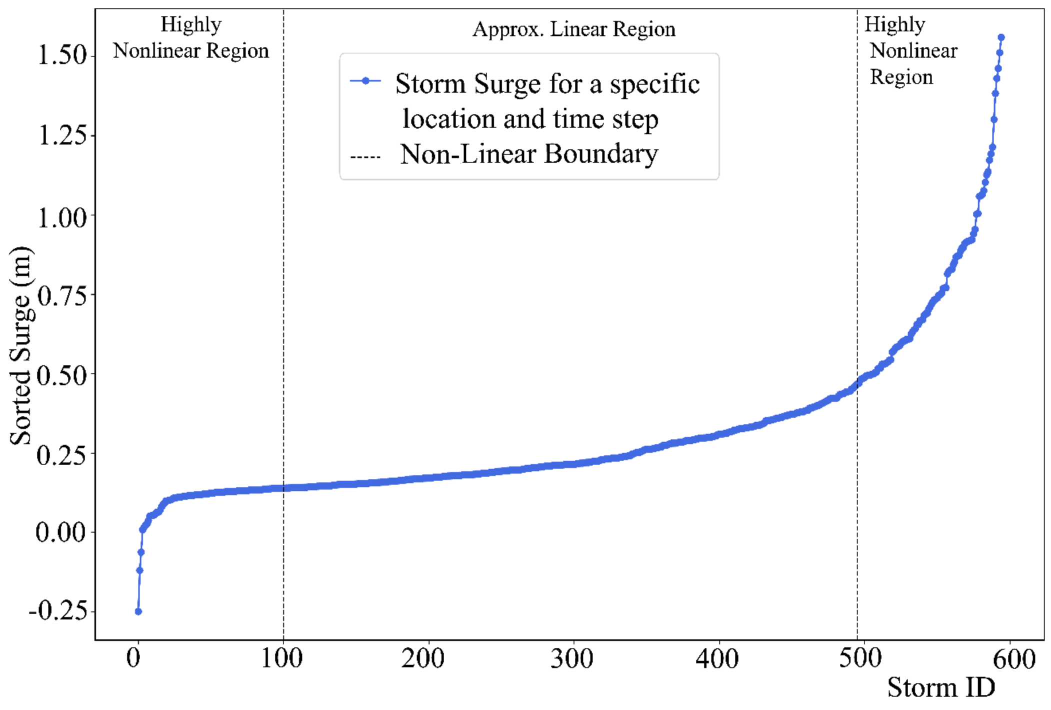


Figure 2-4: Storm surge behavior for a specific location within the region of interest, for a certain time instance across the different storms within the database. The nonlinearity in the recorded behavior is evident across the various storm scenarios.

Figure 2-3 provides a demonstration of the available data in the NACCS database, showing the surge response within the geographic domain of interest for a sample storm of the database and for a specific time instance [part (a)]. On the lower panel [part (b)], the time-series response for four different nodes is also presented, with the geographic location of these nodes highlighted in part(a). The geographic distribution of the available nodes and location sparsity within the database, due to the save point definition, is also evident from the map that is plotted in part (a) of the figure.

As already mentioned in the introduction, NACCS database has been extensively investigated for the development of surrogate models to predict either peak (maximum) surge or time-series evolutions in the storm space [50, 55, 56, 59, 62, 64, 75]. In most of those investigations, the spatial, and in some cases the temporal correlations, are leveraged to deploy an invertible dimensionality reduction technique in order to develop the storm predictive surrogate model with the latent space as the target quantity. Linear techniques, although much easier to deploy (for example PCA relies on matrix decompositions) fail to capture nonlinear patterns in the data that are related to the across storms behavior of the surge. As demonstrated in Figure 2-4 the locations on each time step are experiencing a nonlinear behavior that intensifies for lower and higher surge values that are associated with either the intensity of the storm or the storm's landfalling proximity to the location of interest. Either way, this nonlinearity in the storm space, where the surrogate model is called to make predictions, should be captured in the dimensionality reduction technique that is used to ensure high accuracy predictions for such locations or even locations with complex geomorphology that might be hampering the surrogate model's accuracy.

2.5 Time-series Clustering

The number of locations (SPs) coming through the synthetic storm database is 12,603, and for each, 170 steps (time stamps) are identified across a total of 595 storms. Although the problem might not seem initially of large dimension, if someone attempts to reduce the dimensionality of the spatio-temporal data keeping the storm number intact, they will quickly realize that it is challenging, especially for Autoencoder networks like the ones that will be examined here. To overcome the computational burden that is associated with the large dimensionality of the data at the dimensionality reduction step, a decision was made to further focus on a smaller subset of the initial available 12,603 SPs. Instead of randomly selecting that subset, clustering will be employed to identify it. Hence, for identifying the desired subset of nodes, a grouping into different clusters is employed, selecting the closest to the centroid node to represent each cluster. For the features that should be used in clustering, one has different options, with the most intuitive one, to be using a purely geospatial criterion, which will be grouping locations based on distances among them. Due to the nature of the geospatial grid though, that would be generating challenges for the adequate representation of the entire region. Instead, the spatial features (latitude and longitude) are augmented with features that are extracted from the response of these locations (experienced storm surge). A similar approach for peak surges was examined in [56] and proven to be effective for peak storm surge responses. Here, the time-series surge responses are clustered across all storms. Based on the earlier on established notation, assuming that the complete storm 3-dimensional matrix is $\mathbf{Z} \in \Re^{n_t \times n_s \times n}$ and $\zeta(n_t, \mathbf{s}, \mathbf{x})$ is denoting the storm surge for a location, the data is re-arranged in a 2-dimensional matrix $\mathbf{Z_{con}}$ of dimension $n_s \times (n_t \cdot n)$. This flattening process restructures the data by creating a single continuous row vector from the storm surge time-series for each location over all storms. For example, the data for firth location n_t time steps exist for the first storm, n_t time steps for the second storm, and so forth up to the n-th storm. All temporal and storm data for every location are combined into a single row using this format making it ready for analysis. In order to partition the n_s nodes into n_c cluster sets, the K-means algorithm is adopted [87]. Here for the feature vector **v**, the coordinates of each location are used (n_s = 2) along with the first 10 eigenvectors from the Principal Component Analysis (PCA) decomposition of the concatenated 2-dimensional matrix $\mathbf{Z_{con}}$. K-means algorithm aims to minimize the weighted variance within each identified cluster. Each cluster is represented by a single node, that in the feature space **v** is closest to the cluster centroid. The centroid of each cluster is determined by taking the mean of **v** for all the nodes within the cluster. In order to identify the n_c centroids, defining the gth component of the n_v – dimensional feature vector as v_g, the objective function that provides them, can be written as:

$$\arg\min_D \sum_{q=1}^{n_s} \sum_{v \in D_q} \left(\sum_{g=1}^{n_v} w_g (v_g - \bar{v}_q^g)^2\right) \tag{12}$$

where, $\mathbf{D} = \left\{D_1, \ldots ..D_{n_c}\right\}$ is the complete domain for the n_s nodes divided into n_c clusters, D_q is referring to the nodes whose features belong to the q-th cluster, with the summation for $\mathbf{v} \in D_q$ referring to the nodes that belong to the q-th cluster, $\bar{v}_q^g$ being the centroid cluster of the q-th cluster and the g-th component of **v**. Here w_g is used to indicate the potential existence of a generic weight factor for each feature vector that can accommodate the standardization and/or prioritization of a specific component within the weighted distance definition.

Here the weights were chosen for each feature vector separately, which means for the first two vectors that contained information on the geospatial coordinates a scaling with respect to the variance of each feature was enforced. This means that for v_g^i where g = 1, 2 the weight is set as:

$$w_g = \frac{1}{Var(v_g)} \text{ for } g = 1,2 \tag{13}$$

where the variance $Var(v_g)$ over the nodes is calculated as:

$$Var(v_g) = \frac{1}{n_s}\sum_{i=1}^{n_s}(v_g^i - \frac{1}{n_s}\sum_{i=1}^{n_s} v_g^i)^2 \tag{14}$$

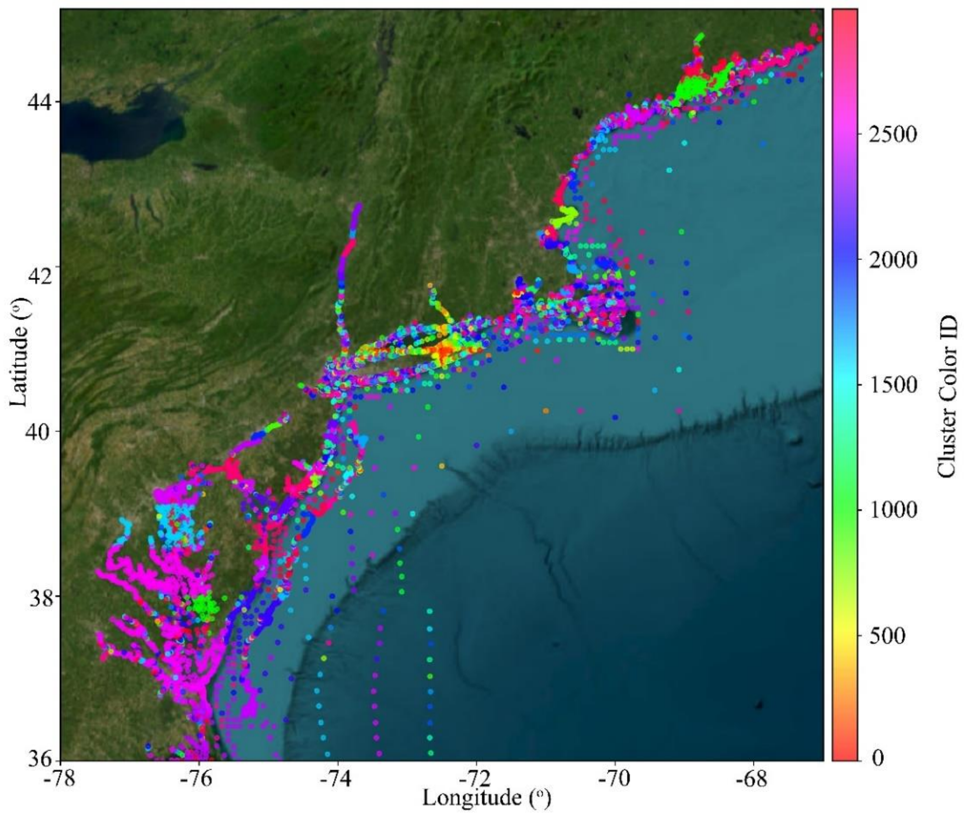


Figure 2-5: The identified groups of locations for each of the 3000 clusters colored in a unique color for each group.

For the feature vectors that are stemming from the storm surge response, through the PCA decomposition, assuming that n_{ret} eigenvectors are retained, the weights used in this case were related to the eigenvalues ω that provide the importance in each component. This means that:

$$w_g = \frac{\omega_{g-2}}{Var(v_g)} \text{ for } g = 3,\dots,n_{ret}+2 \tag{15}$$

Once the clusters were identified, the centroid for each cluster is calculated. The distance between the centroid of the cluster and neighboring nodes belong within the same cluster are calculated so that the centroid can be directly assigned to the nearest SP that exists. For the distance computation between for example the first centroid and the points belonging in the first cluster, the Euclidean formula is applied:

$$d = \sqrt{\left(x_{lat}^{n_c=1} - x_{lat}^{j}\right)^2 + \left(x_{long}^{n_c=1} - x_{long}^{j}\right)^2} \text{ for } j \in n_c = 1 \tag{16}$$

The combined case here, includes both geospatial information and response driven features, with the total number of features being 12 (10 from the eigenvalue decomposition through PCA and 2 containing the geospatial coordinates). Here the number of clusters was set to n_c = 3,000, effectively selecting representative points out of the initial n_s = 12,603. Figure 2-5 shows the clusters and the point that belong to each one of them (colored with the same color at their centroid representative), while Figure 2-6 shows the complete original region (before clustering) and the locations that were retained for further training (centroids). From this point on, and primarily to reduce the computational complexity of the problem the centroid nodes will be only used. If the framework is built around those locations a direct extension to a larger region will be easy to establish, depending on the available computational resources either through the training of the framework on a larger number of locations of interest or by building a geospatial interpolator that will be capable of retrieving information for each time step across the geospatial domain of interest as for example in [56].

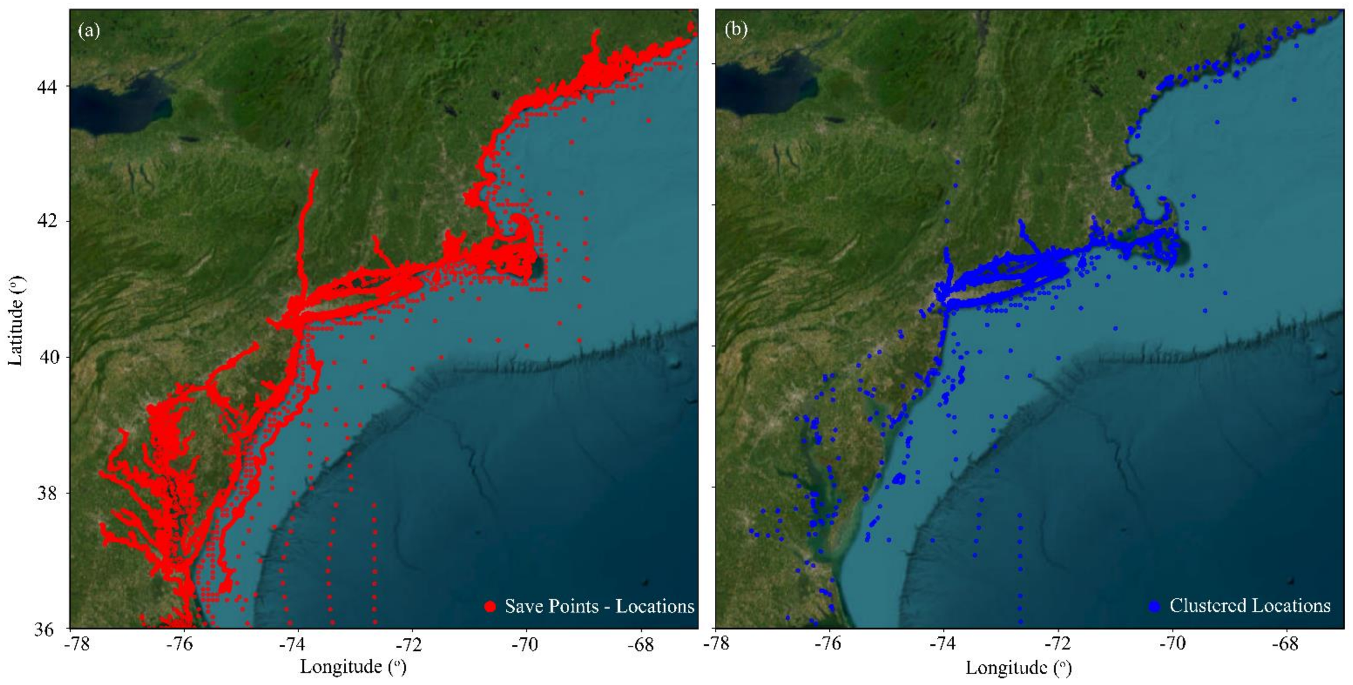


Figure 2-6: The geospatial locations (SPs) of the NACCS database that have remained wet across all storms (a) before clustering, totaling to n_s = 12,603, and (b) after the clustering where n_c = 3,000 were selected.

CHAPTER 3:
NONLINEAR DIMENSIONALITY REDUCTION TECHNIQUES: AUTOENCODERS

In this chapter, a comprehensive literature review on nonlinear dimensionality reduction (NLDR) techniques will be presented, explaining the advantages and disadvantages of some of the most popular techniques thus far, along with a detailed presentation of AutoEncoders (AE) and the different variations that can be encountered, which will be the nonlinear mapping technique of choice for this thesis and the subsequent investigations for the time-series storm surge output dimensionality reduction.

3.1 Overview

In today's information- and data-rich era, handling the high dimensionality of the data is an ongoing challenge [88], that is also a critical task when it comes to solving real-life problems with many parameters and constraints. Reducing the dimensionality of available data without losing any important information is a challenging task [89]. Dimensionality reduction has a binary goal, which is not only to compress information, but also to ensure that all trends (linear and nonlinear) are captured appropriately without any loss of crucial information regarding the problem, its description and potential constraints it might have [90]. Most real-world problems are rarely governed by linear relationships between cause and effects, and storm surge manifestation, as a highly complex physical phenomenon, is no exclusion to this.

Nonlinear dimensionality reduction (NLDR) techniques are defined as methods that are able to transform high dimensional data to lower dimension representations while capturing complex nonlinear relationships among the data and their features. Unlike techniques like PCA that focus on developing a linear latent space representation (new coordinate system) that is capturing the largest variation in the data, NLDR techniques are able to capture complex, curved and manifold structures which might be portraying more realistically the actual structure of the data in high dimensional space [91-94]. Here the focus is on exploring NLDR techniques and their application to spatio-temporal data, ensuring that a balance is achieved between maintaining essential information for the data description without overfitting the available data for the investigated problem.

3.2 Exploration of Nonlinear Dimensionality Reduction (NLDR) Techniques

To support projections of high-dimensional data, that potentially lie along nonlinear manifolds, different methods and techniques have been developed that typically have as ultimate goal either the visualization of the data in the low dimensional space or the learning of the mapping itself, so that this training can be later used on new data generated the same way as the original ones. As primarily, unsupervised, data-driven methods, their effectiveness in compressing the data and maintaining a high level of information relies heavily on the intrinsic characteristics of the data (assumptions, constraints, etc.) as well as the formulation (specialization) of each NLDR method in handling different structures of data (images, time-series, etc.) [91, 94, 95]. Some of the most popular nonlinear dimensionality techniques are: t-distributed Stochastic Neighbor Embedding (t-SNE) [96], Isomap [97], Locally-linear Embedding (LLE) [98], Multidimensional scaling (MDS) [99], Uniform Manifold Approximation and Projection (UMAP) [100], and AutoEncoders (AE) [101]. A brief description for all the above can be further found in [95]. Some of the above techniques are variations of others and specialize in certain tasks. For example, MDS is one of the most fundamental ones in visualizing the level of similarity in data, utilizing an optimization routine to identify a configuration that provides the best reconstruction possible [102]. Due to this optimization, a large computational burden may arise due to multiple local minima, and potential divergence issues from the identification of the global optimum. For new input data, MDS needs to recalculate the entire distance matrix and re-adjust the weights, something that for large dimensional data becomes computationally inefficient [103-105]. Isomap is an enhanced version of MDS that aims to understand the origin of the input assuming that it is sampled from a low dimensional manifold that is embedded inside a higher-dimensional vector space. It assumes smooth manifolds, but struggles with time-series data and has limitations regarding reversing the data to their original space (invertibility property) [106, 107]. Additionally, it has limitations related to the size of the dataset, since the calculation of distances in higher dimensions is becoming challenging (curse of dimensionality challenges) [97]. This makes Isomap a relatively computationally expensive method, similarly to the classical MDS algorithm that Isomap originates from [108]. t-SNE is part of the family of stochastic neighbor embedding methods and it is widely used for visualizing high-dimensional data by preserving local characteristics [109, 110]. As a dimensionality reduction technique, it has a significant limitation with respect to invertibility: although it can achieve efficient nonlinear embeddings, it is unable to develop an invertible mapping that would allow the latent representation to return to its original space. Furthermore as a NLDR, is inherently non deterministic, due to the random initialization of the process, which makes its efficiency intractable and case dependent [111]. t-SNE is primarily considered a visualization technique aiming to identify hidden patterns in the high dimensional data and it seems to be able to provide good results for time-series visualizations [112]. It clusters neighboring points together, allowing for localized structures to be preserved, but it is unable to maintain high level of information at a global level as it focus on local relationships between the data points[109, 113], making it ideal for small scale information retention [114]. Just like t-SNE, LLE also works great in low dimensional spaces, as it is capable of preserving structures locally. It actually works better than nonlinear PCA in low dimensional spaces. LLE appears to have further advantages, taking advantage of sparse matrix algorithms but performs poorly in non-uniform sample densities, since it relies on identifying and preserving local data structures by using a set of nearest neighbors of each sample point [115]. UMAP has been consistently used for both visualization and dimensionality reduction and proven effective in preserving the local structure but faces challenges similar to t-SNE. It can easily find manifolds in high dimensional datasets but it might struggle in low dimensional datasets due to excess of noise [100].

Kernel PCA [116] maps the data to a higher-dimensional space using kernel functions and subsequently performs PCA, with the choice of the kernel function being crucial in the model’s accuracy, making the process computationally expensive for large datasets, with invertibility being dependent on the kernel that was selected [107, 117-119].Studies show that Kernel PCA is capable of reducing the dimensionality of the data while at the same time captures non-linearity patterns, successfully extracting the important features from a complex structure. It can also be employed to remove noise from the data and identify outliers, but it typically requires a large number of computational resources due to intensive kernel processing [120, 121].

Gaussian Process Latent Variable Model (GPLVM) are leveraging Gaussian Process formulations for nonlinear dimensionality reduction and data visualization tasks. They are highly efficient due to the flexibility that the kernel function selection can offer but scale poorly in large datasets due to necessity to invert a large covariance matrix. This makes GPLVMs computationally expensive with high dimensional data, and they may experience sensitivity to initial conditions and hyperparameter identification, as well as challenges in the associated optimization and selection of kernel functions (positive definite kernels should be only considered for the development of a valid covariance matrix and numerical stability) [105, 122, 123].

Most of these methods, based on the assumptions they carry, are able to successfully identify a low-dimensional latent space representation, but many of them are unable to transform the data back to the original space by learning the mapping connecting the two spaces (either because of the stochastic nature of the representation or the lack of an inverse process). Invertibility, for cases like the one examined here, is a strong requirement, since the latent space is only used to facilitate the surrogate model development, but the final results need to be communicated in the original high dimensional physical space. Most of the nonlinear techniques that were briefly discussed above do not have this property, so they are automatically deemed as inappropriate for the problem considered in this investigation.

3.3 AutoEnconders as a NLDR technique

AutoEncoders (AEs) belong in the family of artificial neural networks (ANNs) and they are designed for unsupervised learning to perform tasks like dimensionality reduction, feature extraction, anomaly detection, data denoising and many more [124, 125]. Various types of AEs exist with some of the most popular ones being: Sparse AutoEncoder, Denoising AutoEncoder, Contractive Autoencoder, Variational Autoencoder [126]. AEs can learn unique patterns (including temporal and spatial characteristics) from the available data and extract useful information for the application at hand. They are able to efficiently encode data into lower-dimensional spaces, capturing non-linear patterns effectively, offering significant advantages in dimensionality reduction [127].

AEs have the ability to produce high quality low dimensional data, something that makes them ideal for dimensionality reduction tasks. They are flexible and can be seamlessly combined with other models to enhance the performance of certain tasks [128]. For example, in image embedding, they can be further combined with other Neural Network models like Generative Adversarial Neural Networks to enhance image reconstruction and provide further resolution [126]. Research has demonstrated AE ability to extract important features from high dimensional datasets without significant losses in the reconstruction process [119]. Comparisons of AEs to other nonlinear dimensionality reduction techniques, like kernel PCA, proved AEs superiority in reconstructing the data more accurately, reducing the reconstruction loss by 19-40%. Studies shown that AEs work well in embedding information when it comes to time-series data which is the type of data that will be considered in this thesis [94, 129].

Studies show that AEs are able to handle both the dimensionality reduction stage, as well as the data reconstruction efficiently, something that most of the nonlinear dimensionality reduction techniques are struggling with. Their ability to handle large scale data makes them ideal for high dimensional data like time-series data [130]. AEs are able to invert the latent representation back to the original space using a Decoder routine that is trained and optimized to reconstruct the representation with high precision [131]. In addition to this, the flexibility that AEs offer in crafting customizable architectures (multiple hidden layers activation function, etc.) is much higher when compared to other nonlinear dimensionality techniques that have a predefined, rigid, formulation. Furthermore, their ability to adapt and train over large and complex datasets, handle noisy or incomplete data, is making AEs the perfect method to handle real-world data that typically carry inherent challenges [128, 132].

3.4 Different Variations of AutoEncoders

As discussed in the previous section, there are multiple advantages in using AEs as a nonlinear dimensionality reduction technique [127, 133]. AEs are a special type of the artificial neural network family that can compress high dimensional data into a low dimensional space (encoding) and then perform the reverse process again to reconstruct it back from the low dimensional space to the original high dimensional space (decoding process) [124, 128]. They are able to perform well with both structured and unstructured data [134, 135], building robust abilities by effectively learning patterns, and by identifying and extracting meaningful features [134-136]. A comprehensive review of the different types of AEs and their advantages and disadvantages can be found in [133]. Here only some brief descriptions will be provided to facilitate the explorations that are presented in later chapters, along with justifications on the reasons why certain AEs might be appropriate or not for the problem formulation examined in this thesis which is the efficient dimensionality reduction of spatiotemporal storm surge time-series.

The selection of the most appropriate AE depends on the task at hand and the nature of the available data [124, 136]. Vanilla Autoencoders are a very basic form of AE which are designed with very few parameters. They are typically ideal for low dimensional datasets, but they are unable to handle architectural complexities such as spatial and temporal dependencies, efficiency and scalability [137]. If used for large datasets, they can lead to overfitting. Thus, Vanilla AEs are not ideal for large and complex datasets, like the one here, and in cases where advanced functionality is warranted [138]. **Denoising Autoencoders (DAE)** are designed to handle noisy datasets by intentionally corrupting the input data and learning to reconstruct the original, clean, data. However, the dataset that will be considered here, although might contain noisy observations (due to numerical convergence and other inconsistencies related to geospatial interpolations for the SPs) the observations are considered truth to their values, so the use of a DAE is not a great match for the application investigated here [139]. **Convolutional Autoencoders (CAEs)** combine neural networks with convolutional layers and are able to handle effectively various tasks. They work well with both spatial and non-spatial data as well as image processing tasks. Beyond dimensionality reduction, CAEs are capable of denoising data and of detecting data anomalies [132, 140, 141]. They have the capability of capturing hierarchical patterns from the data and reduce their dimensionality while preserving the important features. As all other AEs, they can also facilitate the reconstruction of the compressed data back to its original state. Recently, CAEs were implemented on storm surge data for the dimensionality reduction of the experienced surge and have shown great results [63], so they should be further investigated and compared against other AEs types to seek the best one in combination with the surrogate model development. **Sparse Autoencoders** encourage sparsity in their hidden layers, meaning they force only a small subset of neurons to be active at a time. Sparsity, although a desired property in many cases, for the problem posed here is not a significant concern since the dimensionality reduction within the overall framework of time-series surge predictions will be enforced offline for the region of interest [142, 143]. Besides, that, such AEs are shallow in their architecture, meaning that they are able to handle a small number of layers, which restricts their ability to learn complex and hierarchical features [105]. **Variational Autoencoders (VAEs)** introduce a probabilistic approach, where the latent space is treated as a distribution, allowing the model to generate new data samples. VAEs are powerful for tasks that involve generative modeling or for understanding the underlying distribution of the data. They are used in various tasks something that demonstrates their versatility, like image generation, data denoising, anomaly detection and for forecasting, dimensionality reduction and predictions [144, 145]. They work well when combined with Recurrent Neural Networks (RNNs) in handling time-series data and can capture latent output in more effective way [146]. However, the focus of this research is on dimensionality reduction, and not on new data generation or on learning probabilistic representations, so the additional complexity introduced by VAEs is not necessary for this task [138, 147]. **Deep Autoencoders (DAEs)** are one of the most fundamental AE types that is an extension of the vanilla autoencoders with the addition of multiple hidden layers, allowing them to capture more complex, non-linear relationships in the data. This makes them particularly suitable for high-dimensional datasets, as they can scale efficiently and learn hierarchical features. They are efficient in anomaly detection and can be integrated with various neural network typologies for advanced analysis [148]. Their ability to reconstruct the data back to its original dimension, makes them a good fit for the data that will be used in this thesis [149-151]. Since the thesis focuses only on dimensionality reduction and on capturing important features a deep autoencoder model makes an ideal choice for further exploration. **Convolutional Recurrent AEs** are a result of the combination of convolutional layers with recurrent layers. They work well with spatially structured data by identifying the patterns efficiently. They effectively handle high dimensional data while preserving its temporal and spatial coherence [152, 153]. This is one of the type of AEs which can be further explored for the application that is examine here. **Spatiotemporal AEs** are specialized neural networks designed to extract spatial and temporal correlations from complex datasets. They can be integrated with convolutional layers or recurrent layers. They are used for various purposes like time-series predictions, dimensionality reduction and anomaly detection, spatiotemporal representation such as training on audiovisual data [154, 155]. These Autoencoders can also be used for the type of data that will be examined here.

Due to time limitations, the DAEs will be explored here, but Convolutional AEs, Convolutional Recurrent AEs and Spatiotemporal AEs should be explored in the future as viable alternatives, comparing their ability to effectively retain information in the latent space for the surrogate model to be trained upon.

3.5 AutoEncoders in Time-series Storm Surge Predictions

In this thesis, Deep AEs will be investigated to reduce the dimensionality of the time-series data in combination with a GP surrogate model to provide storm surge predictions. Existing literature on this topic [62], on storm surge time-series data, has worked with only a small number of locations of interest (a couple of hundred) combining a Deep AutoEncoder (DAE) with Deep Neural Network (DNN) to successfully predict time-series storm surge over those locations. Two approaches have been used thus far to model the relationship between storm parameters and the high dimensional storm surge data: the first one was the hybrid model and the second one utilized a decoupled model. In the hybrid model both the DAE and DNN were trained simultaneously, with the DAE reducing the high dimensional data and DNN mapping the storm parameters directly to the latent space. The decoder then reconstructs the latent space predictions back to the original space to obtain the predicted surge. In the decoupled approach on the other hand, both the DAE and DNN models are separately trained and DNN maps to pre-trained latent space and then the decoder reconstructs it back to the original dimension. The hybrid model was found to outperform the decoupled model due to its ability to optimize the dimensionality reduction and the regression simultaneously when compared to decoupled which was developed sequentially [62]. Further work on this topic, was extended to Convolutional AutoEncoders (CAEs), this time for a slightly larger domain of interest (around 600 unique spatial locations) [63]. The data from dimensionality reduction was further used to train a

Hierarchical Deep Neural Network (HDNN) for storm surge predictions [63] demonstrating extremely good accuracy. Multiple HDNNs were trained on different time intervals (states), allowing for subsequent predictions of surge to provide the correct temporal resolution.

This thesis aims to accommodate a much larger number of locations (couple of thousands) to provide predictions simultaneously for larger domains of interest with a single trained surrogate model. Although such a robust model would be beneficial from a practical aspect when it comes to emergency response decision making and in understanding localized surge patterns and their evolution over time, it generates challenges in the training and predictive abilities of both the AE and the subsequent surrogate model due to the increased dimensionality that needs to be handled. In addition to this, given the complexity of the physical phenomenon of storm surge, larger areas are expected to experience higher variabilities in their responses, something that inherently challenges the generation of a robust latent space representation and the performance of any kind of predictive modeling (during training, testing and implementation). Once the dimensionality reduction is performed, the latent data will be used as the output of interest in Gaussian Process (GP) surrogate model, with the input describing the storm evolution through an appropriate unique parametrization. The goal is to develop a data-driven model that in a new, never-seen-before storm parametrization will be able to initially predict a latent space that later on will be passed through the AE decoder to get the final, time-series surge estimates in the original space. This functionality will allow for the combination of the trained AE and GP model to offer accurate, real-time predictions for the storm surge evolution over a relatively large geospatial area of interest.

CHAPTER 4:

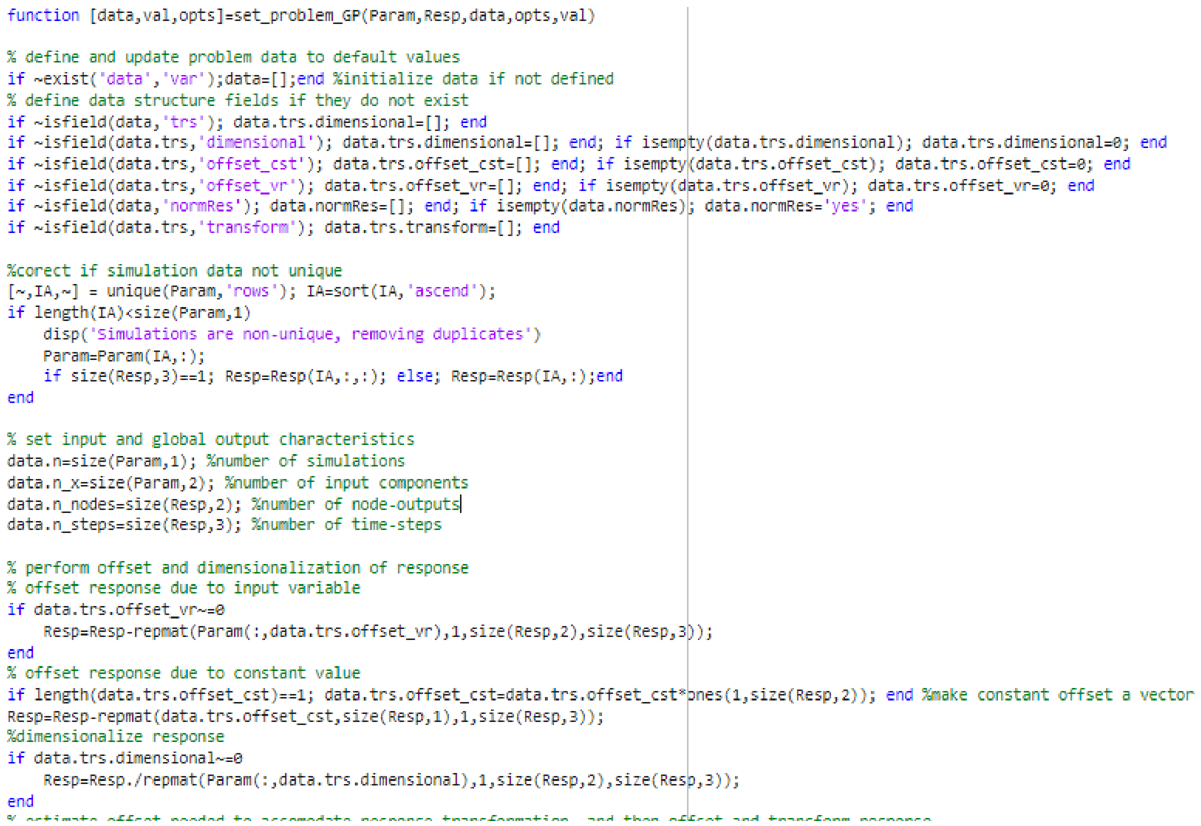
CODE TRANSLATION OF SURROGATE MODELING FORMULATION FROM MATLAB TO PYTHON

4.1 Reasons to Convert Surrogate Model Codes

This thesis focuses on the nonlinear dimensionality reduction of the data, so the surrogate model development is directly adopted by Kyprioti et. al. [59] with most of the coding work done in MATLAB. This pre-existing work was converted to PYTHON programming language, as a side task to the primary investigation, to facilitate a seamless integration between the NLDR technique and the surrogate model development and implementation. Due to PYTHON's versatility in computations, flexible environment, and increased dominance in scientific computing, machine learning and data science applications, it was considered a better option and closer to the researcher's skills and experience. Being new to MATLAB, gave the student an opportunity to dive deeper into ways each language/software has enhanced abilities for different tasks, providing an opportunity to work on a new software in detail, being able to compare commands and ensure that equivalences are consistent between the two environments. PYTHON is versatile and can be easily integrated with other databases and programming languages and web Application Programming Interfaces. With the technological and computational growth that has been seen in the last decade, along with the data-abundancy and plethora of information, these features are important for pushing further advancements in research, computations and large-scale applications. Thus, the decision to convert the MATLAB code to PYTHON is expected to enhance scalability, flexibility, cost effectiveness and computational efficiency.

4.2 Comparisons between MATLAB and PYTHON Interfaces

As expected, there are a variety of differences in syntax, mathematical manipulations, function naming and function versatility as well as discrepancies in graphical user interface and debugging. For example, MATLAB has a more structured environment for "for-loop" with the word "end" used to identify and close repetitive loops efficiently whereas PYTHON works around indentation to indicate different levels of for loops. Another challenging, and to some extent confusing characteristic that can potentially generate significant discrepancies and misinterpretations is the indexing convention that is used. In PYTHON counters start from 0 whereas in MATLAB start from 1. As expected, MATLAB has a significant advantage in performing matrix manipulations in an implicit way that accommodates large dimensional data in a fast and efficient manner, something that PYTHON relies on readily available libraries (like Numpy) that are not necessarily optimized to handle with the same efficiency. Handling of multidimensional arrays is also done inherently in MATLAB, with easy structures (arrays) where information can be packed explicitly, enabling the communication between different functions in a clear and unique way. PYTHON on the other hand, is not as versatile, forcing the restructure of entire functions that were coded in MATLAB, such that information is handled correctly through the use of libraries such as Numpy for reshaping and re-packaging data. Different math operations are established, with some especially from MATLAB being tailored to matrix manipulations (i.e. elementwise matrix multiplications) or using different syntax (based on the PYTHON library of choice), something that creates further challenges in developing direct analogies. Some of the snippets from conversion area given in Figure 4-1 and Figure 4-2 .

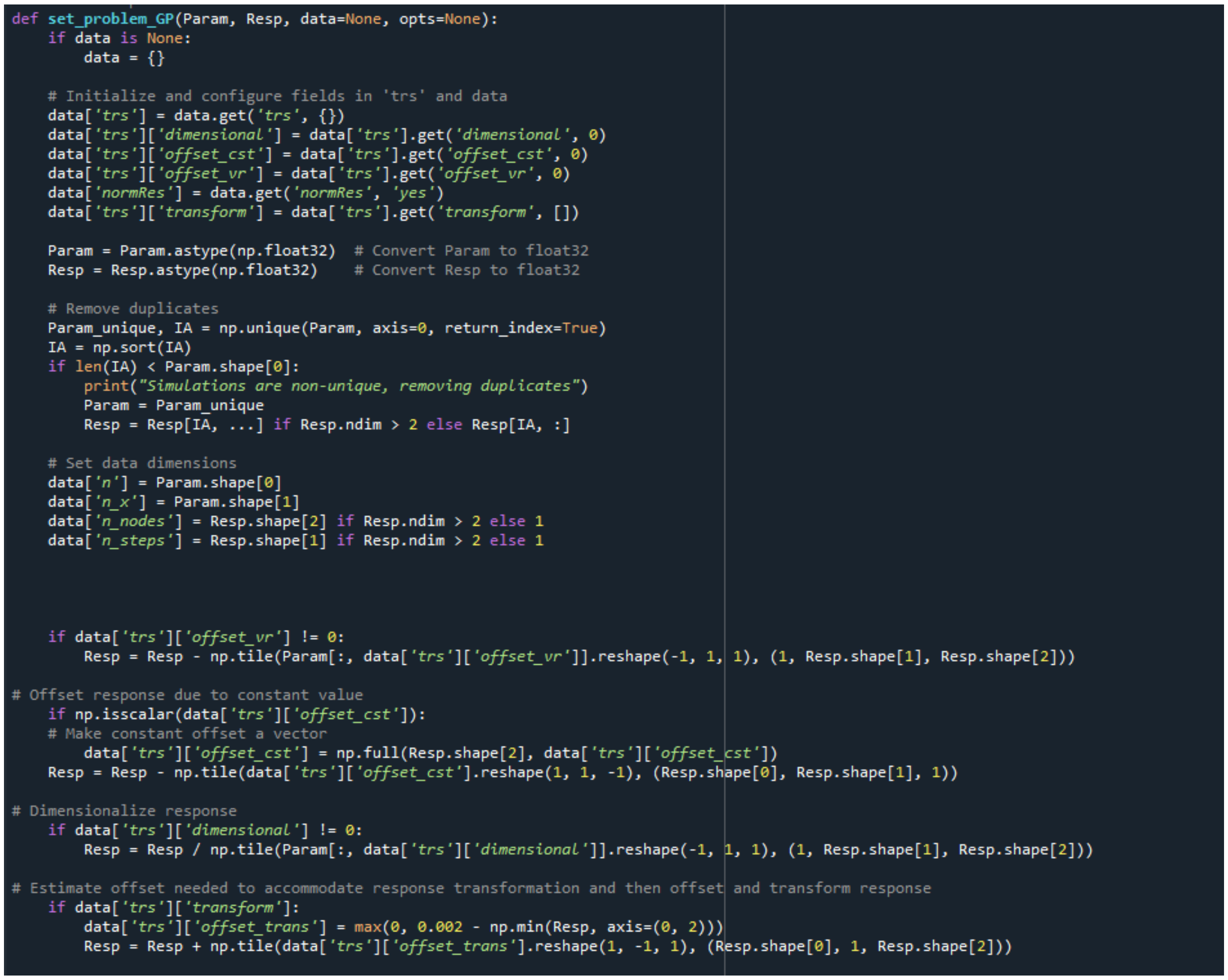


```
function [data,val,opts]=set_problem_GP(Param,Resp,data,opts,val)

% define and update problem data to default values
if ~exist('data','var');data=[];end %initialize data if not defined
% define data structure fields if they do not exist
if ~isfield(data,'trs'); data.trs.dimensional=[]; end
if ~isfield(data.trs,'dimensional'); data.trs.dimensional=[]; end; if isempty(data.trs.dimensional); data.trs.dimensional=0; end
if ~isfield(data.trs,'offset_cst'); data.trs.offset_cst=[]; end; if isempty(data.trs.offset_cst); data.trs.offset_cst=0; end
if ~isfield(data.trs,'offset_vr'); data.trs.offset_vr=[]; end; if isempty(data.trs.offset_vr); data.trs.offset_vr=0; end
if ~isfield(data,'normRes'); data.normRes=[]; end; if isempty(data.normRes); data.normRes='yes'; end
if ~isfield(data.trs,'transform'); data.trs.transform=[]; end

%corect if simulation data not unique
[~,IA,~] = unique(Param,'rows'); IA=sort(IA,'ascend');
if length(IA)<size(Param,1)
    disp('Simulations are non-unique, removing duplicates')
    Param=Param(IA,:);
    if size(Resp,3)==1; Resp=Resp(IA,:,:); else; Resp=Resp(IA,:);end
end

% set input and global output characteristics
data.n=size(Param,1); %number of simulations
data.n_x=size(Param,2); %number of input components
data.n_nodes=size(Resp,2); %number of node-outputs
data.n_steps=size(Resp,3); %number of time-steps

% perform offset and dimensionalization of response
% offset response due to input variable
if data.trs.offset_vr~=0
    Resp=Resp-repmat(Param(:,data.trs.offset_vr),1,size(Resp,2),size(Resp,3));
end
% offset response due to constant value
if length(data.trs.offset_cst)==1; data.trs.offset_cst=data.trs.offset_cst*ones(1,size(Resp,2)); end %make constant offset a vector
Resp=Resp-repmat(data.trs.offset_cst,size(Resp,1),1,size(Resp,3));
%dimensionalize response
if data.trs.dimensional~=0
    Resp=Resp./repmat(Param(:,data.trs.dimensional),1,size(Resp,2),size(Resp,3));
end
% estimate offset needed to accomodate response transformation, and then offset and transform response
```

Figure 4-1: MATLAB script for surrogate model development.

4.3 Challenges Faced and Tackled

While converting the code to python the major challenge was understanding the data structures in MATLAB (arrays, structures, matrices, etc.). The ways the data is stored in PYTHON are entirely different, so choosing the right data structure to maintain consistent data and shapes (matrix and vector dimensions) was the biggest challenge. In addition to that, MATLAB has a number of built-in functions (such as "repmat") that do not directly have an equivalent one in PYTHON. Coding such equivalent functions manually also proved to be challenging, since many times handling large data as the ones in this project, was leading into memory issues. The third major challenge that was faced, was related to converting the already coded in MATLAB functions, as the complexity of model with multiple nested function was high. Many nested functions that were coded manually for efficiency without using MATLAB built-in functions needed to be coded again in PYTHON using explicitly elementwise operations. This created a lot of complications in getting the right results with lots of errors that needed to be debugged and tackled. Another major setback for effectively understanding the way the code was handling information was related to debugging and effectively coding things in PYTHON, in an interactive and fast way, so that visualization and checks are instant. Adding to this, learning how to run jobs in OU Supercomputing Center for Education and Research (OSCER) Supercomputer and efficiently handle time-out restrictions, job submission and correct filing and saving results, generated challenges in efficiently handling computations and understanding how to work differently when sending things in the Supercomputer or running things locally through the lab computers. Overall, converting the code was challenging and is still a work in progress (trying to debug all different options and paths to make sure that they replicate to the extent possible results provided by MATLAB), but doing it efficiently and carefully will result in a successful conversion accommodating both the AE and surrogate model development in the same computational environment.



```python
def set_problem_GP(Param, Resp, data=None, opts=None):
    if data is None:
        data = {}

    # Initialize and configure fields in 'trs' and data
    data['trs'] = data.get('trs', {})
    data['trs']['dimensional'] = data['trs'].get('dimensional', 0)
    data['trs']['offset_cst'] = data['trs'].get('offset_cst', 0)
    data['trs']['offset_vr'] = data['trs'].get('offset_vr', 0)
    data['normRes'] = data.get('normRes', 'yes')
    data['trs']['transform'] = data['trs'].get('transform', [])

    Param = Param.astype(np.float32)  # Convert Param to float32
    Resp = Resp.astype(np.float32)    # Convert Resp to float32

    # Remove duplicates
    Param_unique, IA = np.unique(Param, axis=0, return_index=True)
    IA = np.sort(IA)
    if len(IA) < Param.shape[0]:
        print("Simulations are non-unique, removing duplicates")
        Param = Param_unique
        Resp = Resp[IA, ...] if Resp.ndim > 2 else Resp[IA, :]

    # Set data dimensions
    data['n'] = Param.shape[0]
    data['n_x'] = Param.shape[1]
    data['n_nodes'] = Resp.shape[2] if Resp.ndim > 2 else 1
    data['n_steps'] = Resp.shape[1] if Resp.ndim > 2 else 1



    if data['trs']['offset_vr'] != 0:
        Resp = Resp - np.tile(Param[:, data['trs']['offset_vr']].reshape(-1, 1, 1), (1, Resp.shape[1], Resp.shape[2]))

# Offset response due to constant value
    if np.isscalar(data['trs']['offset_cst']):
    # Make constant offset a vector
        data['trs']['offset_cst'] = np.full(Resp.shape[2], data['trs']['offset_cst'])
    Resp = Resp - np.tile(data['trs']['offset_cst'].reshape(1, 1, -1), (Resp.shape[0], Resp.shape[1], 1))

# Dimensionalize response
    if data['trs']['dimensional'] != 0:
        Resp = Resp / np.tile(Param[:, data['trs']['dimensional']].reshape(-1, 1, 1), (1, Resp.shape[1], Resp.shape[2]))

# Estimate offset needed to accommodate response transformation and then offset and transform response
    if data['trs']['transform']:
        data['trs']['offset_trans'] = max(0, 0.002 - np.min(Resp, axis=(0, 2)))
        Resp = Resp + np.tile(data['trs']['offset_trans'].reshape(1, -1, 1), (Resp.shape[0], 1, Resp.shape[2]))
```

Figure 4-2: Converted PYTHON script from the MATLAB one shown in Figure 4-1.

4.4 Skills Developed

Converting code to PYTHON helped in developing valuable skills, related to revisiting theoretical mathematical concepts, understanding how to efficiently handle and manipulate data, debug codes, and ensure that they work properly, visualize results, and of course work with another software (MATLAB) beyond the ones that the student already knew, to expand their coding abilities and knowledge. This translation improved the student’s ability to understand complex code, and to write efficient code in PYTHON using existing libraries. It was a useful experience that broadened the knowledge they had on statistics, and the ways PYTHON is handling complex calculations. This code migration was a learning step to handling bigger codes and improved the student’s problem-solving skills, debugging skills, validation skills. It overall improved their ability of analytical thinking and understanding principles beyond coding commands.

CHAPTER 5:

DEEP AUTOENCODER ARCHITECTURE AND METHODOLOGY FOR CALIBRATION

5.1 Basic Architecture of Autoencoder

The Autoencoder architecture in its general description, consists of a variety of layers that include an input layer, encoder layers, a latent space layer, decoder layers and an output layer. In Figure 5-1 a generic representation of the structure of an AE is presented. Each layer can have multiple neurons that work as weights in the model's mathematical definition [126].

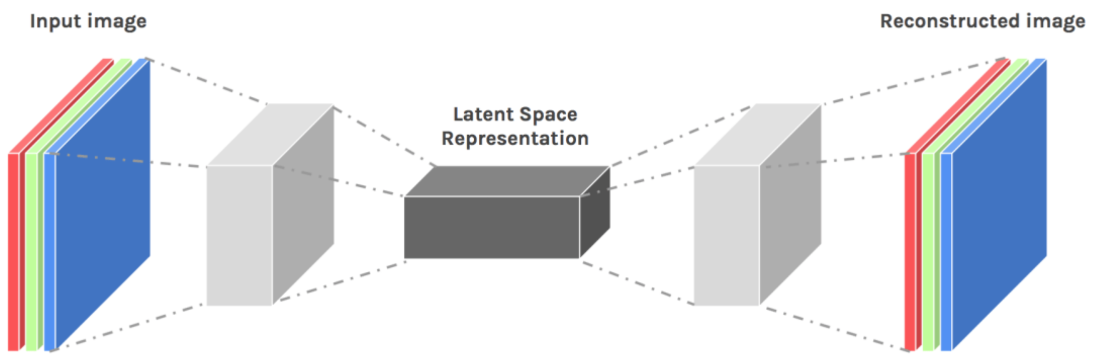


Figure 5-1: Generic representation of the architecture of an AutoEncoder [156].

The data is inserted into the input layer and then pass through a series of encoding layers that contain hidden neurons. After this, the process reaches the latent space, where the most compressed version of the data is stored, with a data dimension, typically being much lower than the initial one at the input stage. If the data needs to be restructured to the original space, then a decoding routine is built through a series of layers and hidden neurons to bring back the data from the latent representation to its original dimension, with the output layer processing the final data. In Figure 5-2 a schematic is offered of how the data flows from one layer to the other. Ultimately in such AE NLDRs routines, the input and output of the model are the exact same data, with some discrepancy between them, since the output data has gone through a nonlinear embedding and has been reconstructed back, based on the information that was embedded in the lower dimensional space. Of course, the smaller the discrepancy between the input and the output (reconstructed data) the more successful the mapping is, unless overfitting issues arise.

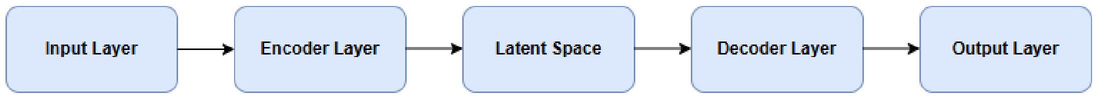


Figure 5-2: Layer order for an AE.

5.2 Deciding the Architecture and Hyperparameters of the DAE

Finalizing the architecture of the autoencoder depends on the aim of the research and the nature of the data. Thus, it is important to explore almost exhaustively different AE architectures and typologies, something that makes this exploration challenging since the numerous combinations can be established. For this reason, and relying on existing literature, certain selections will be part of the explored and further refined below AEs. As already mentioned above, different variants of autoencoder exist and DAE is chosen in this thesis for further investigation. It should be noted though that even within the family of DAEs more subcategories are available [157]. Depending on the comparison between the size of latent space and of the original input, an AE can be considered as undercomplete or overcomplete. In an undercomplete AE, the dimensionality of the latent space (the encoding) is smaller than that of the input space. Such a network is forced to learn a compact, compressed representation of the input, capturing the most important features [158]. On the contrary, an overcomplete AE, has a latent space size that is larger than (or equal to) the dimensionality of the input space. In that case, the model can potentially learn a highly detailed or even redundant representation of the input, since the latent space has enough capacity to encode the data without significant compression [158].

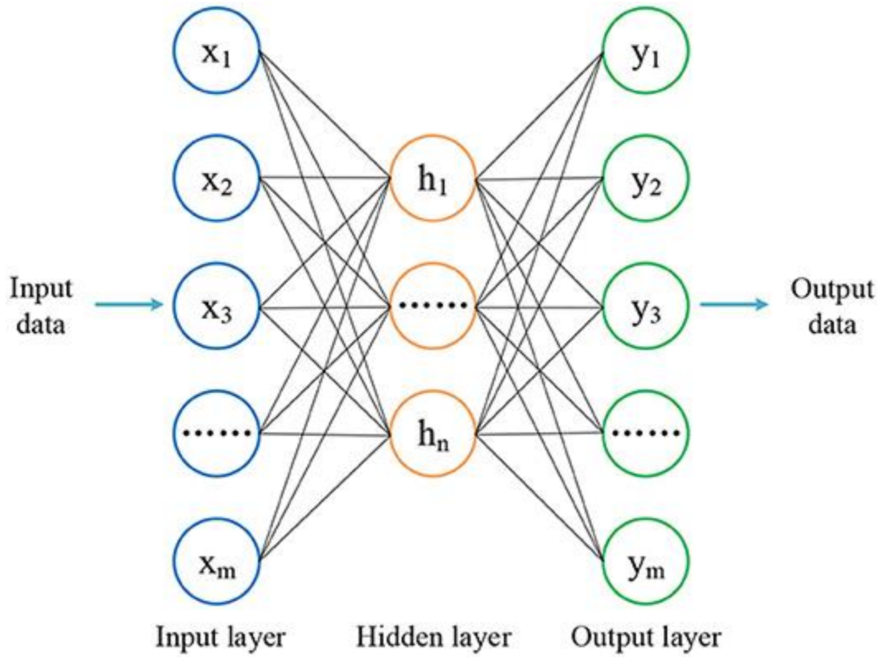


Figure 5-3: The connections between layers in an AE for a fully connected architecture [159].

For this investigation and the task at hand, an undercomplete AE will be selected for further training as the research aims at reducing the dimensionality. Another frequent distinction in the architectural shape of the AE is the layer and neuron configuration for the encoding and decoding parts. AEs can be symmetrical and non-symmetrical. Symmetric AEs are those that their encoder and decoder layers are a mirror image to one another, which means the number of layers and the number of neurons in each layer in the encoder side are the same as in the decoder. Non-symmetric AEs are those where the number of layers and number of neurons in both encoder and decoder may vary without any specific relationship between the two [160]. Asymmetric AEs have proven to be a resourceful solution, specifically for data compression, where have limited computational memory is available. Allowing for the independent optimization of the encoder and decoder layers is more effective and typically results in lower reconstruction errors compared to symmetric configurations [161]. In this thesis unsymmetrical AE configurations are explored to enhance the robustness of the architecture.

In order for the data to flow across the neuron and different layers, AEs have three different types of connectivity relations: fully connected, convolutional and recurrent. Fully connected AEs are AEs where every neuron of the layer is connected to every neuron of the immediate next layer. Figure 5-3 shows below the picture of a fully connected generic representation for an AE. Convolutional AEs are networks which are not fully connected across the entire architecture, but only for small, localized regions of the input which are called receptive fields. Figure 5-4 shows the architecture of Convolutional AE that has partially connected layers. Recurrent AEs are the autoencoders that are not only connected to the other layers but also connected to themselves. Figure 5-5 shows the basic architecture of the Recurrent AEs and the feedback connection that they have in each layer.

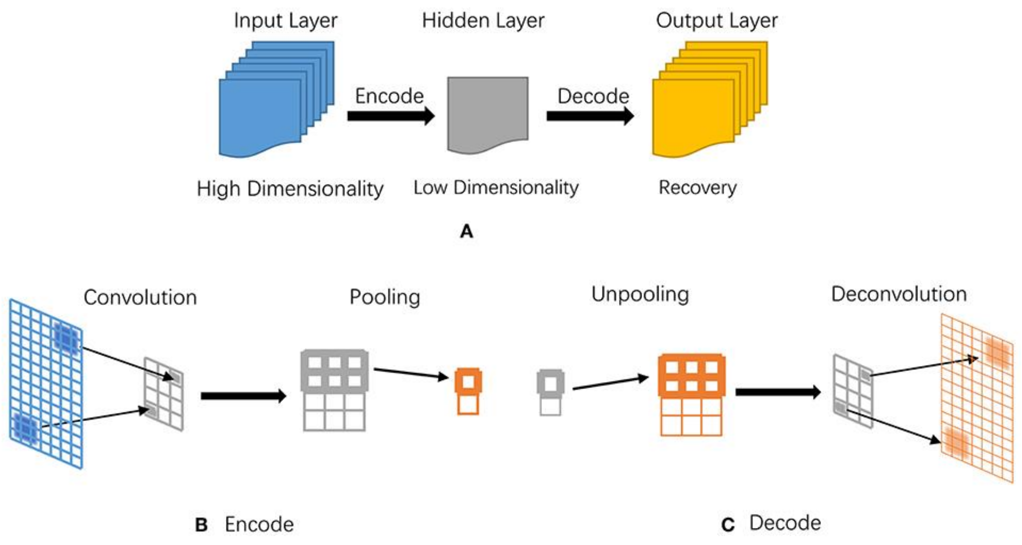


Figure 5-4: The connections between layers in an AE for a convolutional architecture [162].

The autoencoder explored in this thesis is a fully connected one since the aim of the research is the dimensionality reduction of the time-series storm surge response, and while exploring all kinds of autoencoder, it was found that fully connected AEs are one of the foundational architectures for AEs [163]. Fully connected AEs were chosen here to limit the scope of the investigation for the purposes of the thesis, since it is always better to investigate and understand the basic framework that the data patterns provide first and then proceed with the exploration of more complex architectures.

After deciding on the connections among the layers, further decisions need to be made regarding the AE architecture, with respect to the activation function, method of implementation, network architecture, training technique, ways to implement the selected training technique, reconstruction score, regularization methods and sparsity constraints [133, 157]. Below a brief overview of the above AE features will be provided, but more general details on the development of AEs can be found in [133].

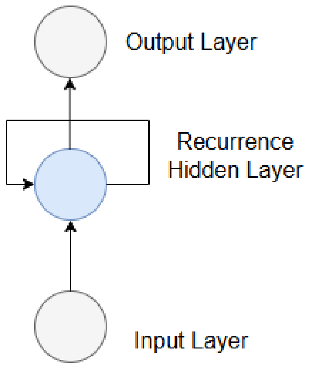


Figure 5-5: The connections between layers in an AE for recurring architecture.

This thesis focuses on parametric exploration of deep autoencoder to understand how the deep autoencoder accuracy varies with the change of a variety of parameters in their architecture. The activation function (AF) for the intermediate layers is one of the parameters that is extensively explored here [151]. The AF is the mathematical operation through which the data is processed defining the output of the neuron. This function is the deciding factor on whether the developed network will be able to capture the linearity or non-linearity within the available data. The nonlinear AFs that were explored in this thesis are: the Hyperbolic Tangent (Tanh), the Sigmoid, the Softplus and the Scaled Exponential Linear Unit (SELU) [164]. Every AF varies with their interval of possible output values with respect to the given input values. The hyperbolic tangent output values in the range [-1,1], allowing for both negative and positive values. The output for the sigmoid is defined in range [0,1] but a modified version of the sigmoid was considered here with values within range [-1, 1] [163]. The Softplus AF original range is [0,∞) which means that is positively defined with a saturation to zero for any negative values that might arise. To enable a wider range for the function region of definition, a modified (shifted) version was considered with limits [-10, ∞) forcing the saturation to -10. The SELU AF original range is [-1.758,

∞), and a similar shift as for Softplus was applied, leading to a new range [-11.758, ∞). This was an arbitrary selection to help in better informing a latent space that can be uniquely mapped by the GP surrogate model in the later step within the framework.

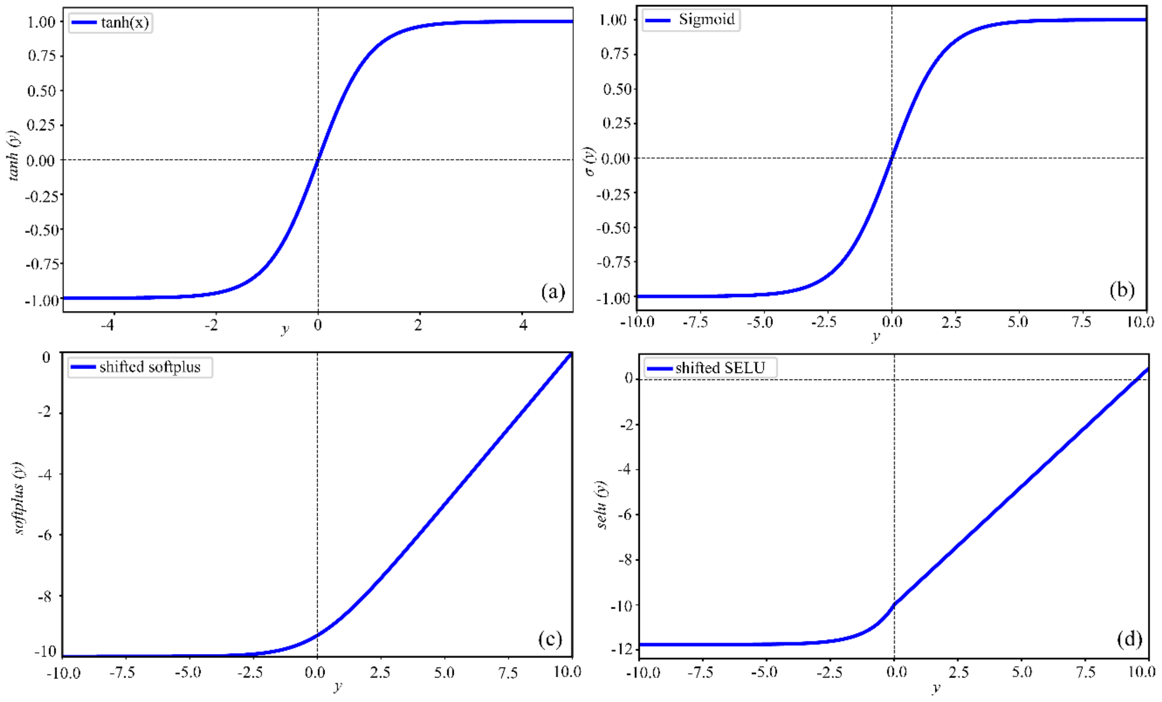


Figure 5-6: The different activation function that were explored for the development of the AE: (a) hyperbolic tangent (Tanh), (b) Sigmoid, (c) Softplus and (d) SELU activation functions.

The mathematical equations for the hyperbolic tangent (Tanh), the Sigmoid, the Softplus and the SELU activation functions are as follows:

$$\tanh(y) = \frac{e^{y} - e^{-y}}{e^{y} + e^{-y}} \tag{17}$$

$$\sigma(y) = \frac{2}{1+e^{-y}} - 1 \tag{18}$$

$$Softplus(y) = \ln(1+e^{y}) - 10 \tag{19}$$

$$selu(y) = \begin{cases} \lambda y - 10 & \text{if } \ y > 0 \\ \lambda(\beta e^{y} - \beta) & \text{if } \ y \leq 0 \end{cases} \text{if } \ y \leq 0 \tag{20}$$

where λ is 1.0507 and β is 1.67326. The plots for all the functions are provided in Figure 5-6.

As mentioned above, a fully connected deep autoencoder with dense layers was used here, which means that every neuron in each layer is connected to every neuron in the immediate next layer. Figure 5-7 below shows the generic architecture of the DAE that was developed. Each layer in the encoder, latent space, and decoder is fully connected and the output data from a layer *i*, can be expressed mathematically as:

$$h_i = f(W_i h_{i-1} + b_i) \tag{21}$$

where h_i is the output of the current hidden layer, h_{i-1} is the input of the current layer, W_i is the weight matrix of the layer *i*, b_i is the bias vector and $f(.)$ is the activation function of choice.

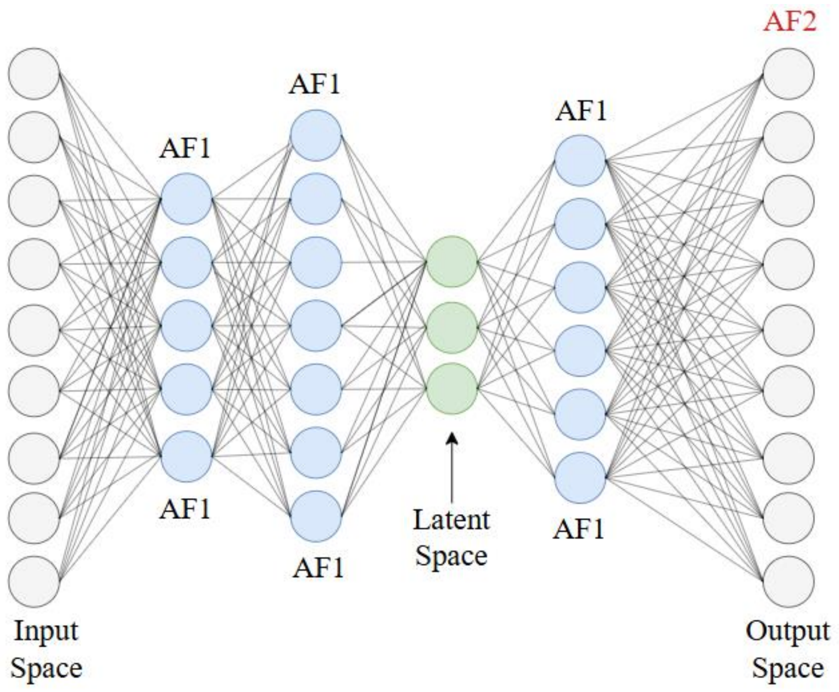


Figure 5-7: The general architecture of the explored DAE. The Afs are explicitly denoted to indicate their distribution within the network. The output AF will be in general different from the ones used internally in the AE.

Figure 5-7 shows the generic architecture of the AE designed in this thesis. Two activation functions are used in each AE configuration, the first activation function (AF1) which is used for all the intermediate layers, and the second activation function (AF2) which is used in the output layer. AF2 for all the cases will be the same (linear activation function) and it will not be explored further in this thesis.

The encoding dimension decides the size of the latent space, and since the aim is to compress the data, the dimension of latent space should be selected to be smaller than that of the input data. The larger the target encoding dimension, the more features are retained, but the lesser the reduction in the dimensionality is, that will then increase the computational complexity for the subsequent GP surrogate model development [163].

The AEs follow forward propagation and backward propagation methods to process the data and adjust their accuracy through the calculation of error metrics. The forward propagation starts with passing the input data through the network encoding layers, the latent space, and the decoding layers to generate the output. Once the output is generated, it is compared with the input and the error is calculated using the following generic equation:

$$Error = o - r \tag{22}$$

where '*o*' is denoting the original data and '*r*' the reconstructed data.

In order to provide a scalar value for the model accuracy, the error is passed into a loss function, that compares the original data (non-processed) with the ones derived from the forward propagation process. Different error metrics may be used for this, with the most popular one, being the Mean Squared Error (*MSE*), which can be calculated using the following equation:

$$MSE = L = \frac{1}{n}\sum_{i=1}^{n}(o_i - r_i)^2 \qquad (23)$$

where *n* here is the number of data points, while o_i is the input data, r_i is the reconstructed output.

The weights are further adjusted to minimize the loss function building the back propagation step. The gradient of the loss function (*L*) with respect to weights and biases is computed. The error signals are then calculated for all layers using:

$$\delta_l = (\delta_{l+1}.W_{l+1}^T).f'(z_l) \qquad (24)$$

where δ_l and δ_{l+1} denote the error signal for the current layer and the next layer, respective, while W_{l+1}^T denotes the vector of weights for the subsequent layer with $.^T$ denoting the transpose of a vector. Finally, this equation $f'(z_l)$ denotes the current layer activation function derivative.

Once the error signals are computed, the gradient is calculated for all weights and biases in all layers. The gradients for the weights can be calculated using:

$$\frac{\partial L}{\partial W_l} = h_{i-1}.\delta_l \qquad (25)$$

where h_{i-1} is the output of the previous layer, *L* is the loss function as defined in Equation (23).

And the gradients for the biases can be given by:

$$\frac{\partial L}{\partial b_l} = \delta_l \qquad (26)$$

Once all the gradients are computed, the weights and biases are updated to minimize the loss. The weight update can be mathematically written as:

$$W_l \leftarrow W_l - \eta . \frac{\partial L}{\partial W_l} \quad (27)$$

where η is the learning rate for the optimization process. Similarly, the bias update can be defined as:

$$b_l \leftarrow b_l - \eta . \frac{\partial L}{\partial b_l} \quad (28)$$

This is an iterative process, and it continues until the model converges. Based on the above description, it becomes evident that the forward propagation starts with the data propagation, while the backward propagation starts with error propagation with the ultimate goal to minimize the loss (discrepancy between the original and reconstructed data).

The weights and biases above are adjusted using optimizing algorithms. Some of the most popular optimizers, are the Stochastic Gradient Descent (SGD), the Momentum Optimization, the Adaptive Moment Estimation (ADAM), the Adaptive Gradient Algorithm (Adagrad), and the Root Mean Square Propagation (RMSprop) [165]. The optimizers used in this thesis are ADAM and SGD. Both are proved to be very effective in neural network optimization and hence both were explored in the parametric analysis that will be described in subsequent section. Its adaptive nature towards the learning rates especially in the case of DAEs during the early training stages gives more stable convergence, makes it ideal in handling complex tasks which leads to faster and stable convergence when compared to other optimizers [166-168]. Research shows that in the case of dimensionality reduction of high dimensional and complex data it performs extremely well [168], and this is the primary reason that it was selected here.

In order to process any data through the AE neural network, multiple different layers of hidden neurons need to be accessed. Deciding an optimal number of layers and neurons within each layer depends upon the data complexity, performance of the AE, and of course, the task at hand. It quickly though, becomes evident that the potential combinations, especially when it comes to these two AE architecture parameters, are infinite. One though, should be aware that the more complex the selected architecture, the harder the training of the model becomes, necessitating a large amount of data for training, to avoid overfitting issues. For that reason, model parsimony should be always applied when such architectures are explored aiming to get the simplest architecture that still serves the target (dimensionality reduction in this case).

While working with large datasets the training may become slow and inefficient. For that reason, dividing the samples into smaller batches facilitates a more manageable training time and ensures the generalization of the code. For these reasons, selecting an optimal batch size plays a critical role in model training, since it balances the need for efficient training speed with the ability to learn from variations within the dataset, preventing overfitting and enabling better generalization; essentially, it strikes a balance between computational efficiency and accuracy by processing data in manageable chunks rather than all at once.

The number of times during the training phase that the model will see the entire dataset depends on the number of epochs. Each epoch is one complete pass through the entire training dataset which is already split into predefined batches as described in the previous paragraph. The number of iterations of the dataset through the model, is a very critical factor, since the more times the model is allowed to learn the data (larger number of epochs), the higher the chances of the model starting to overfit the available dataset. On the other hand, if the model is not trained sufficiently, it might underfit the data simply because there were not enough iterations (epochs) provided for the model to stabilize its structure. In every epoch the model goes through the data, different learning patterns, weight adjustments (back propagations) are performed utilizing new batch arrangements, ultimately providing the calculation of a loss between the initial data (original dataset) and the predictions offered for them by the model in training [169]. The learning rate, which is another important hyperparameter that needs to be identified, controls the speed of the training (learning) process. It is responsible for the convergence and generalization of the model and has been proven to be impactful on the performance of the neural network overall [170, 171].

The DAE model here is trained with the purpose of minimizing the reconstruction error between the input data and the output (reconstructed prediction of data). To measure the performance of the model on how well it can reconstruct the data, different error metrics can be deployed. Here the Root Mean Squared Error (*RMSE*), the coefficient of determination (R^2) and the Mean Squared Error (*MSE*) will be calculated to examine also any potential bias that certain error metrics might have in assessing the AEs predictive accuracy. The mathematical formulations for *RMSE* and R^2 are provided in Section 5.5. For the *MSE,* the formula, in its generic description for *n* storms is given by:

$$MSE = \frac{1}{n \cdot n_s} \sum_{h=1}^{n} \sum_{j=1}^{n_s} \frac{1}{n_t} \sum_{i=1}^{n_t} [\tilde{\zeta}_{ij}(\mathbf{x}^h) - \zeta_{ij}(\mathbf{x}^h)]^2 \tag{29}$$

where *n* is the number of storms, n_s is the number of locations and n_t is the number of time instances, while $\zeta_{ij}(\mathbf{x}^h)$ is the storm surge for the *i*th time instance and the *j*th location for the $\mathbf{x}^h$ storm parametrization. The notation $\tilde{\zeta}$ is used to indicate the reconstructed surge either from AE. Later on, in Section 5.5 the same notation will be used for the reconstructed, predicted surge through the surrogate modeling framework.

5.3 Data Preprocessing

The time-series data for the representative nodes selected through clustering (as described in Chapter 2), were split in training (n = 535 × n_t = 170 × n_s = 3000) and testing (n_{test} = 60 × $n_{t,test}$ = 170 × $n_{s,test}$ = 3000) datasets. In both cases (testing and training), a flattening step is introduced to reshape the data in a two-dimensional matrix, by stacking time-series for each location, resulting in dimensions of (n = 535 × $n_t \cdot n_s$ = 510,000) for the training and (n_{test} = 60 × $n_{t,test} \cdot n_{s,test}$ = 510,000) for the testing set. A single continuous vector for each storm scenario is created by sequentially concatenating the 170 time steps recorded at each of the 3000 clustered locations. By representing each storm as a complete vector this flattened structure preserves temporal and spatial information and makes the data easier to handle especially in NN structures like AE that are not necessarily tailored towards time-series data. The training data for the AE, was further divided into a training and a validation sets with 80 percent and 20 percent respectively. This led to a randomly selected 429 storms to serve in the training dataset and 106 storms that were used as the validation dataset.

5.4 Hyperparameter Optimization

Architecture and hyperparameter brief descriptions were provided above, but the main challenge is achieving the best combination of hyperparameters that will reduce the reconstruction loss and will maximize the model performance without overfitting the available training data. To efficiently seek such nearly optimal combinations, the optimization of the model hyperparameters is necessary. Many different AE architecture optimizing routines exist, and here the Optuna framework [172] will be used which employs a search for the most effective model hyperparameters, based on user-defined constraints. Optuna calculates the validation loss using an error metric (for example *MSE*) for a pre-defined number of trials. Optuna effectively explores different hyperparameter combinations to identify the best performing set [172, 173]. Various frameworks exist for this fine tuning of the AE like Hyperopt and SMAC, that are all equally capable of performing this task, but Optuna was selected to be as effective as the others in such a preliminary investigations, so it was adopted here [174]. A random sampling was used here for the exploration of different architectures and combinations of hyperparameters for a couple of hundred trials for each different activation function.

The search space was designed to explore hyperparameters with different ranges, based on prior experience and literature, but also based on the target latent representation size. The ranges of the parameters that were fine-tuned using Optuna are presented in Table 5-1. The number of layers (either for encoding or decoding) was set to one to five to enforce simplicity, acknowledging that the size of the database is not able to accommodate deeper architectures (with higher number of layers and consequently neurons), priority was given to simpler architectures when it comes to selecting the most promising ones. For the batch size, a relatively wide range was set to ensure that the network is appropriately trained based on the size of the database (four to six hundred unique datapoints). For the latent dimension the range was set to lead to a dimension that would be comparable to the PCA implementation (between 80 and 120 latent dimensions). This ensures that the GP computational complexity is approximately the same, and thus a direct comparison between the two surrogate model developments can be established. The number of epochs for the Optuna implementation was kept constant to provide an idea of how well each hyperparameter combination is performing (in terms of model loss and performance). The combinations that appeared to have an accelerated performance, minimizing the train and validation losses, were selected later for further training.

Table 5-1: Ranges Considered for hyperparameter optimization using Optuna.

Hyperparameters	**Lower Limit**	**Upper Limit**
Encoding Dimension	80	140
Learning Rate	1.00E-05	1.00E-01
Batch Size	8	100
Epoch	150	
Num of Encoding Layers	1	5
Number of Decoder Layers	1	5
Optimizer	SGD	ADAM
Number of Neurons in Encoder Layers	200	2500
Number of Neurons in Decoder Layers	200	2500
Number of Trials	100	

Different Optuna runs were performed for the different AFs that were described above to get the best combination(s) of hyperparameters. Table 5-2 here shows some of the Optuna architectures that appeared to perform well for the available number of epochs that was provided. More than 500 Optuna explorative trials were performed, and the architectures for the trials with the lower losses were isolated and selected for further training as shown in Table 5-3 and in more detail in Table 5-4. Trials that might appeared to perform relatively well, but had parameters close to the limit ranges were eliminated (for example, cases where a very low batch number was identified) to avoid overfitting issues.

The identified cases (combinations of hyperparameters yielding the lowest loss error) that were selected for further exploration, were ran independently for at least 5000 epochs, employing also an early stopping criterion (in cases where the validation loss did not improve for more than 50 consecutive epochs), that prevented the model from over-training (overfitting). Once the models were fully trained, the latent representations were retrieved to be used as the output in the surrogate model formulation. Figure 5-8, Figure 5-9 and Figure 5-10 present the training and validation losses for the different metrics that are explained in detail in the next section. As it can be seen from the plots, different AE architectures with different activation functions achieve different levels of accuracy, and for the majority of them, early stopping was enforced. All of the trained AEs achieve relatively low error rates and very close to one coefficients of determination, with the validation error also following similar trends to the training, but without any signs of overfitting (increasing error trends for the validation curves – dashed lines).

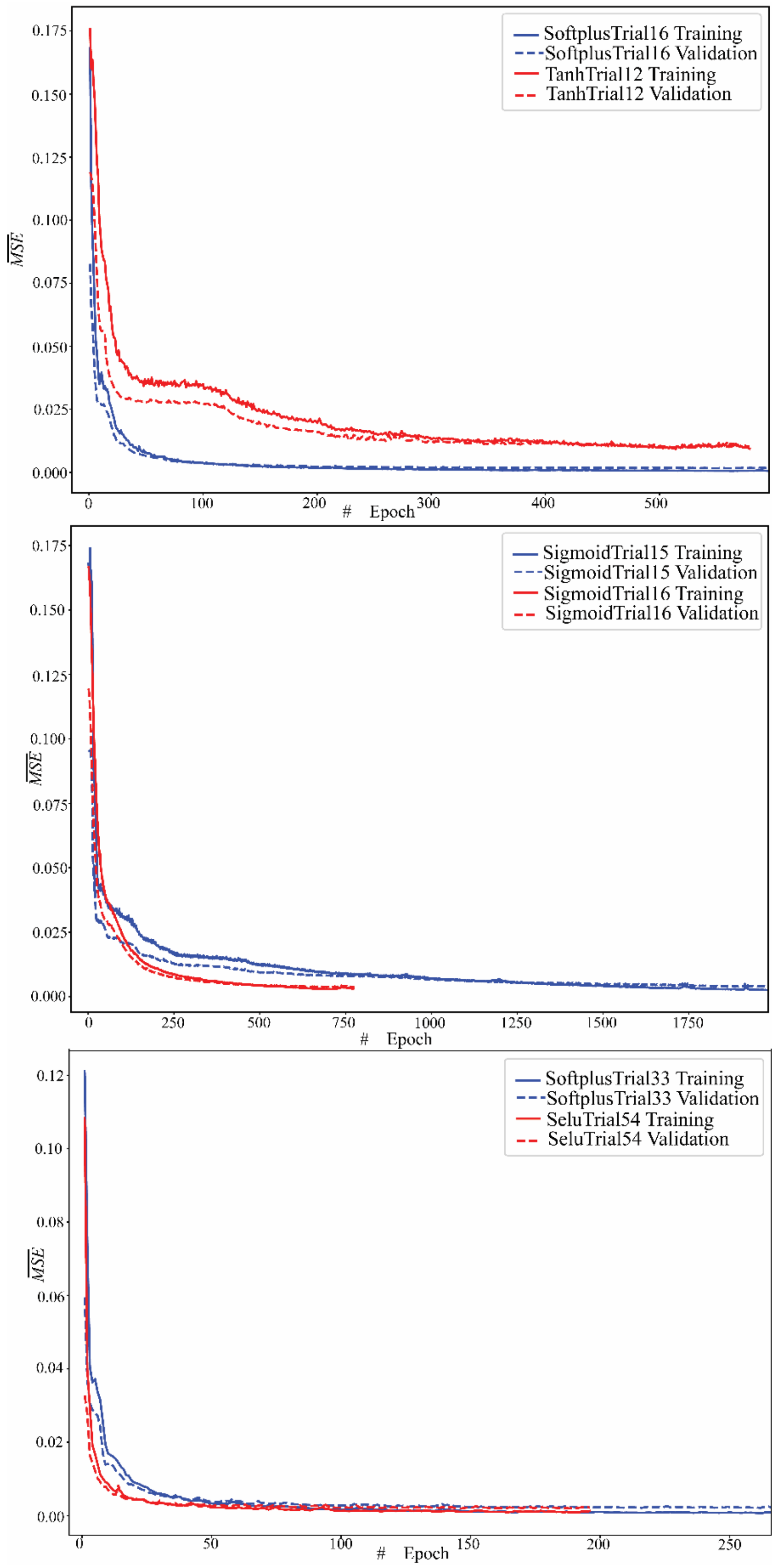


Figure 5-8: The AE training and validation $\overline{MSE}$ *error* for different architectures.

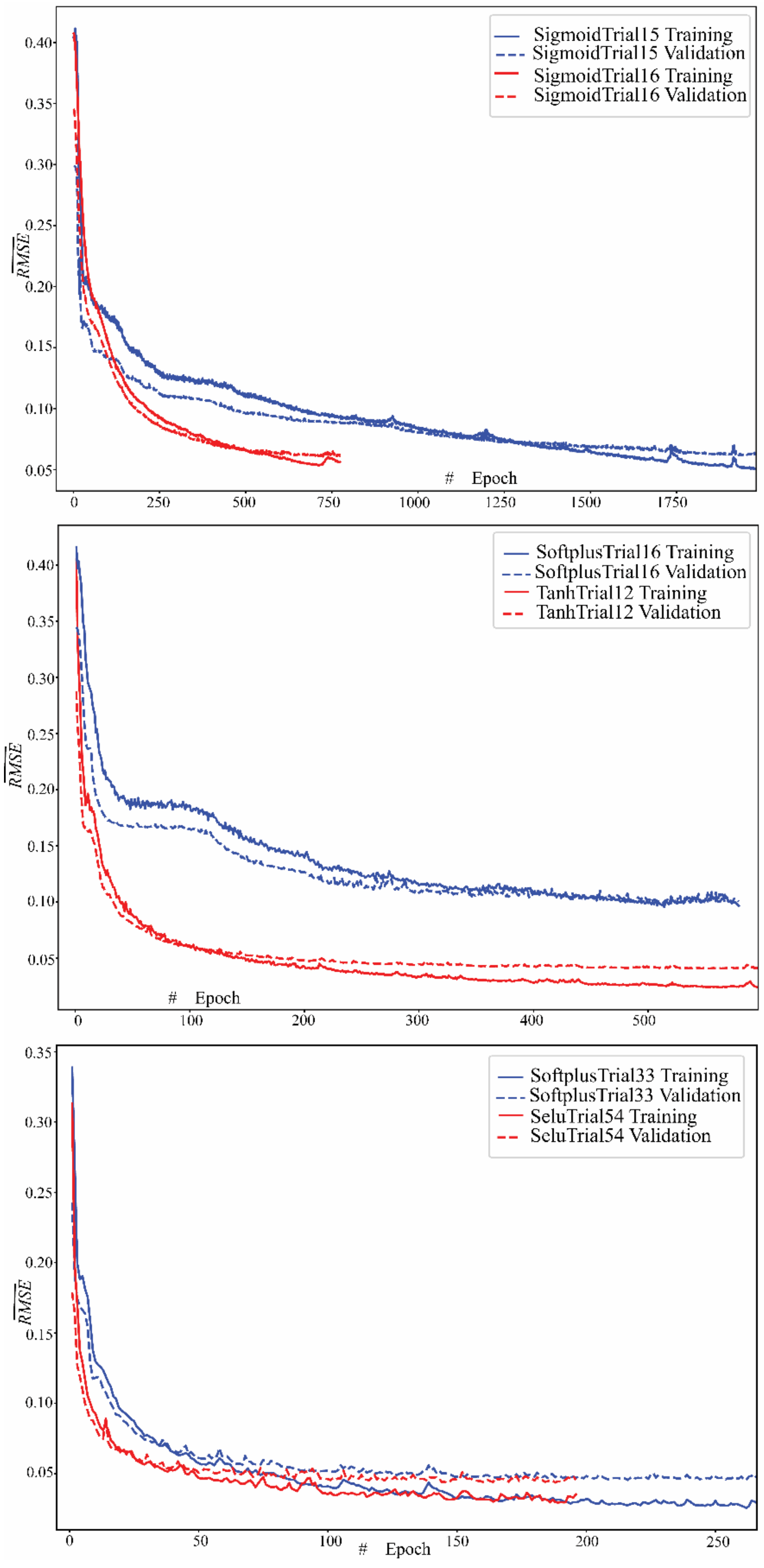


Figure 5-9: The AE training and validation $\overline{RMSE}$ *error* for different architectures.

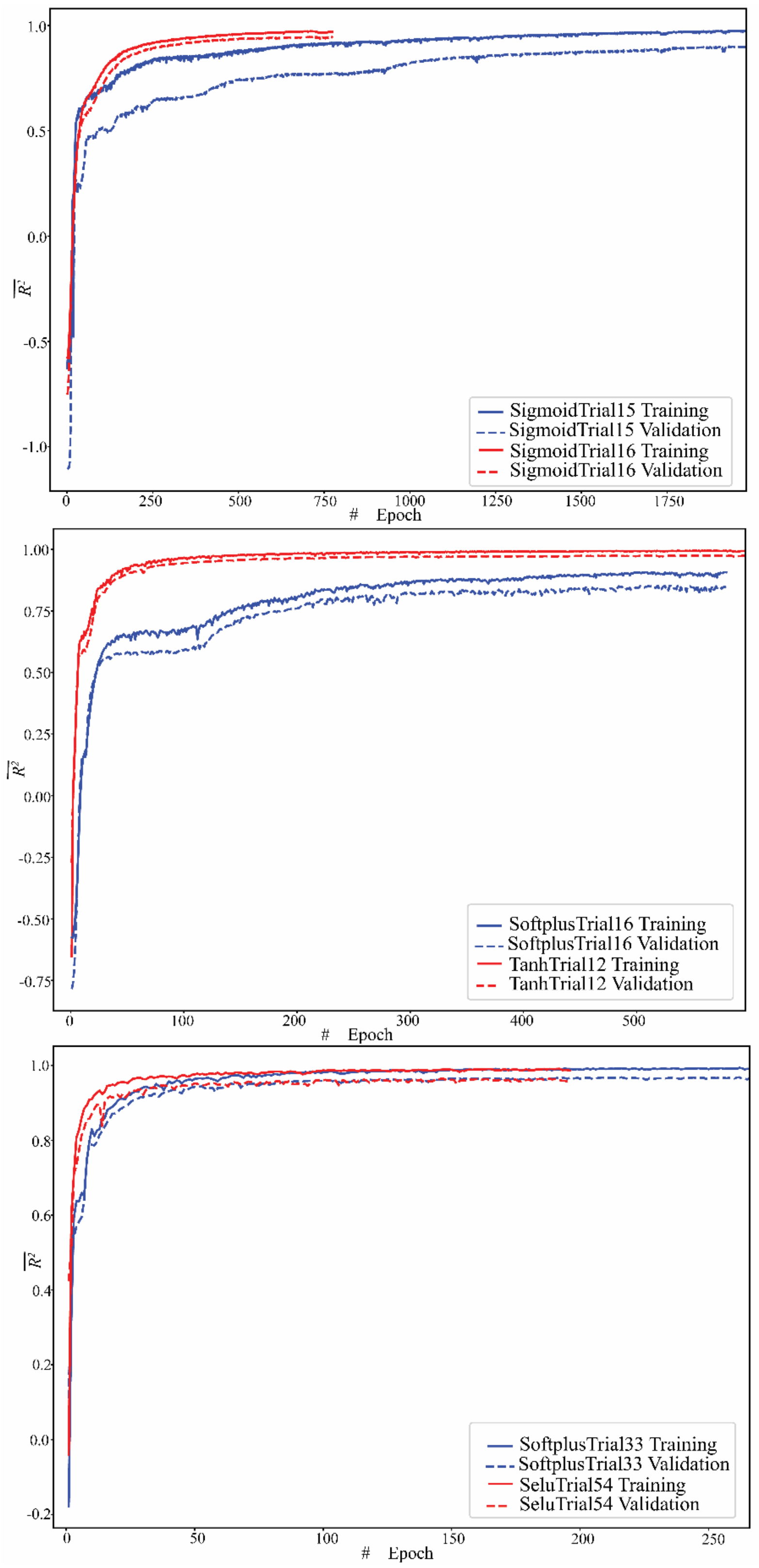


Figure 5-10: The AE training and validation $\overline{R}^2$ *error* for different architectures.

Table 5-2: Some representative optimized AE architectures (from Optuna) performed for different activation functions for 150 epochs.

Trial	AF	Bs	Optmz	Lr	Ls	En Lr1	En Lr2	En Lr3	En Lr4	Dn Lr1	Dn Lr2	Dn Lr3	Dn Lr4	Opt rs	Ef	Tr RMSI	Val RMS	Tr MSE	Val MSE	Tr R sq	Val R sq
5	SELU	42	sgd	0.044100	122	1266				1151				0.018	141	1.982	2.318	4.000	5.373	-44.217	-81.998
54	SELU	24	adam	0.000070	122	800	1572			1407	489			0.002	196	0.035	0.046	0.001	0.002	0.986	0.961
15	Sigmoid	51	adam	0.027558	122	515	1896	2059		1478	2107			0.028	1981	0.051	0.062	0.003	0.004	0.973	0.899
16	Sigmoid	91	adam	0.000021	90	1675				2013				0.021	774	0.056	0.062	0.003	0.004	0.970	0.943
79	Sigmoid	57	adam	0.000019	101	1819	454			1389				0.020	802	0.067	0.071	0.005	0.005	0.955	0.925
16	Softplus	83	adam	0.000032	86	916				2231	557	1804	216	0.004	596	0.024	0.042	0.001	0.002	0.995	0.973
21	Softplus	90	adam	0.000178	122	1185				2454	437	440	703	0.003	241	0.035	0.050	0.001	0.002	0.988	0.963
33	Softplus	30	adam	0.000041	97	1861	1618	2143	329	1521	1286	291		0.003	266	0.029	0.047	0.001	0.002	0.991	0.966
3	Tanh	29	adam	0.022357	132	1538	631	2066		1633	1480			0.064	2110	0.110	0.101	0.012	0.010	0.877	0.849
12	Tanh	67	adam	0.000010	118	433	1529	2313		2422	1412	1055	1147	0.025	579	0.097	0.100	0.009	0.010	0.907	0.849
	Tanh	97	adam	0.000046	133	343				1544	185			0.030	158	0.189	0.146	0.036	0.022	0.661	0.488
36	Tanh	98	adam	0.000029	133	1541	1833	340	905	516	1898			0.017	483	0.095	0.081	0.009	0.007	0.915	0.845
21	Tanh	94	adam	0.000051	115	1487				1738				0.013	496	0.091	0.079	0.008	0.006	0.920	0.866
33	Tanh	63	adam	0.000012	103	859	1935	1234	652	758	2287			0.014	455	0.081	0.082	0.007	0.007	0.935	0.900

Table 5-3: Optimized architectures with the smallest reconstruction losses that were selected for further investigation.

Trial	AF	Bs	Optmz	Lr	Ls	En Lr1	En Lr2	En Lr3	En Lr4	Dn Lr1	Dn Lr2	Dn Lr3	Dn Lr4	Opt rs	Ef	Tr RMSE	Val RMS	Tr MSE	Val MSE	Tr R sq	Val R sq
54	SELU	24	adam	0.000070	122	800	1572			1407	489			0.002	196	0.04	0.05	0.001	0.002	0.99	0.96
15	Sigmoid	51	adam	0.027558	122	515	1896	2059		1478	2107			0.028	1981	0.05	0.06	0.003	0.004	0.97	0.90
16	Sigmoid	91	adam	0.000021	90	1675				2013				0.021	774	0.06	0.06	0.003	0.004	0.97	0.94
16	Softplus	83	adam	0.000032	86	916				2231	557	1804	216	0.004	596	0.02	0.04	0.001	0.002	0.99	0.97
33	Softplus	30	adam	0.000041	97	1861	1618	2143	329	1521	1286	291		0.003	266	0.03	0.05	0.001	0.002	0.99	0.97
12	Tanh	67	adam	0.000010	118	433	1529	2313		2422	1412	1055	1147	0.025	579	0.10	0.10	0.009	0.010	0.91	0.85

Table 5-4: Overview of key features for the optimized architectures with the smallest losses as produced by Optuna fine-tuning routine.

Model	**Number of Decoder Layers**	**Number of Encoder Layers**	**Latent Space**
selutrial5	1	1	122
selutrial54	2	2	122
sigmoidtrial15	2	3	122
sigmoidtrial16	1	1	90
sigmoidtrial79	1	2	101
softplustrial16	4	1	86
softplustrial21	4	1	122
softplustrial33	3	4	97
tanhtrial3	2	3	132
tanhtrial12	4	3	118
tanhtrial25	2	1	133
tanhtrial36	2	4	133
tanhtrial21	1	1	115
tanhtrial33	2	4	103

Explanation of column headers for Table 5-2 and Table 5-3.
Tr = Trial number in Optuna results generated
AF - Activation Function
Ei – Number of Epoch initially set to train the architecture
Bs – Batch size
Optmz - Optimizer Used
Lr – Learning Rate
Ls – Latent Space
En Lr1 – Neurons in Encoder Layer1 / En Lr2 – Neurons in Encoder Layer2
En Lr3 - Neurons in Encoder Layer3 / En Lr4 - Neurons in Encoder Layer4
En Lr5 – Neurons in Encoder Layer5 / Dn Lr1 – Neurons in Decoder Layer1
Dn Lr2 - Neurons in Decoder Layer2 / Dn Lr3 - Neurons in Decoder Layer3
Dn Lr4 - Neurons in Decoder Layer4 / Dn Lr5 - Neurons in Decoder Layer5
Opt rs – Loss calculated by Optuna
Ef – Epoch at which the model stopped learning
Tr RMSE - Training RMSE
Val RMSE – Validation RMSE
Tr MSE – Training MSE
Val MSE- Validation MSE
Tr R square- Training R square
Val R square- Validation R square

5.5 Surrogate Model Development and Storm Surge Predictions

The surrogate model that was used here is the same GP-based model that has been extensively used in storm surge assessments [31, 50, 56, 59, 75], and as already mentioned above it was kept the same to allow for an direct comparison with work that has been done for the same database on linear dimensionality reduction (PCA).

Each of the trials that were selected for further training provided a latent space representation that was used as the output of interest in the surrogate model formulation as detailed in Chapter 2. For the input, the six-dimensional storm parametrization was used, resulting in the development of multiple predictive models, one for each AE trial. The surrogate model was developed for each of the latent dimensions explicitly as a parallel model implementation was deemed unnecessary, since there were not inherent characteristics that should be explicitly maintained through the correlation function (for example positive definiteness, symmetry etc.). The model was validated in a Leave-One-Out Cross Validation (LOO-CV) setting using the error metrics that were presented in Section 5.2. Once the latent space predictions are provided (either in a validation or test setup), the already AE decoding part in each case is called to reconstruct the surrogate model predictions back to the original storm surge space. Once this step is performed, the final predictions (coming from the surrogate model and the dimensionality reduction process) are compared against the original NACCS data, using formulas that can quantify the accuracy in different ways. The validation, as in [59], is performed with respect to the same locations and time instances, to simplify the presentation of the considered validation metrics. The overall model performance will be evaluated using normalized and unnormalized metrics. Let n_{uv} be the test sample storm size, and $\zeta_{ij}^{a}(x^{h})$ denotes the surge for *i*th-series instance and *j*th node for the *h*th storm having as storm input $\mathbf{x}^h$. with the predictions with respect to above surge defined as $\tilde{\zeta}_{ij}(x^h)$, the unnormalized error metric will correspond here to the square root of the average mean squared error:

$$\overline{RMSE} = \sqrt{\frac{1}{n_{uv}n_s}\sum_{h=1}^{n_{uv}}\sum_{j=1}^{n_s}\frac{1}{n_t}\sum_{i=1}^{n_t}[\tilde{\zeta}_{ij}(\mathbf{x}^h) - \zeta_{ij}(\mathbf{x}^h)]^2} \tag{30}$$

This metric has the same units as the quantity of interest (surge), and lower values would correspond to higher accuracy. It should be noted that for $\overline{RMSE}$ the order of the error summation across the locations, storms and time instances does not matter. For normalized statistics though, the order should be explicitly defined [59], leading to an average across nodes and storms normalized series-error statistics. The coefficient of determination (R^2) and the correlation coefficient (*cc*) for the *j*th node and the $\mathbf{x}^h$ storm scenario are given, respectively, by:

$$cc_j(\mathbf{x}^h) = \frac{\frac{1}{n_t}\sum_{i=1}^{n_t}[\tilde{\zeta}_{ij}(\mathbf{x}^h) - \frac{1}{n_t}\sum_{i=1}^{n_t}\tilde{\zeta}_{ij}(\mathbf{x}^h)][\zeta_{ij}(\mathbf{x}^h) - \frac{1}{n_t}\sum_{i=1}^{n_t}\zeta_{ij}(\mathbf{x}^h)]}{\sqrt{\frac{1}{n_t}\sum_{i=1}^{n_t}[\tilde{\zeta}_{ij}(\mathbf{x}^h) - \frac{1}{n_t}\sum_{i=1}^{n_t}\tilde{\zeta}_{ij}(\mathbf{x}^h)]^2 \frac{1}{n_t}\sum_{i=1}^{n_t}[\zeta_{ij}(\mathbf{x}^h) - \frac{1}{n_t}\sum_{i=1}^{n_t}\zeta_{ij}(\mathbf{x}^h)]^2}} \tag{31}$$

$$R_j^2(\mathbf{x}^h) = 1 - \frac{\frac{1}{n_t}\sum_{i=1}^{n_t}[\tilde{\zeta}_{ij}(\mathbf{x}^h) - \zeta_{ij}(\mathbf{x}^h)]^2}{\frac{1}{n_t}\sum_{i=1}^{n_t}[\zeta_{ij}(\mathbf{x}^h) - \frac{1}{n_t}\sum_{i=1}^{n_t}\zeta_{ij}(\mathbf{x}^h)]^2} \tag{32}$$

The average weighted values of the above statistics are respectively defined as :

$$\overline{cc} = \frac{\sum_{h=1}^{n_{uv}}\sum_{j=1}^{n_s} v_j^h cc_j(\mathbf{x}^h)}{\sum_{h=1}^{n_{uv}}\sum_{j=1}^{n_s} v_j^h} \tag{33}$$

$$\overline{R^2} = \frac{\sum_{h=1}^{n_{uv}} \sum_{j=1}^{n_s} v_j^h R_j^2(\mathbf{x}^h)}{\sum_{h=1}^{n_{uv}} \sum_{j=1}^{n_s} v_j^h} \tag{34}$$

where v_j^h defines the weights for *j*th location and the *h*th storm scenario in order to establish an importance factor in the normalized error as done previously in [59]. The motivation behind this was to ensure that small discrepancies in the predictive behavior for small magnitude experienced surges won't impact significantly the metrics, since the higher accuracy should be evaluated for large surge values, where over- or under-predicting is of crucial importance. The weight definition, was selected to be directly related to the surge difference, which for the series response with respect to the *j*th node and the storm input $\mathbf{x}^h$ is calculated as:

$$\Delta\zeta_j^h = \max_i(\zeta_{ij}(x^h)) - \min_i(\zeta_{ij}(x^h)) \tag{35}$$

where $\max_i(.)$ and $\min_i(.)$ refers to the maximum and minimum over index *i*. A binary definition is further adopted here, where a weight is activated if the surge different is larger than a certain value Δ_ζ, as in:

$$v_j^h = I[\Delta\zeta_j^h > \Delta_\zeta] \tag{36}$$

where I[.] is the indicator function, which is 1 if the condition inside the brackets holds else it is considered as zero.

The above equations will be used to quantify the accuracy of the surrogate modeling framework developed on both the linear (PCA) and the nonlinear (AEs) dimensionality reduction techniques. The $\Delta\zeta$ will be taken here as 0.5 m to also account for any steric adjustments in the numerical model (ADCIRC) on top of just the large surge changes across the time-series.

The surrogate model developments were tuned explicitly on each latent representation, and the same kernel and basis functions were used in all cases (the power exponential and $\mathbf{f}(\mathbf{x}) = \begin{bmatrix} 1 & x_1 & x_2 & \ldots & x_{n_x} \end{bmatrix}^T$, respectively) for consistency. Maximum Likelihood Estimation (MLE) calibration is used in all cases for the hyperparameter optimization and a Leave-One-Out-Cross-Validation setting was used to evaluate the surrogate model performance during training.

CHAPTER 6:

RESULTS AND DISCUSSION

In this Chapter, the results from the various investigations will be presented and commented on in detail. The AE (nonlinear DR) and PCA (linear DR) performances and their ability to reconstruct the original output will be compared for new storms that were not used during the training of the models, along with results from their combination with the GP surrogate model will be presented. Extensive comparisons across the entire geospatial domain of interest as well as explicit time-series predictions for certain challenging locations within the domain will be quantified, presented and commented.

6.1 Problematic Storm Surge Locations within the Domain of Interest

As mentioned above, this thesis is set to explore nonlinear dimensionality reduction methods to capture the inherent nonlinearity of the storm surge data across the storm space. The latent representation is then used to train a surrogate model to predict storm surge at the different locations. Aiming to establish a consistent comparison between nonlinear and linear DR techniques, the already established framework of Kyprioti, Irwin [59] was leveraged to develop a surrogate model on PCA latent dimensions. Following the published work, 72 latent components were kept capturing approximately 97% of the data variance. Independent surrogate models were trained for each latent dimension, utilizing the settings of the pre-existing investigation, leading to an average model accuracy (across all latent outputs) of $\overline{R}^2 = 0.77$, $\overline{cc} = 0.85$ and $\overline{RMSE} = 0.0405$.

Despite the overall satisfactory performance of the surrogate model built on PCA latent components, a more in-depth investigation along the time-series for each location and each storm was performed to identify cases where the high nonlinearities in the storm space, hamper the performance of this approach (PCA with GP surrogate modeling). This closer examination led to the identification of what will be called as “problematic locations” across multiple storm scenarios, where the established surrogate model framework relying on linear DR techniques failed to capture and predict accurately.

For every storm, the total absolute difference between the predicted (reconstructed through the PCA inverse mapping) and actual values is calculated, and the locations are ranked in the decreasing order. For each storm the top 100 places with the largest discrepancies are marked. This process is repeated for all the different storms. Once the process for all the storms is completed, the locations that systematically appear to experience larger surge predictive errors across their time-series evolution are isolated and flagged. This group of locations, for short called problematic location, is then identified as the target group that nonlinear DR should be able to capture better and inform more accurately the subsequent development of the surrogate model. The problematic locations that were identified through this process, out of the clustered 3000, were around 130 and are plotted in Figure 6-1. To better demonstrate those discrepancies, some selected time-series for those locations and some specific storm scenarios are further plotted in Figure 6-3, with the locations put also on a map in Figure 6-2.

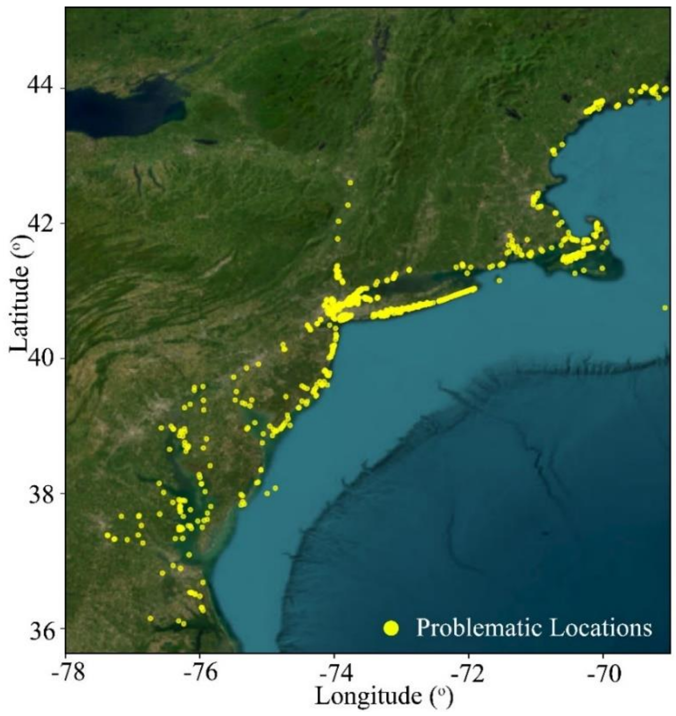


Figure 6-1: The identified problematic nodes (where the predictive error of the surrogate model developed on the PCA components was large across multiple storms) on the map. As is evident, most of these locations are onshore where potential geomorphological challenges exist, generating a nonlinear behavior for the surge across different storm scenarios.

6.2 Test Sample Dataset and Benchmark PCA Surrogate Model Development

The models were validated using test-sample dataset that was randomly selected as the 10% of the original database, leading to 60 storm scenarios. The proportion of 90-10% was selected here for training and testing dataset definition respectively to tackle issues that might arise when it comes to smaller databases like the one here, where only 600 data points are available (the model might underperform due to inadequate training if a large portion of the dataset is used for testing). These 60 scenarios were entirely hidden from all models and were never used for the training of either the AE or the surrogate model. The same storms were also hidden from the PCA development to establish a fair comparison, although it should be noted that underlying behavioral features within the test dataset might be slightly favorable for either linear or nonlinear storm surge patterns.

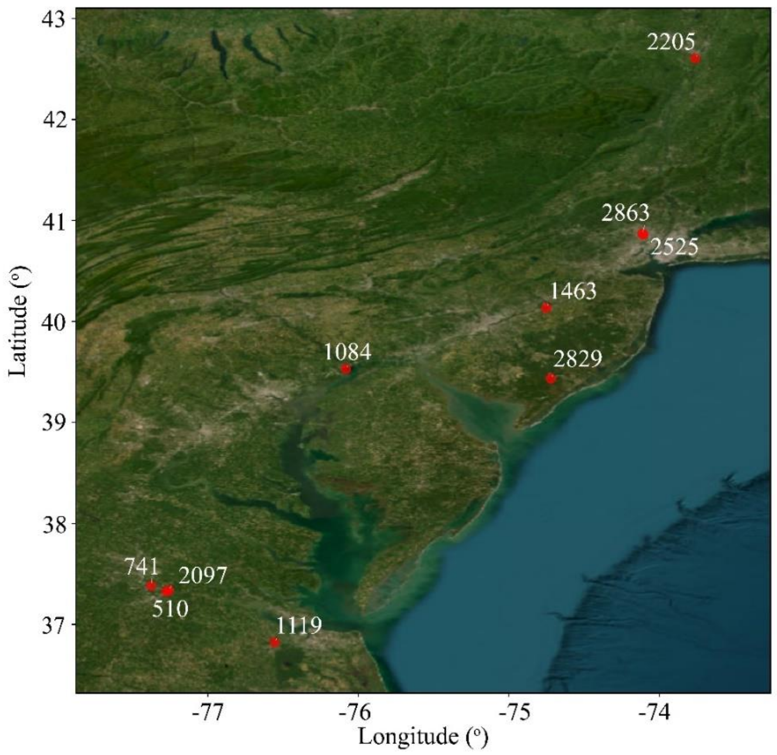


Figure 6-2: The geographical locations of the problematic nodes that are plotted in Figure 6-3 in the form of time-series.

For the establishment of a consistent comparison between the different surrogate model frameworks a similar case to the one presented in [59] was generated, utilizing PCA and a GP surrogate model to create a benchmark case that would serve as the point or reference for the comparisons established in subsequent sections.

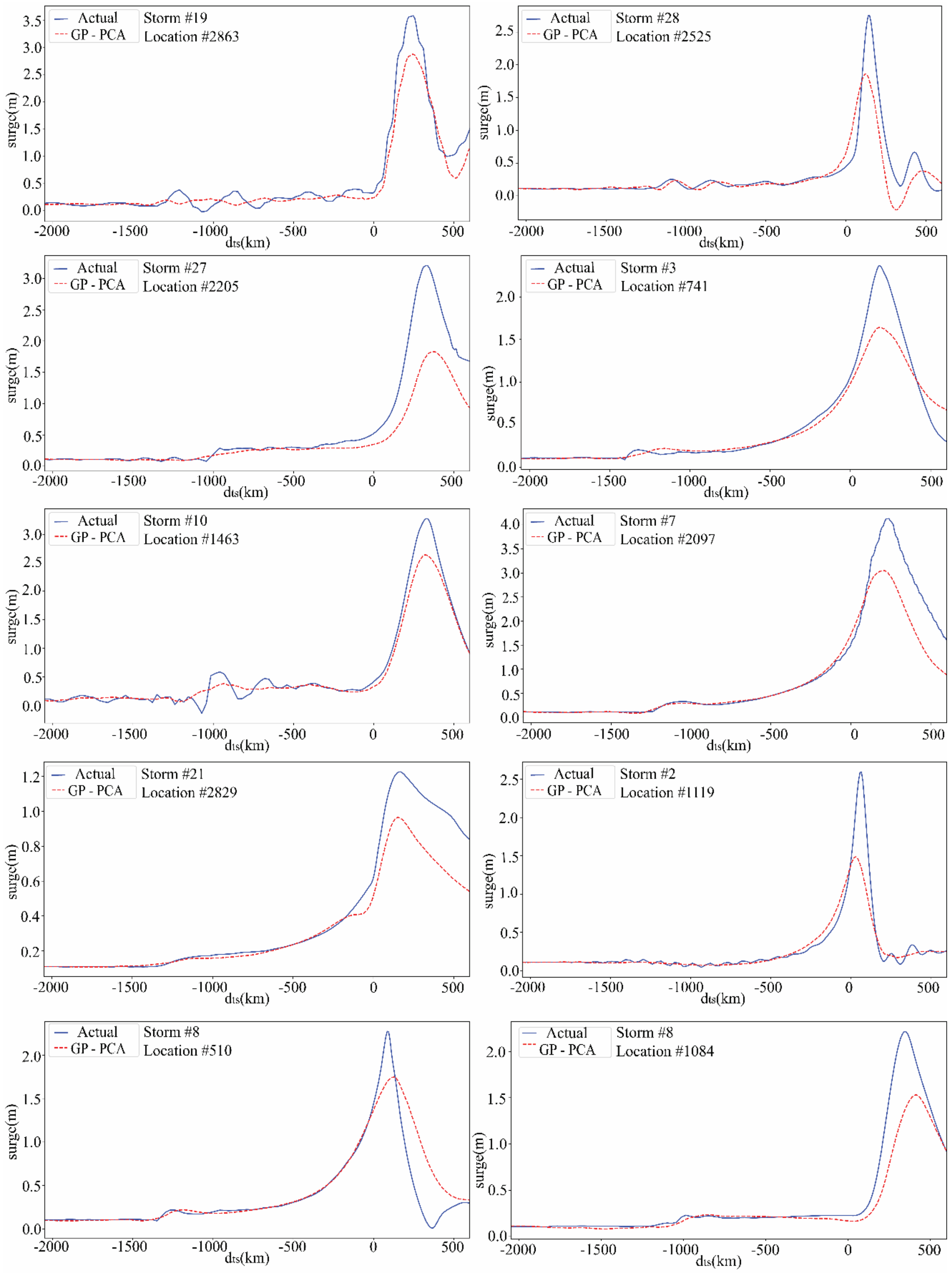


Figure 6-3: Time-series of problematic locations for certain storm scenarios. The actual and the predictive surge is plotted here for comparison.

6.3 Test Data Performance Comparison for Linear (PCA) and Nonlinear (AE) DR Reconstructions

Here the reconstruction abilities for the linear and nonlinear DR techniques will be compared. For PCA, the latent space was obtained for the test data (keeping the same percentage of variance for the 60 storms – 97% – as in the training, which results in only 12 latent vectors) and was reconstructed back (with some simple matrix manipulations). For the AE, the test data was passed through the trained AE architectures that were presented in Table 5-3 to acquire the reconstructed data. The reconstruction performance of both PCA and AE is compared at a global level across all storms and locations using the equations presented in Section 5.2 and is presented in Table 6-1. The accuracy is quantified through different validation metrics of all the optimized AEs. Looking at the metrics, Softplus AF seems to be outperforming for the most part all other AF in reconstruction, but by not a significant margin (all AEs seem to be performing equally well). For further validation on a time-series basis, Figure 6-4 shows 10 random locations within the domain, and the reconstructed time-series using PCA and two different AEs.

Table 6-1: Training and validation results for the different AEs that were selected for further training in Chapter 5.

Trial#	**Activation Function**	**Train** $\overline{RMSE}$	**Validation** $\overline{RMSE}$	**Train** $\overline{MSE}$	**Validation** $\overline{MSE}$	**Train** $\overline{R}^2$	**Validation** $\overline{R}^2$
54	SELU	0.04	0.05	0.001	0.002	0.99	0.96
15	Sigmoid	0.05	0.06	0.003	0.004	0.97	0.90
16	Sigmoid	0.06	0.06	0.003	0.004	0.97	0.94
16	Softplus	0.02	0.04	0.001	0.002	0.99	0.97
33	Softplus	0.03	0.05	0.001	0.002	0.99	0.97
12	Tanh	0.10	0.10	0.009	0.010	0.91	0.85

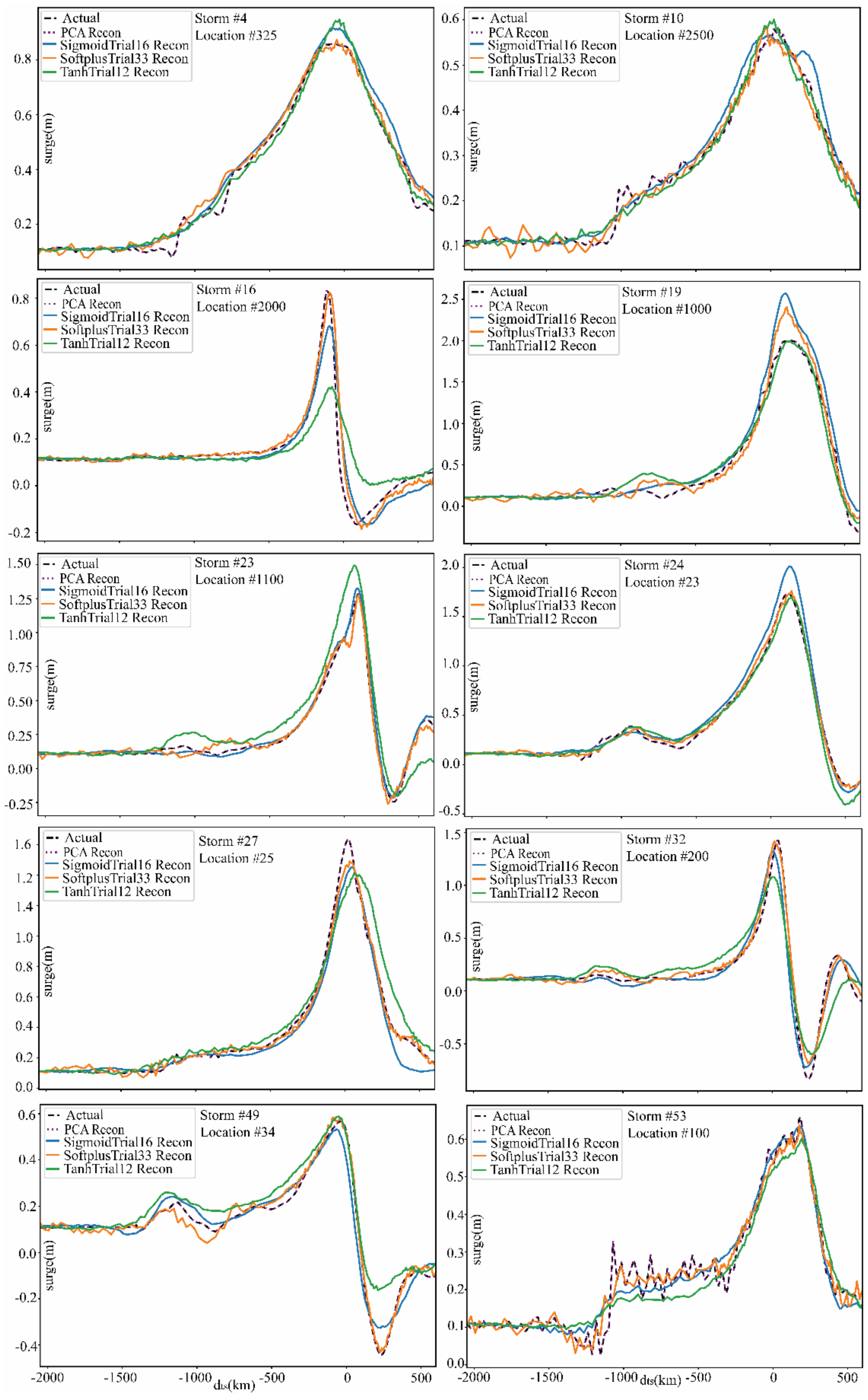


Figure 6-4: PCA and AE reconstruction comparison with the original data across 10 different locations and storms.

The overall reconstruction approximation for most of the cases as shown in Figure 6-4 are really good for PCA and the AE with Softplus as its AF, but both the hyperbolic tangent and the sigmoid, appear to face some challenges in cases especially where negative surge is developed or in some cases for larger experienced surges, they appear to over-predict.

6.4 Surrogate Model Performance Comparisons for Predicting Storm Surge Time-Series

The latent spaces generated from each AE were used as the output for the training of the surrogate model in predicting across different storm scenarios. Once the model was trained, the latent representation was predicted based on the input storm features. These were subsequently passed through the trained AE decoding routine, to reconstruct the original space (surge estimates). The same process was followed for PCA, but both processes of reducing the dimensionality and reconstructing the latent representation to the original space involve minimal computational burden (simple matrix decomposition and matrix multiplications). In every case, to evaluate the predictive accuracy of the established framework, the reconstructed surge data were compared against the original test data, in both global scales (across all storms, locations and time instances) and explicitly looking at surge time-histories or surge spatial distributions.

Table 6-2 is providing the global metrics for all the different surrogate models that were built on AEs (for short these will be denoted as GP-AE with the specific AF mentioned next to it), and with PCA (mentioned as GP-PCA). As it is evident, the discrepancies are not large among any of the different approaches, since GP-PCA is overall performing well across the entire domain, especially when linearity is prevailing in the storm space. For the problematic nodes though, that were identified earlier, the discrepancies are expected to be more obvious, with a much better match across the time-history. For that reason, results are plotted either in the form of time series (for certain storms and locations) or in the form of spatial maps (for certain storms and time instances). Figure 6-5 and Figure 6-6 show twenty distinct locations (with 10 belonging in the identified problematic locations subset) that AE when combined with a GP surrogate model outperforms the GP-PCA model in predicting the storm surge in many cases, by capturing more closely the entire time-history. All the locations are also presented on a map for reference in Figure 6-7. Different magnitudes and behaviors of surge (with some showing also negative surge past the landfall) were captured accurately by most of the AE investigated architectures. If a single model had to be chosen out of all the investigated AE implementation, it seems that the one using as AF the Softplus (trial 33 to be exact) seems to be able to capture the surge behavior most effectively, although the global statistics presented in Table 6-2 seem to favor also the hyperbolic tangent AE.

Focusing finally on the spatial surge distribution, one can see that both the GP-PCA and the GP-AE for the Softplus (trial 33) AF, seem to be performing equally well when compared to the original data for that specific time instance and storm. This proves that PCA is a good enough approach for locations that are not experiencing high nonlinearities in the storm space, with more elaborate techniques becoming efficient for certain locations within the domain.

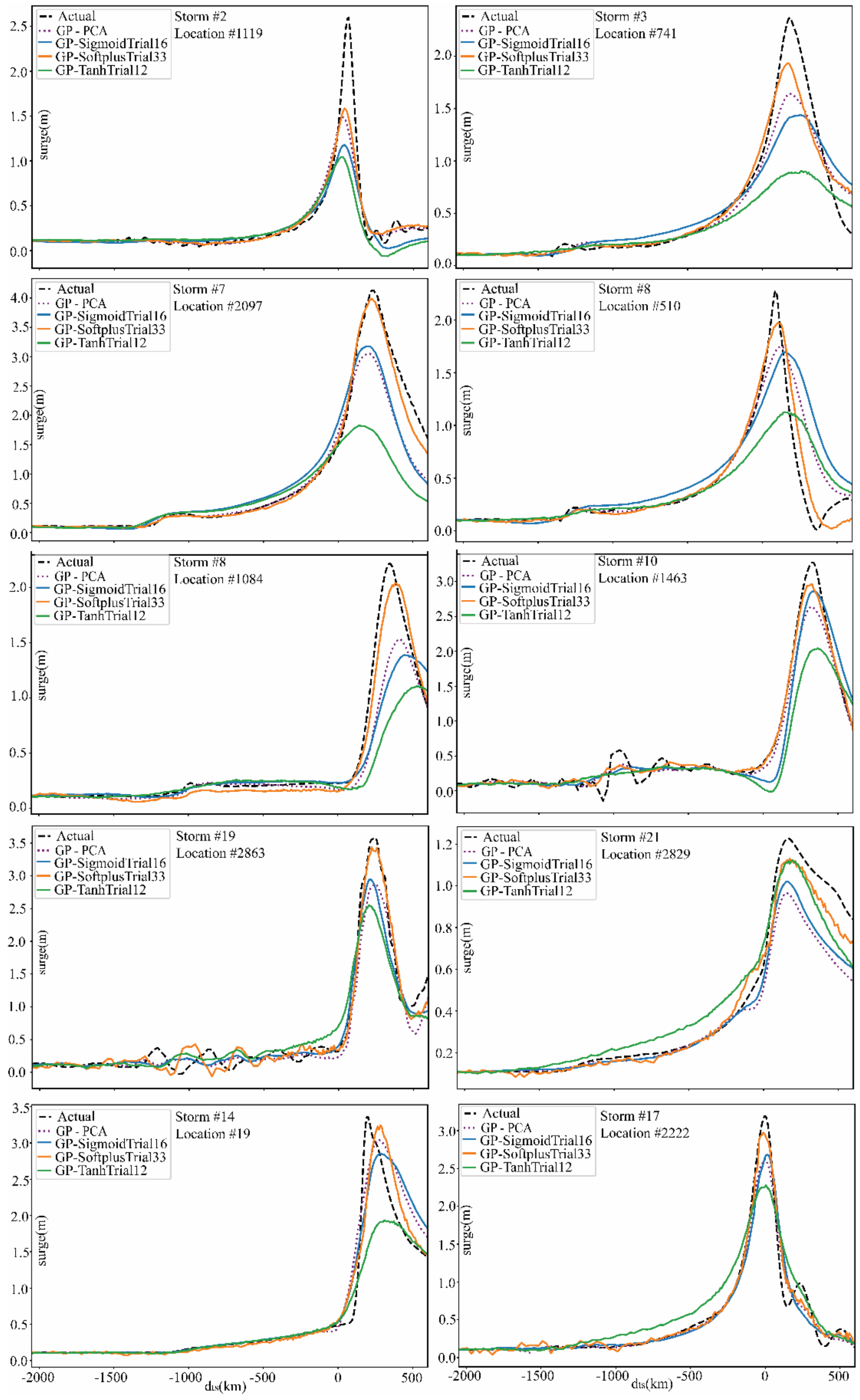


Figure 6-5: Comparison of GP-PCA reconstructed with GP-AE reconstructed cases for 10 problematic locations.

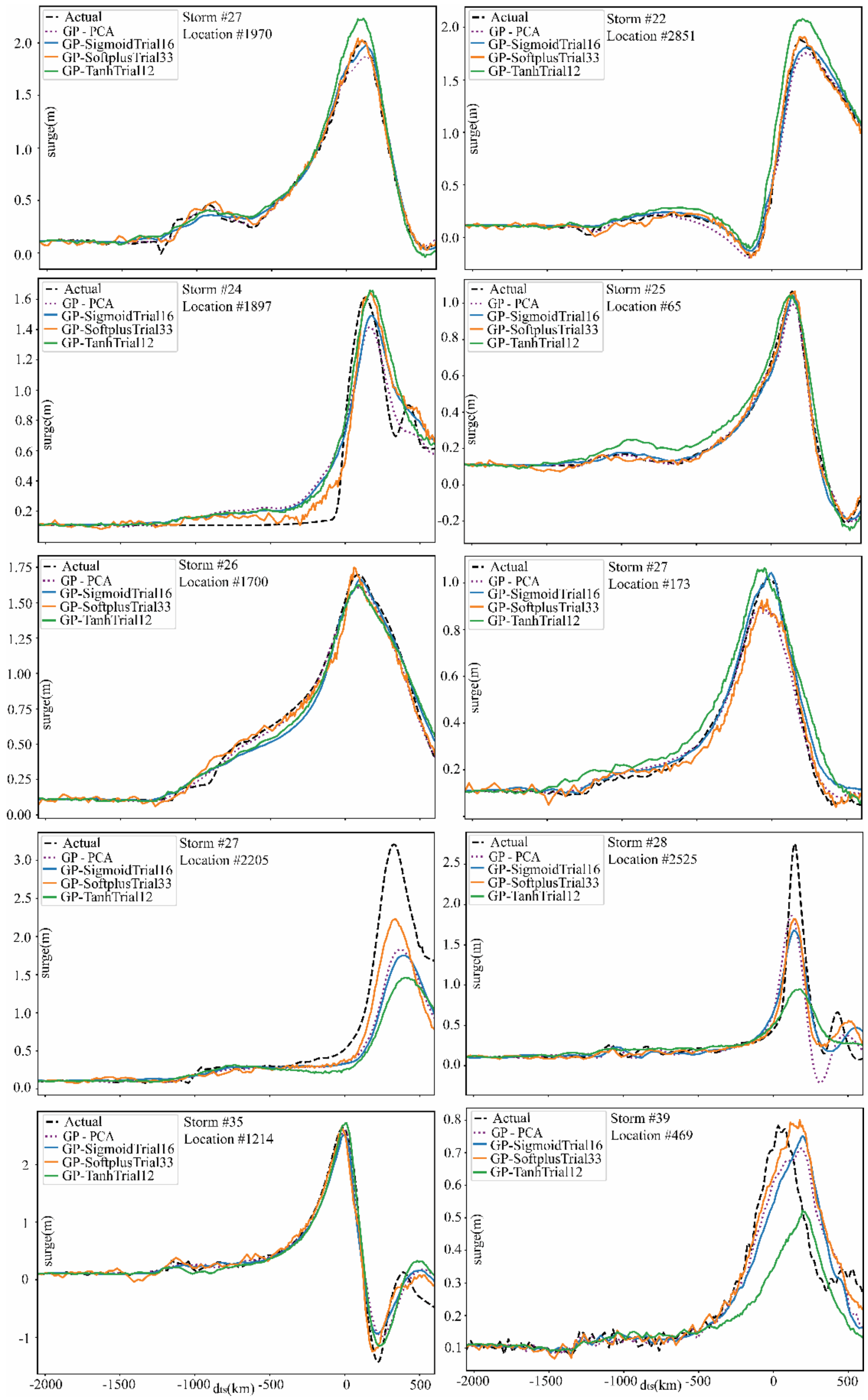


Figure 6-6: Comparison of GP-PCA reconstructed with GP-AE reconstructed cases for 10 (different than Figure 6-5) problematic locations.

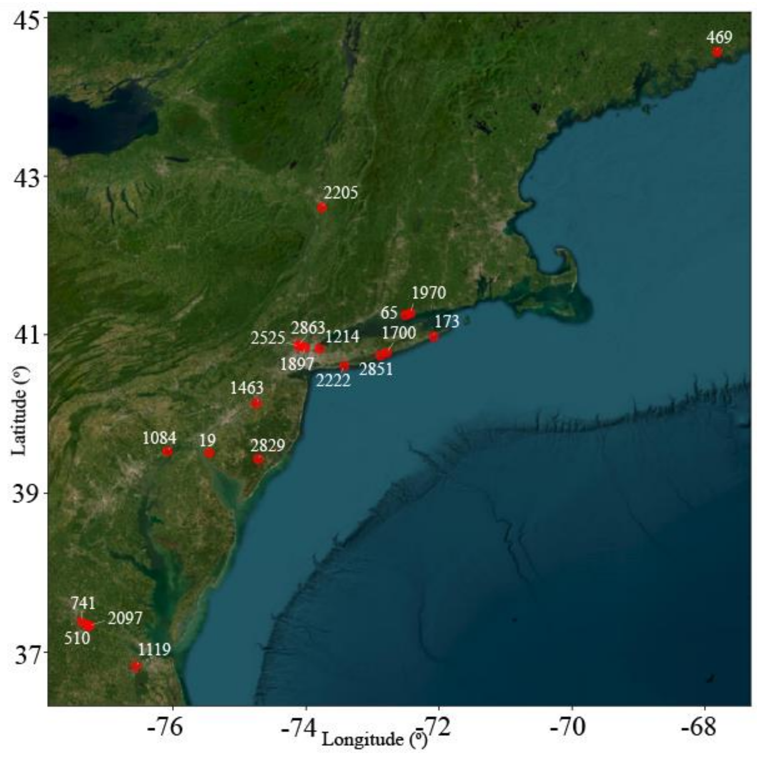


Figure 6-7: Locations explored for the comparison of GP-AE and GP-PCA in Figure 6-6 and Figure 6-5.

Table 6-2: Validation results for all the Autoencoder models with different activation functions along with GP.

Model	$\overline{RMSE}$	$\overline{MSE}$	$\overline{R}^2$	$\overline{cc}$
Softplustrial16	0.0414	0.0028	0.7974	0.9229
Softplustrial33	0.0438	0.0031	0.7782	0.9147
tanhtrial12	0.0689	0.0059	0.9094	0.9575
Selu54	0.0449	0.003	0.7497	0.9056
Sigmoidtrial15	0.0531	0.0047	0.7371	0.9053
Sigmoidtrial16	0.0515	0.0046	0.7705	0.9181
PCA	0.0442	0.0036	0.8250	0.9305

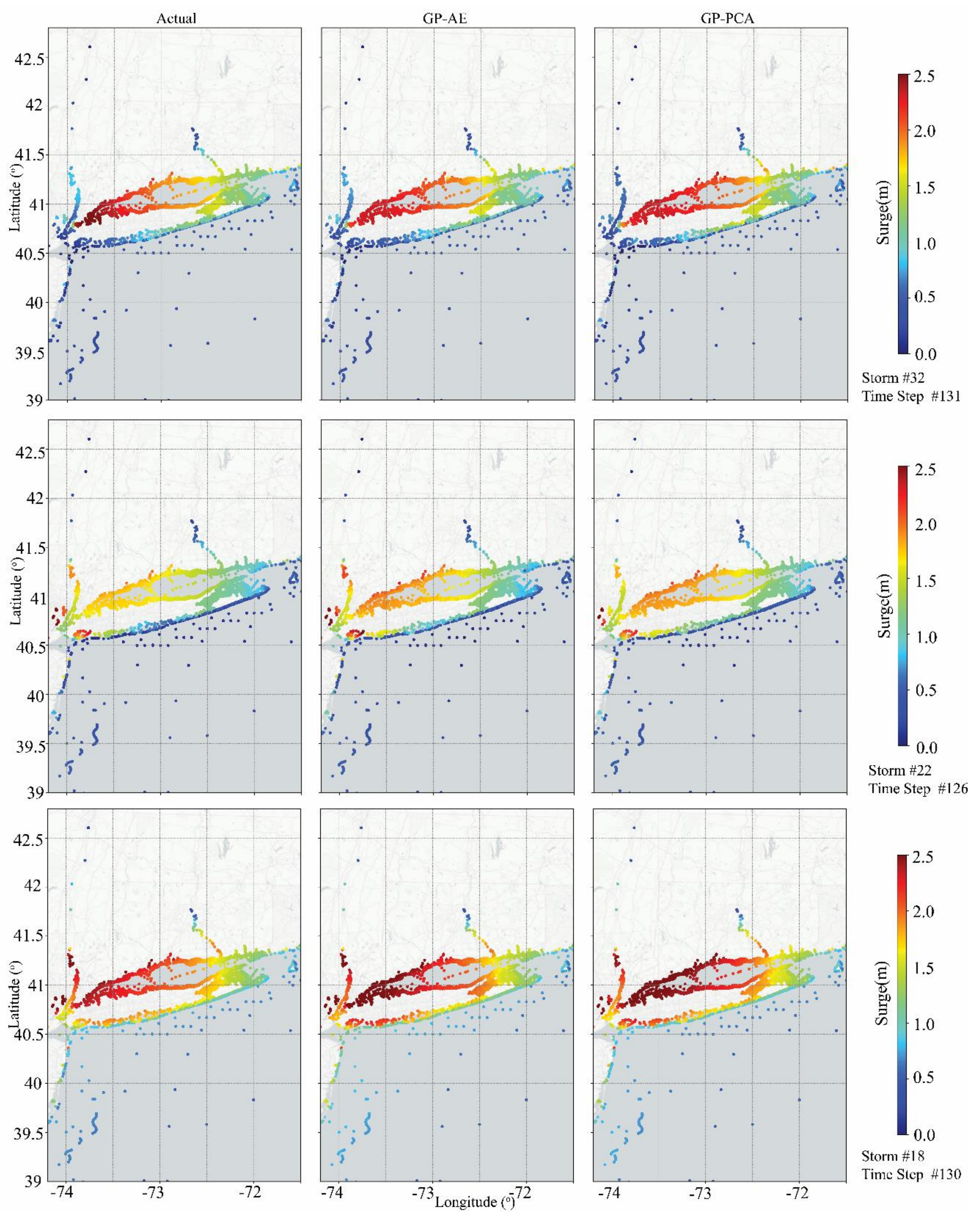


Figure 6-8: Spatial distribution for a specific region within the domain of interest (Manhattan) for surge response at a specific instance for one test-sample storm (each row shows a different storm and different time instance). The original response is compared to the predictive response using PCA and AE (softplustrial33) as the dimensionality reduction techniques.

CHAPTER 7:

CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

The aim of the thesis was to explore nonlinear dimensionality reduction techniques as a means to condense spatiotemporal information that would be later used for the development of a surrogate model to predict storm surge time series. Such a combined framework could be useful in emergency response management efforts, since its development (training of the individualized model components) can be performed offline enabling its use for real-time decision making, when time constraints for evacuation and resource allocation are limited as a hurricane event approaches landfall.

Relying on existing literature that had extensively explored linear dimensionality reduction techniques (Principal Component Analysis) for large geospatial domains and some that had used nonlinear techniques for small number of locations, this thesis investigated Deep AutoEncoders extensively for reducing dimensionality of time-series surge data for large geospatial domains (with a couple of thousand locations of interest).

The introduction of a nonlinear dimensionality reduction was advocated here in an effort to improve the predictive abilities of the surrogate model, for certain locations within the domain that might experience high nonlinearities in the storm space, something that linear techniques might not be able to embed appropriately. This can happen for many reasons, including local geomorphology and spatial features, as well as proximity to the coastline and water (ocean or riverine). The North Atlantic Comprehensive Coastal Study was used as the database to demonstrate the potential of the developed framework, and a hybrid clustering (involving geospatial and response features) was enforced to reduce the locations to a manageable number for the computational resources that were available.

Autoencoder was chosen due to its flexibility in architecture and computational abilities among other NLDR techniques. A brief architecture exploration of the Autoencoder types was performed that led to the selection of the family of Deep AutoEncoders. Different hyper-parameters for a fully connected, dense layered, Deep AutoEncoder were explored extensively using optimization algorithms to retrieve the most appropriate architectures for the model. Among other, different nonlinear, smooth activation functions were investigated, that did not have any significant saturation features (such as the rectified linear). This systematic exploration revealed certain architectures with various activations functions (hyperbolic tangent, SELU, Sigmoid and Softplus) were found to perform really well for optimized architectures, minimizing the reconstruction error. When compared to the linear dimensionality reduction technique (PCA) their superiority for certain time-histories within the domain was significant as expected.

The latent representations from the different dimensionality reduction techniques and the different AE models, were used subsequently to train a Gaussian Process surrogate model. High accuracy was achieved during training, and the ability of each framework to provide predictions was evaluated on a test dataset. The surrogate model predictions were reconstructed back to the original output (surge) and a comparative analysis utilizing different error metrics was performed. As expected, the performance of all frameworks was satisfactory, with global metrics indicating a good performance across all different variants, including the one that used PCA. Given the computational complexity that is associated with the training of an AutoEncoder, its use should be only tailored towards locations with high nonlinear behavior, since the PCA surrogate model is performing relatively well for most of the domain of interest. From an AutoEncoder perspective, the performance of all the examined alternatives was really close, but in a more detailed assessment Softplus was found to be able to follow the overall surge time-series more faithfully than others (the hyperbolic tangent was found to be globally the best), with maybe larger errors for some small surge manifestations.

7.2 Future Work

This research focused explicitly on a single nonlinear dimensionality reduction technique, with most of the traditional architecture features kept as is (fully connected, dense Deep Autoencoders). Although the results were satisfactory, the computational burden of such architectures is significant, and prevents the handling of large number of locations (tens of thousands). There is still more to be explored in this domain, that are related to different AutoEncoder architectures like Convolutional, Variational and Sparse Autoencoders that are able to handle time-series data with more innovative architectures that might be more versatile and computationally less expensive.

Combining dimensionality reduction with surrogate modelling does introduces some level of error which impacts the overall predictions. Quantifying further this error is critical to assessing the percentage of contribution for each component in the overall predictive behavior. By Exploring different AutoEncoder typologies and applying the developed framework to different databases, we will be able to improve the understanding of how this predictive error is distributed among the different models that are combined for the final time-series surge predictions. This refinement process could result in more precise and actionable storm surge predictions.

BIBLIOGRAPHY

1. Fraser, S., et al., *The making of a riskier future: How our decisions are shaping future disaster risk*. 2016: Global Facility for Disaster Reduction and Recovery.
2. Gu, D., *Exposure and vulnerability tonatural disasters for world'scities.* 2019.
3. Nations, U., *United Nations Department of Economic and social affairs.* United Nations, New York, 2018.
4. Cutter, S.L., B.J. Boruff, and W.L. Shirley, *Social vulnerability to environmental hazards*, in *Hazards vulnerability and environmental justice*. 2012, Routledge. p. 143-160.
5. Nicholls, R.J. and A. Cazenave, *Sea-level rise and its impact on coastal zones.* science, 2010. **328**(5985): p. 1517-1520.
6. Nicholls, R.J., et al., *Climate change and coastal vulnerability assessment: scenarios for integrated assessment.* Sustainability Science, 2008. **3**: p. 89-102.
7. Cooper, J.A.G. and O.H. Pilkey, *Sea-level rise and shoreline retreat: time to abandon the Bruun Rule.* Global and planetary change, 2004. **43**(3-4): p. 157-171.
8. Arkema, K.K., et al., *Coastal habitats shield people and property from sea-level rise and storms.* Nature climate change, 2013. **3**(10): p. 913-918.
9. Rose, A., *Defining and measuring economic resilience to disasters.* Disaster Prevention and Management: An International Journal, 2004. **13**(4): p. 307-314.
10. Cutter, S.L., C.G. Burton, and C.T. Emrich, *Disaster resilience indicators for benchmarking baseline conditions.* Journal of homeland security and emergency management, 2010. **7**(1).
11. Godschalk, D.R., *Urban hazard mitigation: Creating resilient cities.* Natural hazards review, 2003. **4**(3): p. 136-143.
12. Rappaport, E.N., et al., *Advances and challenges at the National Hurricane Center.* Weather and Forecasting, 2009. **24**(2): p. 395-419.
13. Harrison, S., *Hurricane Evacuation Research: A Systematic Review.* 2012.
14. DeBastiani, S.D., et al., *Preparedness perceptions, sociodemographic characteristics, and level of household preparedness for public health emergencies: behavioral risk factor surveillance system, 2006-2010.* Health security, 2015. **13**(5): p. 317-326.
15. Greer, A., et al., *Navigating dual hazards: Managing hurricane evacuation and sheltering operations Amidst COVID-19.* International Journal of Disaster Risk Reduction, 2024. **112**: p. 104773.
16. Gladwin, H., et al., *Social science research needs for the hurricane forecast and warning system.* Natural Hazards Review, 2007. **8**(3): p. 87-95.
17. Westerink, J.J., et al., *A basin-to channel-scale unstructured grid hurricane storm surge model applied to southern Louisiana.* Monthly weather review, 2008. **136**(3): p. 833-864.
18. Taflanidis, A.A., et al., *Development of real-time tools for hurricane risk assessment*, in *Vulnerability, Uncertainty, and Risk: Quantification, Mitigation, and Management*. 2014. p. 1341-1350.

19. Ferrier, N. and C.E. Haque, *Hazards risk assessment methodology for emergency managers: A standardized framework for application.* Natural hazards, 2003. **28**: p. 271-290.
20. Cauzzi, C., et al., *Earthquake early warning and operational earthquake forecasting as real-time hazard information to mitigate seismic risk at nuclear facilities.* Bulletin of Earthquake Engineering, 2016. **14**: p. 2495-2512.
21. Calkin, D.E., et al., *A real-time risk assessment tool supporting wildland fire decisionmaking.* Journal of Forestry, 2011. **109**(5): p. 274-280.
22. Cremen, G., C. Galasso, and J. McCloskey, *Modelling and quantifying tomorrow's risks from natural hazards.* Science of The Total Environment, 2022. **817**: p. 152552.
23. Westermann, P. and R. Evins, *Surrogate modelling for sustainable building design–A review.* Energy and Buildings, 2019. **198**: p. 170-186.
24. Alizadeh, R., J.K. Allen, and F. Mistree, *Managing computational complexity using surrogate models: a critical review.* Research in Engineering Design, 2020. **31**(3): p. 275-298.
25. Sacks, J., et al., *Statistical Science.* Design and Analysis of Computer Experiments, 1989. **4**(4): p. 409-423.
26. Williams, C.K. and C.E. Rasmussen, *Gaussian processes for machine learning*. Vol. 2. 2006: MIT press Cambridge, MA.
27. Drucker, H., et al., *Support vector regression machines.* Advances in neural information processing systems, 1996. **9**.
28. Dyn, N., D. Levin, and S. Rippa, *Numerical procedures for surface fitting of scattered data by radial functions.* SIAM Journal on Scientific and Statistical Computing, 1986. **7**(2): p. 639-659.
29. Lophaven, S.N., H.B. Nielsen, and J. Søndergaard, *DACE: a Matlab kriging toolbox*. Vol. 2. 2002: Citeseer.
30. Martin, J.D. and T.W. Simpson, *Use of kriging models to approximate deterministic computer models.* AIAA journal, 2005. **43**(4): p. 853-863.
31. Jia, G. and A.A. Taflanidis, *Kriging metamodeling for approximation of high-dimensional wave and surge responses in real-time storm/hurricane risk assessment.* Computer Methods in Applied Mechanics and Engineering, 2013. **261-262**: p. 24-38.
32. Bachoc, F., *Cross validation and maximum likelihood estimations of hyper-parameters of Gaussian processes with model misspecification.* Computational Statistics & Data Analysis, 2013. **66**: p. 55-69.
33. Mehmani, A., et al., *Concurrent surrogate model selection (COSMOS): optimizing model type, kernel function, and hyper-parameters.* Structural and Multidisciplinary Optimization, 2018. **57**: p. 1093-1114.
34. Zijlema, M., *Computation of wind-wave spectra in coastal waters with SWAN on unstructured grids.* Coastal Engineering, 2010. **57**(3): p. 267-277.
35. Westerink, J.J., R.A. Luettich, and N.W. Scheffner, *ADCIRC: An advanced three-dimensional circulation model for shelves, coasts, and estuaries, Report 3: Development of a tidal constituent database for the western north Atlantic and Gulf of Mexico.* Dredging Research Program Technical Report DRP-92, 1993. **6**.

36. Tanaka, S., et al., *Scalability of an unstructured grid continuous Galerkin based hurricane storm surge model.* Journal of Scientific Computing, 2011. **46**: p. 329-358.
37. Taylor, A.A. and B. Glahn. *Probabilistic guidance for hurricane storm surge*. in *19th Conference on probability and statistics*. 2008.
38. Irish, J.L., D.T. Resio, and M.A. Cialone, *A surge response function approach to coastal hazard assessment. Part 2: Quantification of spatial attributes of response functions.* Natural hazards, 2009. **51**: p. 183-205.
39. Das, H.S., et al., *An efficient storm surge forecasting tool for coastal Mississippi.* Coastal Engineering Proceedings, 2011(32): p. 21-21.
40. Taflanidis, A.A., et al., *Implementation/optimization of moving least squares response surfaces for approximation of hurricane/storm surge and wave responses.* Natural hazards, 2013. **66**: p. 955-983.
41. Kim, S.-W., et al., *A time-dependent surrogate model for storm surge prediction based on an artificial neural network using high-fidelity synthetic hurricane modeling.* Natural Hazards, 2015. **76**: p. 565-585.
42. Smith, J.M., et al., *SWIMS Hawaii hurricane wave, surge, and runup inundation fast forecasting tool*, in *Solutions to Coastal Disasters 2011*. 2011. p. 89-98.
43. Kijewski-Correa, T., et al., *Geospatial environments for hurricane risk assessment: applications to situational awareness and resilience planning in New Jersey.* Frontiers in Built Environment, 2020. **6**: p. 549106.
44. Resio, D.T., T.G. Asher, and J.L. Irish, *The effects of natural structure on estimated tropical cyclone surge extremes.* Natural hazards, 2017. **88**: p. 1609-1637.
45. Resio, D.T., J. Irish, and M. Cialone, *A surge response function approach to coastal hazard assessment–part 1: basic concepts.* Natural hazards, 2009. **51**: p. 163-182.
46. Niedoroda, A., et al., *Analysis of the coastal Mississippi storm surge hazard.* Ocean Engineering, 2010. **37**(1): p. 82-90.
47. Kennedy, A.B., et al., *Tropical cyclone inundation potential on the Hawaiian Islands of Oahu and Kauai.* Ocean Modelling, 2012. **52**: p. 54-68.
48. Cialone, M.A., et al., *North Atlantic Coast Comprehensive Study (NACCS) coastal storm model simulations: Waves and water levels.* 2015.
49. Guise, M.A., et al., *North Atlantic Coast comprehensive study: Resilient adaptation to increasing risk*. 2015, Technical report. National Planning Center for Coastal Storm Risk Management ….
50. Jia, G., et al., *Surrogate modeling for peak or time-dependent storm surge prediction over an extended coastal region using an existing database of synthetic storms.* Natural Hazards, 2016. **81**: p. 909-938.
51. Rohmer, J., et al., *Dynamic parameter sensitivity in numerical modelling of cyclone-induced waves: a multi-look approach using advanced meta-modelling techniques.* Natural Hazards, 2016. **84**: p. 1765-1792.
52. Kleijnen, J.P., *Kriging metamodeling in simulation: A review.* European journal of operational research, 2009. **192**(3): p. 707-716.
53. Bass, B. and P. Bedient, *Surrogate modeling of joint flood risk across coastal watersheds.* Journal of Hydrology, 2018. **558**: p. 159-173.

54. Parker, K., et al., *Emulation as an approach for rapid estuarine modeling.* Coastal Engineering, 2019. **150**: p. 79-93.
55. Adeli, E., et al., *An advanced spatio-temporal convolutional recurrent neural network for storm surge predictions.* Neural Computing and Applications, 2023. **35**(26): p. 18971-18987.
56. Kyprioti, A.P., et al., *Storm hazard analysis over extended geospatial grids utilizing surrogate models.* Coastal Engineering, 2021. **168**: p. 103855.
57. Chen, X., et al. *An Attention-Based Temporal and Spatial Convolution Recursive Neural Network for Surrogate Modeling of the Production Curve Prediction*. in *International Field Exploration and Development Conference*. 2023. Springer.
58. Zhan, W. and A. Datta, *Neural networks for geospatial data.* Journal of the American Statistical Association, 2024(just-accepted): p. 1-21.
59. Kyprioti, A.P., et al., *Spatio-temporal storm surge emulation using Gaussian Process techniques.* Coastal Engineering, 2022: p. 104231.
60. Nadal-Caraballo, N.C., Melby Jeffrey A, and G.V. M., *Coastal Storm Hazards from Virginia to Maine*. 2015, U.S. Army Engineer Research and Development Center.
61. Jolliffe, I.T., *Principal component analysis for special types of data*. 2002: Springer.
62. Naeini, S.S. and R. Snaiki, *A novel hybrid machine learning model for rapid assessment of wave and storm surge responses over an extended coastal region.* Coastal Engineering, 2024. **190**: p. 104503.
63. Naeini, S.S., R. Snaiki, and T. Wu, *Advancing Spatio-temporal Storm Surge Prediction with Hierarchical Deep Neural Networks.* arXiv preprint arXiv:2410.12823, 2024.
64. Kyprioti, A.P., *Natural Hazard Risk Assessment: Promoting Computational Efficiency Through Advances in Surrogate Modeling*. 2022: University of Notre Dame.
65. Jelesnianski, C.P., *SLOSH: Sea, lake, and overland surges from hurricanes*. Vol. 48. 1992: US Department of Commerce, National Oceanic and Atmospheric Administration ….
66. Vickery, P.J. and D. Wadhera, *Statistical models of Holland pressure profile parameter and radius to maximum winds of hurricanes from flight-level pressure and H* Wind data.* Journal of Applied Meteorology and climatology, 2008. **47**(10): p. 2497-2517.
67. Vickery, P.J., et al., *Coastal storm surge analysis: storm forcing; Report 3: Intermediate submission no. 1.3.* 2013.
68. Hanson, J.L., et al., *Coastal storm surge analysis: storm surge results; Report 5: Intermediate submission no. 3.* 2013.
69. Schwerdt, R.W., F.P. Ho, and R.R. Watkins, *Meteorological criteria for standard project hurricane and probable maximum hurricane windfields, Gulf and East Coasts of the United States.* 1979.
70. Scheffner, N.W., L.E. Borgman, and D.J. Mark, *Empirical simulation technique based storm surge frequency analyses.* Journal of waterway, port, coastal, and ocean engineering, 1996. **122**(2): p. 93-101.

71. Toro, G.R., et al., *Efficient joint-probability methods for hurricane surge frequency analysis.* Ocean Engineering, 2010. **37**(1): p. 125-134.
72. Ho, F.P. and V.A. Myers, *Joint probability method of tide frequency analysis applied to Apalachicola Bay and St. George Sound, Florida.* 1975.
73. Myers, V.A., *Storm tide frequencies on the South Carolina Coast*. Vol. 16. 1975: Office of Hydrology, National Weather Service.
74. Resio, D.T., et al., *White paper on estimating hurricane inundation probabilities.* 2007.
75. Zhang, J., et al., *Advances in surrogate modeling for storm surge prediction: storm selection and addressing characteristics related to climate change.* Natural Hazards, 2018. **94**: p. 1225-1253.
76. Igarashi, Y. and Y. Tajima, *Application of recurrent neural network for prediction of the time-varying storm surge.* Coastal Engineering Journal, 2021. **63**(1): p. 68-82.
77. Ramos-Valle, A.N., et al., *Implementation of an artificial neural network for storm surge forecasting.* Journal of Geophysical Research: Atmospheres, 2021. **126**(13): p. e2020JD033266.
78. Gramacy, R.B. and H.K. Lee, *Cases for the nugget in modeling computer experiments.* Statistics and Computing, 2012. **22**: p. 713-722.
79. Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P., *Design and analysis of computer experiments.* Statistical Science, 1989. **4**(4): p. 409-435.
80. Sundararajan, S. and S.S. Keerthi, *Predictive approaches for choosing hyperparameters in Gaussian processes.* Neural computation, 2001. **13**(5): p. 1103-1118.
81. Schobi, R., B. Sudret, and J. Wiart, *Polynomial-chaos-based Kriging.* International Journal for Uncertainty Quantification, 2015. **5**(2).
82. Dubrule, O., *Cross validation of kriging in a unique neighborhood.* Journal of the International Association for Mathematical Geology, 1983. **15**(6): p. 687-699.
83. Nadal-Caraballo, N.C., et al., *Coastal hazards system: a probabilistic coastal hazard analysis framework.* Journal of Coastal Research, 2020. **95**(SI): p. 1211-1216.
84. Melby, J.A., et al. *Coastal hazards system*. in *Coastal Structures and Solutions to Coastal Disasters Joint Conference 2015*. 2015. American Society of Civil Engineers Reston, VA.
85. Smith, J.M., A.R. Sherlock, and D.T. Resio, *STWAVE: Steady-state spectral wave model user's manual for STWAVE, Version 3.0*. 2001: Citeseer.
86. Nadal-Caraballo, N., et al., *North Atlantic Coast Comprehensive Study (NACCS): Coastal Storm Hazards from Virginia to Maine.* US Army Engineer Research and Development Center (ERDC), Technical Report. ERDC-CHL-TR-15-5, 2015.
87. Alsabti, K., S. Ranka, and V. Singh, *An efficient k-means clustering algorithm.* 1997.
88. Jia, W., et al., *Feature dimensionality reduction: a review.* Complex & Intelligent Systems, 2022. **8**(3): p. 2663-2693.
89. Ahmad, N. and A.B. Nassif. *Dimensionality reduction: Challenges and solutions*. in *ITM Web of Conferences*. 2022. EDP Sciences.

90. Fernández-Martínez, J.L. and Z. Fernández-Muñiz, *The curse of dimensionality in inverse problems.* Journal of Computational and Applied Mathematics, 2020. **369**: p. 112571.
91. Van Der Maaten, L., E.O. Postma, and H.J. Van Den Herik, *Dimensionality reduction: A comparative review.* Journal of machine learning research, 2009. **10**(66-71): p. 13.
92. Sorzano, C.O.S., J. Vargas, and A.P. Montano, *A survey of dimensionality reduction techniques.* arXiv preprint arXiv:1403.2877, 2014.
93. Ayesha, S., M.K. Hanif, and R. Talib, *Overview and comparative study of dimensionality reduction techniques for high dimensional data.* Information Fusion, 2020. **59**: p. 44-58.
94. Sumithra, V. and S. Surendran, *A review of various linear and non linear dimensionality reduction techniques.* Int J Comput Sci Inf Technol, 2015. **6**(3): p. 2354-60.
95. Lee, J.A. and M. Verleysen, *Nonlinear dimensionality reduction*. Vol. 1. 2007: Springer.
96. Balamurali, M., *T-Distributed stochastic neighbor embedding*, in *Encyclopedia of mathematical geosciences*. 2023, Springer. p. 1527-1535.
97. Yousaf, M., T.U. Rehman, and L. Jing, *An extended isomap approach for nonlinear dimension reduction.* SN Computer Science, 2020. **1**(3): p. 160.
98. Saul, L.K. and S.T. Roweis, *An introduction to locally linear embedding.* unpublished. Available at: http://www. cs. toronto. edu/~ roweis/lle/publications. html, 2000.
99. Hout, M.C., M.H. Papesh, and S.D. Goldinger, *Multidimensional scaling.* Wiley Interdisciplinary Reviews: Cognitive Science, 2013. **4**(1): p. 93-103.
100. McInnes, L., J. Healy, and J. Melville, *Umap: Uniform manifold approximation and projection for dimension reduction.* arXiv preprint arXiv:1802.03426, 2018.
101. Wang, W., et al. *Generalized autoencoder: A neural network framework for dimensionality reduction*. in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2014.
102. Nanga, S., et al., *Review of dimension reduction methods.* Journal of Data Analysis and Information Processing, 2021. **9**(3): p. 189-231.
103. Yin, H., *Nonlinear dimensionality reduction and data visualization: a review.* International Journal of Automation and Computing, 2007. **4**: p. 294-303.
104. Kruskal, J.B., *Nonmetric multidimensional scaling: a numerical method.* Psychometrika, 1964. **29**(2): p. 115-129.
105. Hinton, G. and R. Salakhutdinov, *Reducing the dimensionality of data with neural networks. science, 313 (5786), 504-507.* 2006.
106. Silva, V. and J. Tenenbaum, *Global versus local methods in nonlinear dimensionality reduction.* Advances in neural information processing systems, 2002. **15**.
107. Ghodsi, A., *Dimensionality reduction a short tutorial.* Department of Statistics and Actuarial Science, Univ. of Waterloo, Ontario, Canada, 2006. **37**(38): p. 2006.
108. Ashraf, M., et al., *A survey on dimensionality reduction techniques for time-series data.* IEEE Access, 2023. **11**: p. 42909-42923.

109. Van der Maaten, L. and G. Hinton, *Visualizing data using t-SNE.* Journal of machine learning research, 2008. **9**(11).
110. Cai, T.T. and R. Ma, *Theoretical foundations of t-sne for visualizing high-dimensional clustered data.* Journal of Machine Learning Research, 2022. **23**(301): p. 1-54.
111. Wattenberg, M., F. Viégas, and I. Johnson, *How to use t-SNE effectively.* Distill, 2016. **1**(10): p. e2.
112. Wong, K.Y. and F.-l. Chung. *Visualizing time series data with temporal matching based t-SNE*. in *2019 International Joint Conference on Neural Networks (IJCNN)*. 2019. IEEE.
113. Kobak, D. and P. Berens, *The art of using t-SNE for single-cell transcriptomics.* Nature communications, 2019. **10**(1): p. 5416.
114. Ali, M., et al., *TimeCluster: dimension reduction applied to temporal data for visual analytics.* The Visual Computer, 2019. **35**(6): p. 1013-1026.
115. Accamma, I., H. Suma, and M. Dakshayini, *Decoding Multiple Subject fMRI Data using Manifold based Representation of Cognitive State Neural Signatures.* International Journal of Computer Applications, 2015. **115**(15).
116. Schölkopf, B., A. Smola, and K.-R. Müller. *Kernel principal component analysis*. in *International conference on artificial neural networks*. 1997. Springer.
117. Weinberger, K.Q., F. Sha, and L.K. Saul. *Learning a kernel matrix for nonlinear dimensionality reduction*. in *Proceedings of the twenty-first international conference on Machine learning*. 2004.
118. Raschka, S., et al., *Kernel tricks and nonlinear dimensionality reduction via rbf kernel pca.* Blog, September, 2014.
119. Lawrence, N. and A. Hyvärinen, *Probabilistic non-linear principal component analysis with Gaussian process latent variable models.* Journal of machine learning research, 2005. **6**(11).
120. Das, R., A. Golatkar, and S.P. Awate. *Sparse kernel PCA for outlier detection*. in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. 2018. IEEE.
121. Mika, S., et al., *Kernel PCA and de-noising in feature spaces.* Advances in neural information processing systems, 1998. **11**.
122. Lawrence, N., *Gaussian process latent variable models for visualisation of high dimensional data.* Advances in neural information processing systems, 2003. **16**.
123. Wang, J.M., D.J. Fleet, and A. Hertzmann, *Gaussian process dynamical models for human motion.* IEEE transactions on pattern analysis and machine intelligence, 2007. **30**(2): p. 283-298.
124. Wang, Y., H. Yao, and S. Zhao, *Auto-encoder based dimensionality reduction.* Neurocomputing, 2016. **184**: p. 232-242.
125. Pinaya, W.H.L., et al., *Autoencoders*, in *Machine learning*. 2020, Elsevier. p. 193-208.
126. Bank, D., N. Koenigstein, and R. Giryes, *Autoencoders.* Machine learning for data science handbook: data mining and knowledge discovery handbook, 2023: p. 353-374.
127. Michelucci, U., *An introduction to autoencoders.* arXiv preprint arXiv:2201.03898, 2022.

128. Baldi, P. *Autoencoders, unsupervised learning, and deep architectures*. in *Proceedings of ICML workshop on unsupervised and transfer learning*. 2012. JMLR Workshop and Conference Proceedings.
129. Ryu, S., et al., *Convolutional autoencoder based feature extraction and clustering for customer load analysis.* IEEE Transactions on Power Systems, 2019. **35**(2): p. 1048-1060.
130. Abolhasanzadeh, B. *Nonlinear dimensionality reduction for intrusion detection using auto-encoder bottleneck features*. in *2015 7th Conference on Information and Knowledge Technology (IKT)*. 2015. IEEE.
131. Bengio, Y., et al., *Greedy layer-wise training of deep networks.* Advances in neural information processing systems, 2006. **19**.
132. Vincent, P., et al. *Extracting and composing robust features with denoising autoencoders*. in *Proceedings of the 25th international conference on Machine learning*. 2008.
133. Li, P., Y. Pei, and J. Li, *A comprehensive survey on design and application of autoencoder in deep learning.* Applied Soft Computing, 2023. **138**: p. 110176.
134. Kusner, M.J., B. Paige, and J.M. Hernández-Lobato. *Grammar variational autoencoder*. in *International conference on machine learning*. 2017. PMLR.
135. Liou, C.-Y., et al., *Autoencoder for words.* Neurocomputing, 2014. **139**: p. 84-96.
136. Zhai, J., et al. *Autoencoder and its various variants*. in *2018 IEEE international conference on systems, man, and cybernetics (SMC)*. 2018. IEEE.
137. Ronald, J., et al., *Learning internal representations by error propagation.* Parallel distributed processing: Explorations in the microstructure of cognition, 1986. **1**.
138. Thakkar, S., et al. *Autoencoder and evolutionary algorithm for level generation in lode runner*. in *2019 IEEE Conference on Games (CoG)*. 2019. IEEE.
139. Miranda-González, A.A., et al., *Denoising Vanilla Autoencoder for RGB and GS Images with Gaussian Noise.* Entropy, 2023. **25**(10): p. 1467.
140. Masci, J., et al. *Stacked convolutional auto-encoders for hierarchical feature extraction*. in *Artificial Neural Networks and Machine Learning–ICANN 2011: 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I 21*. 2011. Springer.
141. Baur, C., et al. *Deep autoencoding models for unsupervised anomaly segmentation in brain MR images*. in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries: 4th International Workshop, BrainLes 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 16, 2018, Revised Selected Papers, Part I 4*. 2019. Springer.
142. Ng, A., *Sparse autoencoder.* CS294A Lecture notes, 2011. **72**(2011): p. 1-19.
143. Makhzani, A. and B. Frey, *K-sparse autoencoders.* arXiv preprint arXiv:1312.5663, 2013.
144. An, J. and S. Cho, *Variational autoencoder based anomaly detection using reconstruction probability.* Special lecture on IE, 2015. **2**(1): p. 1-18.
145. Higgins, I., et al., *beta-vae: Learning basic visual concepts with a constrained variational framework.* ICLR (Poster), 2017. **3**.
146. Chung, J., et al., *A recurrent latent variable model for sequential data.* Advances in neural information processing systems, 2015. **28**.

147. Pinheiro Cinelli, L., et al., *Variational autoencoder*, in *Variational Methods for Machine Learning with Applications to Deep Networks*. 2021, Springer. p. 111-149.
148. Cheng, Z., et al., *Improved autoencoder for unsupervised anomaly detection.* International Journal of Intelligent Systems, 2021. **36**(12): p. 7103-7125.
149. Farina, M., Y. Nakai, and D. Shih, *Searching for new physics with deep autoencoders.* Physical Review D, 2020. **101**(7): p. 075021.
150. Wang, J., H. He, and D.V. Prokhorov, *A folded neural network autoencoder for dimensionality reduction.* Procedia Computer Science, 2012. **13**: p. 120-127.
151. Hinton, G.E. and R.R. Salakhutdinov, *Reducing the dimensionality of data with neural networks.* science, 2006. **313**(5786): p. 504-507.
152. Mallik, W., R.K. Jaiman, and J. Jelovica, *Predicting transmission loss in underwater acoustics using convolutional recurrent autoencoder network.* The Journal of the Acoustical Society of America, 2022. **152**(3): p. 1627-1638.
153. Gonzalez, F.J. and M. Balajewicz, *Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems.* arXiv preprint arXiv:1808.01346, 2018.
154. Feichtenhofer, C., Y. Li, and K. He, *Masked autoencoders as spatiotemporal learners.* Advances in neural information processing systems, 2022. **35**: p. 35946-35958.
155. Jia, Y., C. Zhou, and M. Motani. *Spatio-temporal autoencoder for feature learning in patient data with missing observations*. in *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. 2017. IEEE.
156. *Generic representation of the architecture of an AutoEncoder (AE)* Available from: https://i2.wp.com/sefiks.com/wp-content/uploads/2018/03/convolutional-autoencoder.png?resize=1140%2C381&ssl=1.
157. Pawar, K. and V.Z. Attar, *Assessment of autoencoder architectures for data representation.* Deep learning: concepts and architectures, 2020: p. 101-132.
158. Kabil, S.H. and H. Bourlard, *From Undercomplete to Sparse Overcomplete Autoencoders to Improve LF-MMI Speech Recognition.* Interspeech 2022, 2022: p. 1061-1065.
159. *The connections between layers in an AE for a fully connected architecture*. Available from: https://www.frontiersin.org/files/Articles/546769/fnsys-14-00043-HTML/image_m/fnsys-14-00043-g002.jpg.
160. Sun, Y., et al., *Learning a good representation with unsymmetrical auto-encoder.* Neural computing and applications, 2016. **27**: p. 1361-1367.
161. Gilbert, M.S., M.L. de Campos, and M.E.M. Campista, *Asymmetric Autoencoders: An NN alternative for resource-constrained devices in IoT networks.* Ad Hoc Networks, 2024. **156**: p. 103412.
162. *The connections between layers in an AE for a convolutional architecture* Available from: https://www.frontiersin.org/files/Articles/420104/fgene-10-00214-HTML/image_m/fgene-10-00214-g006.jpg.
163. Berahmand, K., et al., *Autoencoders and their applications in machine learning: a survey.* Artificial Intelligence Review, 2024. **57**(2): p. 28.
164. Szandała, T., *Review and comparison of commonly used activation functions for deep neural networks.* Bio-inspired neurocomputing, 2021: p. 203-224.

165. Bengio, Y., A. Courville, and P. Vincent, *Representation learning: A review and new perspectives.* IEEE transactions on pattern analysis and machine intelligence, 2013. **35**(8): p. 1798-1828.
166. Keskar, N.S. and R. Socher, *Improving generalization performance by switching from adam to sgd.* arXiv preprint arXiv:1712.07628, 2017.
167. Diederik, P.K., *Adam: A method for stochastic optimization.* (No Title), 2014.
168. Ruder, S., *An overview of gradient descent optimization algorithms.* arXiv preprint arXiv:1609.04747, 2016.
169. Hoffer, E., I. Hubara, and D. Soudry, *Train longer, generalize better: closing the generalization gap in large batch training of neural networks.* Advances in neural information processing systems, 2017. **30**.
170. Smith, L.N. *Cyclical learning rates for training neural networks*. in *2017 IEEE winter conference on applications of computer vision (WACV)*. 2017. IEEE.
171. Jacobs, R.A., *Increased rates of convergence through learning rate adaptation.* Neural networks, 1988. **1**(4): p. 295-307.
172. Akiba, T., et al. *Optuna: A next-generation hyperparameter optimization framework*. in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019.
173. Agrawal, T. and T. Agrawal, *Optuna and autoML.* Hyperparameter Optimization in Machine Learning: Make Your Machine Learning and Deep Learning Models More Efficient, 2021: p. 109-129.
174. Feurer, M., et al., *Efficient and robust automated machine learning.* Advances in neural information processing systems, 2015. **28**.