Deep learning techniques for forecasting compound coastal flooding due to tropical cyclones

D.H.H. Oudenes

Deep learning techniques for forecasting compound coastal flooding due to tropical cyclones

Thesis report

by

D.H.H. Oudenes

to obtain the degree of Master of Science
at the Delft University of Technology
to be defended publicly on June 25, 2025 at 16:00

Thesis committee:

Chair:	Dr. ir. J.A.A. Antolínez
TU Delft supervisors:	Dr. ir. R. Gelderloos
	Dr. ir. A. Heinlein
Deltares supervisors:	ir. K. van Asselt
	Dr. ir. P. Athanasiou
	Dr. ir. K. Nederhoff
Place:	Faculty of Civil Engineering and Geosciences, Delft
Student number:	4864239

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

Faculty of Civil Engineering · Delft University of Technology

Preface

Before you lies the master thesis ”Deep learning techniques for forecasting compound coastal flooding due to tropical cyclones”. It has been written to obtain a master’s degree in Hydraulic Engineering at Delft University of Technology. I have been working on this thesis for the past ten months, together with a group of supervisors from Delft University of Technology and the company Deltares.

I noticed the upcoming urge to step outside my comfort zone during the master’s and I have attempted to combine my interest in machine learning with my study civil engineering in this thesis. This combination required skills that I did not yet have, therefore providing the challenge needed to keep my interest and enthusiasm throughout the project. I have gained a better understanding of machine learning and how it can be applied in the engineering field. Also, I have learned that setbacks are part of the process. Therefore, this thesis has taught me valuable lessons both professionally and personally.

I would like to thank my supervision team, consisting of 6 (!) people. In particular Koen, with whom I had weekly chats to keep me going on the subject. I would like to thank Deltares for providing me with the opportunity to carry out my thesis in such a knowledge-rich environment.

Finally, I would like to thank my friends and family, who have supported me from the beginning. I would also like to thank you, my reader: I hope you enjoy your reading.

Derby Oudenes
Delft, June 2025

Abstract

Accurate flood prediction is essential for effective risk management, but simulating with physics-based models is computationally intensive and time-consuming, limiting their use in operational use cases. To address this challenge, this study develops a deep learning surrogate model that replicates spatial flood depth predictions of the physics-based model SFINCS, with significantly reduced computational cost.

The surrogate model is trained with a synthetic dataset of 13,000 tracks generated by TCWiSE, that have been computed with SFINCS. Input for the surrogate model are key drivers of compound coastal flooding, including cumulative precipitation, maximum wind speed, and water depth at the river outlet, which are given to the model as scalar values. The results demonstrate that the model can reproduce flood depth patterns with a RMSE of 0.054 m on the original dataset and 0.069 m on a case study application. The CSI values of 0.829 and 0.776, respectively, are comparable to values reported in recent literature. Visual inspection of spatial predictions shows that the model performs well overall but shows localized error patterns, particularly in the western part of the domain, indicating that additional input features may be needed to better capture interactions in physical processes that occur in compound coastal flooding. From a computational perspective, the deep learning model runs approximately 100 times faster than the physics-based model SFINCS on CPU, with further gains on GPU. This computational efficiency allows for running more simulations within a given timeframe, enabling the exploration of an increased number of potential flood scenarios and providing decision-makers with more time to evaluate and respond.

Overall, the study demonstrates that deep learning surrogate models offer a promising alternative to physics-based models by providing accurate and faster predictions. Although the current set-up shows accurate results, there remains room for improvement through enhanced input representation, such as incorporating spatiotemporal information of the drivers or including additional hydrodynamic features. Further gains in performance and generalizability can also be achieved by refining the model architecture and training strategies.

Contents

Nomenclature

List of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
DL	Deep Learning
DNN	Deep Neural Network
EWS	Early Warning System
GPU	Graphics Processing Unit
MAE	Mean Absolute Error
ML	Machine Learning
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
NaN	Not a Nubmer
NHC	National Hurricane Center
NN	Neural Network
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
RQ	Research Question
SGD	Stochastic Gradient Descent
SWE	Shallow Water Equations
TC	Tropical Cyclone

List of Figures

List of Tables

1

Introduction

1.1. Background

Coastal flooding is one of the most dangerous natural hazards, posing significant risks to human life, infrastructure, and ecosystems every year (Jonkman et al., 2008), especially in areas prone to Tropical Cyclones (TCs) (P. Hu et al., 2018). TCs contain a variety of drivers that contribute to flooding, with the main drivers being wind, which causes storm surge and consequently coastal flooding, and precipitation around the eye of the storm, which causes inland and riverine flooding (Edmonds et al., 2020).

Compound coastal flooding occurs when multiple flood drivers such as storm surge, heavy rainfall, river discharge, and high tides coincide (Wahl et al., 2015). Figure 1.1 illustrates the individual processes of coastal flooding (A) and riverine flooding (B), which can occur simultaneously. When these drivers combine, their effects can amplify, resulting in compound coastal flooding with greater severity than any individual flooding process. Some of these drivers are correlated to each other, as can be seen in a TC. TCs can cause extreme winds and precipitation rates, making them an interesting subject of study for coastal areas.

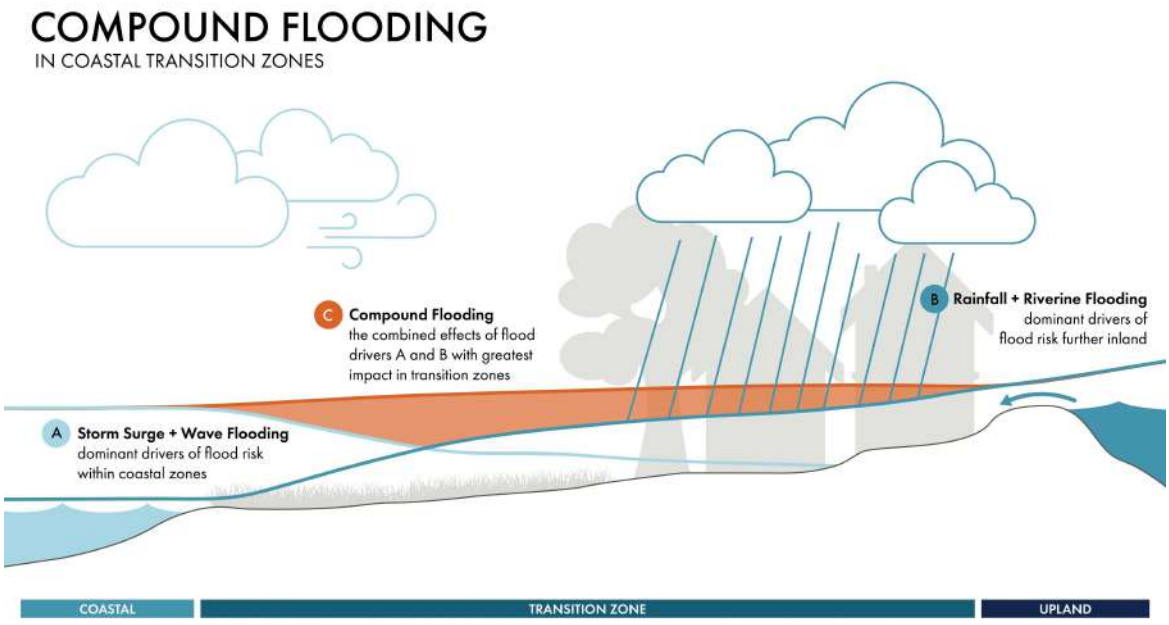

Figure 1.1: Illustration of multiple triggers for compound coastal flooding. Source: (The Water Institute, 2025).

Public advisories by the National Hurricane Center (NHC) are issued for disturbances, TCs, or systems not yet classified as such, which pose a threat to land areas within 72 hours. The NHC updates its TC

advisories at least every 6 hours. These updates include forecasts for TCs, including track and intensity forecasts for the next five days (National Hurricane Center, 2025).

Simulating a tropical cyclone brings difficulties due to uncertainties inherent in TCs (Nederhoff et al., 2023). Not only is the location of landfall uncertain, but also propagation speed and TC intensity. Figure 1.2 illustrates an example of a prediction by the National Oceanic and Atmospheric Administration (NOAA), for the future trajectory of the TC. Uncertainty in the path of the TC, illustrated by the white cone for the 1-3 day forecast and the larger semi-transparent cone with a white border for the 3-5 day forecast, can be in the order of hundreds or even thousands of kilometers.

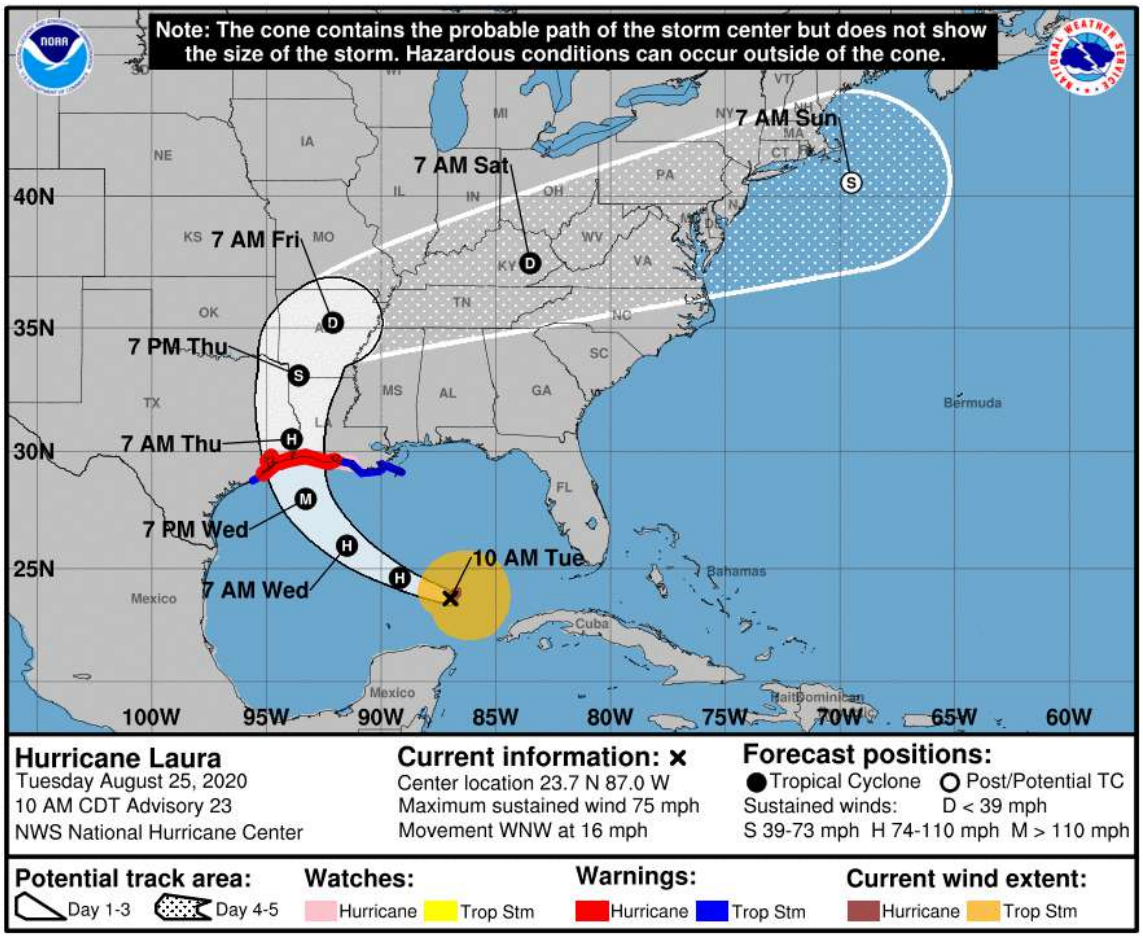

Figure 1.2: Prediction of future path of the Tropical Cyclone. Source: (National Hurricane Center, 2020).

Forecasting flooding due to TCs in time is crucial for saving human lives and reducing economic loss (Roy et al., 2012). Rapid flood predictions serve as an input for Early Warning Systems (EWSs), facilitating the evacuation of populations in areas susceptible to floods. Accurate and quick predictions facilitate more response time, thereby improving the preparations and minimizing the possible devastating impacts of such natural disasters.

The frequency and intensity of TCs are expected to increase in the coming years due to the effects of climate change (Emanuel, 2013). Moreover, the population in low-lying coastal areas will grow in the future (Curtis et al., 2011). Increasing strength and frequency of TCs in combination with a growing population in coastal areas leads to a higher risk of mortalities during a flood event.

1.2. Problem statement

To capture the uncertainty in forecasts, a method called ensemble forecasting is applied. Ensemble forecasting is a technique that uses a set of multiple scenarios, rather than a single scenario, to represent

the range of possible future outcomes. This approach gives insight into the uncertainty in forecasts and provides a better understanding of potential future events, especially in flood prediction.

An ensemble of synthetic tracks is simulated to capture the range of possible tracks of the TC. The ensemble is used by a physics-based model to simulate flooding, capturing the range of possible flood scenarios. An example of a physics-based model is SFINCS (Leijnse et al., 2021). These models, founded in physical equations, often take several minutes per simulation, making it a time-consuming process when the number of simulations increases significantly.

To get an understanding of the possible impacts of an incoming TC, first the TC advisory from the NHC needs to be collected, then a set of synthetic tracks needs to be created, which serve as an input for the physics-based model that needs to run every track individually. In an operational setting, the number of simulations that can be run with the physics-based model is limited to the time until the new TC advisory arrives.

Physics-based models are limited by high computational costs, which restrict their operational applicability. Addressing this challenge can involve either increasing computational resources or developing surrogate models that offer similar predictive capabilities at significantly lower computational costs. Given the availability of large datasets from tools developed by Deltares, deep learning offers a promising alternative. By learning complex patterns in the data, deep learning models can serve as efficient surrogates for physics-based models, greatly reducing prediction time while maintaining a high level of accuracy.

1.3. Literature review

The review of the literature is structured in two parts, with the aim of providing a comprehensive understanding of both conventional and emerging modeling approaches relevant to compound coastal flooding. The first part examines how numerical modeling approaches have been applied to forecast compound coastal flooding. The second part explores the use of deep learning techniques in related fields and identifies current gaps and challenges in applying these methods to the present research problem.

1.3.1. Numerical Modeling

Numerical flood modeling is widely used in flood risk assessment and real-time forecasting. These physics-based hydrodynamic models, such as SFINCS and Delft3D-FM, solve the Shallow Water Equations (SWE) to simulate compound coastal flooding. They provide spatially distributed outputs of water levels and inundation extent with high accuracy. The accuracy of these models depends on the level of detail of the input data, including topography, bathymetry, land use, and meteorological forcing.

A key challenge is the computational cost associated with these models. High-resolution simulations, especially over large coastal regions or for long-duration storm scenarios, can take hours to run, limiting their use in real-time contexts. Real-time forecasting must balance accuracy with computational speed. This remains a significant challenge given the computational demands of physics-based models (Suh et al., 2015). This is especially problematic for ensemble forecasting, where many scenarios must be evaluated quickly to capture uncertainty.

SFINCS (Super-Fast INundation of CoastS) is a two-dimensional (2D) reduced-physics model, which aims to simulate compound coastal flooding with limited computational cost and good accuracy. In contrast to full-physics models like Delft3D-FLOW (Lesser et al., 2004) or ADCIRC (Luettich et al., 1992), which solve the full SWE or even the Navier-Stokes equations, SFINCS uses simplified equations to reduce computational costs. By neglecting the Coriolis force, viscosity terms, and atmospheric pressure gradients, SFINCS simplifies the SWE to form the Simplified Shallow Water Equations (SSWE). This reduces numerical complexity while retaining the physical processes necessary for modeling coastal flooding (Leijnse et al., 2021).

The core equations that are solved in SFINCS are the continuity and momentum equation. In the model, it is optional to include the wind shear and advection terms in the momentum equation.

Continuity equation:

$$\zeta_{m,n}^{t+\Delta t} = \zeta_{m,n}^{t} + \left(\frac{q_{x,m-1,n}^{t+\Delta t} - q_{x,m,n}^{t+\Delta t}}{\Delta x} + \frac{q_{y,m,n-1}^{t+\Delta t} - q_{y,m,n}^{t+\Delta t}}{\Delta y} + S_{m,n} \right) \Delta t \tag{1.1}$$

Here, ζ is the water level, q_x and q_y are the unit discharges in x and y directions, and $S_{m,n}$ represents the source term (e.g. precipitation or infiltration).

Momentum equation (with advection and wind shear):

$$q_x^{t+\Delta t} = q_x^t - \left(g h_x^t \frac{\Delta \zeta}{\Delta x} + \text{adv}_x - \frac{\tau_{w,x}}{\rho_w} \right) \Delta t \left(1 + \frac{g \Delta t n^2 q_x^t}{(h_x^t)^{7/3}} \right)^{-1} \tag{1.2}$$

Where h_x^t is the average water depth at the cell interface, n is the Manning roughness coefficient, $\tau_{w,x}$ is the wind shear stress, and ρ_w is the water density. The term adv_x represents the advection term, which can be included or omitted depending on the model configuration.

Advection term (in x-direction, conservative form):

$$\text{adv}_x = \frac{\left(q_{x,m,n}^t\right)^2 / h_{x,m,n}^t - \left(q_{x,m-1,n}^t\right)^2 / h_{x,m-1,n}^t}{\Delta x} \tag{1.3}$$

This upwind scheme approximates the advection due to momentum gradients in the x-direction. A similar term is used in the y-direction.

Despite these efforts, computational demands of numerical models remain high, with SFINCS simulations typically requiring one or more minutes per run. Faster real-time forecasting methods are needed to facilitate ensemble forecasting (Bakker et al., 2022).

1.3.2. Deep Learning Techniques

Deep learning methods can aid in rapid forecasting of floods induced by TCs. Several DL architectures have been employed in flood hazard mapping, including Multi-Layer Perceptrons (MLPs), Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs) (Bentivoglio et al., 2022). Among these, CNNs have demonstrated effectiveness due to their ability to capture spatial features and patterns from input data, particularly in predicting urban pluvial flood water depth (Löwe et al., 2021). CNNs can extend their application to prediction of flood water depth induced by tropical cyclones, potentially improving forecast accuracy in coastal regions.

Deep learning methods have the potential to aid in rapidly forecasting TC-induced compound coastal floods, providing an advantage over physics-based models due to their speed and computational efficiency (Löwe et al., 2021). By leveraging complex patterns in data, deep learning models can speed up the prediction process, particularly attractive for real-time applications such as early warning systems.

However, a common challenge faced by deep learning models is the issue of generalization across different case studies (Bentivoglio et al., 2022). While these models may perform well on training datasets, their ability to generalize to unseen data can be limited. Therefore, careful consideration must be given to the variability and variety within the training data to ensure robust model performance across various scenarios.

Furthermore, data preparation is crucial in deep learning applications for flood prediction (Huang et al., 2021). Comprehensive datasets are essential for training accurate and reliable models. Data collection and preprocessing steps, including cleaning, and normalization are necessary to ensure that input data effectively captures the relevant information needed for accurate predictions.

On a local scale, DL-based flood hazard maps have been used to capture the extent of compound coastal flooding (Bentivoglio et al., 2022). However, scaling these approaches to regional domains introduces new challenges. Recent studies demonstrate that techniques like Reduced Order Modeling (ROM) in combination with Long Short-Term Memory (LSTM) networks show potential, but require further exploration and validation in the context of TC-induced compound flooding. (R. Hu et al., 2019).

1.4. Objectives

This study aims to develop a data-driven surrogate model that can replicate the predictions of physics-based flood models, while reducing computational demands. This approach seeks to combine preservation of the spatial resolution and accuracy of numerical models with significantly faster run times, enabling potential use in real-time applications such as early warning systems for TC-induced flooding. The research objectives are:

- Assess Model Performance: Assess the accuracy, computational efficiency, and robustness of DL models compared to physics-based models. These models are trained using flood map data, computed with SFINCS, of more than 10.000 synthetic events.
- Compare and Contrast Deep Learning Architectures: Compare DL models with varying architectures to evaluate the contrasts between models.
- Address Prediction Uncertainties: Investigate strategies to account for uncertainties in DL predictions to ensure reliability and robustness.

By achieving these objectives, the research seeks to demonstrate that DL techniques can provide a viable solution to the time-consuming simulation times of current models. In this way, early warning systems have improved efficiency and the impact of TC-induced floods on communities can be reduced.

1.5. General framework

The main focus of the research is on the development and understanding of a DL model that has a lower simulation time and that will give results with approximately the same accuracy as the physics-based model. In this research, the focus will be on comparing and contrasting different DL architectures and applying them to the available data.

The historical dataset on TCs is limited to a small number of TCs. To work around this issue, a synthetical dataset is created with the TCWiSE tool (K. Nederhoff et al., 2021), containing about 13.000 synthetic events. The creation of the flood maps due to the synthetic events, computed with SFINCS, is not part of the scope of this research. Therefore, the validity of these tracks is presumed and will not be evaluated.

In developing a deep learning model, various challenges emerge. The effectiveness of DL models depends on many aspects, including the dataset, the input layers, the output layers, and the architecture configuration. The focus of this research is to provide insight into the steps of a deep learning model, combined with finding the optimal architecture configuration for a DL model to predict compound coastal flooding.

The efficiency and accuracy of the deep learning model will be compared to a physics-based model. The physics-based model used in this study, SFINCS, is assumed to be valid, as its calibration or modification falls outside the scope of this research.

1.6. Research questions

Based on the research objectives as defined above, research questions are formulated. By providing an answer to these research questions, the general objectives of this research will be fulfilled.

The main question for this research is the following:

- How do deep learning models compare to physics-based models on forecasting floods induced by tropical cyclones in terms of accuracy and speed?

To answer the main question, several subquestions are listed below, in logical order:

- What data needs to be selected as input for a DL model and how does the data need to be pre-processed?
- What performance metric can be used to judge the performance of different DL techniques?
- How can the model architecture be optimized to capture the processes needed to predict compound coastal flooding?
- How well does the deep learning model replicate the spatial flood patterns predicted by the physics-based model?

- How does the simulation time of DL models compare to the simulation time of the physics-based model?

1.7. Research theory

The theoretical foundation of this research is based on the hypothesis that DL models can significantly improve the speed of flood forecasting induced by tropical cyclones, compared to physics-based models. Deep learning techniques excel in pattern recognition and predictive analytics and can leverage the extensive amounts of flood data to provide faster and more reliable forecasts.

2

Theoretical Background

2.1. Neural Networks

2.1.1. Definition

A neural network is a computational model based on the structure of the human brain, designed to detect patterns in data (McCulloch et al., 1943). Neural networks consist of multiple layers of nodes or neurons. All of these neurons can perform a simple calculation given input data. Processing input data allows the neural network to perform various tasks, such as classification and pattern recognition.

Figure 2.1 gives a schematic overview of a simple neural network. The ten neurons depicted in orange are stacked in three fully connected layers, where the links make the connections between neurons visible. Nodes are interconnected, giving nodes a dependency on all nodes in the previous layer. Data flows from the input layer to the hidden nodes layer, to be outputted by the output nodes layer.

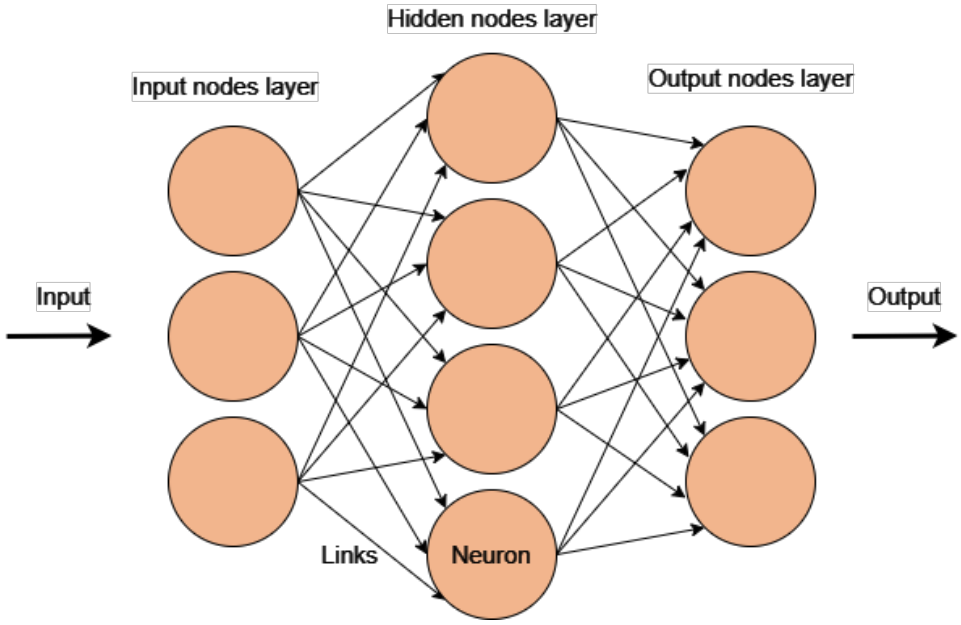

Figure 2.1: An example of a Neural Network

Trainable parameters for nodes are weights and biases, where the weight is the strength of the relation between the neuron and a neuron from the previous layer. Biases are constants associated with a neuron. Biases add an additional layer of flexibility to the deep learning model. The neuron input, a, is calculated using the following formula (Apicella et al., 2021):

$$a = \sum_j w_{ij} z_j + b_i$$

where w_{ij} is the weight of the connection from the neuron j belonging to the previous layer to i, the neuron belonging to the current layer, z_j is the output of the neuron belonging to the previous layer, b_i is the bias, and j runs on the indices of the neurons of the previous layer.

The neuron output z is computed by a differentiable (usually non-linear) function, also known as activation function (Apicella et al., 2021):

$$z_i = f\left(\sum_j w_{ij} z_j + b_i\right)$$

An activation function defines the internal transformation of the input to a neuron. Activation functions introduce non-linear transformations to the model, allowing the model to learn more complex patterns (Suttisinthong et al., 2014). A number of activation functions are available such as; linear or identity function, step function, sigmoid, hyperbolic-tangent function, etc. Several widely used activation functions are shown in Figure 2.2. However, the activation function can vary according to the architecture of the neural network, the input configuration, and application of the problem.

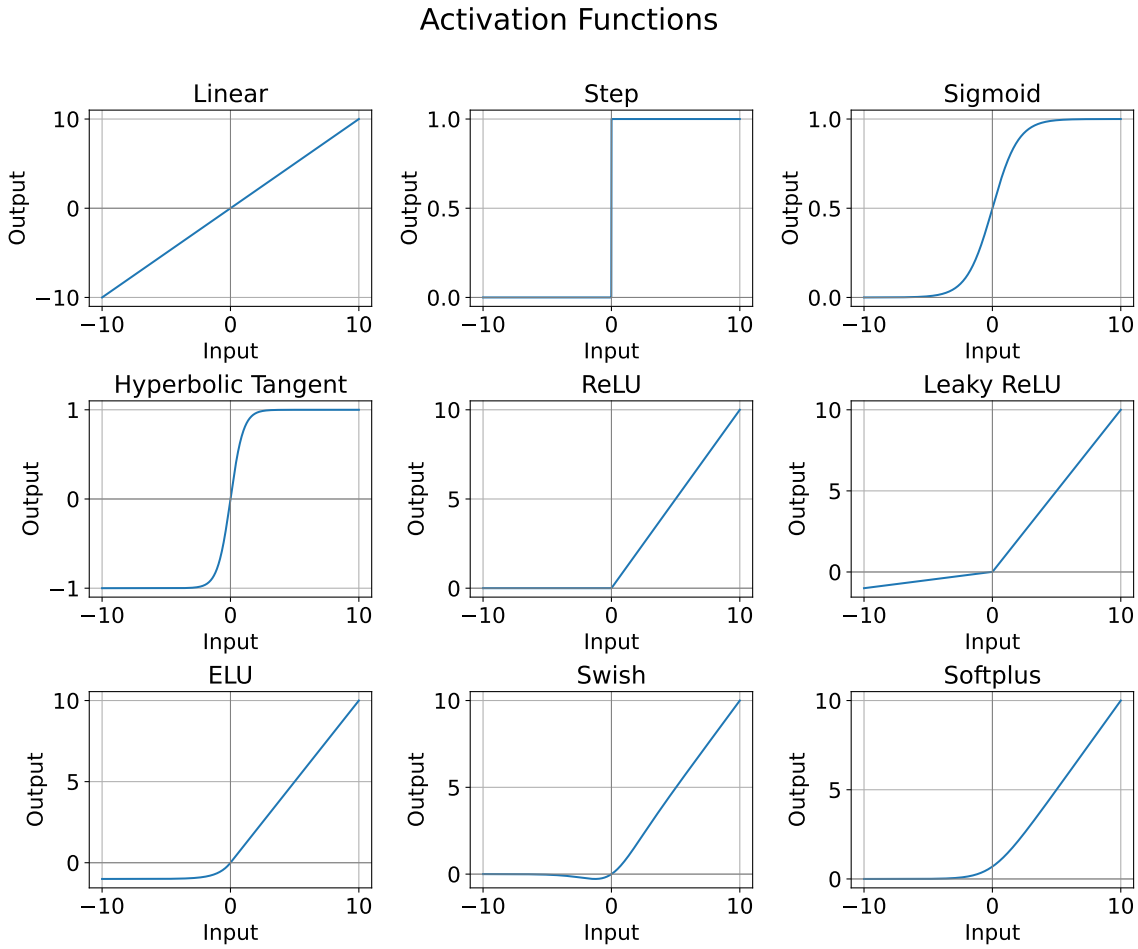

Figure 2.2: Examples of classic activation functions

Training the model means varying the weights and biases of the neurons in the hidden layer(s), thereby optimizing the output value. Based on a loss function, the accuracy of the prediction will be evaluated.

The loss function measures the difference between the predicted value and the true value. Loss functions guide the model's training process, as the goal of training the model is either minimizing or maximizing the loss. Examples of loss functions are the Mean Squared Error (MSE) and Mean Absolute Error (MAE)

functions. The MSE is commonly used for regression tasks, as the error is calculated by taking the average of the squared differences between the actual value and the predicted value. The MSE calculates the square root of the variance of the residuals. The effect of each error is proportional to the size of the squared error, and thus the MSE is sensitive to outliers. In MAE the errors are not squared, making this loss function less sensitive to outliers compared to MSE.

2.1.2. Training

The first step in training the neural network is initializing the weights and biases randomly. Data flows through the network, which is known as a forward pass, with each node performing its own calculation. Layer by layer the calculations continue, until the output layer is reached and the model gives its predicted results. The predicted output is then compared to the true value, which is done using a loss function.

The cost function calculates the discrepancy between the target and the predicted output. The average cost function is the average over the errors over a set of input and output pairs, and is defined as:

$$E(\Theta) = \frac{1}{m} \sum_{m=1} L(y, \hat{y})$$

Clearly, a model with randomly initialized weights and biases will not give good results. In the training process, adjusting weights and biases iteratively improves the accuracy of the network.

After calculating the loss, the weights and biases are updated by propagating the error backward through the network, an operation called backpropagation. Weights and biases are updated individually, because either one of the contributions may have a larger impact on the calculated loss.

$$w_{t+1} = w_t - \epsilon \frac{\partial E(\Theta)}{\partial w},$$

$$b_{t+1} = b_t - \epsilon \frac{\partial E(\Theta)}{\partial b},$$

where the weights and biases are updated using a learning rate ϵ and a time step indicated by t. The learning rate determines at which rate the model optimizes the weights and biases. A small learning rate will cause the algorithm to converge slowly during training, while a large learning rate will cause the algorithm to diverge. Common learning rates have a value between 0 and 1 (Smith, 2015). Negative learning rates should be avoided, as they move the parameters in the direction that increases the loss, thus preventing convergence.

The process of optimizing weights and biases is governed by an optimization procedure, which determines how the gradients are used to update the weights and biases.

A problem in a deep neural network occurs when gradients become too large or too small during back-propagation (Ali et al., 2024). Optimizers like Adam Optimizer address these issues, as do architectural changes such as batch normalization or residual connections (Kingma, 2014). An Adam Optimizer uses the momentum of the loss function and adaptive learning rates per parameter.

2.2. Convolutional Neural Networks

Building on the foundations of neural networks, an architecture exists which is designed to work effectively on structured data such as images or time-series, called Convolutional Neural Networks (CNNs). This architecture leverages the spatial or temporal relationship within the data to capture patterns of nearby points in the data more easily compared to a standard neural network.

CNNs were first introduced by LeCun et al. (1998) using the LeNet-5 architecture, a model designed for handwritten digit recognition. This architecture showed the ability of CNNs to extract hierarchical features, empowering the pattern recognition capabilities of the architecture.

The novelty of CNNs can be found in the convolutional layer, which makes the network capable of detecting patterns such as edges, textures, or shapes in the input data. Patterns can be learned through applying

filters, also called kernels, on the input data, which are, in essence, small matrices containing a weight for every pixel in the matrix. The filter moves over the input data in an operation called convolution, multiplying the filter with the window of the input data that the filter is covering, thus focusing on smaller regions of the data at the same time.

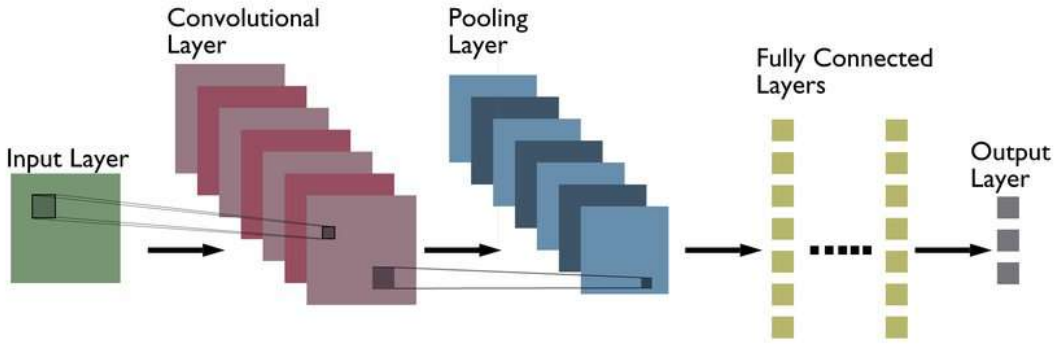

Figure 2.3: An example of a Convolutional Neural Network. Source: (Yazdani Abyaneh et al., 2018).

A schematic overview of an example CNN is given in Figure 2.3. As can be seen in the figure a CNN consists of different layers. These layers will be explained in more detail in the following subsections.

2.2.1. Input Layer

Data enters the network in the input layer of the architecture. In the case of an image this will be a multi-dimensional array or tensor where the dimensions are the height, width and depth of the image. A color (RGB) image contains three channels, whereas a black and white image contains only one channel. In the input layer, data can be passed to the DL model to be processed further.

2.2.2. Convolutional Layer

The convolutional layer is the foundation of the neural network, designed to extract key features from the input data by applying filters or kernels. As described above, these kernels are small weight matrices, which can slide over the input data, performing convolution to extract features. This process results in feature maps in which specific features, such as edges or patterns, are highlighted

Kernels will be of a smaller size than the input data, but the kernel size must be large enough to capture spatial patterns. Larger kernel sizes allow the convolutional layers to make use of spatial information from larger areas, but increase the computational window, hence increasing computational expense. A kernel with size 1 will only consider a single pixel, and does not capture spatial features beyond the individual pixel. A kernel size of [3,3] is large enough to capture spatial patterns between three pixels in both directions. An example of a kernel of size [3,3] moving over the input data is illustrated in Figure 2.4. The red box in the input data is the convolutional window, which is then multiplied by the kernel to result in the circled red value in the feature map. Many popular recent network architectures employ a kernel size of 3 units (Chen et al., 2018).

Feature maps are the output of applying the convolutional kernel on the input layer. The number of feature maps determines the amount of features that can be learned by the model. Increasing the number of feature maps means that more features can be learned, increasing the flexibility of the model and ability to capture complex patterns. Therefore, the number of feature maps depends on the input data complexity and computational limitations.

2.2.3. Pooling Layer

Following the convolutional layer, the pooling layer serves to reduce the spatial dimensions of the feature maps. During a pooling operation, a small filter is applied on each feature map, where the type of pooling determines to which value the area over which the filter is applied reduces. The filter summarizes the features that are present in the filter window, reducing the dimensions of the feature map, thus reducing the computational complexity and making the model more robust to variations in input data. Max and average pooling are commonly applied pooling operations (Indolia et al., 2018). Max pooling selects the maximum

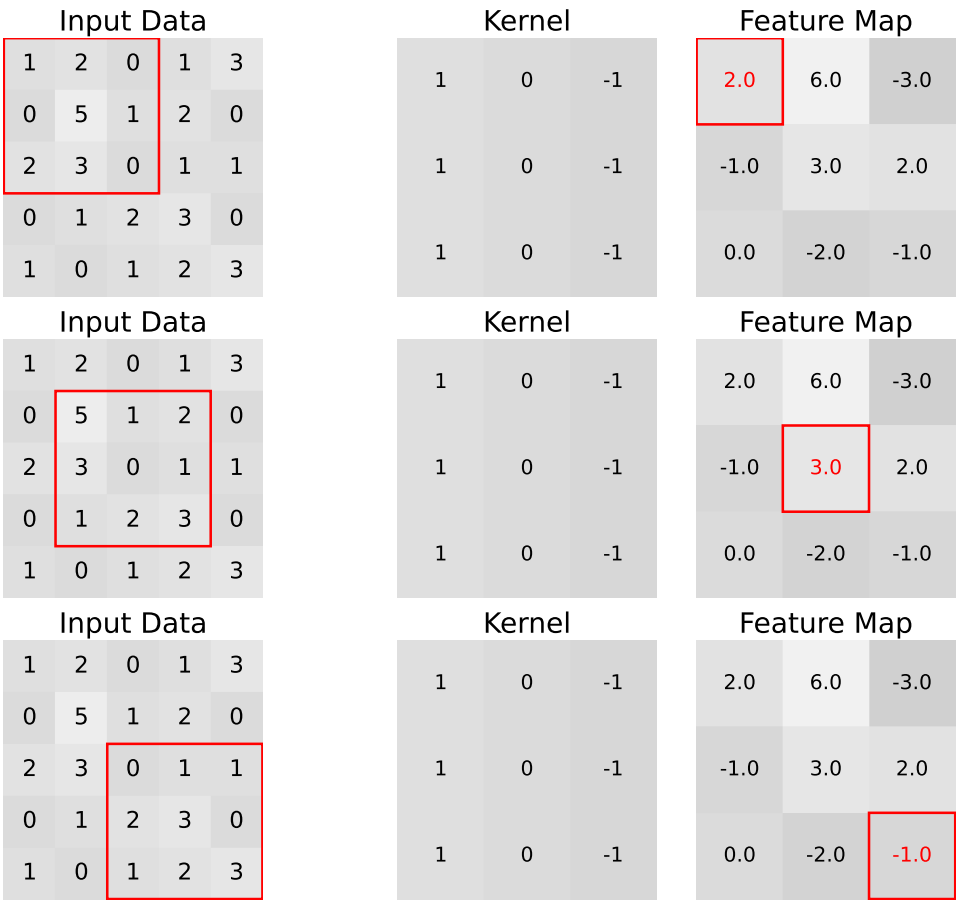

Figure 2.4: An example of a kernel moving over the input layer

element or value that is present in the filter window, while average pooling computes the average over the region. An example of a max pooling operation is shown in Figure 2.5.

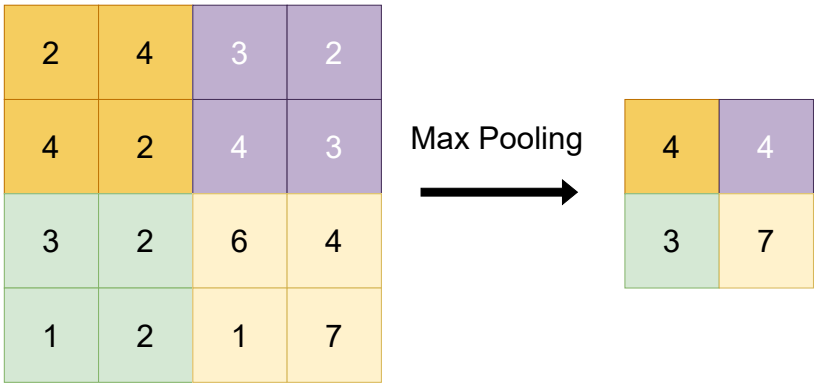

Figure 2.5: An example of a pooling operation

2.2.4. Fully Connected Layer

In a convolutional layer, each neuron is only connected to input in the window of the convolutional kernel, allowing the model to learn spatially localized patterns such as edges, or shapes.

However, in the fully connected layer (also known as a dense layer), there is no computational window.

Each neuron in the fully connected layer receives input from every neuron in the previous layer, regardless of its location in the layer. This means that the fully connected layer aggregates all the extracted features from earlier layers, combining them into a global understanding of the input. These dense connections allow the model to learn complex high-level representations that are crucial for making final predictions or classifications.

2.2.5. Output Layer

Following the fully connected layers, the output layer produces the final prediction of the model. Its structure depends on the specific task of the model. For classification, the output layer typically contains one neuron per class, with a softmax activation to convert the outputs into probabilities that sum to one. For regression tasks, it might consist of a single neuron with a linear activation to produce a continuous value. The output layer interprets the aggregated features of the fully connected layer.

2.3. Application to civil engineering

In the field of civil engineering, neural networks can be used to tackle a variety of challenges, especially those involving spatially distributed data. Many civil engineering problems require the analysis of inputs such as Digital Elevation Models (DEMs), rainfall maps, wind fields, flood maps, or land use data, which are inherently spatial.

CNNs are particularly well-suited for these tasks, as they are designed to extract and learn spatial patterns from grid-like inputs. By leveraging the local connectivity of convolutional layers, CNNs can capture fine-grained features related to terrain structure, hydraulic connectivity, and flow accumulation, which are essential for accurate modeling of phenomena such as flooding, erosion, or sediment transport (Xie et al., 2024).

Depending on the specific engineering task, the structure of the output layer in a CNN varies, as demonstrated by Yessoufou et al. (2023). For example, when predicting flood extent, the model performs a binary classification to determine for each grid cell if it is classified as flooded or not. In contrast, when estimating flood depth, the task becomes a regression problem, where the CNN predicts a continuous depth value for each pixel. In this case, the output layer typically uses a linear activation function.

Ultimately, CNNs offer a flexible and powerful framework for civil engineering problems that involve spatial prediction, enabling data-driven solutions that complement or enhance traditional modeling techniques.

3

Data and Study Area

This chapter describes the geographical location and available data of the study. It introduces the research area and provides an overview of the dataset available for model development. A description of the data analysis is given, as well as the preprocessing steps required to prepare the data for input into the deep learning model. These elements form the basis for the modeling approach described in the subsequent methodology chapter. Spatial data is processed and visualized using the Python packages Matplotlib (Hunter, 2007), GeoPandas (Jordahl et al., 2020) and Contextily (Arribas-Bel, 2023).

3.1. Study area

The study area is Charleston, presented in Figure 3.1, situated on the southeastern coast of the United States, in the state of South Carolina. The city is home to approximately 150,000 inhabitants, with a larger metropolitan population exceeding 800,000 inhabitants.

Charleston has a landscape defined by a mix of urban and natural features. The city's coastal location includes a network of rivers, estuaries, tidal marshes, and inland mountains, creating a diverse topography. This setting makes Charleston particularly vulnerable to flooding events caused by both coastal and inland drivers, including storm surges, riverine flooding, and heavy rainfall.

The topology of Charleston is complex and varied. Riverine areas such as the Ashley, Cooper, and Wando Rivers traverse the region, creating an intricate system of waterways, illustrated in the right panel of Figure 3.1. Islands located at the water-side of the city, including James Island and Johns Island, act as natural buffers, offering protection to the mainland from coastal flooding. However, the combination of sea level rise and urbanization increases the region's susceptibility to flood risks (Emanuel, 2013; Curtis et al., 2011).

The combination of these system components makes Charleston an interesting study area for understanding flood dynamics, particularly the interactions between geomorphic features and hydrodynamic processes in coastal settings.

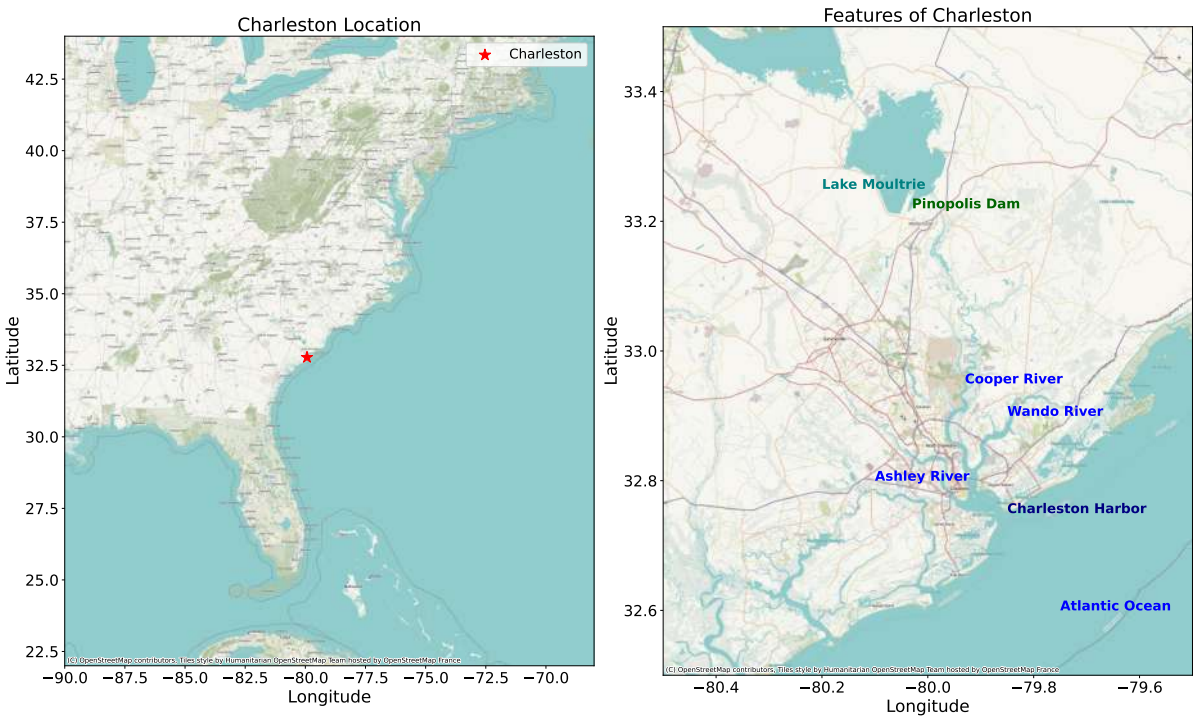

Figure 3.1: Left panel: Location of Charleston in the United States. Right panel: Features of the Charleston area.

3.2. Available data

A collection of historical flood events is available for the study area. By means of the Tropical Cyclone Wind Statistical Estimation Tool (TCWiSE) the historical data is processed and extended to create a synthetic dataset of 13,275 samples. The tool uses the statistics of historical TCs to create thousands of synthetic tracks, helping to determine the probability of a particular event (K. Nederhoff et al., 2021).

In the synthetic track, meteorological data is captured in the form of a spiderweb, which contains spatially varying input in a polar coordinate system. Available variables in the spiderweb are the TC location, precipitation, wind speed, and atmospheric pressure. For the TC, the location of the eye is given by longitude and latitude coordinates. These coordinates are given in a time series with a time step of three hours.

Precipitation data is available in precipitation rates per hour, varying over the distance from the eye of the TC and the direction (azimuth). A visual representation of the precipitation data is given in Figure 3.2. An increase in range (distance from the eye of the TC) shows lower precipitation rates. The highest precipitation rates can be seen around time step 22, close to the eye of the TC. After the peak of rain intensity, the intensity decreases again.

The wind data is presented in the spiderweb in the same manner as the precipitation data, but varying over the azimuth. All meteorological data serves as primary diver for flooding during TC events.

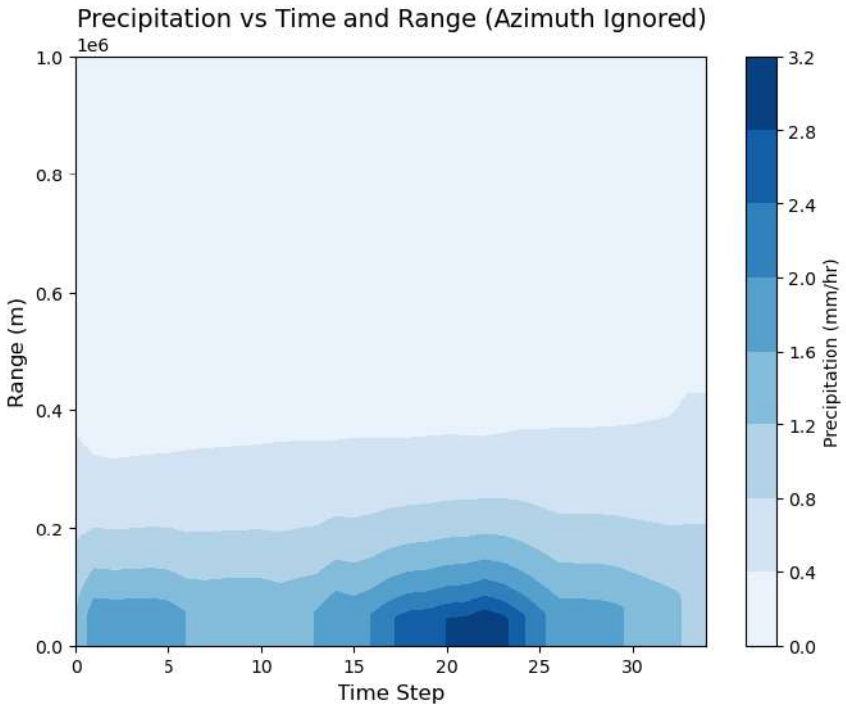

Figure 3.2: Visual representation of the available precipitation data. The precipitation varies per timestep over the range (distance from TC) and azimuth (direction). The variation over the azimuth has been ignored in this plot.

Synthetic events have been used by the model SFINCS to produce flood maps (Leijnse et al., 2021). SFINCS uses a physics-based approach to calculate flood levels during the flood event, as explained in Chapter 1. SFINCS uses two nested models to predict flooding in Charleston, a large-scale (overall) model and a detailed model. The overall model solves the equations with a resolution of about 2 kilometers, while the detailed model has a resolution of 200 meters. The number of active z points is 28,337 for the overall model, while in the detailed model this number is 119,864.

These models are coupled, where the boundary conditions for the detailed model are collected from the output of the overall model. An overview of the model phases is given in Figure 3.3. The overall model uses the input from the spiderweb in combination with generated boundary conditions. The overall model passes on water level boundary conditions to the detailed model, where it is used together with the spiderweb, to simulate the detailed model.

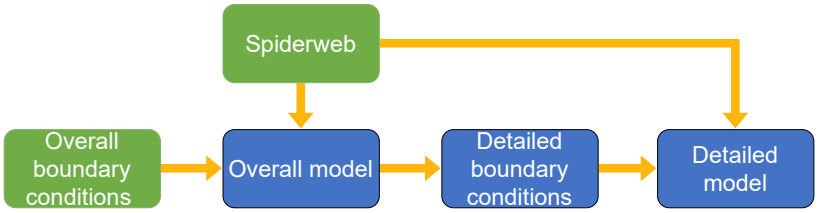

Figure 3.3: Flowchart of the SFINCS model

The overall model is presented in Figure 3.4, where observation locations are indicated in red. The detailed model obtains water level and wave boundary conditions from the observation points in the overall model.

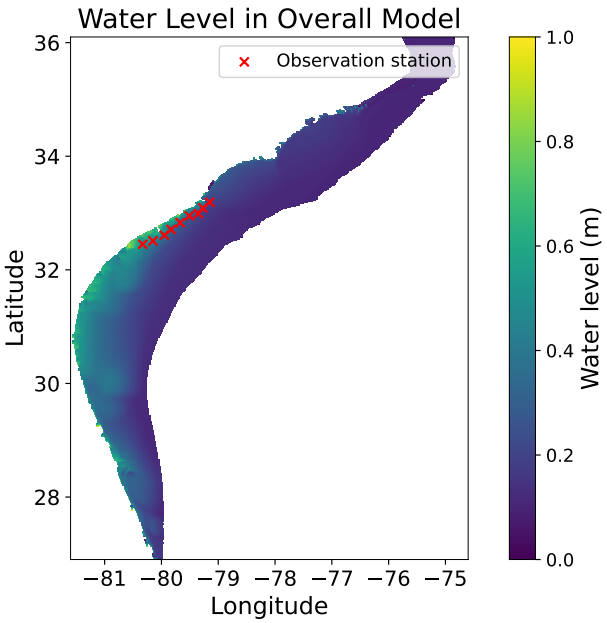

Figure 3.4: Output of the overall model. The observation locations are indicated in red.

After converting the overall model's output to boundary conditions for the detailed model, this model can be simulated. Its output contains the maximum water level for the duration of the TC. To account for elevation, the bottom level is subtracted from the water level, resulting in a flood map containing the water depth. For the deep learning model, flood maps containing water depths are the model target output, improving interpretability of the results.

An example of a flood depth map is given in Figure 3.5, where a higher value indicates a greater depth, illustrated by a lighter color. A dark colormap is used so that areas with zero water depth appear white, making it easy to distinguish between flooded and non-flooded regions. Riverine areas are clearly visible because the water depth in a river is larger than the water depth in a flooded area. In addition, many areas are colored white in the flood map, indicating that these areas are not wetted.

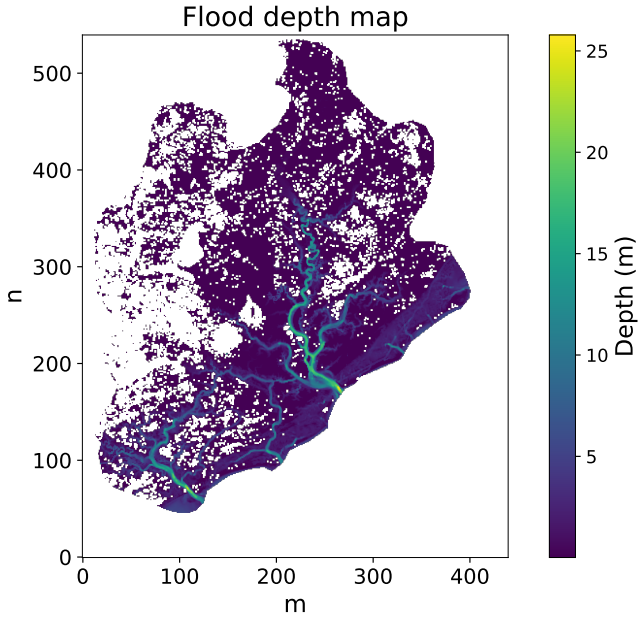

Figure 3.5: Example of a flood depth map

A Digital Elevation Map (DEM) is available, covering the area of Charleston in 540 x 440 pixels with a resolution of 200 meters. In total, the coverage of the DEM is approximately 100 square kilometers. A visual representation of the terrain is given in Figure 3.6. DEMs capture the elevation profile of a region, allowing for identification of flood-prone areas. In the figure, these areas are identifiable by their relatively low elevations compared to the surrounding terrain. Water tends to accumulate in these zones, either due to runoff from higher elevations or as a result of river or coastal flooding.

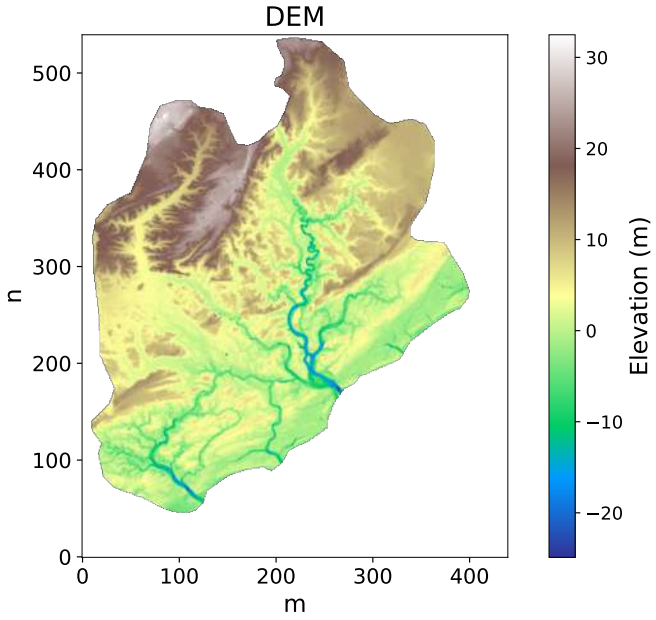

Figure 3.6: Digital Elevation Map of the study area of Charleston

3.3. Data analysis

The parameters cumulative precipitation, maximum wind speed, water depth at estuary entry, and flood extent are analyzed in this section. First the extraction of these parameters is explained, then distributions and correlations are explored.

In the spiderweb, precipitation is provided in 3-hour timesteps. For each timestep, with a calculated distance from the TC to the study area, the total precipitation can be computed. Summing precipitation across all timesteps yields the total cumulative precipitation used as input.

Wind data in the spiderweb has the same temporal and spatial structure as the precipitation data, however, in contrast to precipitation, wind speed varies over the azimuth. To derive a representative wind speed, the components in the x- and y-directions are combined, and the maximum wind speed over the duration of the TC is selected as the input parameter.

Data from the physics-based model SFINCS is extracted to obtain the water depth at the estuary mouth. The water depth is sampled from the flood depth map, at the entry of the estuary, presented in Figure 3.7, the border between river and ocean, where the storm surge effect is most pronounced. However, this data is dependent on the physics-based model, resulting in a limitation to the deep learning model, where a reliable water depth estimate needs to be computed first.

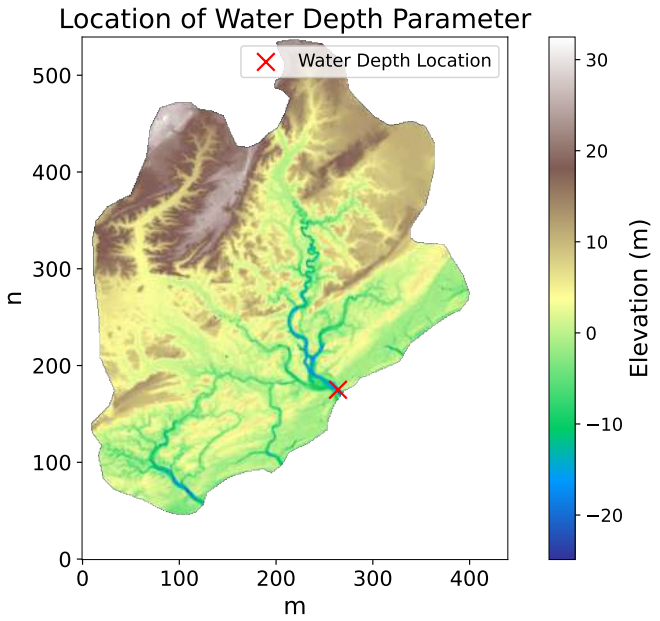

Figure 3.7: Location in the study area where the water depth variable is sampled from

Figure 3.8 displays the distribution of the flood extent variable. Flood extent is the amount of cells flooded in the flood depth map, where flooded means a water depth greater than a threshold. A common value for the threshold of flooding is 0.05 to 0.15 meters (Jamali et al., 2019; Löwe et al., 2021). In this study, a threshold of 0.1 meters is applied to capture water depths that represent minor but tangible impacts, while excluding surface wetting.

The distribution of the flood extent is notably right-skewed, with a large number of samples having a relatively small flood extent. As flood extent increases, the frequency of occurrence decreases, indicating that minor flood events are considerably more common than severe floods. The most extreme flood event in the dataset affects nearly the entire study domain, with approximately 80,000 flooded cells out of a total of 120,000 valid (non-NaN) cells in the DEM.

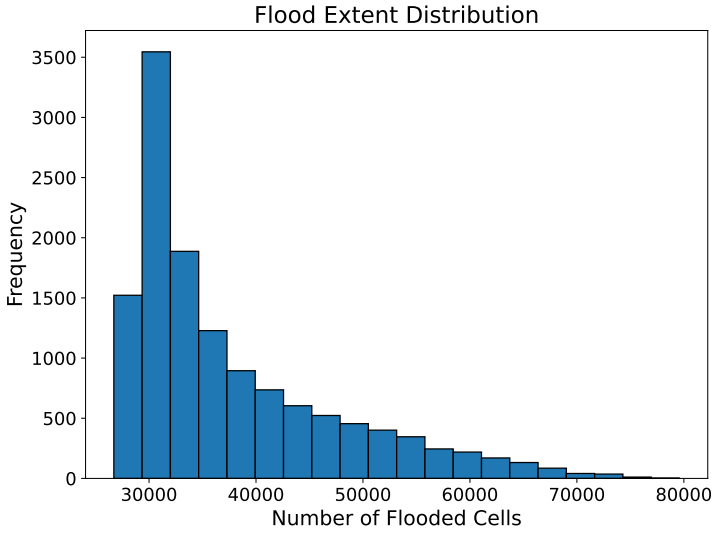

Figure 3.8: Histogram of flood extent across all samples

The distributions of the three potential input parameters are illustrated in Figure 3.9. These distributions exhibit similar characteristics to the flood extent variable, with each showing a decreasing frequency as the parameter values increase. This pattern suggests that extreme values for these input parameters are less common, mirroring the distribution of severe flood events.

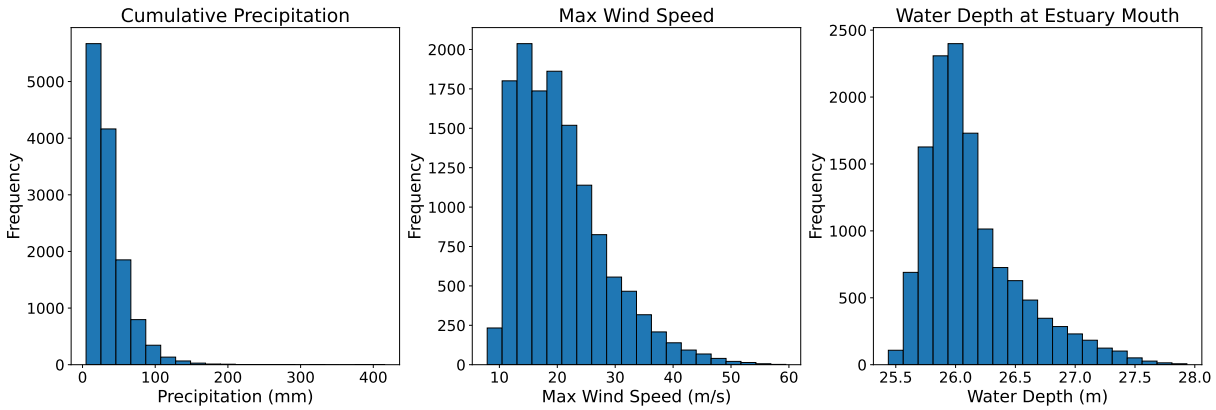

Figure 3.9: Histograms of input parameters: precipitation, wind speed, and water depth

In order to illustrate how the variables relate to each other, a joint distribution is made. The distribution of samples between two values becomes visible, and as additional parameter a hue color can be used to visualize the effect of two variables on a third variable.

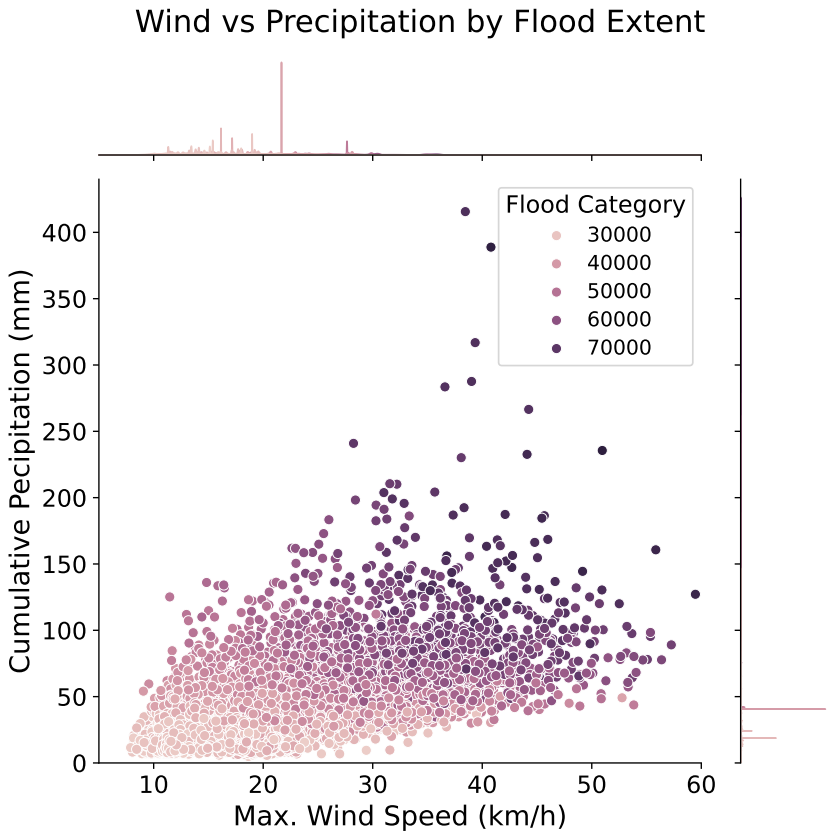

Figure 3.10: Joint distribution of wind speed against precipitation

In Figure 3.10 the variables wind and precipitation are plotted, showing a positive correlation between the two variables. As the maximum wind speed increases, the total cumulative precipitation increases. This can be attributed to more intense TCs, which have higher maximum wind speeds and higher precipitation rates compared to smaller TCs.

This correlation indicates that wind and precipitation are not fully independent parameters. Although neural networks can handle correlated inputs effectively for prediction, this correlation may affect the interpretability of the model results, making it more difficult to distinguish the individual contributions of each parameter to the predicted water depth.

Another conclusion that can be drawn from Figure 3.10 is higher cumulative precipitation and higher maximum wind speed result in a greater flood extent. In the bottom left of the figure, parameters take on low values, corresponding with a small flood extent, whereas in the top right the flood extent is greater for samples with high parameter values.

In Table 3.1 the Pearson and Spearman correlation coefficients can be seen between the number of flooded cells and different input parameters. Precipitation has the highest correlation with the flood extent. From a civil engineering perspective this makes sense, because the inland patches are flooded by precipitation. Wind and water depth are strongly correlated to the flood extent. Wind and precipitation are correlated to each other through the intensity of a TC, contributing to the correlation between wind and flood extent. A high water depth in the river means that river banks overflow more quickly, resulting in a greater flood extent.

Figure 3.11 presents the distribution of the three input parameters, precipitation, wind speed, and water depth in relation to flood extent. A positive correlation is observed for all three parameters, indicating that higher values are generally associated with a larger number of flooded cells. Among these parameters, precipitation has the strongest correlation with flood extent, as can be observed by the narrow spread of data points along the trend line. This suggests that increased precipitation is directly related to the

Table 3.1: Correlation coefficients between parameters and flooded cells

Parameter	Pearson Correlation	Spearman Correlation
Wind	0.795	0.744
Precipitation	0.876	0.846
Water depth	0.719	0.788

flood extent. Notably, several outliers in the precipitation data are associated with extreme flood events, characterized by a high number of inundated cells.

Wind speed and water depth display a wider spread of data points, implying a less direct, though still positive, correlation with flood extent. In the case of water depth, a distinct lower threshold can be observed. Floods with depths exceeding this threshold tend to breach riverbanks and result in flooding of adjacent low-lying areas. This pattern illustrates the importance of water depth in controlling the spatial extent of flooding.

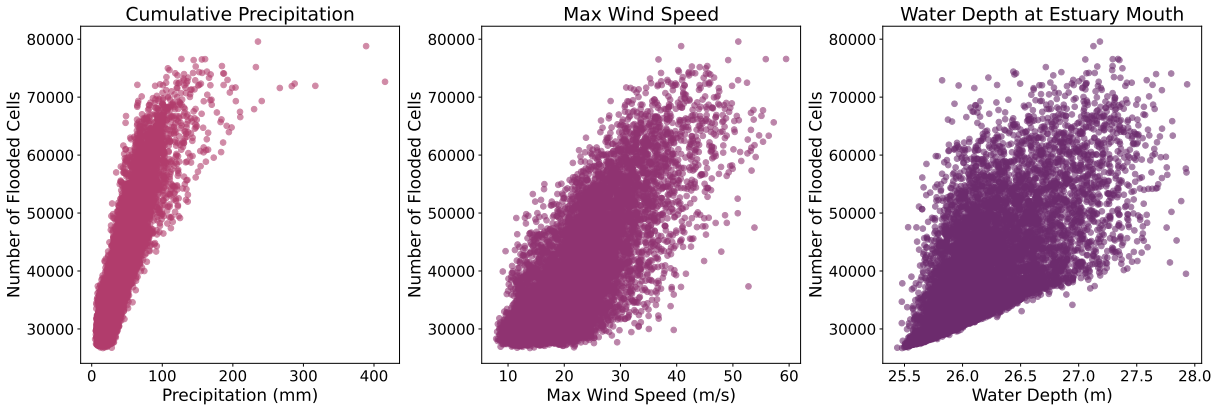

Figure 3.11: Distributions of the available parameters

3.4. Data preparation

Neural networks are rarely applied to a raw dataset. Data needs to be processed in order for a neural network to apply statistical methods. Preprocessing steps include cleaning and normalizing the raw data. Missing values can cause errors during model training or testing because a neural network cannot handle undefined inputs. Furthermore, input and target variables need normalization to improve model efficiency and performance. Normalization rescales variables to a range of [0, 1] or [-1, 1], depending on the target variable and its distribution. The normalization is done over the entire dataset. Although not all samples are used in the training, validation and testing of the model, these samples need to be scaled in the same manner as would be done for the entire dataset.

Generally, for positive data, normalization to [0,1] is the preferred approach. This is used to normalize precipitation, wind speed and flood depth. Additionally, this approach can also be used for the DEM, because consistency across features is conserved, retaining the physical meaning of elevations. The model learns that values closer to zero are the lowest lying areas. Furthermore, mapping all inputs to the same range avoids issues related to different magnitudes between features.

Figure 3.12 shows the DEM before and after normalization. To enable the model to process the data, cells with missing elevation values (originally marked as Not a Number, NaN) were replaced with zeros during preprocessing. However, since a value of zero may correspond to actual terrain elevation (e.g. sea level or river bed level), this substitution introduces ambiguity between low-lying terrain and originally missing data.

To mitigate this issue, a binary mask was used during model training and evaluation. This mask identifies the originally valid and invalid (NaN) regions of the DEM, ensuring that loss calculations are restricted

only to valid areas. By excluding masked-out regions from the loss function, the model is not penalized for predictions in areas where no elevation data was available. This approach helps prevent the model from learning patterns from artificially introduced zero values and focuses on learning from reliable input data.

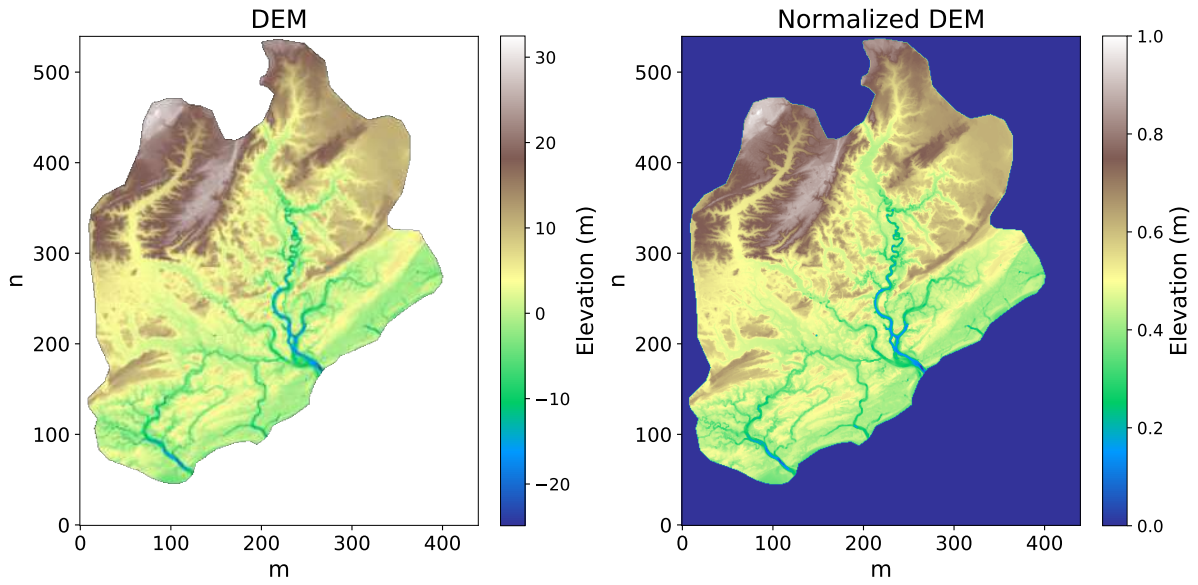

Figure 3.12: DEM before and after normalization

Missing and incorrect values in the data are removed from the dataset. This involved two samples, leaving a dataset consisting of 13273 samples.

Water depth is primarily determined by the bathymetry and storm surge, with the deepest areas reaching about 25 meters below reference level near the estuary mouth. The maximum water depth in the domain generally ranges from 25 to 28 meters. However, some samples include a water depth of 35 in local spots, as can be seen in Figure 3.13. After discussing these observations with SFINCS experts, it is concluded that this is a model error. Due to a high gradient near the river, the rainfall runoff accumulates in the riverine area. The model experiences difficulties with spreading the water, so that a difference of water depth of 10 meters can occur within a few pixels. As a result, 190 samples, representing less than 2% of the total dataset, were removed from the dataset.

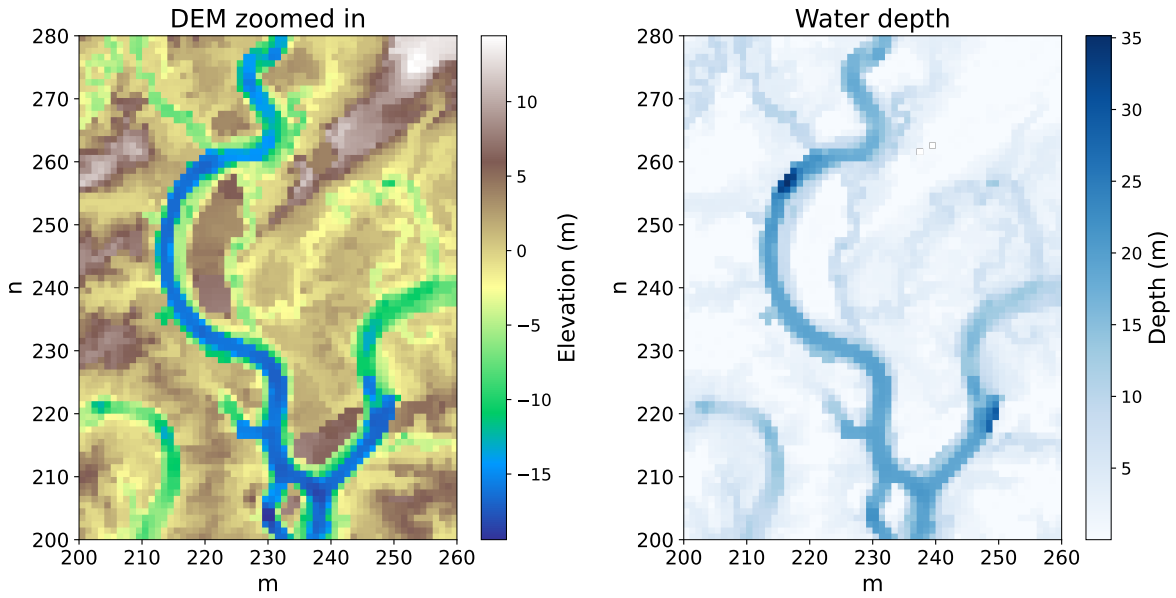

Figure 3.13: Flood depth anomalies in the SFINCS flood map

4

Methodology

4.1. Model setup

This section outlines the configuration of the deep learning model used to predict flood maps based on topographic and meteorological inputs. A U-Net-like Convolutional Neural Network (CNN) architecture was selected as the basis for the deep learning model due to its ability to extract spatial features from the input and reconstruct the output to a flood map. An overview of the deep learning architecture is given in Figure 4.1.

The setup includes the integration of spatial and scalar inputs, the definition of model depth and convolutional parameters, and the configuration of output representations. Each design choice, from kernel size and feature map progression to skip connections and upsampling strategy, is optimized to balance accuracy and computational efficiency. The following subsections describe the architecture and implementation choices in detail.

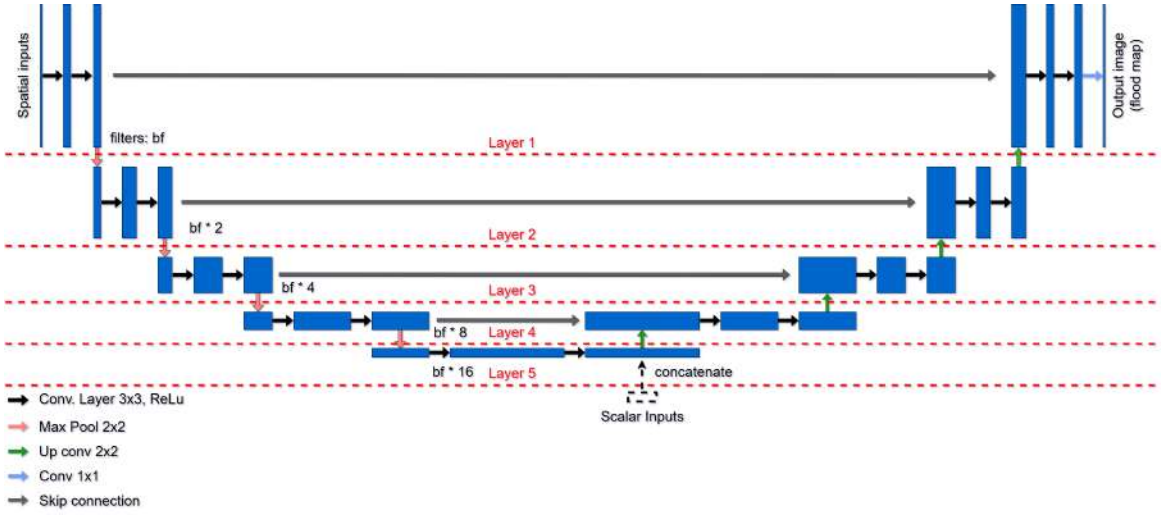

Figure 4.1: Architecture of the deep learning model

4.1.1. Input Configuration

Each model sample consists of spatial and scalar inputs that together define the flood scenario. The spatial input consists of a DEM of the study area, which provides topographic information on a fixed grid (540 × 440). The DEM encodes terrain information such as depressions, edges, and flow paths, all of which influence how water flows and accumulates during a flood.

To represent the forcing conditions, three scalar inputs are included: cumulative precipitation, maximum wind speed, and maximum water depth. These variables are unique for each simulated flood event, but are uniform across the spatial grid. To allow the model to incorporate the forcing conditions spatially, each

scalar variable is broadcast across the spatial grid, creating a uniform map that contains the same value at every grid cell. The spatial uniform maps are then concatenated with the DEM, resulting in a multi-channel input that combines both spatial and event-specific features. This approach allows the model to combine the scalar inputs for each simulated event with the spatial structure of the DEM.

4.1.2. Network depth

The model follows a U-Net-like architecture composed of an encoder–decoder structure. The encoder consists of four levels of downsampling, where in every layer the input is passed through a pair of convolutional layers and a pooling operation. This progressive reduction in spatial resolution allows the network to extract increasingly large-scale spatial features relevant to flood behavior.

The decoder mirrors the encoder, reconstructing the output resolution by upsampling and refining the features at each level. Skip connections are used to concatenate feature maps from the encoder to the corresponding decoder levels, preserving the spatial information that is lost during downsampling.

The depth of a neural network is defined by the number of layers that the neural network is composed of. In the figure, the depth of the network is shown by the dashed red line. Generally, a deeper network allows the model to capture greater detail and extract more features from the data. However, this comes with a loss of spatial information that is present in the data and an increase in computational time. The depths of the network to consider in the analysis differ from 3 to 5 layers, based on frequently used network depths (Löwe et al., 2021).

4.1.3. Feature maps

As mentioned in the previous chapter, the number of feature maps compares to the amount of features that can be learned by the model. Increasing the number of feature maps means that more features can be learned, increasing the flexibility of the model and ability to capture complex patterns. However, increasing the number of feature maps also increases the computational time of the model. The variation in number of feature maps is selected to range between 8, 16, 32, and 64. A study by Löwe et al. (2021) demonstrated its effectiveness with a similar number of feature maps.

4.1.4. Kernel size

In a standard U-Net architecture, a kernel size of 3 is used (Ronneberger et al., 2015). This size has been used frequently in other U-Net applications, for semantic image segmentation by Chen et al. (2018) and flood map prediction by Löwe et al. (2021). Taking into account the resolution of 200 meters for one pixel, models are trained applying different kernel sizes: 3, 5 and 7.

4.1.5. Skip-connections

Skip connections are illustrated by the gray arrows, indicating that the output of that layer is input for a layer further in the deep learning model. With the addition of skip connections, the model resembles the U-Net architecture (Ronneberger et al., 2015).

4.1.6. Loss function

The loss function will be varied as well, to optimize prediction for initially dry areas. The loss function used for training and validation of the data is a RMSE loss function, given by the formula:

$$L_{\text{RMSE}}(y, \hat{y}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{4.1}$$

The problem at hand is predicting flooding. RMSE is not conservative in predicting flooding, as weights and biases will be such that the most prevailing flooding conditions are predicted accurately. In engineering, the goal is to anticipate on floods with lower probability as well. An approach is to adopt a quantile loss function such as the pinball loss function. It is a metric used to assess the accuracy of quantile forecasts.

The pinball loss function, shown in red in Figure 4.2, has been named after its shape, which looks like the trajectory of a ball on a pinball. The function always has a positive value and the farther away from

the target value $\hat{y}$, the larger the loss. The slope is used to reflect the desired imbalance in the quantile forecast. The target quantile is given by q, with the following formulas:

$$L_{\text{pinball}}(y, \hat{y}) = \begin{cases} (y - \hat{y})q, & \text{if } y \geq \hat{y} \\ (\hat{y} - y)(1 - q), & \text{if } \hat{y} > y \end{cases} \tag{4.2}$$

If the target quantile is set to 0.8, the loss penalizes underpredictions more heavily than overpredictions. Specifically, when the prediction $\hat{y}$ is less than the true value y, the loss increases more sharply compared to cases where $\hat{y}$ exceeds y.

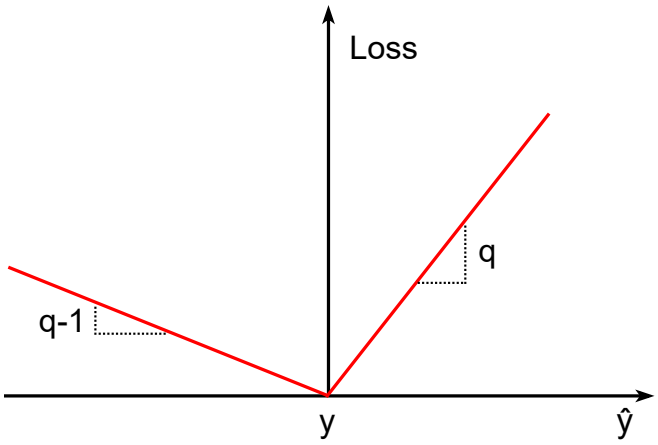

Figure 4.2: Pinball loss function

For the binary classification task of identifying whether individual cells are flooded or non-flooded, the Binary Cross-Entropy (BCE) loss function is used. BCE is a widely adopted loss function for binary classification problems, as it quantifies the dissimilarity between the predicted probability and the actual binary label.

In this case, the model outputs a logit value per cell, which is interpreted as the likelihood of flooding through the sigmoid activation function. The sigmoid function maps the raw output to a probability between 0 and 1. The BCE loss is then computed as:

$$L_{\text{BCE}}(y, \hat{y}) = -\left[y \cdot \log\left(\hat{y}\right) + (1 - y) \cdot \log\left(1 - \hat{y}\right)\right] \tag{4.3}$$

where the true label y has a value of either 0 (non-flooded) or 1 (flooded). The predicted probability $\hat{y}$ is a continuous value between 0 and 1. In practice, the model is trained using a combination of the sigmoid activation and BCE loss in a numerically stable way. This avoids the need to apply the sigmoid function manually prior to loss computation.

While the regression loss captures the magnitude of water depth, it tends to produce over-smoothed outputs, particularly for extreme flood events where sharp flood boundaries are important. To address this, the classification loss serves as a complementary objective to encourage the model to better predict flood extents, especially in marginal areas that would otherwise be underestimated by the regression loss alone. This is critical to accurately capture extreme scenarios where a reliable prediction of the flood extent is crucial.

To effectively train the model on both tasks, a MultiTask loss function is used, combining the regression loss (e.g., RMSE) with the BCE classification loss. This allows the model to simultaneously predict continuous water depths and identify flooded areas. The MultiTask loss function is defined as:

$$L_{\text{MT}}(y, \hat{y}) = w_{\text{reg}} \cdot L_{\text{reg}} + w_{\text{BCE}} \cdot L_{\text{BCE}} \tag{4.4}$$

where w_{reg} and w_{BCE} are predefined weights that control the relative importance of each task during training. The weights remain fixed throughout the training process. One configuration emphasizes regression with $w_{\text{reg}} = 0.7$ and $w_{\text{BCE}} = 0.3$, while the other prioritizes classification by swapping the weights: $w_{\text{reg}} = 0.3$, $w_{\text{BCE}} = 0.7$. This approach draws inspiration from recent advancements in surrogate modeling, such as the Surge-NF framework, which demonstrated that a multi-task learning module can enhance peak storm surge prediction accuracy (Jiang et al., 2024).

4.2. Training and validation

The model receives the preprocessed data as input. The input will be used to train, validate, and test the model. A split needs to be made to divide the available data into three subsets; a training, validation, and test set. The split of these data is chosen with a seed, so that all simulations use the same data in the training, validation, and test set. This ensures that simulations are reproducible and that the predictions of the models can be compared to each other.

4.2.1. Dataset size

It is important to have a balanced sample size for the training and validation of the model. On the one hand, the subset of samples must represent the entire dataset, on the other hand, the subset is preferred small, so that training and validation of the model in all simulations is computationally efficient.

Three models have been trained with different training and validation sets. A small training set, containing 200 samples, a medium size training set of 1000 samples, and a large dataset containing 5000 samples. The difference in coverage of the subsets is shown in Figure 4.3.

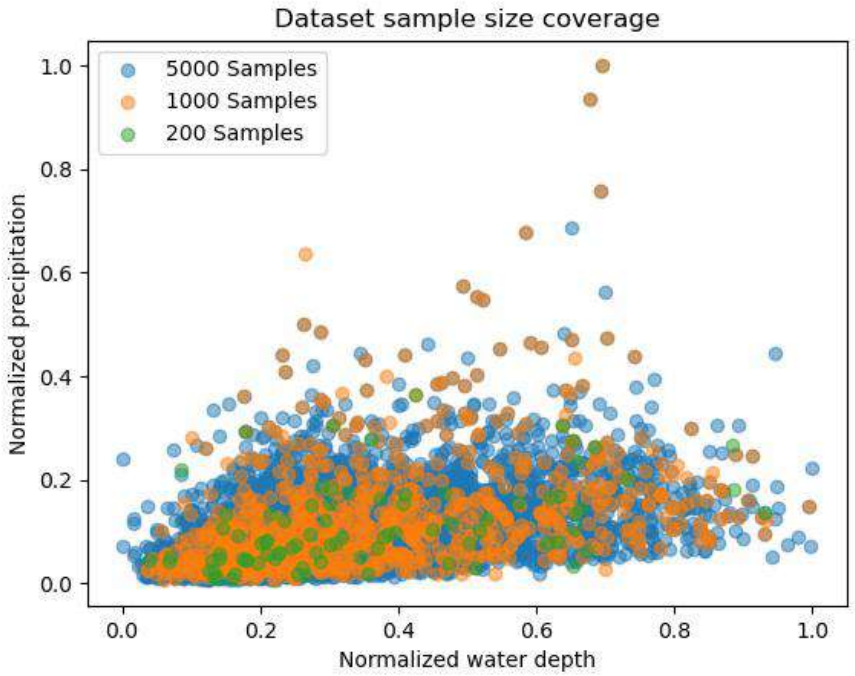

Figure 4.3: Coverage of subsets of data with varying dataset sizes

All training datasets follow the general pattern of the data, a large number of samples with low parameter values. For increasing training dataset size, more variance in the extreme parameter values can be seen.

The influence of the input data will be analyzed in the first round of simulations, meaning that there will be variance in the amount of input data. It is important to create a representative subset of the data to train, validate, and test on, as this subset will be kept small, in order to keep the training time of the models low.

4.2.2. Sampling method

A great way to cover the influence of the input data is to compare two completely different datasets. One dataset containing randomly selected samples and the other dataset samples selected based on a clustering method. The test sets will be the same in both cases, to ensure an equal comparison between both models.

K-means clustering is a widely used technique in unsupervised learning, dividing data into distinct clusters based on similarity. Using K-means clustering, data points can be sampled from different clusters, ensuring that the selected samples represent diverse yet structured patterns within the dataset. This method allows for a more systematic examination of how structured variability in input data influences the simulation outcomes, as opposed to the purely random selection approach. Comparing results from these two datasets will provide insights into the sensitivity of the model to different data distributions and highlight potential biases introduced by random versus structured sampling.

Both training and validation samples are drawn using the K-means clustering method to ensure consistency and maintain diversity across the subsets. This method helps capture the variability in the entire dataset, while working with a smaller, more manageable portion of the data. However, determining the optimal size of these subsets involves a trade-off: larger subsets may yield better model performance but come at a higher computational cost, while smaller subsets speed up training, but may not capture the full complexity of the data. Therefore, the subset size must be carefully balanced to ensure efficient training without compromising the model's accuracy.

The training and validation sets using random samples, shown in Figure 4.4, have less spreading than the samples selected by K-means clustering, illustrated in Figure 4.5.

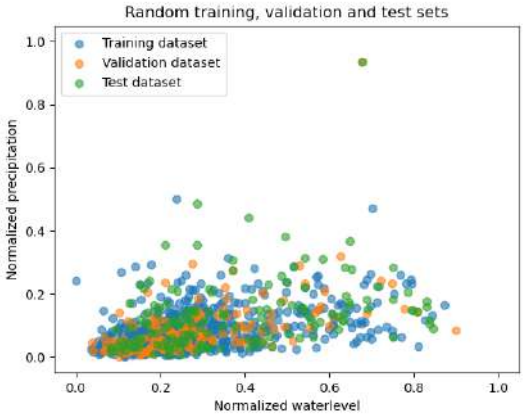

Figure 4.4: Data split distribution after random sampling

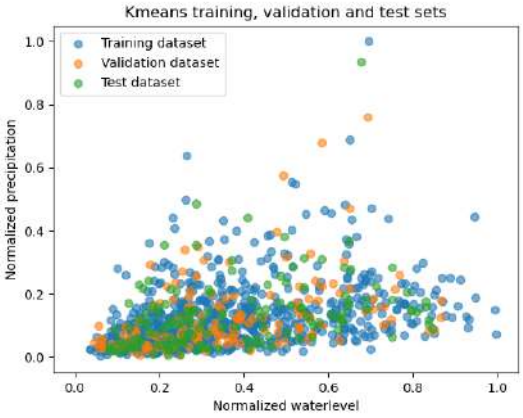

Figure 4.5: Data split distribution after K-means sampling

The test set is designed to represent approximately 15 to 20% of the total available data, with a final selection of about 2000 samples. The distribution of the test dataset is illustrated in Figure 4.6, demonstrating that it provides a representative subset of the entire dataset. The test set covers the entire range of parameter values for both precipitation and water depth, while largely maintaining the shape of the data distribution.

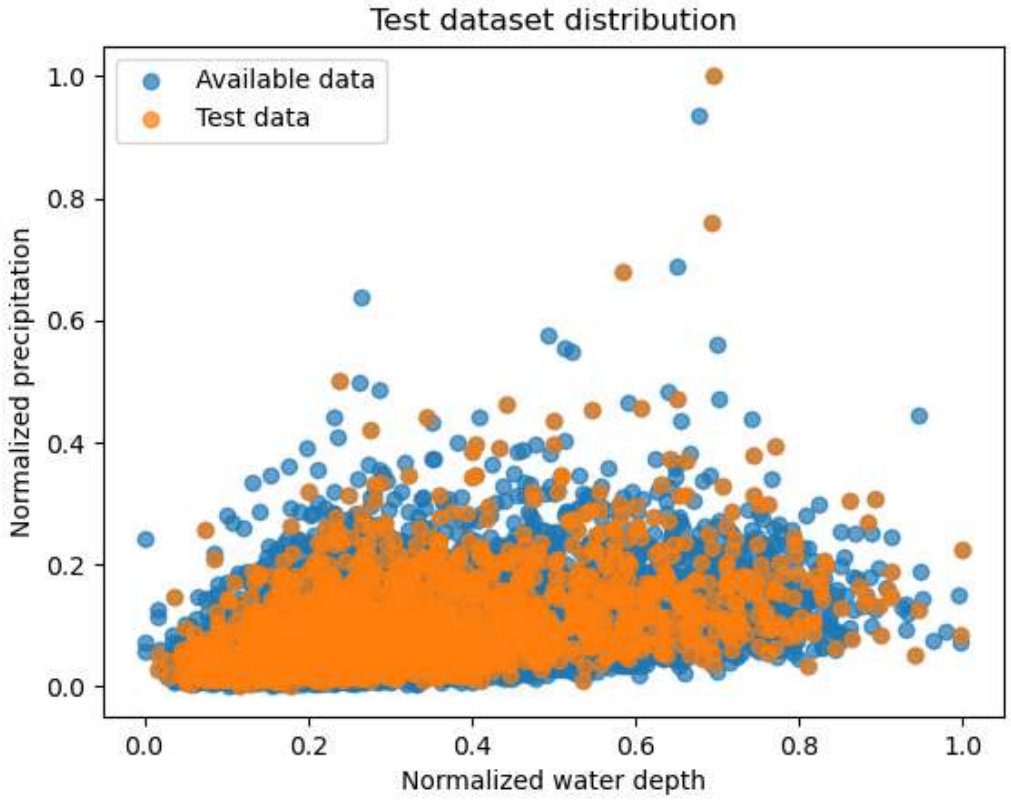

Figure 4.6: The selected test dataset compared to the entire dataset

4.2.3. k-fold cross validation

To obtain a more robust estimate of predictive model performance, the model is trained with a k-fold cross-validation procedure with five folds (k = 5). For this purpose, five non-overlapping training and validation datasets are created randomly.

K-fold cross-validation is used to evaluate a model's performance more reliably and reduce the risk of overfitting to a specific train–validation split. The available dataset is divided into k equal-sized subsets, also called folds. The model is trained k times, each time using k−1 folds for training and the remaining fold for validation.

This process ensures that every data point is used for both training and validation across the different iterations. The average performance across all folds provides a more robust estimate of how the model will generalize to unseen data.

4.3. Simulations

Many simulations are dependent on eachother. However, for the sake of this research, only the individual contributions to the performance will be evaluated.

For the first simulation, we start with parameters as described in Chapter 4, all input parameters, and a K-means clustered training and validation dataset. Model hyperparameters are a depth of 3, a kernel size of 3, and 16 filters. The number of epochs is set to 500, with an early stopping mechanism for no improvement in validation loss for 200 epochs. The learning rate is chosen to be 0.001 and the batch size is 32 samples.

All simulations will be performed in descending order. An overview of all simulations on variables as mentioned before are listed in Table 4.1.

Table 4.1: Table of simulations

Simulation	Value	Reasoning
Training/validation set size	200 samples 1000 samples 5000 samples	Optimal training set size
Input parameters	Precipitation Wind speed Water depth Precipitation, wind speed Precipitation, water depth Wind speed, water depth All 3 parameters	Select input parameters
Input dataset	Random Kmeans	Importance and variance of input data
Mask	DEM River	Importance of prediction
Loss function	RMSE Pinball Binary Cross Entropy (BCE) MultiTask	Optimal model architecture
Network depth	3 4 5	Optimal model architecture
Feature maps	64 32 16 8	Optimal model architecture
Kernel size	3 5 7	Optimal model architecture

4.4. Model performance

4.4.1. Evaluation Metrices

To be able to give a measure of how well a model performs, evaluation metrics are needed. A common evaluation metric is the Root Mean Squared Error (RMSE):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (y_{pred} - y_{target})^2}{N}},$$

By using this metric, the larger outliers are penalized more than smaller errors. If we want to prevent penalizing the larger errors more, the Mean Absolute Error (MAE) can be used:

$$MAE = \frac{\sum_{i=1}^{N} |y_{pred} - y_{target}|}{N},$$

Using RMSE or MAE as an error metric results in a good indication of how close the prediction is to the target. However, a key element that is missing is how well the model predicts dry and wet cells, or in other

words, the flood extent. For this, three classification metrics are introduced.

4.4.2. Classification Metrics

A single value is preferred for model selection, to get an immediate overview of the model performance. The Critical Success Index (CSI), also known as Intersection over Union (IoU), is a metric that measures how well the predicted flood extent overlaps with the actual flood extent. It gives the ratio of correctly predicted flooded cells to the total predicted and actual flooded cells, where a flooded cell is defined as a cell with a flood depth larger than a given threshold. The threshold between dry and wet cells is defined as a water depth of 10 centimeters. The formula of the CSI is given by:

$$CSI = \frac{|WET_{pred} \cap WET_{target}|}{|WET_{pred} \cup WET_{target}|} = \frac{TP}{TP + FP + FN},$$

where a value closer to 1 indicates that the model is closer to the ground truth.

The classification terminology is described as follows:

- **TP (True Positives)**: The number of cells correctly predicted as flooded.
- **FP (False Positives)**: The number of cells incorrectly predicted as flooded (these were actually dry).
- **FN (False Negatives)**: The number of cells incorrectly predicted as dry (these were actually flooded).
- **TN (True Negatives)**: The number of cells correctly predicted as dry.

To provide context for these terms, the classification of each grid cell can be described as follows:

- **True Positives (TP)**: Cells that are flooded in both the ground truth and the model prediction.
- **False Positives (FP)**: Cells that the model predicts as flooded, but are dry in the ground truth.
- **False Negatives (FN)**: Cells that are flooded in the ground truth, but the model predicts as dry.
- **True Negatives (TN)**: Cells that are dry in both the ground truth and the model prediction.

An alternative for regression tasks, the skill score (Murphy, 1988), indicates the predictive capability of the model. The formula is given by:

$$skill = 1 - \frac{RMSE^2}{\sigma^2_{target}},$$

where a skill score of 1 indicates a perfect prediction. The closer the skill is to one, the better the predictive capabilities of the model. Obtaining the skill on a pixel basis can be a challenging task. In case of non-varying truths, in other words a pixel that never floods, the standard deviation of the model truths can be equal to zero, resulting in an infinitely low skill rate. Therefore, modeling the skill in terms of closeness in predicting the amount of wet cells is suggested. The number of flooded cells differs for every training sample, thus a high skill score indicates a model that can capture the exent well based on given input parameters.

Two additional scores, False Wet Rate (FWR) and False Dry Rate (FDR), are used to gain insight into the predictive capacities of the model.

$$FWR = \frac{FW}{DRY_{target}} = \frac{FP}{FP + TN},$$

$$FDR = \frac{FD}{WET_{target}} = \frac{FN}{FN + TP},$$

where FWR is the number of dry cells that the model predicted as wet and FDR the number of wet cells predicted as dry. An overprediction of the flood is defined in a high number of false positives, indicated by a high FWR. The same holds for a high number of false negatives, resulting in a high FDR.

Table 4.2: Score values used to compare the maximum water depths predicted by the neural network and the physics-based model.

Score	Purpose	Equation	Range	Best value
RMSE (m)	Root mean square error to penalize large errors more heavily	$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(y_{pred} - y_{target})^2}{N}}$	$0 - \infty$	0
MAE (m)	Average error	$MAE = \frac{\sum_{i=1}^{N}\lvert y_{pred} - y_{target}\rvert}{N}$	$0 - \infty$	0
CSI	Binary comparison on pixel to pixel basis	$CSI = \frac{\lvert WET_{pred} \cap WET_{target}\rvert}{\lvert WET_{pred} \cup WET_{target}\rvert}$	$0 - 1$	1
skill	Variation of prediction accuracy	$skill = 1 - \frac{RMSE(PredWet, TrueWet)}{StdofTrueWet}$	$-\infty - 1$	1
FWR	Indicator that model is overpredicting flood extent	$FWR = \frac{FW}{DRY_{target}}$	$0 - 1$	0
FDR	Indicator that model is underpredicting flood extent	$FDR = \frac{FD}{WET_{target}}$	$0 - 1$	0

4.4.3. Mask

The focus will be on predicting flood water depth in initially dry areas. Evaluation and classification metrics will be calculated using a mask over the river area. Water depths in the river vary greatly and have a large impact on the accuracy of the model. As no information on the water level before the TC is available, another approach is taken to create the river mask.

The river mask is created by identifying areas that are consistently wet across multiple simulations with minimal rainfall and low flood extent, combined with topographic constraints. Specifically, the water level maps from the sample with the lowest rainfall and the one with the fewest flooded cells are each compared against the bed elevation (DEM), and only areas where the water depth exceeds a small threshold of 0.1 meters are retained. Using both simulations ensures a reliable estimation of the river area.

A connectivity filter is applied to maintain a continuous river-like mask. This approach balances physical constraints with robustness, avoiding isolated patches or local noise in the DEM or water surface data. This results in the mask as shown in Figure 4.7, and subsequently in the area that we are interested in.

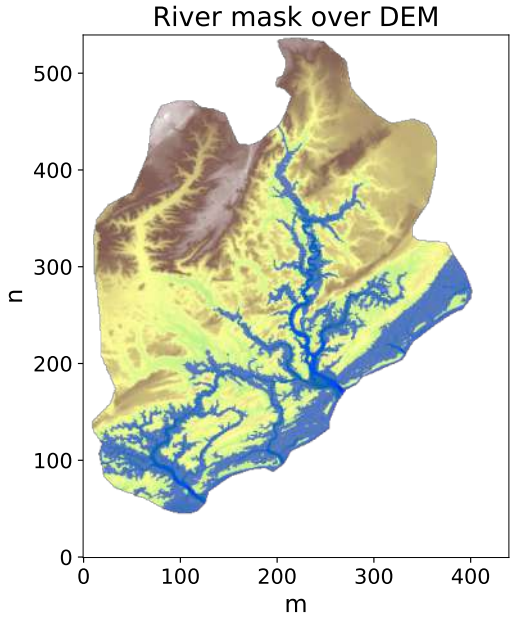

Figure 4.7: DEM area covered by a blue river mask

4.4.4. Spatial Analysis

For every simulation, three areas will be highlighted for analysis. A coastal area, a riverine area, and an inland area. The location of these areas in the DEM are illustrated in Figure 4.8.

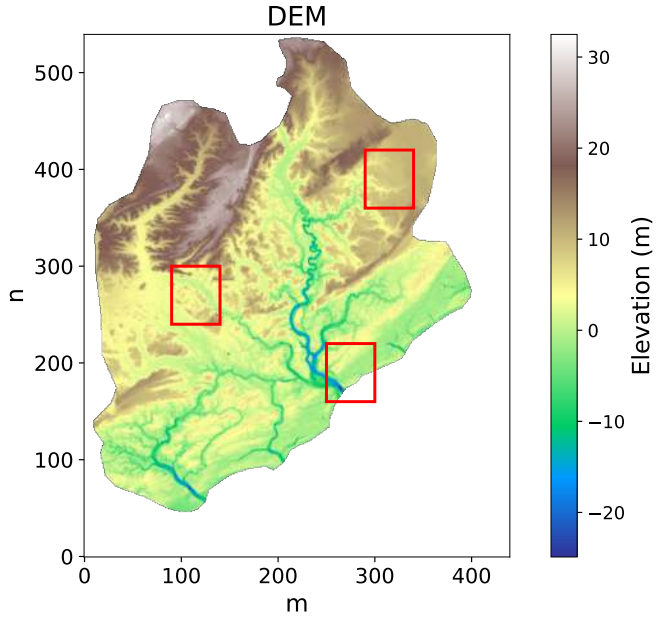

Figure 4.8: DEM containing the zoomed in areas of the river, coast and mountain region

Zooming in on those areas of interest results in the plots displayed in Figure 4.9.

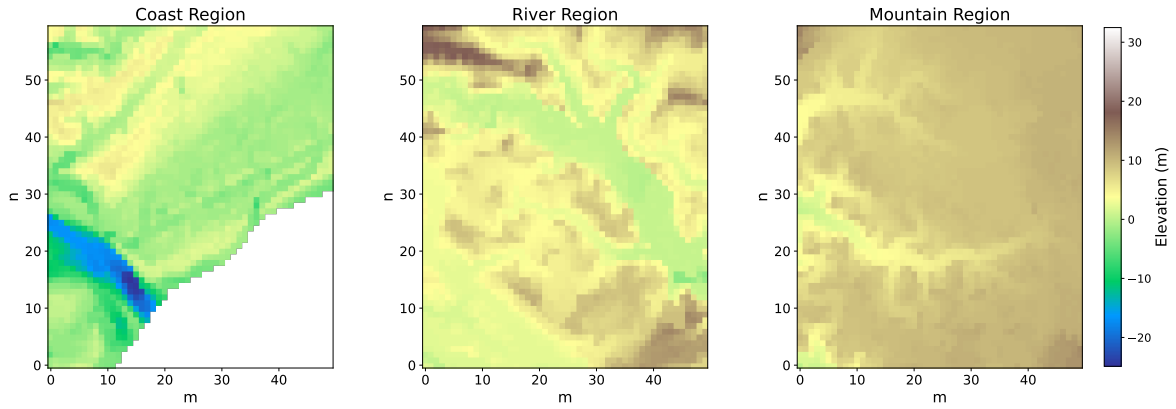

Figure 4.9: Regions river, coast and mountain zoomed in on from the DEM

As is visible in the figure, the mask over the river is present. The river can only be analyzed using a river branch that is not inundated by low floods, but only during more extreme flood cases. It is a low-lying area compared to its surroundings and an extension of an existing river branch.

In the coastal region, many areas are inundated, thus not visible. A small barrier island in front of the coastline has been taken into account to visualize the results in this part of the model.

The mountainous area is taken in the eastern side of the domain. Here, the mountains are relatively equal and a catchment can be seen in the contours of the elevation of the DEM.

4.4.5. Computational Time

The second important performance metric is computational time. The deep learning model will be compared against the physics-based model based on speed. As the incoming TC can change its track at rapid speeds, a shorter computational time is an advantage.

First, we look at the model run time of SFINCS. Two models are created, one overall and one detailed model. The overall model is needed because the output serves as input boundary conditions for the detailed model. The runtime for the overall model is about 20 seconds, whereas the runtime for the detailed model is 110 seconds.

Secondly, the deep learning model is assessed. One can look at the time it takes to make a prediction or at the training time of the model. During the simulations, the model will be optimized with the main focus on reducing training time. The prediction time will be in the order of seconds and is therefore not considered further, as the variation between model configurations is expected to be minimal.

4.5. Case Study

To assess the performance and applicability of the proposed deep learning framework, a case study is carried out for the Charleston region, located in South Carolina. The physics-based model SFINCS will be compared with the deep learning model based on accuracy and computational time.

A single historical TC track is selected from the available database to serve as a baseline event. This track was selected due to its medium to high values in the parameter range, and its track, passing close to the case study area. With the reference track as baseline, 50 synthetic cyclone tracks are generated using CHT Cyclones, a dedicated storm synthesis tool designed to create realistic TC scenarios.

The wind field is modeled using the parametric wind profile based on research of G. J. Holland et al. (2010), while the wind-pressure relationship follows from the formulation by G. Holland (2008). In addition, precipitation is included using empirical relationships such as in the Interagency Performance Evaluation Task Force (IPET) (2006), where a precipitation factor of 4 is used to multiply precipitation rates to obtain representative values for the rainfall.

The synthetic events are designed to represent a plausible range of storm intensities, landfall locations, and trajectories relevant to the Charleston area. An overview of the generated tracks is given in Figure 4.10.

Most TCs make landfall in Florida, USA, but samples that make landfall in Charleston are also captured in the dataset.

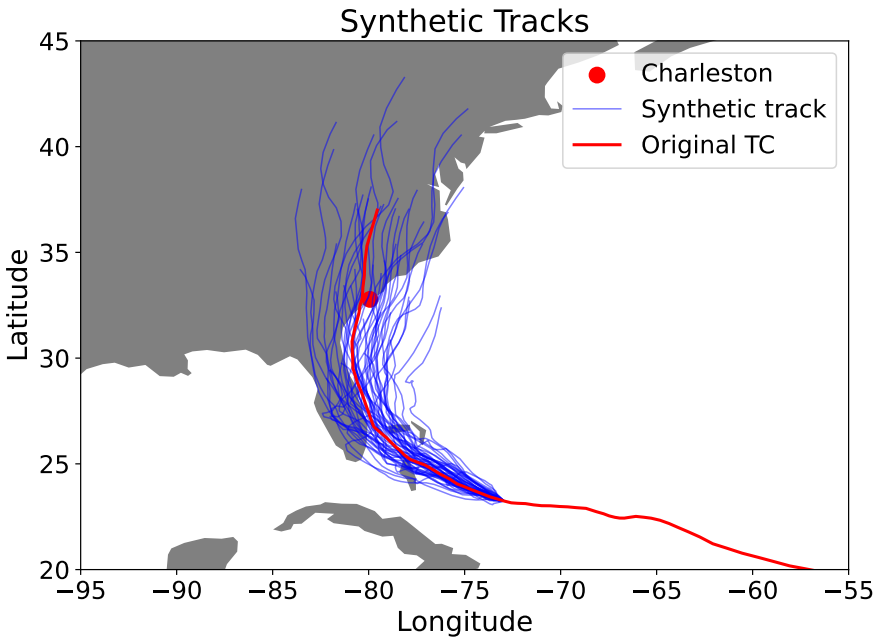

Figure 4.10: Synthetic tracks generated with the CHT Cyclones tool. The TC that is used to sample from is shown in red.

First the SFINCS overall model is run. The output of the overall model serves as input for the detailed model. Computed water levels are used as boundary conditions for the detailed model. This way, large scale water level effects are transferred to a local scale. Lastly, the detailed simulations are done, to produce the output of the SFINCS model that will be compared to the DL model.

From the spiderweb, the cumulative precipitation and the maximum wind speed can be computed. The water level at the estuary mouth is taken from the detailed SFINCS model. An overview of the parameters in the dataset range is given in Figure 4.11 for the parameters cumulative precipitation and maximum wind speed. The figure shows that a large cluster of the samples from the casestudy have little precipitation. The wind speed shows similarity to the distribution of the entire dataset. One sample contains a sample with a wind speed value above the maximum wind speed observed in the entire dataset. This sample will be analyzed in more detail and possibly taken out of the dataset.

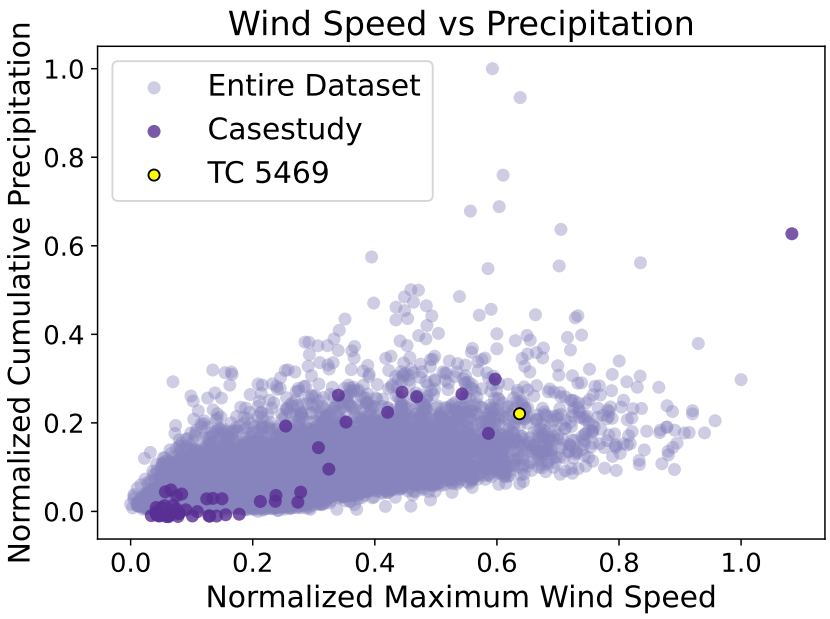

Figure 4.11: Samples from casestudy presented in the entire dataset range

The relation between precipitation and water depth in estuary mouth are shown in Figure 4.12. The computed water depths by SFINCS cluster in the lower range of the parameter range. A clear upward trend is observed in the relation between precipitation and water depth, where the values show a higher precipitation rate compared to the general pattern in the data. A possible explanation can be found in the precipitation factor that multiplies the precipitation rates by a factor of 4.

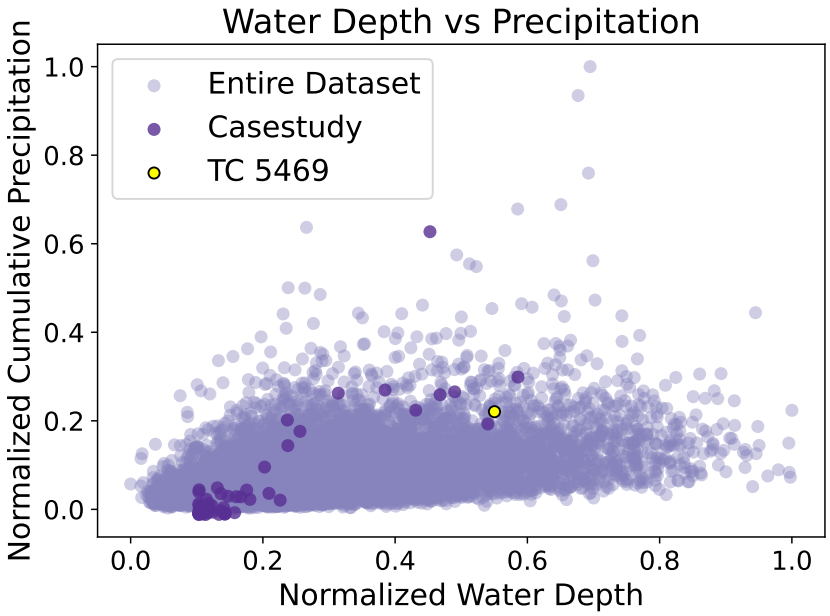

Figure 4.12: Samples from casestudy presented in the entire dataset range

The output of SFINCS will be compared to the output of the DL model. Computed water levels are compared on flood prediction accuracy, flood coverage, and computational time. Since the computational time for the deep learning model is in the order of seconds, the computation speed will differ greatly per simulation.

Therefore, the entire test set will be passed to the deep learning model, so the computational time of one sample can be computed as the average of the total computational time.

4.6. Technical specifications

Simulations will be run on the Deltares H7 Cluster. The cluster has two Graphics Processing Units (GPUs) of the model NVIDIA H100 (80GB VRAM), with both GPUs containing 64 cores, a memory of 512 GB RAM, and a memory bandwidth of 2 TB/s.

5

Results

In this chapter, results of the simulations will be described. Predictions will be compared with the targets and evaluated based on accuracy and computational expense. Firstly, an overview of the performance metrics is given. Secondly, the model performance is analyzed with 2D-visualizations. Lastly, the simulation time of the trained models will be evaluated.

5.1. Dataset sample size

5.1.1. Performance metrics

Table 5.1 shows the results of the simulations performed with varying input training size. Three models have been trained with the model name including the number of samples the model has been trained and validated on, in ascending order 200, 1000, and 5000 samples.

Looking at the regressive performance of the models, the model trained with 1000 samples has the lowest RMSE (0.092 m) and MAE (0.061 m), indicating that it achieves the most accurate predictions in terms of numerical error. Increasing the number of training samples results in similar results, whereas a smaller number of samples results in a higher regression error.

Table 5.1: Performance comparison for models with different training dataset sizes

Model	RMSE (m)	MAE (m)	CSI	Skill	FWR	FDR
Samples_200	0.105	0.072	0.534	0.631	0.070	0.266
Samples_1000	**0.092**	**0.061**	**0.585**	0.682	0.058	**0.239**
Samples_5000	0.093	0.062	0.576	**0.689**	**0.051**	0.277

The classification performance of the models can be evaluated by looking at the last four metrics in the table. The performance is about the same for the models trained on 1000 and 5000 samples. The model trained with 200 samples performs worse on the CSI (about 0.05 lower) and skill (about 0.05 lower) metrics compared to the other two models.

The model trained with 1000 samples achieves the best overall balance between regression accuracy and classification skill. While 200 samples lead to underperformance, increasing the dataset to 5000 samples does not lead to improvements in the metrics. Based on the observed metric scores, the optimal number of training and validation samples appears to be between 200 and 5000, with current results suggesting a likely optimum between 200 and 1000. However, additional testing would be needed to confirm whether the true optimum lies closer to 1000 or elsewhere within this range.

5.1.2. Spatial analysis

Figure 5.1 illustrates the mean RMSE in all test samples for the three different models. In the plots, the DEM and the riverine area have been masked, as we are not interested in predicting the flood depth in these regions. Errors have been binned into bins of 0.1 meters. A depth below 0.1 meters will be plotted with white, as the model will always contain a certain offset and this will not be taken into account. The

color bar limit goes up to 1.0 m to enhance differences, but there some (less than ten) pixels with an error of 1.5 m for the model trained with 200 samples.

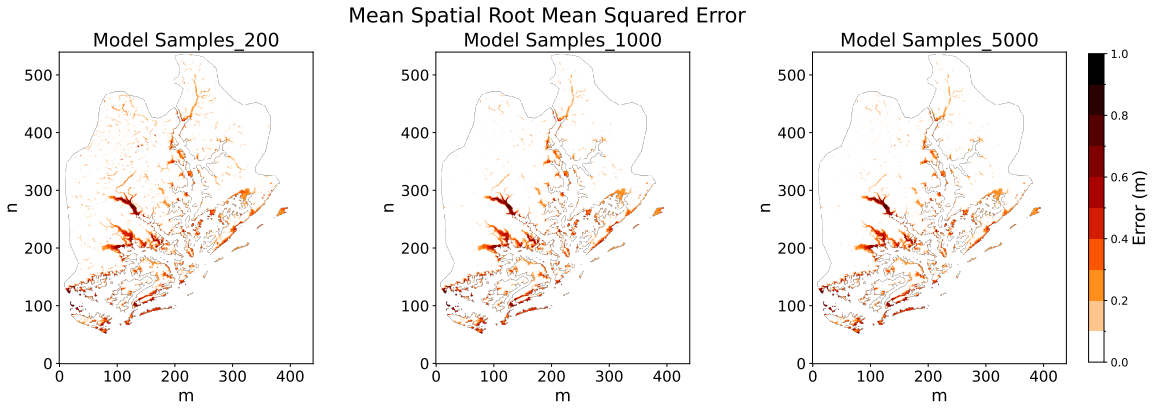

Figure 5.1: Mean RMSE of inundation depth in all test samples for models trained with a different sample size

The model trained on 200 samples shows the largest spatial extent of prediction errors. Higher lying areas in the north side of the domain show errors, while these areas do not show an error for the other two models. Models trained on 1000 and 5000 samples have the same spatial error pattern.

In general, it can be seen that the coastal areas have a high error, as well as the west side of the domain near the river area. The water depth in these areas is expected to be high and contains a lot of variance compared to those of the inland areas.

Zooming in on the specified detailed areas, differences between models become more pronounced. Plotting the zoomed-in area of the mean error will show differences of up to 20 centimeters. However, these differences are hard to interpret visually. Therefore, another approach is taken, presented in Figure 5.2, where the differences between models are plotted. This facilitates easy comparison, also when the magnitude of the errors is comparable for the models. Looking at the river area, the model trained on 1000 samples shows the largest error in the river, as can be seen by the red area in the middle of the plot.

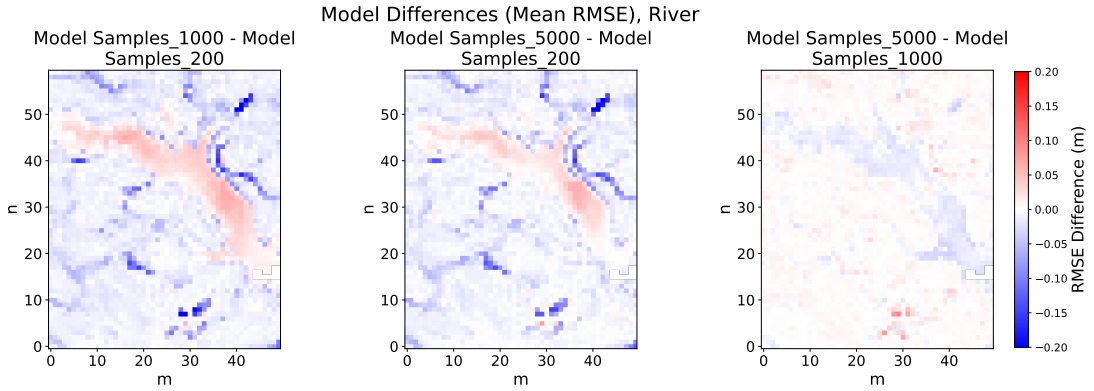

Figure 5.2: Mean RMSE differences in predicted inundation depth across models trained with different sample sizes (river area)

In the coastal region, in Figure 5.3, the models show more or less similar performance. Models trained with 1000 and 5000 samples have similar performance, indicated by the light color of the plot when comparing these models. A handful of pixels show a dark blue color in the plot on the left and in the center, indicating that the model trained with 200 samples has a higher error in these regions.

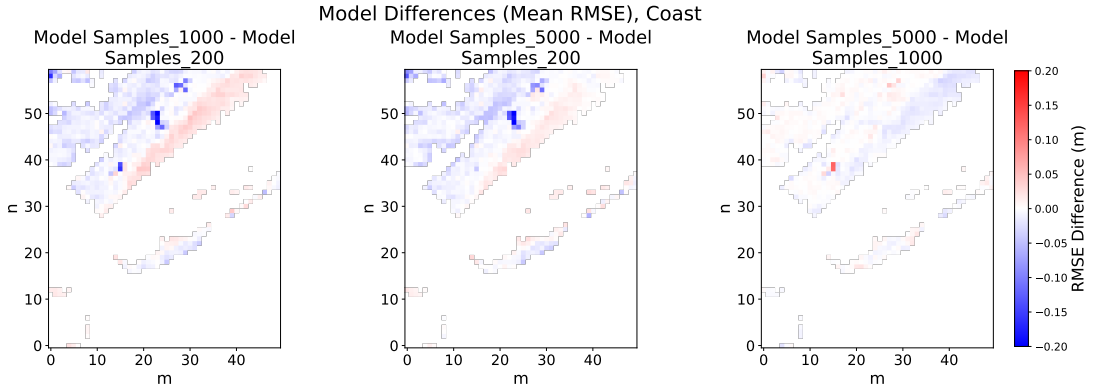

Figure 5.3: Mean RMSE differences in predicted inundation depth across models trained with different sample sizes (coast area)

More errors can be seen outside the river area in the model trained with 200 samples compared to the other two models, as can be seen in the mountain region in Figure 5.4.

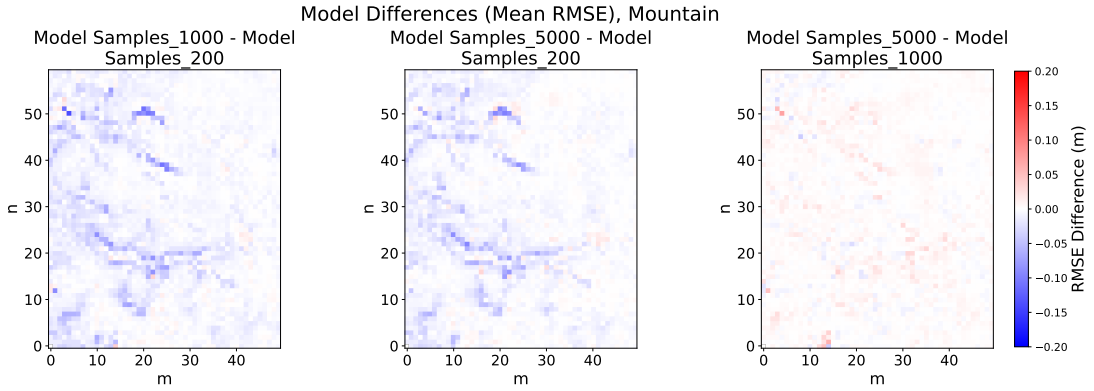

Figure 5.4: Mean RMSE differences in predicted inundation depth across models trained with different sample sizes (mountain area)

5.1.3. Computational time

Increasing the training dataset size significantly increases the training time of the model, as shown in Table 5.2, making it less efficient.

Table 5.2: Average model training times for different training dataset sizes

Model	Training Time [s]
Samples_200	274
Samples_1000	1348
Samples_5000	5436

Using only 200 samples results in poorer predictions, confirming that a small training dataset is insufficient for reliable flood modeling. An optimum with regard to accuracy and computational efficiency in number of training samples can likely be found between 200 and 1000 samples. In this research, we will not explore the optimal number of samples for both accuracy and computational time, but we will use the best performing model from the simulations. Thus, we select 1000 training samples for upcoming simulations as it provides the best trade-off between accuracy and efficiency.

5.2. Input parameters

5.2.1. Performance metrics

Results of the simulations on configurations of the input parameters are shown in Table 5.3. Models are named after the parameters that are used as input for the model. The model named 'All parameters' contains all three parameters: precipitation, wind speed, and water depth.

Looking at the regression error metrics (RMSE and MAE), the model with precipitation, wind speed, and water depth performs best. Flood water depths are predicted with the lowest RMSE (0.069 m) and MAE (0.046 m) compared to other models. Other models perform reasonably well on the regression metrics, especially the model that includes both precipitation and water depth. Including the wind speed as a model parameter seems to reduce the regression error by 0.02 m. However, the wind speed parameter on its own does not show a good performance of the model.

Table 5.3: Performance comparison for models with varying input configuration

Model	RMSE (m)	MAE (m)	CSI	Skill	FWR	FDR
All parameters	**0.069**	**0.046**	**0.651**	**0.700**	**0.052**	0.166
Precipitation	0.097	0.070	0.544	0.426	0.093	**0.152**
Wind speed	0.105	0.078	0.505	0.148	0.113	0.159
Water depth	0.087	0.062	0.489	0.146	0.097	0.167
Wind speed, precipitation	0.092	0.066	0.578	0.535	0.084	0.153
Wind speed, water depth	0.082	0.057	0.556	0.330	0.082	0.166
Precipitation, water depth	0.071	0.048	**0.651**	0.586	0.061	0.159

In predicting flood extent, the model containing all parameters and the precipitation-water depth model have the best performance, indicated by a high CSI. This suggests that leaving the wind speed out of the model input does not influence the flood extent prediction. For the remaining models, the CSI has a value of around 0.5, indicating a moderate performance. A CSI value of 0.5 implies that only half of the total flooded area, both in prediction and target, is correctly predicted.

Furthermore, the skill parameter for the model including all parameters is the highest, indicating that the predictive capabilities of the model are the highest. A model containing precipitation as input parameter is expected to perform better on this metric, as high cumulative precipitation leads to rivers and patches of land being flooded. Adding the water depth and wind speed as input parameters significantly improves the skill, because the model can predict flooding in the coastal area and close to the rivers. Combining water depth, wind speed, and precipitation leads to both inland and coastal flooding being predicted with a high certainty.

The FWR and FDR show whether the model is under- or overpredicting the number of flooded and dry cells. In general, the FDR appears to be around 0.15, with a margin of 0.02 for all models. The model with all parameters as input has the lowest False Wet Rate, meaning that a low number of cells is predicted to be flooded, whereas these cells are actually dry. The low FWR is counteracted by a high False Dry Rate, where the model predicts a large number of dry cells that are wet in the target. However, the FDR remains comparable to that of the other models.

5.2.2. Spatial analysis

Model names are abbreviated as 'PRECIP', 'WIND', and 'WL', and a combination of parameters is illustrated by an underscore.

To assess the predictive power of each individual input parameter, models trained separately using only one input: precipitation, wind, or water depth, are analyzed. Figure 5.5 illustrates the prediction, target, and resulting difference for the model trained solely on precipitation data, evaluated on a randomly selected test sample. This particular sample is characterized by low parameter values and a relatively small flood extent. The model substantially overpredicts flood depths, particularly near the river and coastal zones, as indicated by the red areas in the difference map.

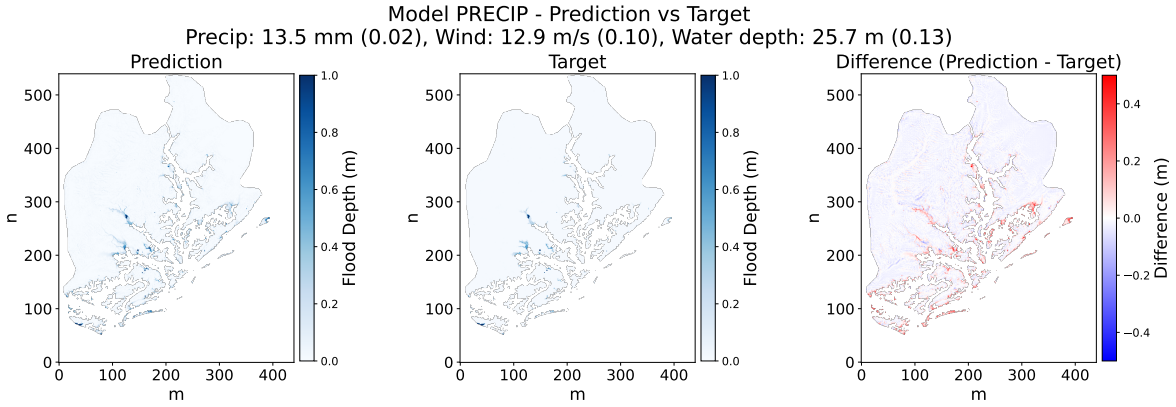

Figure 5.5: Prediction, target, and difference maps for a randomly selected test sample using the model trained with precipitation input only. Corresponding values of the input parameters are listed in the figure title with the normalized value between brackets.

To investigate these spatial patterns more closely, the previously specified regions (river, coast, and mountain) are examined in detail. Figure 5.6 spatial distribution of the error for the three single-input models in the river region. The precipitation-based model significantly overpredicts flood depths in the river area, with errors reaching up to 0.4 meters. The wind-based model shows a mixed pattern, balancing under- and overpredictions with several local extremes. In contrast, the water-depth-based model consistently underpredicts flooding, with localized errors up to 0.5 meters.

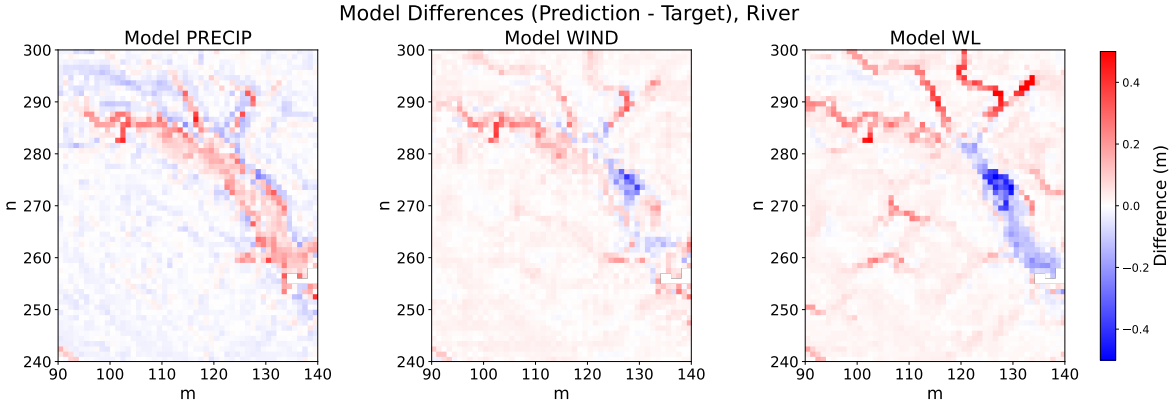

Figure 5.6: Prediction error in the river region for models trained with individual input parameters: precipitation (left), wind (middle), and water depth (right). The parameter values of the corresponding sample are: Precipitation: 13.5 mm (0.00), Wind speed: 12.9 m/s (0.10), Water depth: 25.7 m (0.13).

Figure 5.7 focuses on model performance in the coastal region. Among the three models, the precipitation-based model shows the largest errors, with significant overprediction near the boundary between land and sea. In comparison, the wind- and water-depth-based models show similar overall performance, but the water-depth-based model achieves slightly better accuracy.

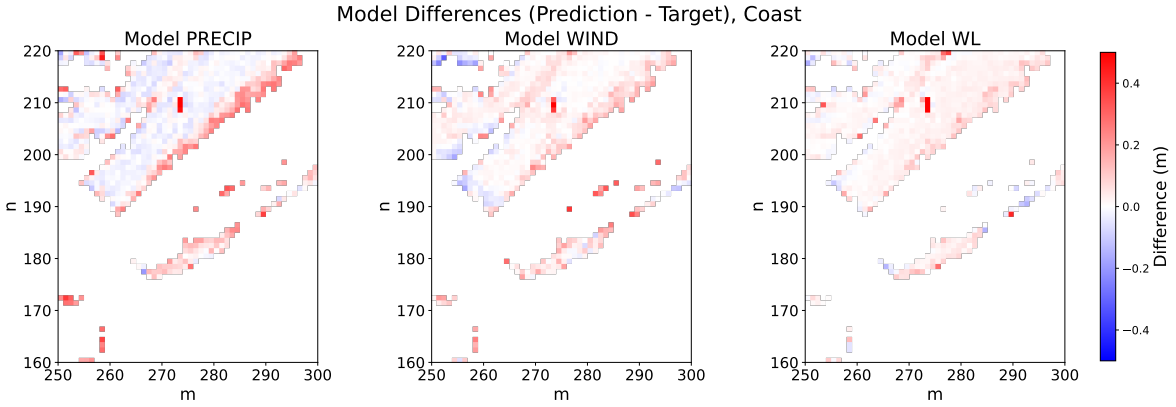

Figure 5.7: Prediction error in the coastal region for models trained with individual input parameters: precipitation (left), wind (middle), and water depth (right). The parameter values of the corresponding sample are: Precipitation: 13.5 mm (0.00), Wind speed: 12.9 m/s (0.10), Water depth: 25.7 m (0.13).

Figure 5.8 presents the prediction errors in the mountainous region. Here, the precipitation-based model shows minor underprediction. The wind-based model exhibits a slight overprediction, and notably, it begins to capture topographic contours of the DEM, shown by the river-like path of the errors. These patterns become more pronounced in the water depth model, which shows more overprediction, indicated by the darker red. Also, the low-lying areas become more notable in the water-depth-based model.

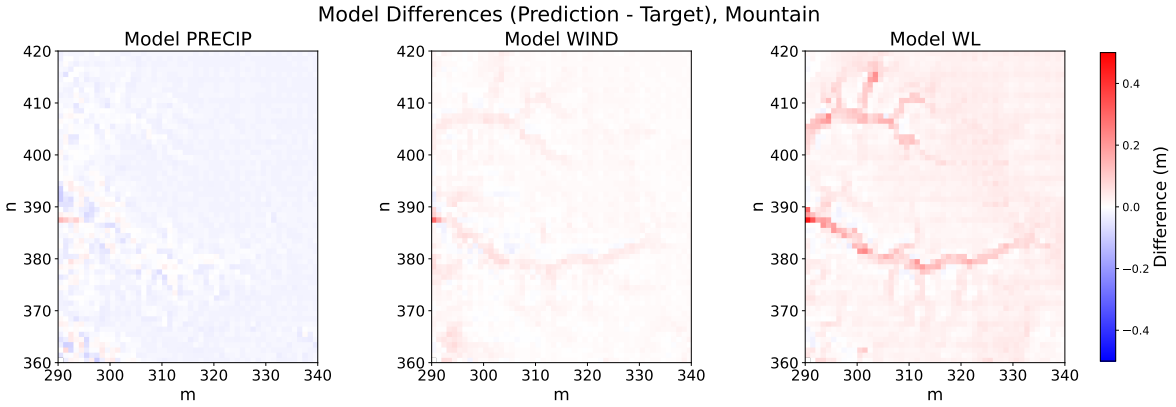

Figure 5.8: Prediction error in the mountain region for models trained with individual input parameters: precipitation (left), wind (middle), and water depth (right). The parameter values of the corresponding sample are: Precipitation: 13.5 mm (0.00), Wind speed: 12.9 m/s (0.10), Water depth: 25.7 m (0.13).

The mean spatial error across all test samples is plotted in Figure 5.9. It is insightful to analyze how different input parameters influence the spatial distribution of error. For the model trained on precipitation data, the inland error is low, but the error near the river and coast is significant. The model trained on wind and water depth data shows better performance on the coast and river areas. However, due to precipitation not being taken into account, high-lying areas show a larger error compared to the precipitation model. The model including all parameters performs best on the combination. Few error locations can be seen in the higher-lying areas, as well as good performance in the coastal en river regions. This highlights how different input variables contribute to model accuracy in different parts of the domain.

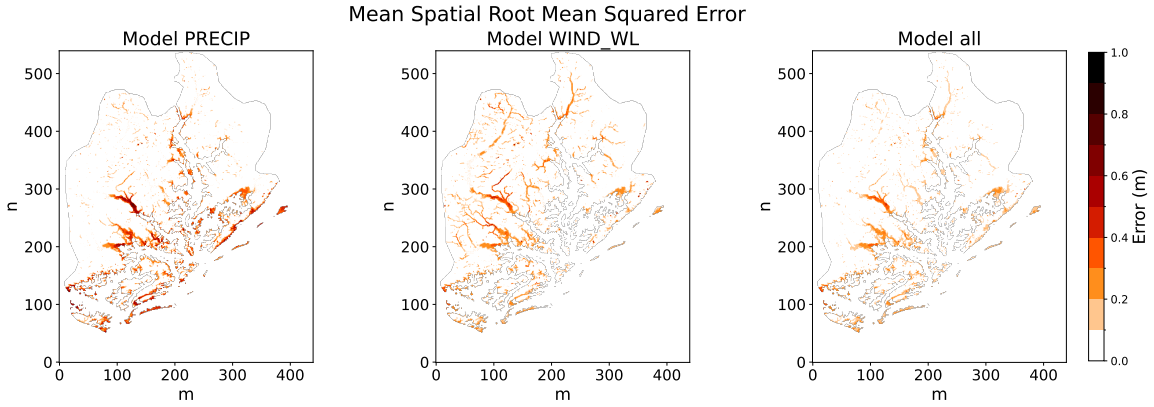

Figure 5.9: Mean RMSE of inundation depth in all test samples for models trained with a different input parameter configuration

Figure 5.10 illustrates the differences between the models, zoomed in on the river area. It can be seen that the model trained on precipitation shows a large difference in error in the river. The model has no information on the water depth, whereas the other models have this information available. The full-parameter model and the model trained on wind and water level show similar performance in the river, illustrated by the light color in the difference plot.

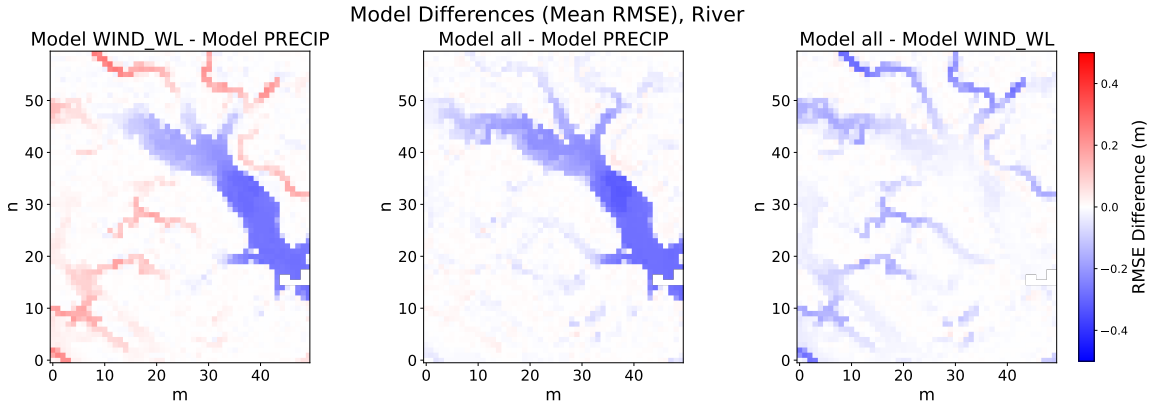

Figure 5.10: Differences between models with varying input parameters in the river area

In Figure 5.11, it can be seen that the prediction of coastal inundation highly depends on the wind speed and water depth input parameters. The models containing both parameters show similar performance, while the model trained on precipitation has a large offset in the coastal region.

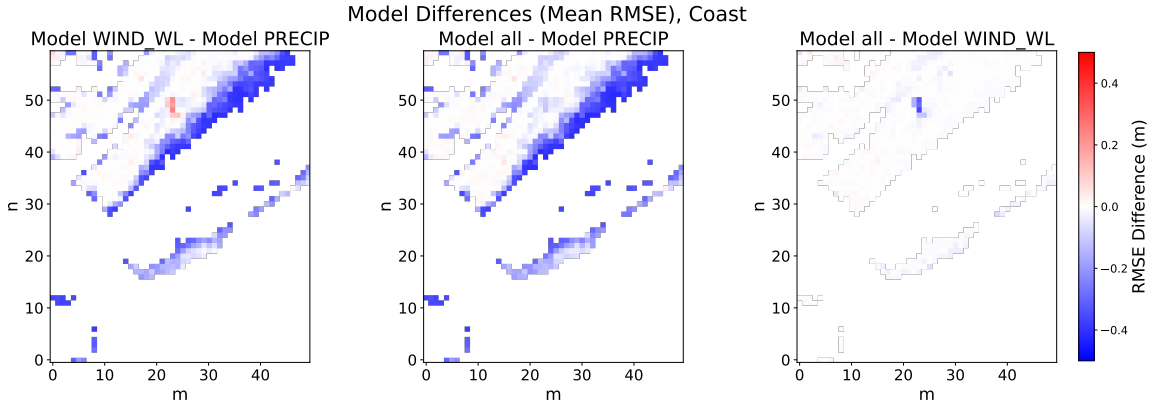

Figure 5.11: Differences between models with varying input parameters in the coastal area

In the mountain region, the precipitation parameter is of great importance. Figure 5.12 shows that there is little difference between the model trained on all input parameters and the model trained using only precipitation.

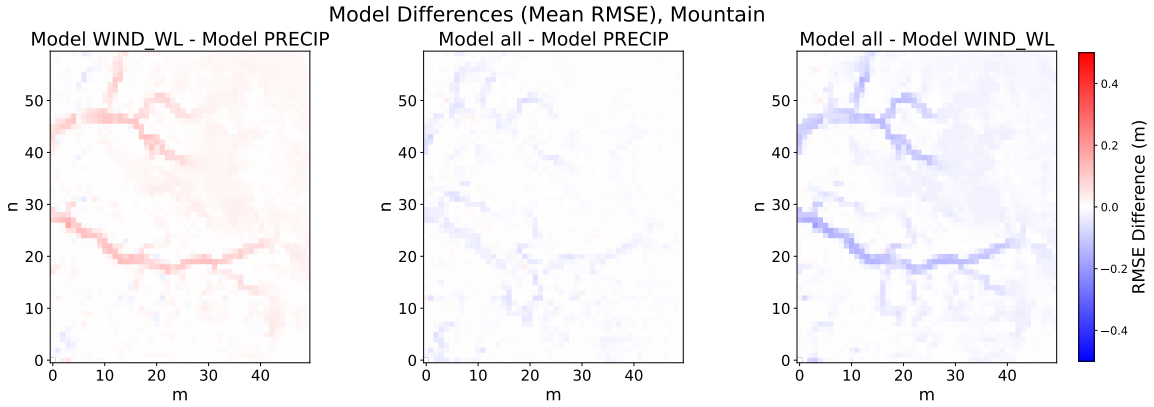

Figure 5.12: Differences between models with varying input parameters in the mountain area

5.2.3. Computational time

Considering computational speed, reducing the number of input parameters results in a faster model. Therefore, a model with fewer inputs is preferred over a model with more inputs.

The model that used all parameters; precipitation, wind speed, and water depth, has the best performance on the metrics. This reflects the combined strengths of both the precipitation and wind and water depth variables. Precipitation contributes to accurate predictions of inland flooding, while wind speed and water depth improve predictions in coastal and riverine areas.

Although a model with fewer inputs is generally preferred for efficiency, the improvement in performance from using the wind speed as an additional input outweighs the increase in computation time. Therefore, for future simulations, models that incorporate precipitation, wind speed, and water depth are selected to ensure accuracy and robustness.

5.3. Dataset sampling

5.3.1. Performance metrics

The results in Table 5.4 highlight differences in model performance based on the sampling strategy used to select samples for training and validation data. In terms of RMSE, the model trained with K-means selected samples outperforms the model trained on randomly selected samples. For the MAE, it is the other way around, which means that training with randomly selected samples results in a better average

prediction, while selecting samples with the K-means method improves model performance at extreme parameter values.

Table 5.4: Performance comparison of models trained on data from K-means and random sampling methods

Model	RMSE (m)	MAE (m)	CSI	Skill	FWR	FDR
Random	0.064	**0.038**	**0.710**	**0.744**	**0.037**	0.167
KMeans	**0.062**	0.040	0.697	0.741	0.044	**0.145**

The model trained on randomly selected samples performs slightly better on the classification metrics than the model trained with K-means selected samples. The CSI value (0.774) is higher than for the K-means model (0.765), as well as the skill metric (0.744 versus 0.740). One downside of the model trained on randomly selected samples is the higher FDR, indicating that the model tends to underpredict the flood extent compared to the K-means model.

5.3.2. Spatial analysis

The two plots of the mean RMSE in Figure 5.13 illustrate similar error patterns in both models. The river area and the coastal zone on the west side of the domain show the highest error patterns with values around 0.4 meters.

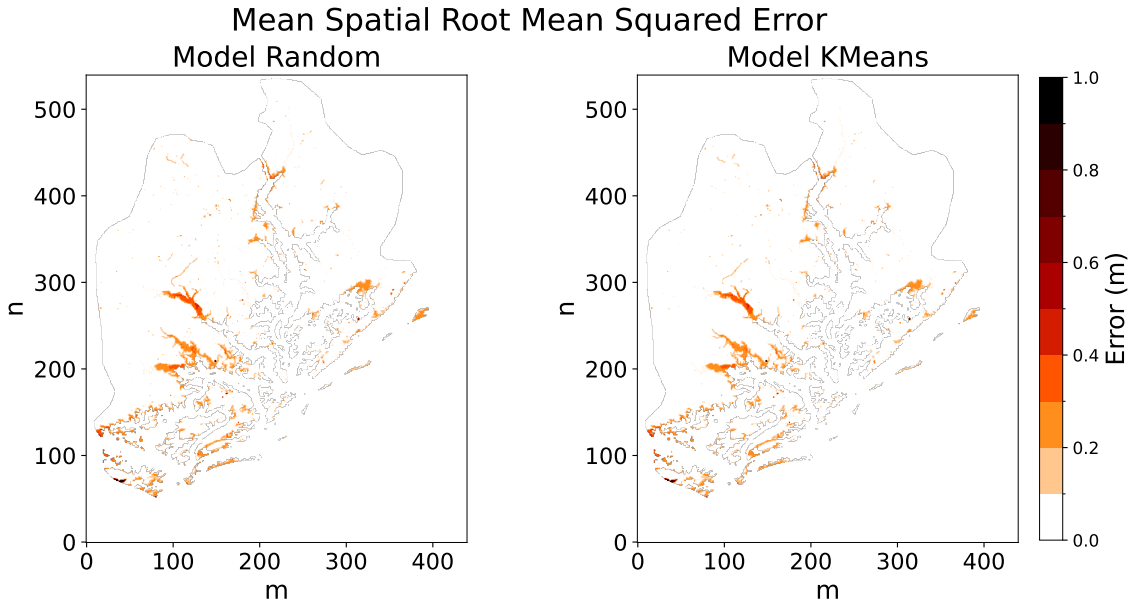

Figure 5.13: Mean RMSE of inundation depth in all test samples for models trained with a differently sampled dataset

Zooming in to the specified areas, as presented in Figure 5.14, differences are not pronounced. A handful of pixels has a difference of 0.1 meters, while most of the differences between the models are less than 0.1 meters. The model with randomly selected samples performs slightly worse in the river and coastal region, illustrated by the blue color of these areas. In the mountain region, the models are quite similar and no clear distinction between models can be made.

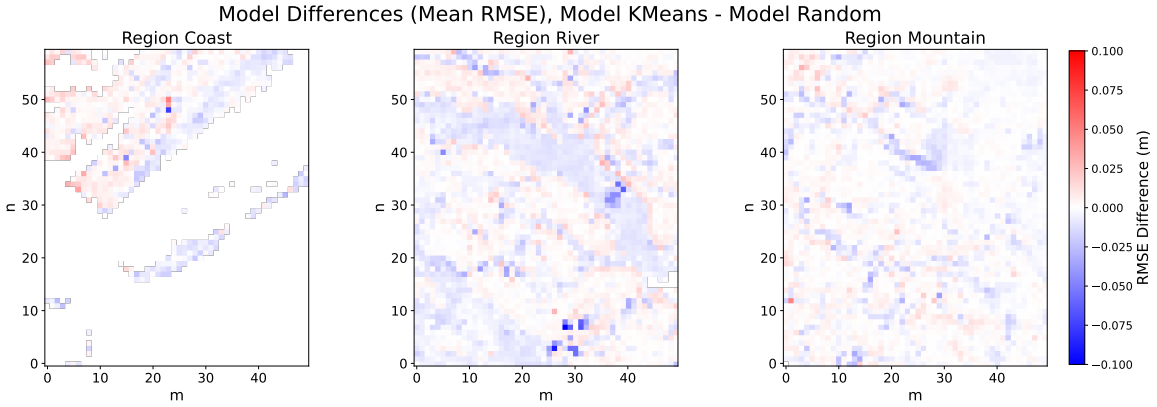

Figure 5.14: Differences between models trained with a differently sampled dataset in three regions, coast, river and mountain

Figure 5.15 presents a scatter plot of the prediction error (RMSE) across all samples, with respect to both precipitation and water depth values. In general, low parameter values are associated with low error values, indicating that the models perform well on low to mild flood conditions. As the parameter values increase, the error also tends to rise. This trend is particularly pronounced for the water depth parameter, suggesting greater model uncertainty in predicting more severe flood scenarios. Notably, the model trained on randomly selected samples shows higher error magnitudes on average than the model trained using K-means selected samples, especially in the higher ranges of the parameter space. This reinforces earlier findings that K-means sampling enhances the model's performance on extreme flood events.

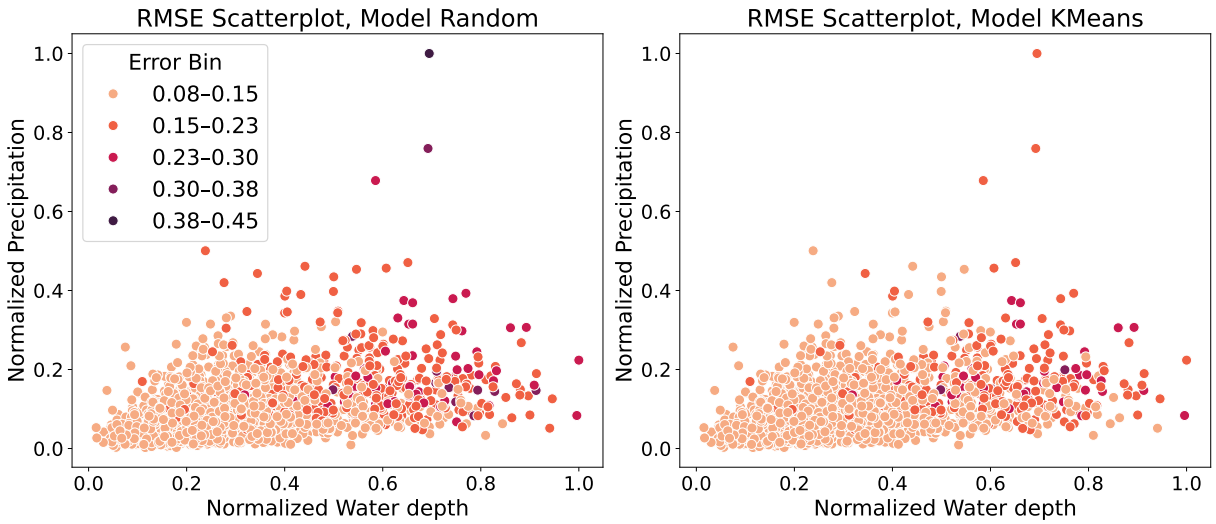

Figure 5.15: Scatter plot of samples in the precipitation and water depth parameter space, with RMSE values indicated by color, for models trained using different sampling methods.

5.3.3. Computational time

From a computational standpoint, there is no measurable difference in training or computational time between the two sampling strategies. Both models are trained on datasets of equal size and dimensionality, with the same number of input features.

Flood forecasting requires not only small flood cases, but also extreme flood events to be modeled with great accuracy. The model's ability to capture rare but severe cases is more critical than minimizing average error. Further simulations are performed with samples selected by the K-means sampling method.

5.4. Mask

In the training and validation phase of the process, a choice can be made to vary the area that needs to be evaluated by the model. The first case covers the entire Charleston area, where the second case excludes areas below sea level or, in other words, all rivers. Since we are interested in areas that are initially not flooded, the already flooded areas can be excluded, to improve the model performance on the dry cells. The other way around, the river depth might contain useful information for the prediction of flood water depth close to the river.

5.4.1. Performance metrics

The results in Table 5.5 show that applying a river mask improves the model performance across all metrics. Specifically the regression metrics, that drop from 0.068 to 0.062 for the RMSE and from 0.044 to 0.040 for the MAE. In terms of classification, the CSI increases from 0.645 to 0.710 and the skill score increases from 0.705 to 0.752, making the model better at classifying the flood extent.

Table 5.5: Performance comparison for models with and without river masking

Model	RMSE (m)	MAE (m)	CSI	Skill	FWR	FDR
no_mask	0.068	0.044	0.645	0.705	0.057	0.166
river_mask	**0.062**	**0.040**	**0.710**	**0.752**	**0.041**	**0.149**

Overall, these results suggest that incorporating a river mask helps the model focus on relevant regions, as the metrics are only calculated for the initially dry cells. Possible noise from initially wet areas is reduced and the model allows for better generalization.

5.4.2. Spatial analysis

The two models illustrate clear differences in RMSE error pattern, as illustrated in Figure 5.16. Especially inland, the model trained without a mask performs worse than the model trained with a river mask. Also, the coastal region on the eastern side of the domain contains more errors for the model without the mask.

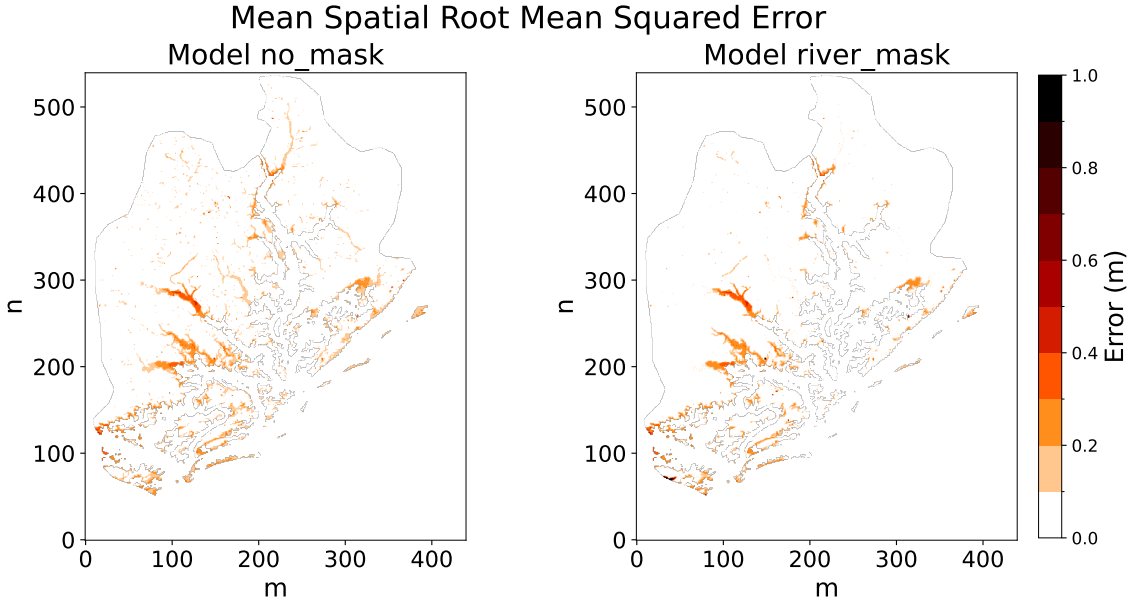

Figure 5.16: Mean RMSE of inundation depth in all test samples for models trained with and without a river mask

Zooming in to the specified areas, as presented in Figure 5.17, differences between models become more

pronounced. In general, the plots are colored blue, indicating that the model trained with the river mask outperforms the model trained without the mask. A few pixels range up to an error difference of 0.15 meters; however, most of the differences are in the range of 0 to 0.1 meters.

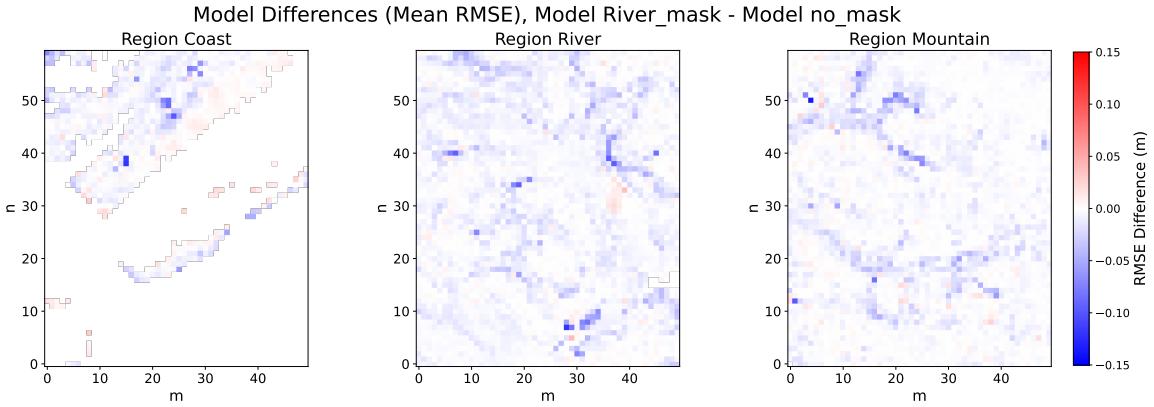

Figure 5.17: Differences between models trained with and without a river mask in three regions, coast, river and mountain

5.4.3. Computational time

Implementing a river mask in the model does not noticeably affect computational time. While the model continues to generate predictions for all spatial locations, the mask is only applied during the evaluation phase to exclude non-relevant areas from the loss calculation. Consequently, the model architecture and forward pass remain unchanged.

Similarly, the prediction time is unaffected, as the masking operation is applied post hoc and does not alter the computational load of the computational process itself.

Combining the results from the performance metrics with the spatial analysis and computational differences, future models are trained and evaluated with the river mask included.

5.5. Loss function

The model names reflect the specific loss functions used during training and validation. Two variants of a multitask loss function were explored, combining regression and classification objectives with different weights. In one configuration, the regression loss is prioritized with a weighting of 0.7, while the classification loss contributes 0.3. In the second configuration, this weighting is reversed, giving more influence to the classification component.

5.5.1. Performance metrics

The results in Table 5.6 clearly show that the RMSE model performs best on regression, with the lowest RMSE (0.062) and lowest MAE (0.040). The multitask model with 70% regression and 30% classification weighting shows slightly degraded regression performance, but maintains a similar CSI and achieves a modest improvement in false flood prediction metrics. Increasing the weight for the classification and decreasing the weight for regression results in better performance in classification task, resulting in an increase in regression error. This suggests that assigning a large value to classification weight in a multitask loss function severely compromises the accuracy of predicted flood depths.

The pinball loss shows slightly higher regression errors and a lower CSI (0.681) and lower skill (0.538), likely due to its emphasis on quantile regression. A low False Dry Rate illustrates that this model has a bias towards overprediction. This could make it useful in uncertainty estimation or extreme event prediction, but it underperforms compared to RMSE and multitask loss functions for flood depth accuracy.

In contrast, the BCE (Binary Cross Entropy) loss function results in high classification metrics (CSI = 0.767, skill = 0.764), but extremely high RMSE and MAE values. This suggests that a BCE loss function is not suitable for regression-based flood depth prediction and likely causes numerical instability due to its design

Table 5.6: Performance comparison of models trained with different loss functions

Model	RMSE (m)	MAE (m)	CSI	Skill	FWR	FDR
RMSE	**0.062**	**0.040**	0.710	0.752	0.041	0.149
MultiTask$_{70reg/30class}$	0.069	0.049	0.709	0.746	0.038	0.138
MultiTask$_{30reg/70class}$	4.296	4.204	**0.778**	**0.764**	0.027	0.118
Pinball	0.072	0.045	0.681	0.538	0.072	**0.072**
BCE	308.781	280.404	0.767	**0.764**	**0.025**	0.142

being intended for classification, not continuous targets.

5.5.2. Spatial analysis

To further evaluate model performance, spatial prediction maps were created for each model trained on a different loss function. The RMSE model, given in Figure 5.18 shows the most balanced prediction, together with the multitask model in Figure 5.19, with areas of blue (underprediction) concentrated in the south of the domain and red (overprediction) in the north, suggesting relatively small and spatially distributed errors. This aligns well with its strong quantitative performances.

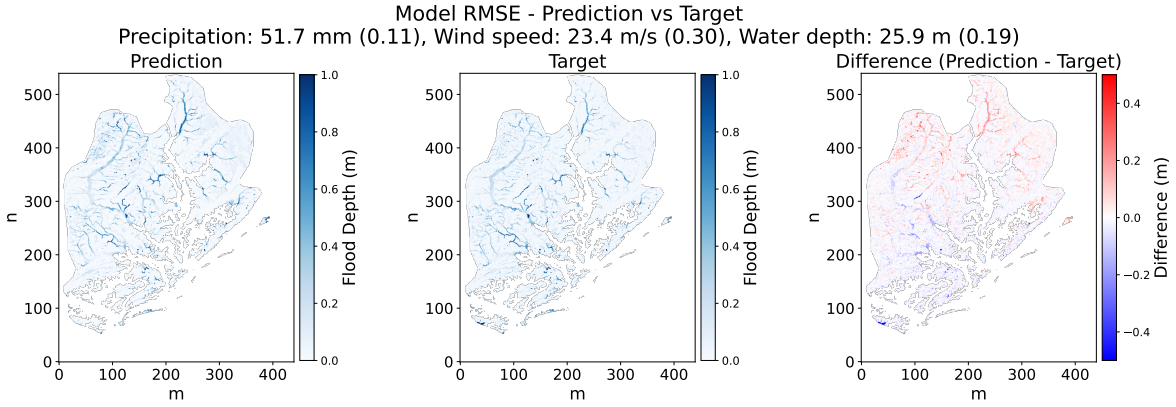

Figure 5.18: Prediction, target, and difference maps for a randomly selected test sample using the model trained with RMSE loss function

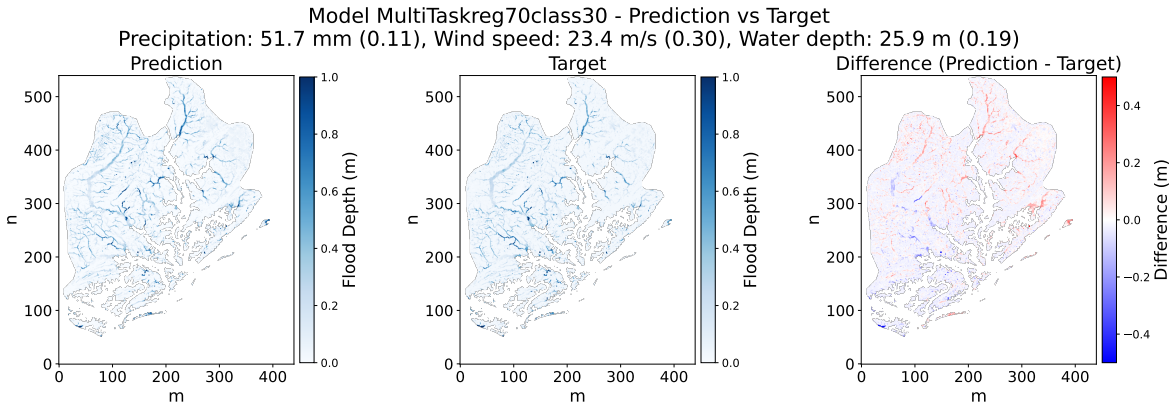

Figure 5.19: Prediction, target, and difference maps for a randomly selected test sample using the model trained with MultiTask$_{70reg/30class}$ loss function

In contrast, the pinball model produced maps with extensive dark red regions, indicating systematic

overprediction, presented in Figure 5.20. This aligns with its higher RMSE and MAE scores and reflects its tendency to bias toward the upper quantiles of the target distribution. The prediction is conservative and far away from the ground truth, which lead to unreliable predictions. Localized regions with underprediction can be seen in blue. These areas are also present in the difference plots of the other two models, but less pronounced in the pinball model.

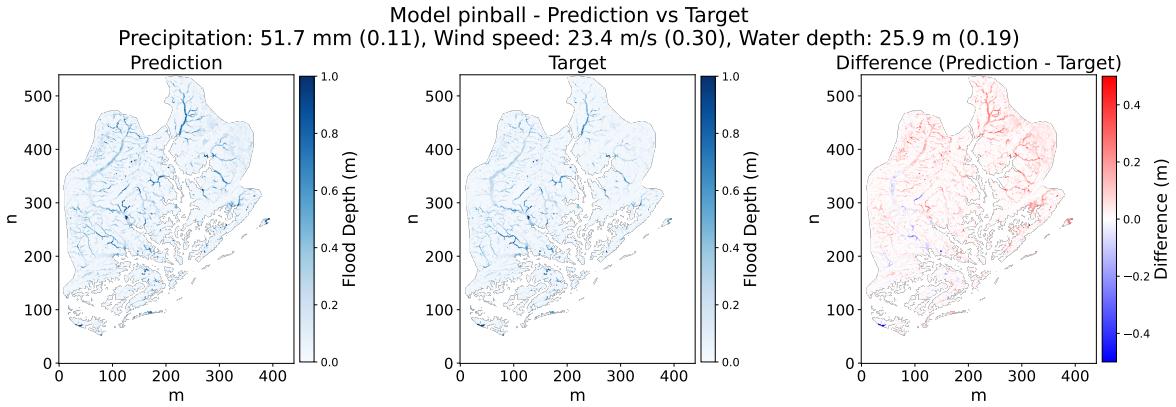

Figure 5.20: Prediction, target, and difference maps for a randomly selected test sample using the model trained with Pinball loss function

Models trained with a classification (dominant) loss function have relatively high regression errors, indicating that the prediction of continuous flood depths is less accurate when classification is prioritized during training. As a result, evaluating these models based on regression metrics does not provide an insightful assessment of their performance. Instead, it is more informative to analyze their output in terms of classification accuracy by comparing the predicted and target flood extent maps.

Figure 5.21 shows the same sample used previously to evaluate the model's regression performance, but the outputs are now classified into binary flood extents. Cells are labeled as flooded if the predicted or target water depth exceeds 0.1 meters. In the prediction and target visualizations, blue indicates flooded cells, and white indicates non-flooded cells. The difference plot highlights areas of agreement and disagreement between the prediction and the target:

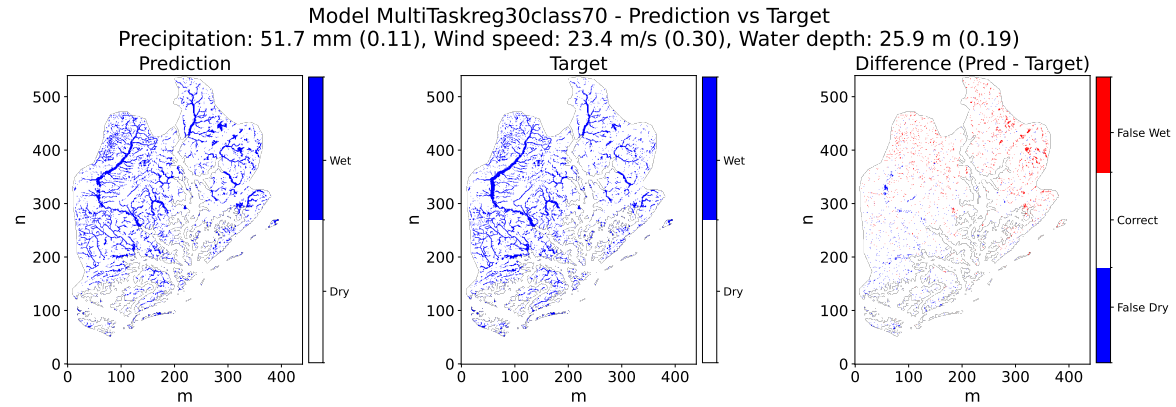

Figure 5.21: Prediction, target, and difference maps for a randomly selected test sample using the model trained with MultiTask$_{30reg/70class}$ loss function. Flooded areas (water depth greater than 0.1 meters) are illustrated by a blue and non-flooded areas by a white color.

- White areas: Agreement between prediction and target (both flooded or both non-flooded).
- Red areas: False Wet — the model predicts flooding, but the target cell is dry.
- Blue areas: False Dry — the target cell is flooded, but the model prediction dry.

The spatial patterns in the difference plot are consistent with those observed in Figure 5.18, which showed the regressive difference in predicted and target water depth. The northern region of the domain again shows signs of overprediction, with more predicted flooded cells than observed in the target. Similarly, the southern region illustrates mainly blue cells, confirming the underprediction of flood extent in that area.

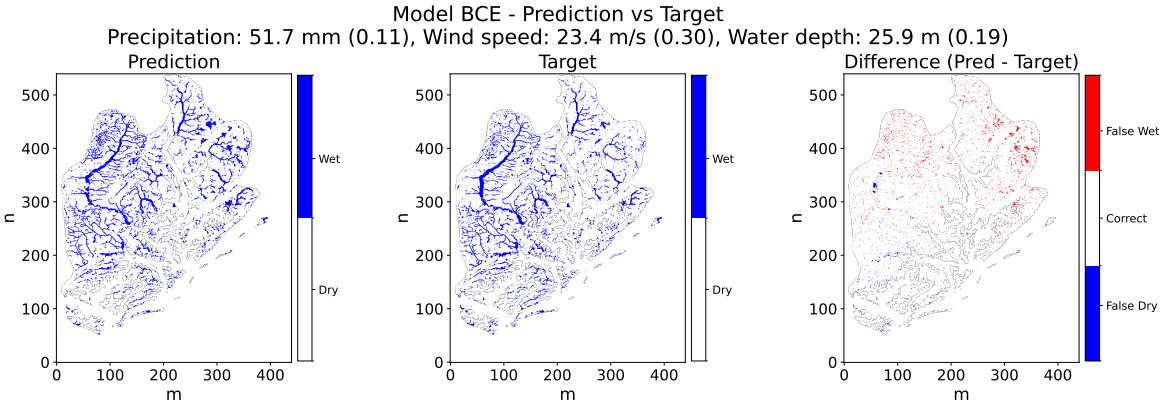

Figure 5.22: Prediction, target, and difference maps for a randomly selected test sample using the model trained with BCE loss function. Flooded areas (water depth greater than 0.1 meters) are illustrated by a blue and non-flooded areas by a white color.

The mean spatial RMSE across all test samples is plotted in Figure 5.23. Models trained with the RMSE and MultiTask loss function have similar performance. The model containing the pinball loss function, plotted on the right, illustrates a higher error, indicated by a darker shade of red, compared to the other two models.

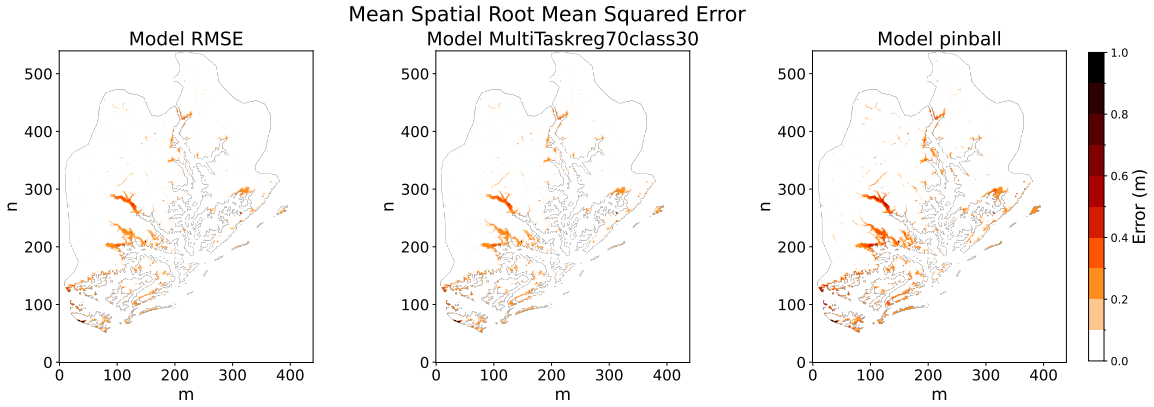

Figure 5.23: Mean RMSE of inundation depth in all test samples for models trained with a different loss function

Figure 5.24 presents a comparison of the model performance, zoomed in on the river area. It can be seen that the models trained with the RMSE and MultiTask loss function show little to no differences in the river area. The model trained with the pinball loss function has a large error in predicting the water depth in the river, illustrated by the dark red color in the difference plot.

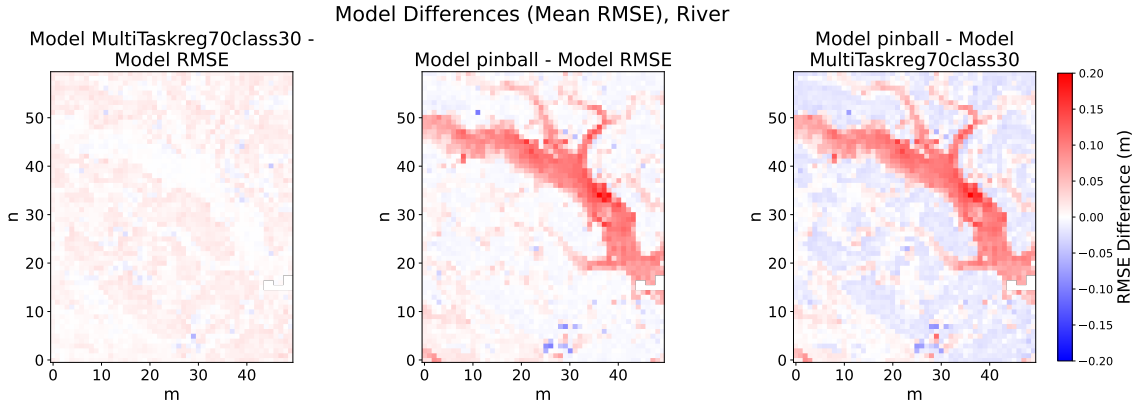

Figure 5.24: Differences between models with varying loss function in the river area

Figure 5.25 shows a similar comparison in the coastal region, where the differences between models are smaller but still present. The pinball model again demonstrates a higher error, particularly along the shoreline and lower-lying coastal zones.

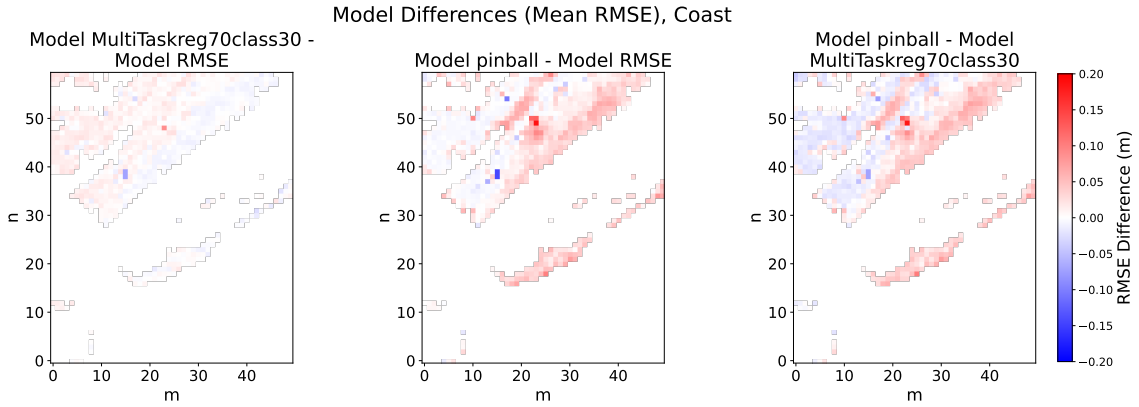

Figure 5.25: Differences between models with varying loss function in the coast area

In the mountain region, models show similar performance, indicated by the light colors in Figure 5.26. The order of the error difference between models is lower than 0.1 meters.

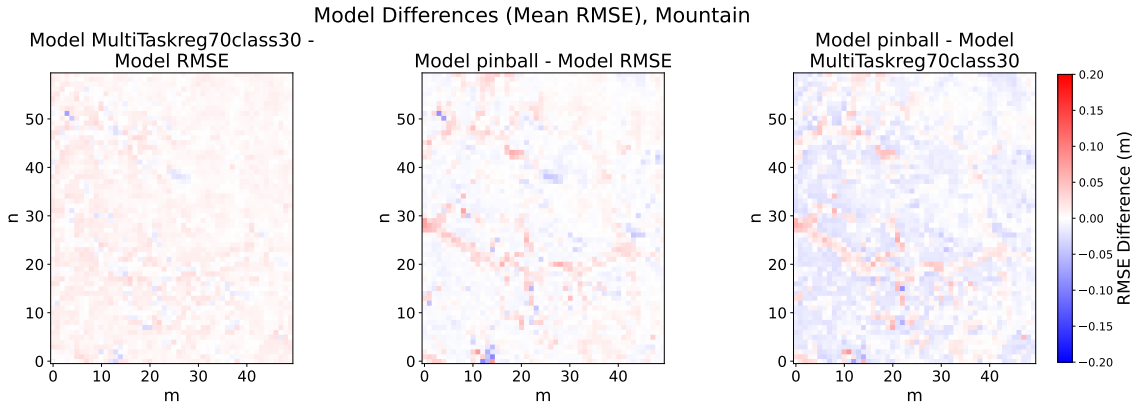

Figure 5.26: Differences between models with varying loss function in the mountain area

5.5.3. Computational time

The results in Table 5.7 show that model training time varies depending on the loss function used. In particular, the multitask models required significantly more training time compared to the single-task models. This increase is expected due to the loss function involving both a regression and classification loss simultaneously. The models using RMSE and pinball loss, which are regression-based, trained faster than the MultiTask models, and slightly faster than the BCE model.

Table 5.7: Training time (in seconds) for each model variant.

Model	Training time [s]
RMSE	1257
MultiTask$_{70reg/30class}$	1835
MultiTask$_{30reg/70class}$	1836
Pinball	1296
BCE	1305

Despite differences during training, prediction time remains unaffected by the choice of loss function. Since the loss function is only used during training to compute gradients and update model parameters, it plays no role during model prediction. As long as the model architecture and output structure remain unchanged, the time required to generate predictions is consistent across all trained models.

Based on the comparative analysis of model performance, the RMSE loss function was selected for the final model configuration. This choice is motivated by the fact that the problem at hand is a regressive problem, aiming to predict flood water depths rather than binary flood extent classifications. Among the regression-based loss functions evaluated, the RMSE model demonstrated the best overall performance in predicting water depths, while still maintaining reasonable classification accuracy when outputs were binarized. Further simulations include models that incorporate the RMSE loss function in the training and validation phase.

5.6. Network architecture

In this section, model names are abbreviated using the depth (d), number of filters in the first convolution (bf), and the kernel size (k).

5.6.1. Performance metrics

This section presents the results of simulations conducted with varying model configurations. Specifically, combinations of network depth (3, 4, or 5 layers), initial number of filters (8, 16, 32, or 64), and kernel size (3, 5, or 7) were tested. Results are sorted by kernel size, starting with results for kernel size 3 in Table 5.8, kernel size of 5 in Table 5.9, and a kernel size of 7 in Table 5.10.

In general, results indicate that a smaller kernel size (3) yields more consistent and reliable performance across all metrics, compared to those with larger kernels. As the kernel size increases, performance metrics tend to vary more widely, suggesting less stability in model behavior.

Among the models using a kernel size of 3, the architecture with a depth of 3 layers and 64 filters performs best. The regressive performance is slightly better than other models, with an RMSE of 0.054 and MAE of 0.032. In terms of classification, this model also outperforms most models, achieving the highest CSI value (0.827) and a high skill metric (0.753).

The results in Table 5.10 show that, as model complexity increases, through larger kernel sizes, deeper architectures, and higher filter counts, a consistent decline in CSI scores can be observed, together with negative skill scores. This performance degradation can be attributed to several factors. Larger kernels capture information over broader spatial regions, which diminishes the model's ability to capture small-scale topographic features.

Similarly, deep networks or those with a high number of filters are prone to overfitting, especially when trained on relatively small datasets. These models may memorize training patterns rather than generalize to unseen data, leading to poor performance on evaluation metrics. The increase in false alarms and

Table 5.8: Performance comparison for models with kernel size = 3

Model	RMSE (m)	MAE (m)	CSI	Skill	FWR	FDR
d3_bf8_k3	0.088	0.067	0.723	0.651	0.065	0.111
d3_bf16_k3	0.060	0.036	0.794	0.759	0.034	0.114
d3_bf32_k3	0.056	0.033	0.815	**0.765**	**0.031**	0.097
d3_bf64_k3	**0.054**	**0.032**	**0.827**	0.753	0.034	0.065
d4_bf8_k3	0.060	0.037	0.779	0.756	0.037	0.111
d4_bf16_k3	0.058	0.035	0.807	0.763	**0.031**	0.101
d4_bf32_k3	0.055	0.033	0.813	0.755	0.037	0.071
d4_bf64_k3	0.063	0.042	0.764	0.662	0.054	0.074
d5_bf8_k3	0.062	0.039	0.779	0.743	0.032	0.129
d5_bf16_k3	0.063	0.043	0.704	0.640	0.050	0.083
d5_bf32_k3	0.061	0.041	0.715	0.638	0.051	0.065
d5_bf64_k3	0.098	0.078	0.548	-0.013	0.129	**0.063**

Table 5.9: Performance comparison for models with kernel size = 5

Model	RMSE (m)	MAE (m)	CSI	Skill	FWR	FDR
d3_bf8_k5	0.070	0.048	0.742	0.695	0.055	0.117
d3_bf16_k5	0.058	0.034	0.803	**0.765**	0.033	0.101
d3_bf32_k5	0.082	0.061	0.659	0.458	0.088	0.063
d3_bf64_k5	0.105	0.077	0.733	-0.936	0.178	0.074
d4_bf8_k5	0.062	0.039	0.781	0.759	0.036	0.113
d4_bf16_k5	0.057	0.034	0.810	0.764	**0.030**	0.098
d4_bf32_k5	**0.054**	**0.032**	**0.829**	0.763	0.032	0.074
d4_bf64_k5	0.134	0.104	0.474	-1.447	0.311	0.020
d5_bf8_k5	0.062	0.041	0.776	0.741	0.045	0.091
d5_bf16_k5	0.104	0.082	0.549	-0.005	0.117	0.086
d5_bf32_k5	0.120	0.100	0.483	-0.439	0.192	0.056
d5_bf64_k5	0.178	0.147	0.266	-3.331	0.646	**0.010**

missed flood predictions in these cases reduces both the CSI and skill scores, suggesting that additional model complexity does not necessarily lead to improved performance.

Figure 5.27 presents the results for simulations on the network architecture in relation to the CSI scores. Among architectural variations, models with a kernel size of 3 consistently yield the highest CSI scores, typically ranging between 0.70 and 0.80, regardless of filter size or network depth. This suggests that smaller kernels are more effective in capturing localized spatial features within the digital elevation model (DEM), which are needed for accurate flood prediction.

Table 5.10: Performance comparison for models with kernel size = 7

Model	RMSE (m)	MAE (m)	CSI	Skill	FWR	FDR
d3_bf8_k7	0.071	0.048	0.707	0.636	0.071	0.101
d3_bf16_k7	0.080	0.060	**0.757**	**0.693**	**0.053**	0.090
d3_bf32_k7	**0.066**	0.047	0.702	0.655	0.054	0.082
d3_bf64_k7	0.145	0.115	0.424	-1.644	0.358	0.033
d4_bf8_k7	0.069	**0.045**	0.724	0.641	0.065	0.101
d4_bf16_k7	0.118	0.092	0.511	-0.156	0.175	0.055
d4_bf32_k7	0.114	0.092	0.505	-0.068	0.163	0.073
d4_bf64_k7	0.163	0.133	0.410	-0.961	0.266	0.070
d5_bf8_k7	0.111	0.087	0.520	0.188	0.141	0.103
d5_bf16_k7	0.115	0.092	0.469	-0.233	0.187	0.080
d5_bf32_k7	0.179	0.148	0.281	-3.327	0.617	0.006
d5_bf64_k7	0.227	0.196	0.200	-5.794	0.994	**0.002**

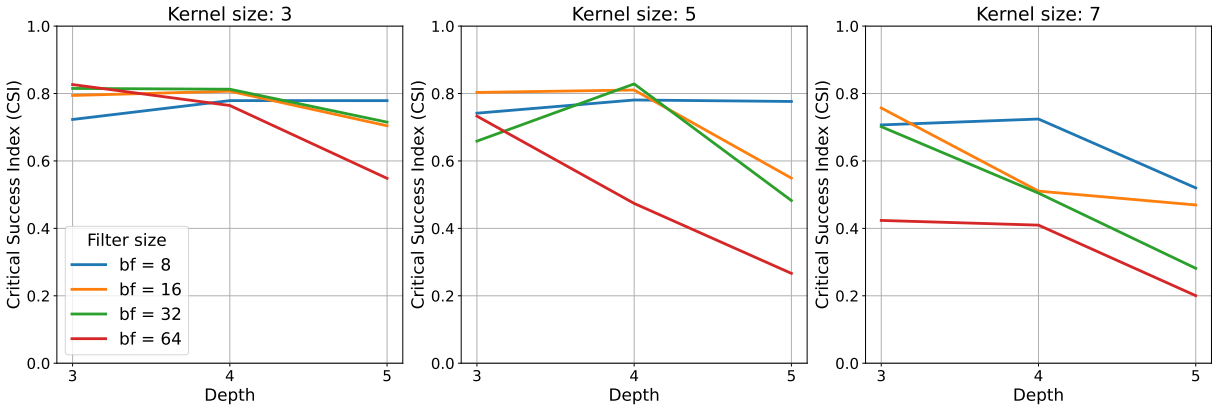

Figure 5.27: CSI scores computed for a water depth threshold of 0.1 m for varying combinations of depth, number of filters (bf) and kernel size

In contrast, increasing the kernel size results in a decrease in the CSI scores. Larger kernels aggregate information over broader spatial areas, which appears to reduce the model's ability to capture local topographic variations that influence flooding.

Additionally, model depth plays a role in performance. For architectures with more than 8 filters, an increase in depth generally corresponds to a decline in CSI. However, for models with 8 filters, performance continues to improve with depth, indicating a more favorable balance between model complexity and capacity.

The best-performing configuration overall consists of 32 filters, a kernel size of 5, and a depth of 4. This suggests that while a small kernel size is generally optimal, the combination of moderate depth and sufficient filter count can compensate for a slightly larger kernel under certain conditions.

The performance of the network architectures based on the RMSE metric is presented in Figure 5.28. Consistent with the findings from the CSI metric, models with a kernel size of 3 achieve the lowest RMSE values, typically ranging between 0.05 and 0.1, independent of filter size or network depth. This further supports the finding that smaller kernels are more effective at capturing localized spatial features in the DEM.

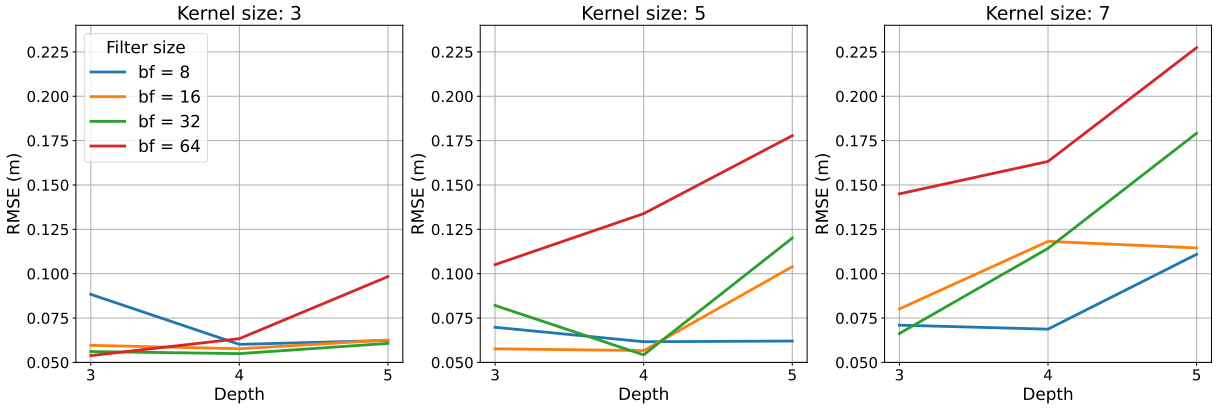

Figure 5.28: RMSE scores computed for varying combinations of depth, number of filters (bf) and kernel size

A clear positive correlation is observed between kernel size and RMSE: as the kernel size increases, the RMSE values tend to increase. This trend is particularly pronounced for kernel sizes of 7, leading to a substantial increase in RMSE, indicating a loss in accuracy with larger kernel sizes.

For kernels of sizes 3 and 5, increasing the depth from 3 to 4 results in improved performance, reflected by a reduction in RMSE. However, increasing the model depth to 5 layers again increases the RMSE, suggesting an optimal network depth beyond which additional layers may introduce overfitting or training instability.

Moreover, large kernels (size 7) perform progressively worse as the depth increases. Combining a large number of filters (64 filters) with a deeper architecture results in an increased RMSE, which may be due to excessive model complexity or overfitting.

The best-performing configuration overall consists of 32 filters, a kernel size of 5, and a depth of 4. This suggests that an optimal balance is found in a small kernel size, small depth, and large number of filters to capture relevant spatial patterns without making the model overly complex.

5.6.2. Spatial analysis

Kernel size comparison

A spatial comparison of models with varying kernel sizes is shown in Figure 5.29, where the depth is fixed at 4 and the number of filters at 16. The remaining models have different kernel sizes, so the effect of increasing the computational window through kernel size can be observed.

The results indicate strong and comparable performance for kernel sizes 3 and 5. However, there is a notable drop in performance at kernel size 7, with increases in regressive error and a decline in classification skill. The spatial RMSE patterns confirm that larger kernels may lead to over-smoothing or reduced sensitivity to localized features. Error patterns follow the pattern of low-lying areas, which are lost at large kernel sizes. While models with kernel sizes 3 and 5 maintain similar spatial error distributions, the model with kernel size 7 shows a clear increase in RMSE.

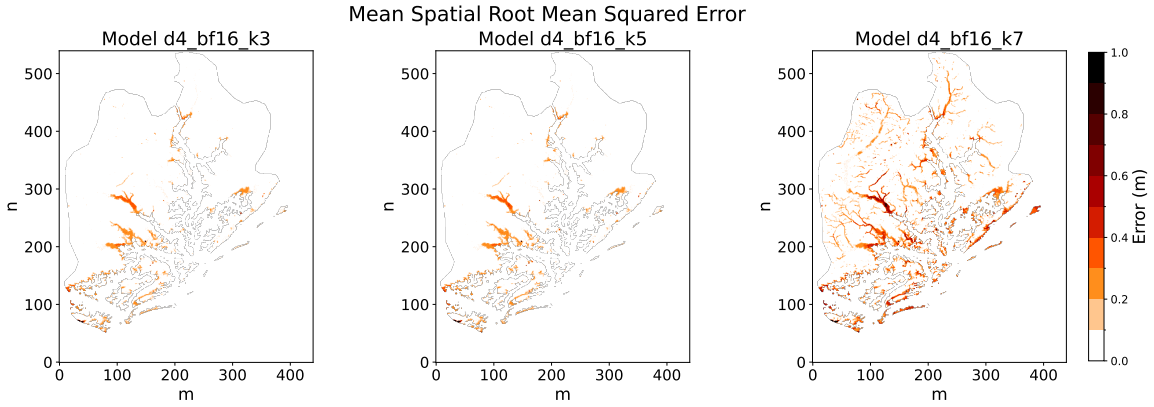

Figure 5.29: Mean RMSE of inundation depth in all test samples for models trained with varying kernel size

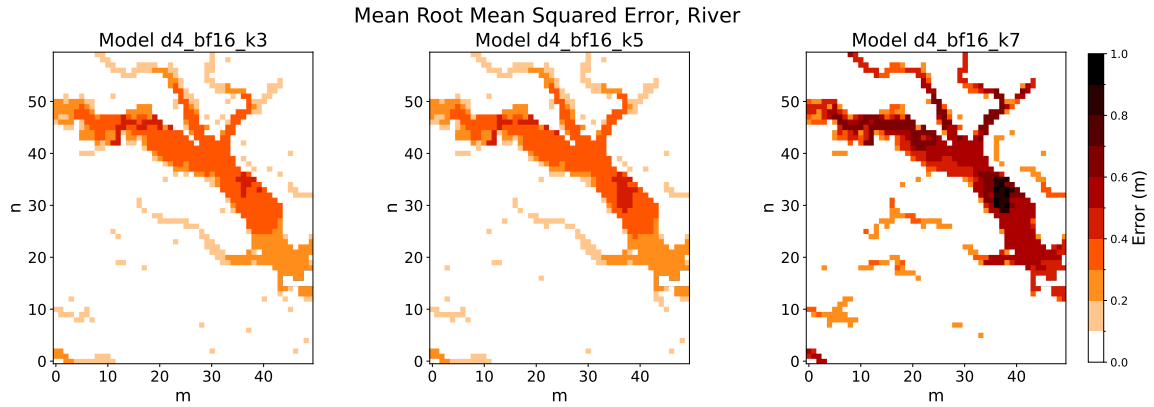

Figure 5.30: Mean RMSE of inundation depth in all test samples for models with varying kernel size in the river area

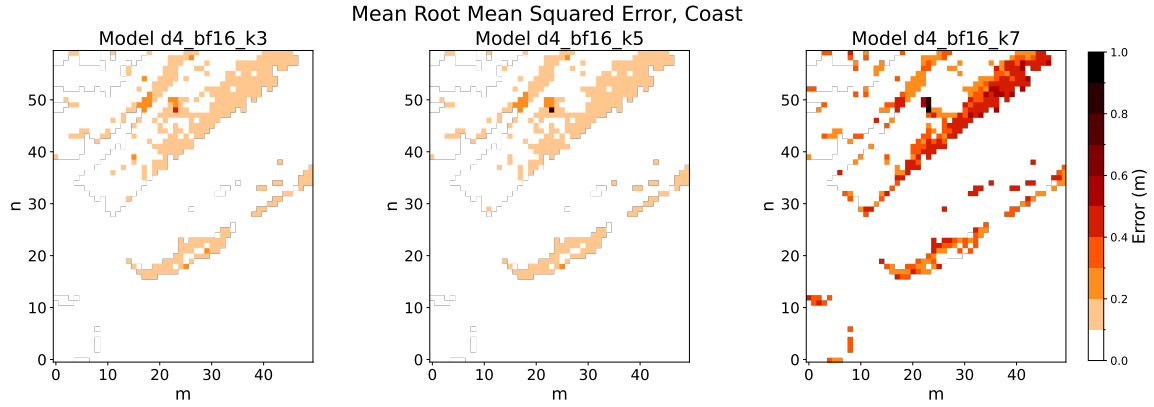

Figure 5.31: Mean RMSE of inundation depth in all test samples for models with varying kernel size in the coast area

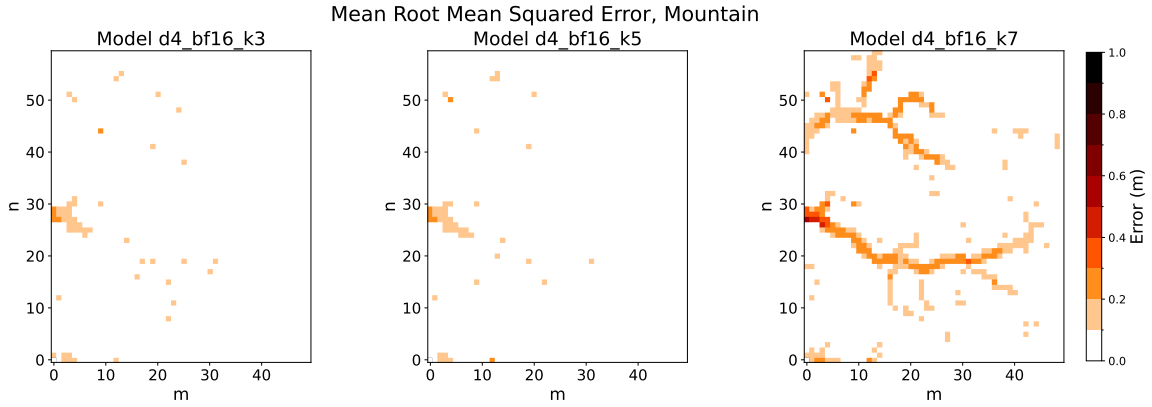

Figure 5.32: Mean RMSE of inundation depth in all test samples for models with varying kernel size in the mountain area

Depth comparison

A spatial comparison of models with varying depth is presented in Figure 5.33, with a fixed kernel size of 3 and number of filters set to 16. The selected models allow for isolating the effect of increasing network depth while keeping other parameters constant.

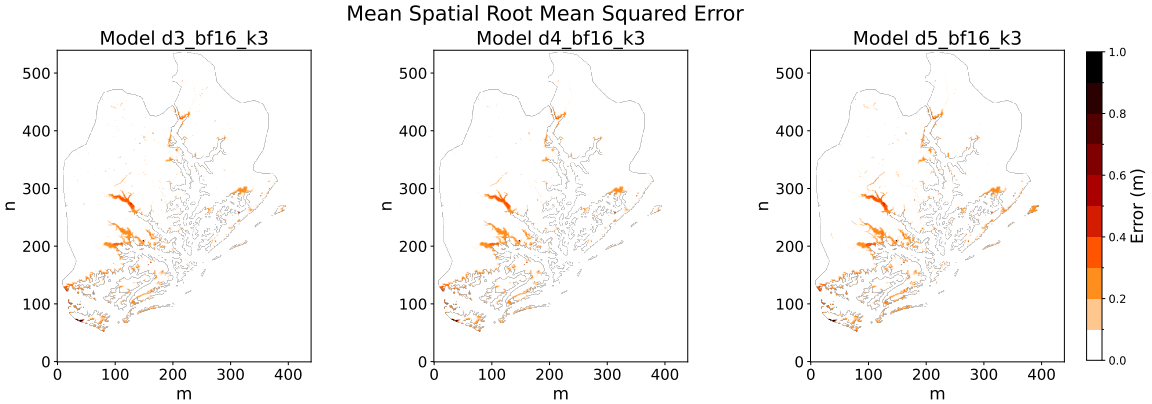

Figure 5.33: Mean RMSE of inundation depth in all test samples for models trained with varying depth

The results suggest that performance peaks at a depth of 4, with a slight degradation at a depth of 5 layers. While the global RMSE plot shows only minor differences between the models, the increase in architecture depth does not yield noticeable improvements in the regressive error.

A closer inspection of the river area, shown in Figure 5.34, reveals that the model with depth 5 has slightly higher errors compared to the shallower models. However, the difference is modest, with a value around 0.1 meters. Models with depths 3 and 4 illustrate similar performance in this region.

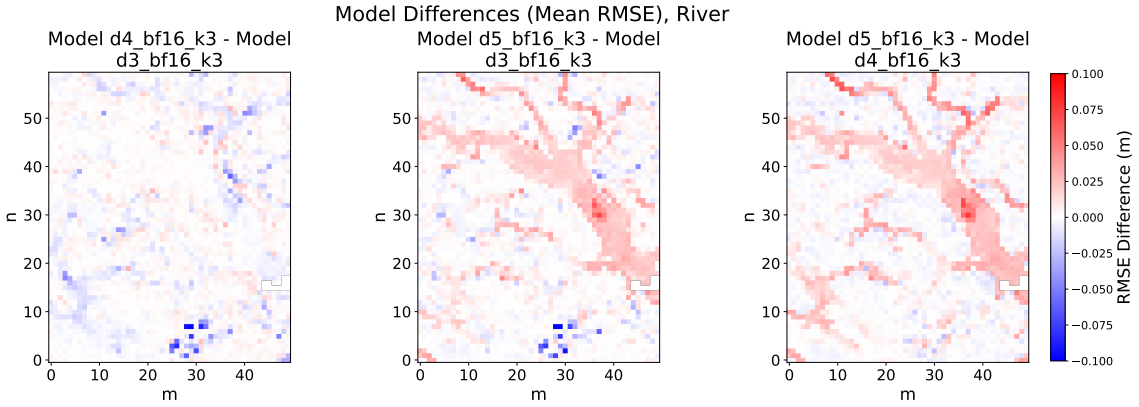

Figure 5.34: Mean RMSE of inundation depth in all test samples for models with varying depth in the river area

The coastal region, illustrated in Figure 5.35, shows a similar pattern as the river area. The model with depth 5 again demonstrates higher errors, on the order of 0.1 meters, while the models with depths 3 and 4 maintain comparable accuracy.

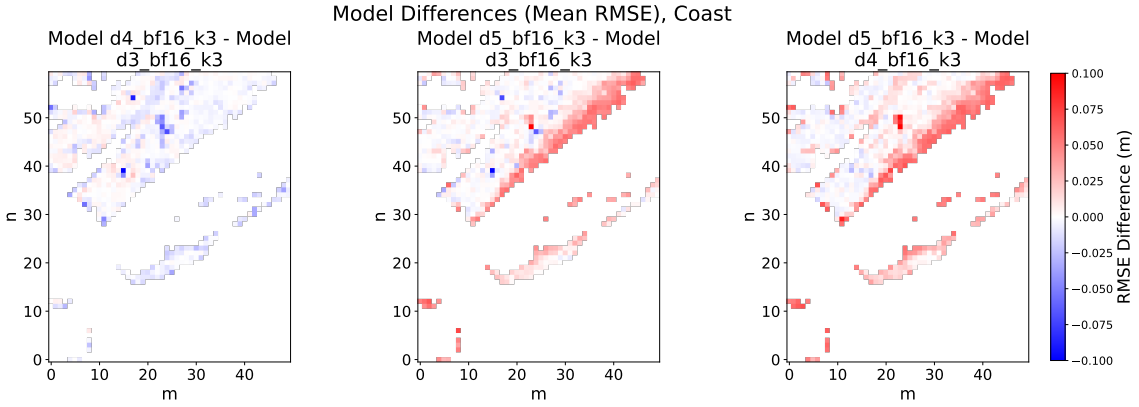

Figure 5.35: Mean RMSE of inundation depth in all test samples for models with varying depth in the coast area

In the mountain region, presented in Figure 5.36, the model with depth 5 shows increased errors in predicting low-lying areas. This suggests that the additional pooling layers in deeper architectures can prevent the model from learning detailed spatial information.

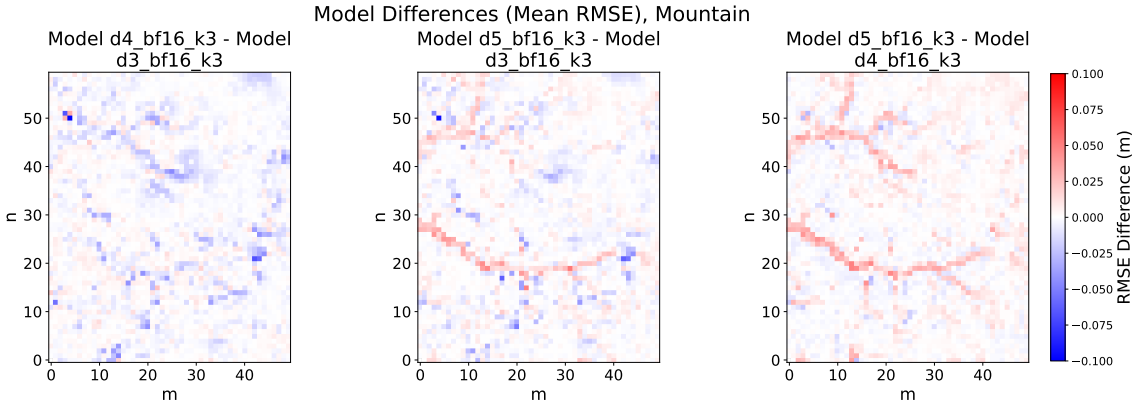

Figure 5.36: Mean RMSE of inundation depth in all test samples for models with varying depth in the mountain area

Filter size comparison

A comparison of models with varying numbers of filters is made with a fixed depth of 4 layers and kernel size of 3, enabling the assessment of how increasing model capacity through the number of filters affects performance.

The metrics showed that performance generally improves when increasing the number of filters from 8 to 32. However, increasing to 64 filters results in a slight performance decline. The global RMSE plot presented in Figure 5.37 does not reveal large differences between the models, suggesting that gains from additional filters are marginal.

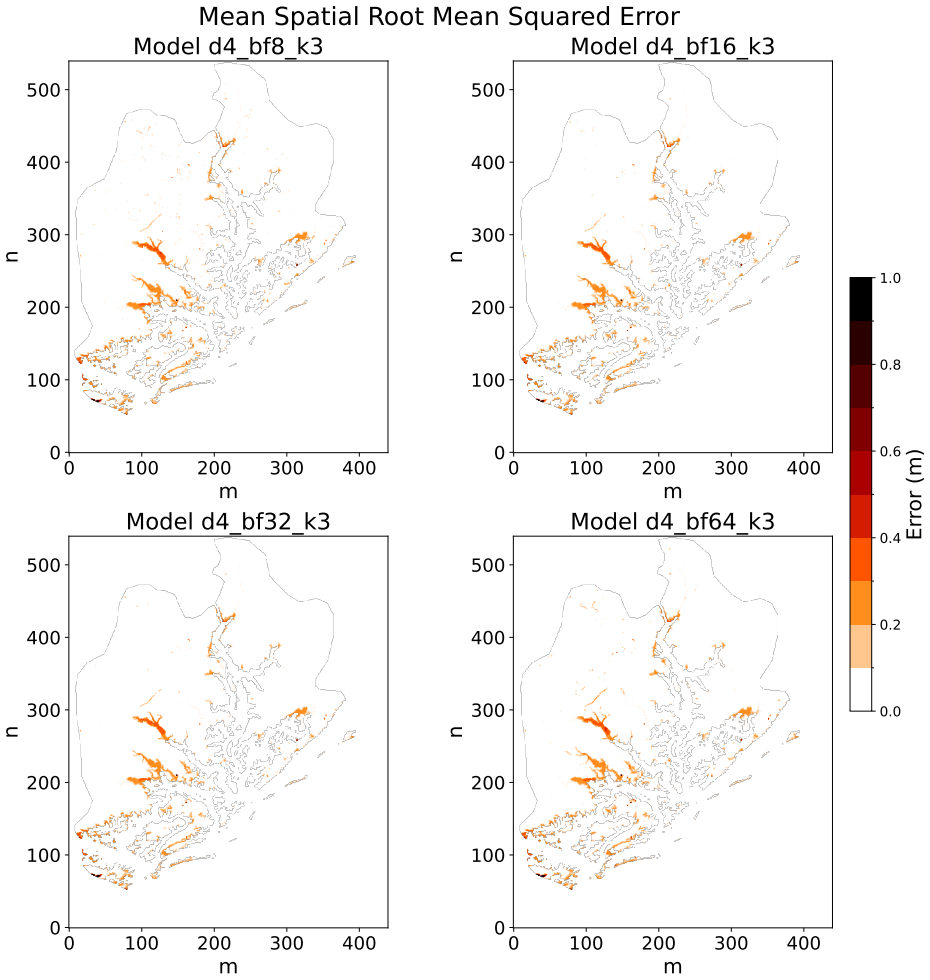

Figure 5.37: Mean RMSE of inundation depth in all test samples for models trained with varying number of filters

To explore local effects, river, coastal, and mountain regions are highlighted and compared to a baseline model containing 8 filters.

In the river region, shown in Figure 5.38 models with 16 and 32 filters show slightly lower errors than the 8-filter baseline, indicating a modest improvement. However, the model with 64 filters shows a small increase in error, performing slightly worse than the baseline. This suggests diminishing returns beyond a certain number of filters.

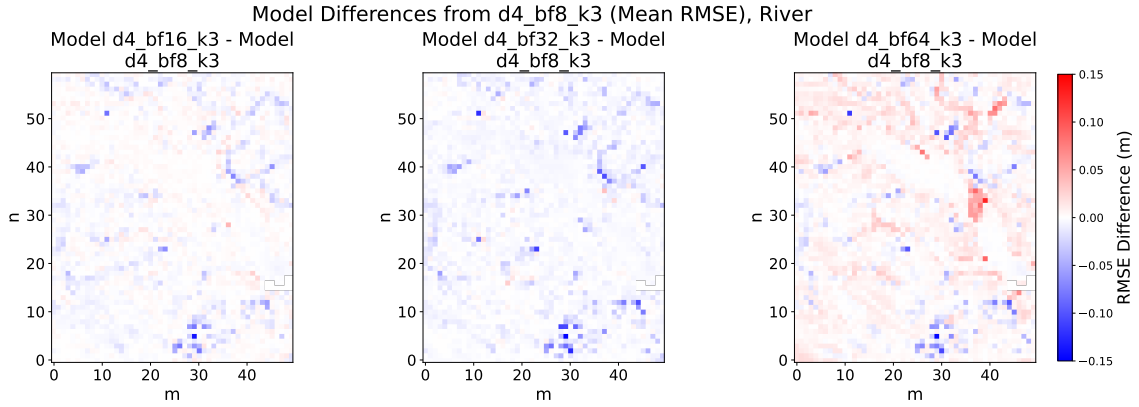

Figure 5.38: Differences in RMSE between models with varying number of filters in the river area

A similar trend is observed in the coastal region, presented in Figure 5.39, where the model with 64 filters again shows higher errors (around 0.1 m), while models with 16 and 32 filters perform slightly better than the baseline.

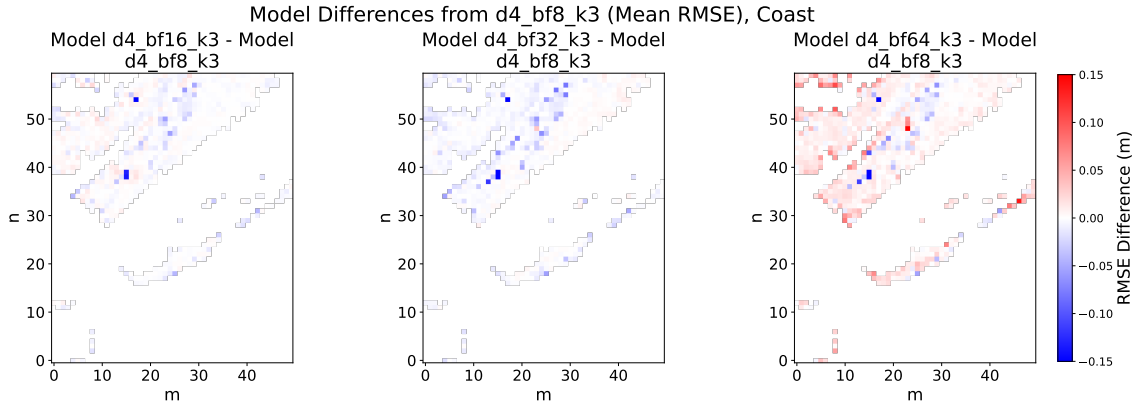

Figure 5.39: Differences in RMSE between models with varying number of filters in the coast area

Figure 5.40 presents the differences in the mountain area, where differences between models are minimal, and all architectures show comparable performance.

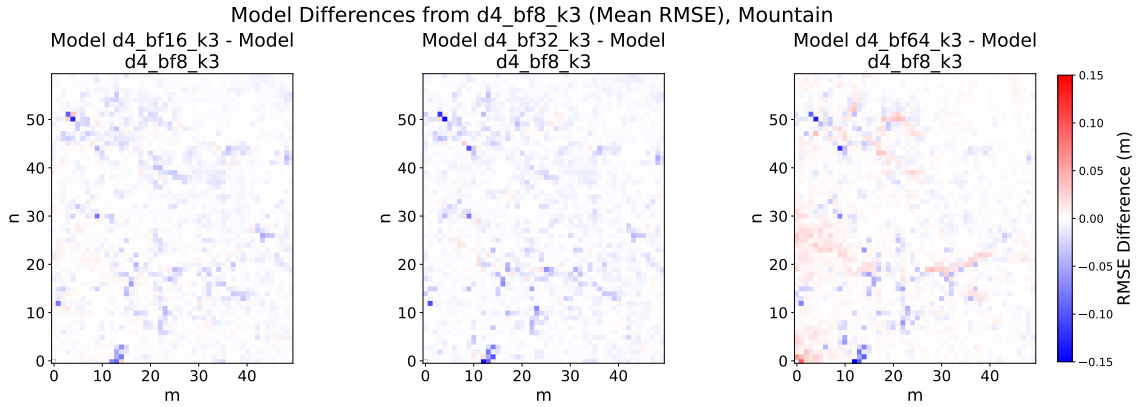

Figure 5.40: Differences in RMSE between models with varying number of filters in the mountain area

5.6.3. Computational time

Training times for the model configurations are presented in Figure 5.41. Results show that larger kernel sizes are associated with longer training durations. A larger kernel increases the computational cost, because the computational domain is larger, which becomes more pronounced as the network depth and number of filters increase.

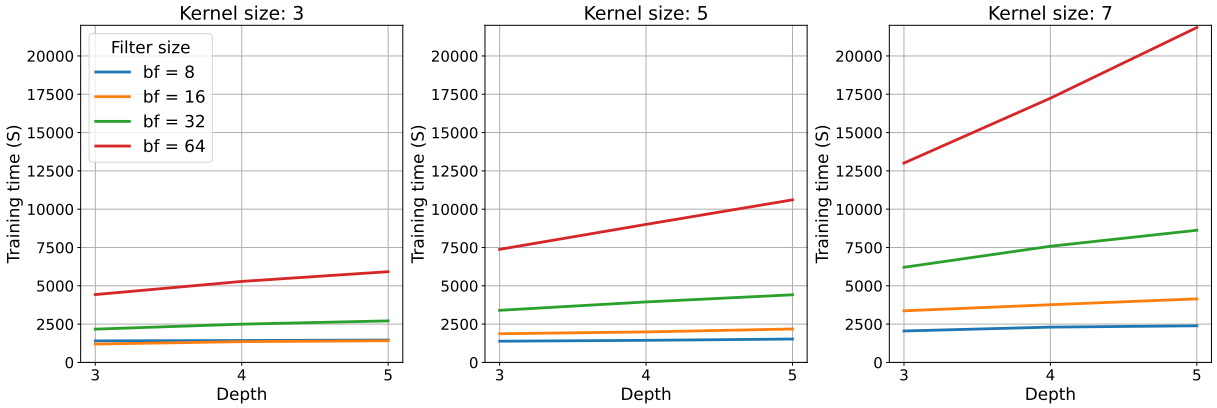

Figure 5.41: Training times, scaled by the number of epochs, for varying combinations of depth, number of filters (bf) and kernel size

Training time increases linearly with model depth across all kernel sizes and number of filters. The slope is greater for larger kernels, indicating a compound effect when both network depth and kernel size grow.

The number of filters has an approximately exponential effect on training time. Training time increases almost exponentially with the number of filters because each additional filter adds a new feature map, and each feature map means a convolution over the input. In CNNs, the number of operations grows proportionally to the number of output filters and the size of the kernel. When the number of filters increases, especially in deeper layers where the input already has many channels from previous filters, the computational cost multiplies rapidly. This effect becomes more pronounced when larger kernel sizes are used, as each filter then involves more operations per position.

As a result, the longest training times are observed in complex architectures that combine deep networks, large kernel sizes, and high filter counts. In contrast, the shortest training times occur in compact architectures with shallow depth, a small number of filters, and a kernel size of 3.

5.7. Detailed analysis

5.7.1. Input parameters

To assess whether adequate input parameters have been selected for the deep learning model, an analysis of the best and worst performing sample will be performed. With the model that contains all parameters, that is, cumulative precipitation, maximum wind speed, and water depth at the river outlet, individual predictions can explain patterns that have been found in computing the mean spatial error.

The prediction for the best and the worst-performing samples are shown in Figure 5.42 and Figure 5.43 respectively.

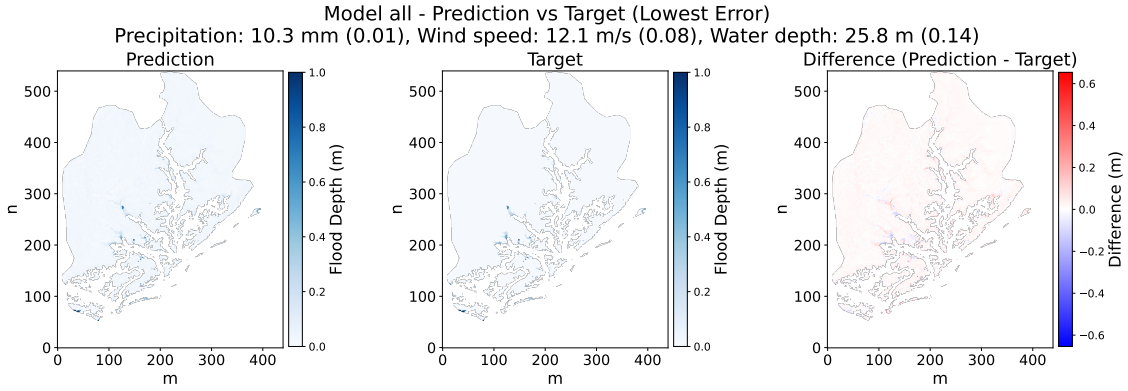

Figure 5.42: Prediction made with the model containing all input parameters for the best performing sample

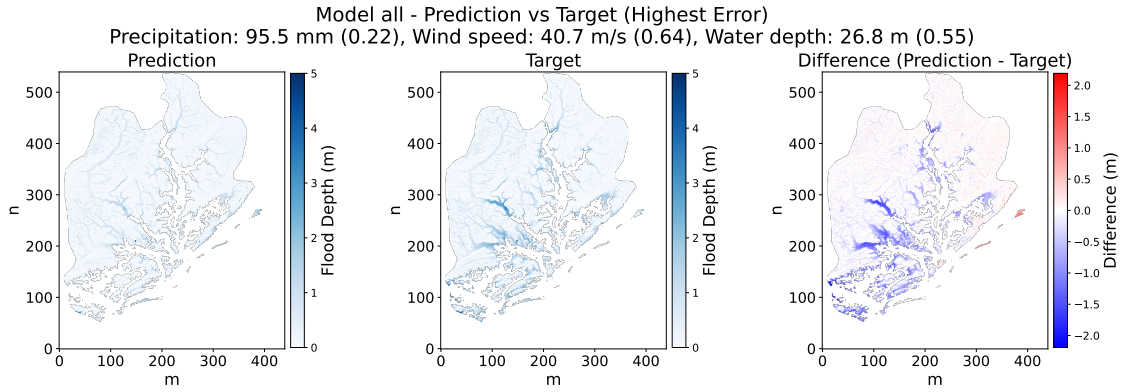

Figure 5.43: Prediction made with the model containing all input parameters for the worst performing sample

In the best-performing sample, the input parameters have low normalized values, resulting in minimal flooding. Both the predicted flood extent and depth are very limited, aligning with the target image. The model exhibits a slight overprediction in most of the domain, with a general offset of less than 10 centimeters, as indicated by the light red shading in the difference plot. Local errors up to 0.6 meters are observed near the river mask, but overall accuracy remains high.

In contrast, the worst-performing sample corresponds to higher input parameter values, leading to a more severe flood scenario. The flood extent and depth are substantially greater than in the best-case scenario. The difference plot reveals a large area in the western part of the domain where the model underestimates flood depths by up to two meters in local regions. In the eastern coastal region, slight overpredictions are present, marked by red tones. These patterns suggest that model performance degrades under more extreme flood conditions, particularly in complex or spatially heterogeneous areas.

In Figure 5.44 the tracks of the best- and worst-performing samples are presented, where TC 1 corresponds the the worst-performing sample and TC 2 to the best-performing sample.

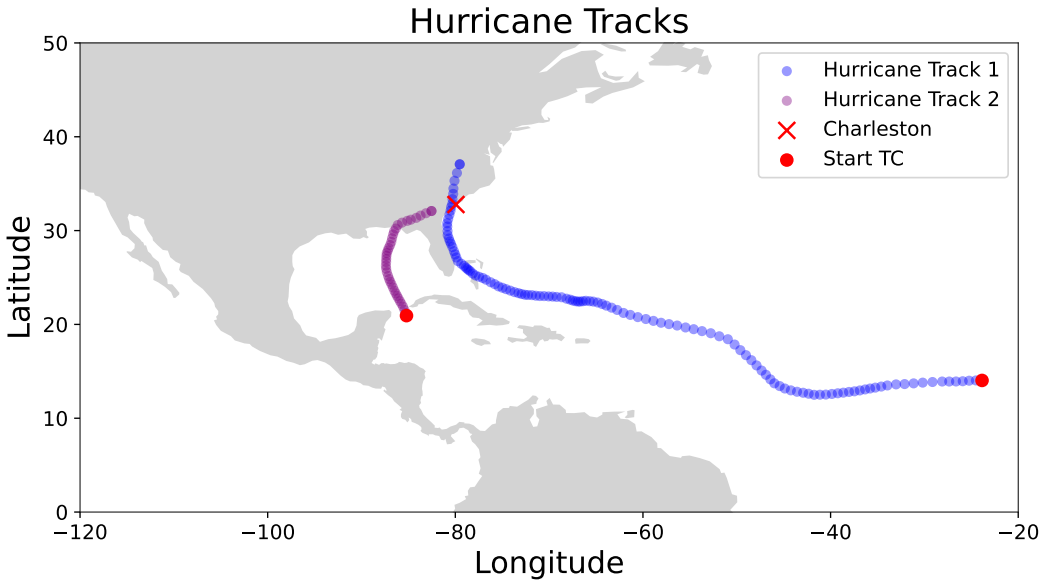

Figure 5.44: Tropical Cyclone tracks of the worst and best performing samples

The track of TC 2 does not get as close to Charleston compared to TC 1. Furthermore, the spacing between the dots, plotted at three-hour intervals, shows a key difference between the two TCs. The TC track points are more densely clustered near Charleston for TC 1, indicating that the cyclone moves more slowly in this region and thus remains overhead for a longer duration. In contrast, TC 2 moves more rapidly towards the study area.

This prolonged time of TC 1 above the domain probably contributes to significantly higher cumulative precipitation. Given that rainfall intensity tends to peak near the eye of a tropical cyclone, an extended duration of the TC over the region results in intense rainfall over a short time span. These conditions can lead to surface saturation and limited runoff capacity, amplifying flood depth and extent.

Importantly, this temporal dimension of rainfall accumulation is not captured by the deep learning model. The model uses cumulative rainfall as input and does not account for how storm movement and rainfall duration dynamically evolve over time. This limitation may explain why the model performs poorly in scenarios such as TC 1, where short, high-intensity rainfall dominates the flooding process.

5.8. Case Study

5.8.1. Physics-based model

In Figure 5.45 the results of a single prediction by the SFINCS model are presented. In the left panel, the water level from the output of the overall model is shown, where an increase in water level close to Charleston can be observed. In the right panel, the water depth in the detailed model is presented. Water depths up to about 26 meters can be observed, with the largest water depths in the river. The SFINCS detailed model outputs contain one sample with a maximum water depth of 33 meters in the detailed model, an issue as mentioned before in Chapter 3. This sample is removed from the dataset, leaving 49 samples for analysis.

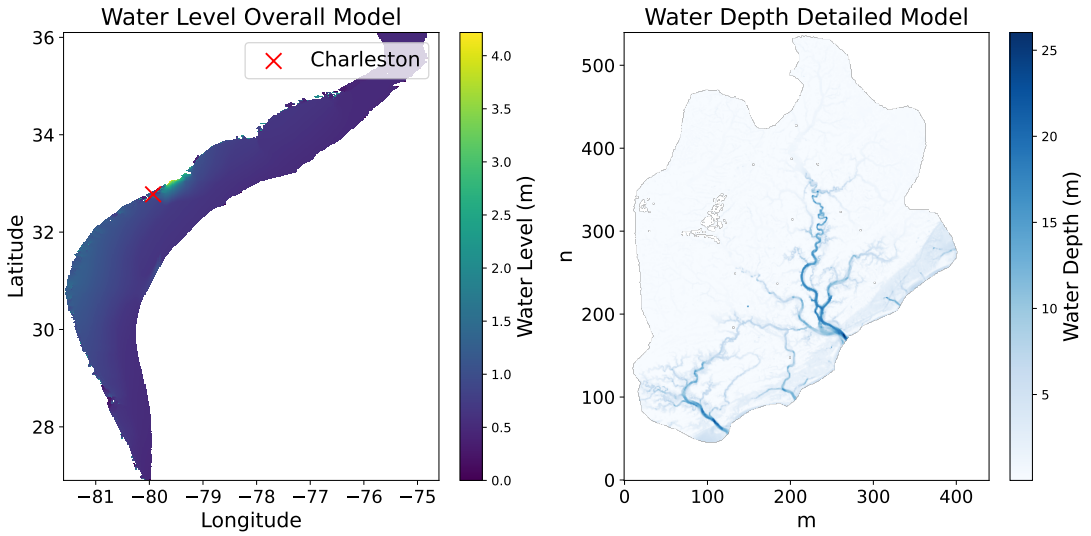

Figure 5.45: Prediction made with the physics-based model SFINCS

5.8.2. Model performance

The deep learning model used in this case study is selected based on the model with the best performance metrics in the network architecture simulations described in Section 5.6. This model has the lowest regression error and the highest classification accuracy among the tested configurations. The optimal model is characterized by a network depth of 4, a base number of filters equal to 32, and a kernel size of 5.

The values of the computed performance metrics are presented in Table 5.11 for both the case study and the original dataset. The performance of the model on the original dataset has been analyzed before, thus will only be compared to the case study in this section.

For the case study, the regression metrics of the deep learning model demonstrate strong performance in estimating continuous flood depths. With an RMSE of 0.069 m and a MAE of 0.033 m, the model shows low error values in reproducing the flood maps. The regression metrics indicate that the RMSE for the case study dataset is notably higher than that for the original dataset (0.054 m), while the MAE values are nearly identical (0.033 m vs. 0.032 m). This pattern suggests the presence of larger individual errors or outliers in the case study predictions, as RMSE is more sensitive to large deviations than MAE.

Table 5.11: Performance metrics of the DL model for the case study and original dataset

Dataset	RMSE (m)	MAE (m)	CSI	Skill	FWR	FDR
CaseStudy_dataset	0.069	0.033	0.776	0.602	**0.021**	0.159
Original_dataset	**0.054**	**0.032**	**0.829**	**0.763**	0.032	**0.074**

On the classification side, the model has a high CSI value (0.776) and skill score(0.602) when trained on the case study data, illustrating good performance in identifying flooded areas and a better performance than an average prediction. The FWR of 0.021 indicates that the model predicts a small number of dry cells incorrectly as flooded. The FDR is higher with a value of 0.159, suggesting that the model still misses flooded areas.

The classification metrics show a decrease in performance for the case study dataset compared to the original dataset. The CSI value drops from 0.829 to 0.776, indicating a reduced ability of the model to correctly identify flooded areas in the case study. Furthermore, the skill decreases from 0.763 to 0.602, suggesting that the model performs worse at capturing the variance in input data. The FDR increases from 0.074 to 0.159, suggesting a higher proportion of predicted flood cells that were not actually

flooded. Interestingly, the FWR is lower in the case study (0.021 vs. 0.032), implying the model was more conservative in generating false alarms.

5.8.3. Spatial analysis

An individual prediction of the deep learning model is presented in Figure 5.46, where the left panel shows the prediction of the deep learning model, in the center the prediction of SFINCS, and in the right panel the difference between the two models. The deep learning model overpredicts the west side of the domain, but underpredicts the eastern side. The differences range up to 1 meter for some pixels, but most differences that can be noted are in the order of 0.5 meters.

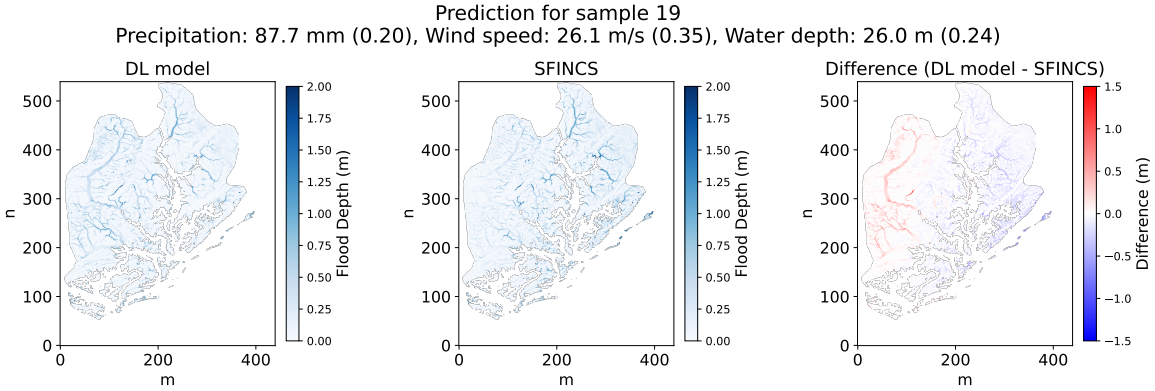

Figure 5.46: DL model prediction, SFINCS prediction, and difference map for a randomly selected test sample. Corresponding values of the input parameters are listed in the figure title with the normalized value between brackets.

The track of the corresponding sample is presented in Figure 5.47. The track makes landfall on the west side of Charleston, the same side as the TC where the synthetic tracks are sampled from. The landfall on the west side of Charleston suggests increased rainfall on the western side of the domain. While the deep learning model does not account for spatial variability of the input data, the physics-based model SFINCS does. As a result, in high-lying areas it is expected that SFINCS has larger flood depths in areas with increased precipitation. However, closer to the center of the TC, the west side of the detailed model, the predicted water depths are lower than on the east side of the domain.

On the west side of the domain, the deep learning model tends to overpredict compared to SFINCS. Although the model does not take spatially varying input into account, the west side of the domain shows larger flood depths than the east side. One possible explanation is that the model was predominantly trained on TCs that make landfall west of Charleston, leading it to associate the western part of the domain with higher flood impacts due to increased precipitation. As a result, the model may implicitly learn a regional bias, even without spatially distributed rainfall or wind data.

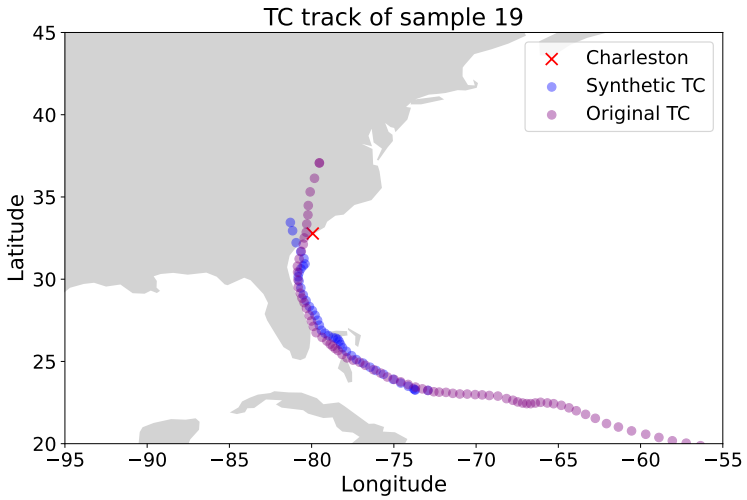

Figure 5.47: Track of the TC corresponding to sample 19 (blue) with parameter values Precipitation: 87.7 mm (0.20), Wind speed: 26.1 m/s (0.35), Water depth: 26.0 m (0.24). The original track where the TCs are sampled from is shown in red.

The mean RMSE of the model is presented in Figure 5.48, where the left panel shows the error on the case study and the right panel on the original dataset. What stands out in the case study are the high-lying areas in the west side of the domain that illustrate the largest error, up to 0.7 meters in two local spots. The east side and the coast show small errors, up to 0.2 meters.

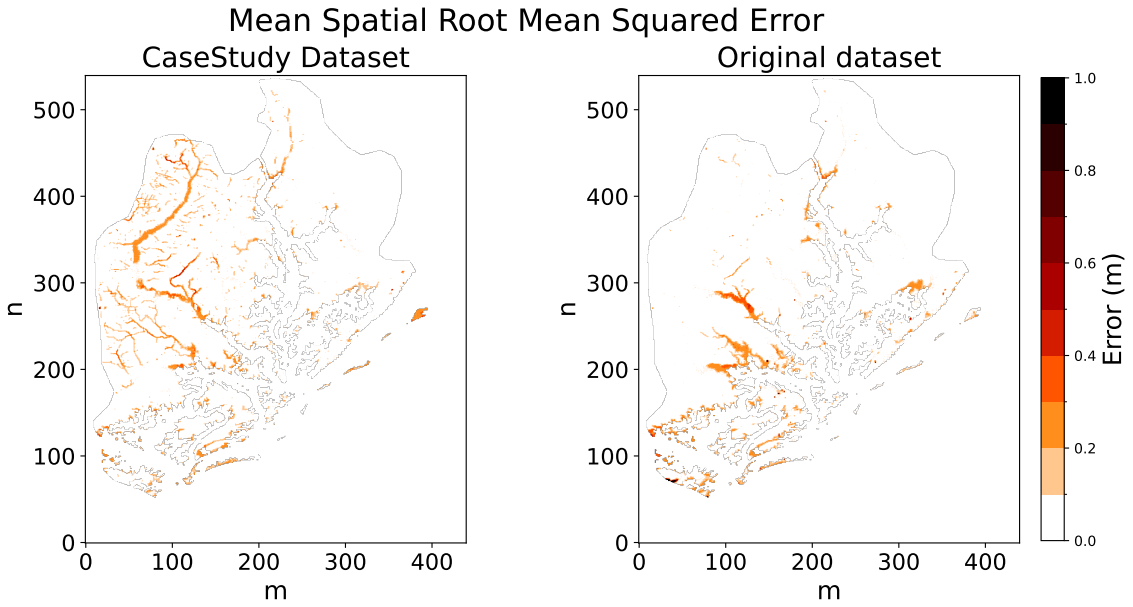

Figure 5.48: Mean RMSE of inundation depth in all test samples of the case study and original dataset

Comparing the spatial distribution of errors in the case study dataset to the original dataset, the patterns differ noticeably. In the original dataset, the largest errors occur near the river branches on the west side of the domain. In contrast, the case study dataset shows most errors in predicting the higher-elevation areas, also located on the western side.

To get an overview of how the error is varying over the parameter range, a scatter plot is created. The scatter plot in Figure 5.49 reveals a general trend of increasing RMSE values with higher input values. Samples with low precipitation and water depth tend to cluster in the lower-left region of the plot and

illustrate the lowest errors, suggesting that the model performs best under mild storm conditions. These findings are in line with the results of Section 5.3. As both the precipitation and water depth increase, the RMSE values rise, signaling that the model has difficulty predicting more extreme conditions. Notably, there is one sample in the middle of the parameter range that shows the highest RMSE value of all samples.

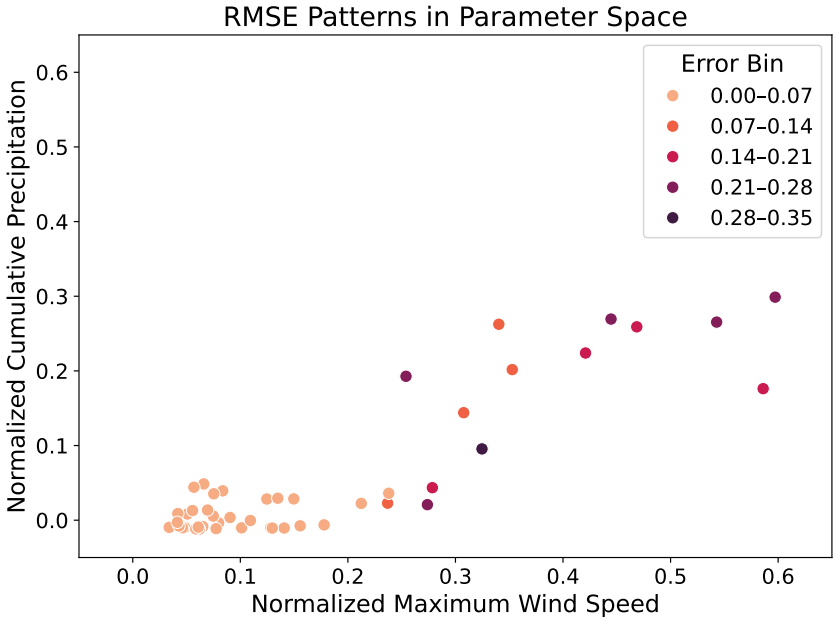

Figure 5.49: Scatter plot of samples in the precipitation and water depth parameter space, with RMSE values indicated by color, illustrating model performance patterns.

5.8.4. Computational Time

The physics-based model SFINCS creates two models, an overall and a detailed model. First the overall model is generated, then the boundary conditions for the detailed model are created in order to run the detailed model. The time required for creating the boundary conditions is not taken into account in the calculation of the prediction time.

SFINCS can only be executed on a CPU unit. A WCF client, which has 4 cores and a memory of 32 GB RAM, was used to run the simulations with SFINCS. The deep learning model predicts the entire batch of 50 samples (later the incorrect sample is removed), where the average is taken to get the prediction time for a single sample. To compare computational performance across platforms, the deep learning model was evaluated both on the WCF and on the GPU.

Table 5.12 presents the average prediction time per sample for the different models. SFINCS is approximately 100 times slower than the deep learning model when both are run on the same CPU. When executed on a GPU, the deep learning model achieves an additional speed-up, with prediction times reduced by a factor of 8.5 compared to its performance on the CPU.

Table 5.12: Average model prediction times per sample

Model	Prediction Time [s]
SFINCS	100-110
DL_WCF	0.85
DL_GPU	0.10

6

Discussion

6.1. Training dataset

The analysis of model performance across varying sizes of the training dataset reveals that a likely optimal number of training samples lies between 200 and 1000. Within this range, a clear improvement in predictive accuracy is measured, while computational efficiency remains relatively high. Beyond 1000 samples, the model does not seem to improve in accuracy, and the computational cost increases. This can be attributed to both the skewed distribution of the data and the use of K-means sampling. The extreme cases in the long tail of the sample distribution are already captured in the dataset with 1000 training samples, representing the general pattern in the data. In this study, the exact trade-off between sample size, accuracy, and computational cost is not optimized.

This study compared two sampling methods for training data selection: random sampling and K-means clustering. The results indicate that the model trained on randomly selected samples performs better in scenarios involving small or minor flooding. This is likely because the distribution of the entire dataset more closely resembles the random samples than the K-means-selected samples.

However, the test set was constructed using K-means sampling to ensure sufficient representation of extreme flood events, which are critical to capture in flood prediction models. As a result, this introduces a bias in model evaluation, since the test set distribution is skewed toward extreme events, with larger consequences. The test set may favor models trained with K-means sampling in their ability to predict extreme flooding. This bias should be considered when interpreting model performance.

Alternative sampling methods such as the Maximum Dissimilarity Algorithm (MDA) can be explored, which can ensure that the chosen events represent the extremes well enough (Camus et al., 2011). Ensuring maximal difference between samples leads to better coverage of the input space and improved performance on rare events, but can come at the cost of slightly lower average performance due to reduced focus on common conditions, ignoring the distribution of the dataset.

The training dataset is derived from a single geographic region, with the result that the generalizability of the model to other regions is uncertain. Deep learning models struggle to transfer to unseen topographies due to terrain differences, where new patterns can emerge. As a result, models typically need to be retrained for new regions. While the same model architecture may provide a useful starting point, it is unclear whether the same parameters will result in an optimal model elsewhere.

Additionally, the dataset contains samples that show instabilities, as discussed in Chapter 3. These numerical instabilities arise in the physics-based model SFINCS, which was run to create the data. Such inconsistencies can introduce noise or bias into the training process, potentially affecting the model's ability to learn correct flood dynamics. While these samples have been filtered, more complex data validation or filtering procedures could help improve data quality and model performance.

6.2. Input parameters

In this study, the model predicts maximum flood depths using input parameters that include the peak values of wind speed and water level. While this approach effectively captures extreme conditions, it does not account for whether these peak values occur simultaneously in reality. The model cannot represent

important interactions between drivers that contribute to compound flooding, because it lacks spatial and temporal information of the input parameters. As a result, the model's predictions may be limited in capturing the full complexity of compound coastal flooding, driven by multiple interacting processes.

Furthermore, the range of input parameters remains limited in this model setup, only looking at cumulative precipitation, maximum wind speed, and water depth at the river outlet. These input parameters provide essential forcing data that drive flooding, but are limited because they do not account for key physical and hydrological processes such as spatial and temporal variability in rainfall and wind, infiltration and soil moisture, or coastal influences like storm surge and tidal effects. Including these parameters gives the deep learning model more parameters from which it can extract patterns, potentially improving accuracy with a slight increase in computational expense.

The current model architecture integrates scalar variables and transforms these variables to spatially uniform images, meaning that each value is uniform across the entire input grid. This approach allows scalar information to be combined with spatially varying data within a convolutional framework, but limits the model's ability to learn from spatial patterns in the forcing conditions. The area of interest covers 100 square kilometers, which is large enough for the input to vary within the region. Spatially varying rainfall data may result in the west side of the domain being flooded, while the east side remains dry. This pattern could be seen in Chapter 5, where the TC passed over the west side of the study area, and the east side was highly overpredicted.

Including spatially or temporally varying meteorological inputs, such as TC distance, precipitation intensity over time, or spatial rainfall distributions may enhance model performance, particularly for events driven by heavy local rainfall where water cannot runoff. Löwe et al. (2021) demonstrated good performance using characteristics from rainfall data to capture the temporal distribution.

Moreover, the water depth parameter is obtained from SFINCS output, which means that SFINCS must be run to generate input for the deep learning model. This contradicts the objective of reducing computational time compared to SFINCS. Therefore, relying on water depth as an input is not ideal. Alternative methods for estimating this parameter, or replacing it with a more readily available estimate, should be explored to maintain model efficiency.

6.3. Neural network architecture

In this research, architecture optimizations were done subsequently. However, the performance of a neural network is not governed by individual architectural components, but by a balance between multiple factors such as input complexity, network depth, number of filters, and kernel size. For instance, increasing model depth or filter size can improve accuracy when sufficient input features are provided, but may lead to overfitting when input data is limited. Conversely, simpler models may outperform more complex architectures when few inputs are used, limiting the number of patterns the model is able to learn, as was found in Chapter 5.

The complexity of the model should reflect the complexity of the underlying input data. The results show that a kernel size of 7, which translates to a spatial convolution window of about 1.4 kilometers, is unable to recognize local topographic features. Similarly, increasing the number of filters to 64 in the first convolutional layers, while only using a single spatial input channel, leads to more model parameters than necessary. This introduced unnecessary complexity and increased training time.

The optimal network configuration balances model accuracy, generalizability, and computational cost. Architectural choices should align with the complexity of the input data and the specific objectives of the modelling task. The optimal configuration cannot be found by tuning each architectural component independently. Instead, a comprehensive strategy that evaluates combinations of architectural components is required. This includes assessing how performance varies across different input setups, architectural depths, and network hyperparameters. Techniques such as grid search or Bayesian optimization can be employed to systematically explore this design space. In addition, cross-validation can be used to ensure that the selected configuration generalizes well across the dataset.

Currently, input data for the deep learning model is not masked, meaning that areas outside the DEM extent are included during training and only masked at the evaluation stage. These regions carry information, but are not used for training and validation and still consume computational resources. Future improvements

could include applying the mask prior to providing input to the model, reducing the number of trainable pixels and thereby enhancing training efficiency, especially when processing large datasets.

The use of both regression and classification loss functions in this study highlights that classification losses underperform when applied to continuous targets such as water depth. Regression tasks require precise numerical outputs, whereas classification losses divide outputs into categories, which can lead to over-smoothed or inaccurate predictions in continuous domains. Future work should treat the regressive and classification objectives separately. Using two specialized models, one for regression and one for classification, may offer a more effective way to model flood dynamics.

6.4. Model evaluation

The river mask used during evaluation of the model performance was derived based on low-lying areas and a small flood event. As a result, some areas within the mask are initially dry, meaning that these areas are not evaluated by the model. In use cases, for example, in an Early Warning System, this means that these areas always need to be evacuated.

While RMSE is a commonly used metric for assessing model performance, this metric alone is insufficient to fully evaluate the predictive capabilities of a flood model. RMSE tends to be dominated by high-error outliers and does not provide information about the spatial distribution or classification quality of predictions.

To better assess how well the model captures flood extent, classification-based metrics should be applied. Metrics such as the Critical Success Index (CSI) or skill score indicate how accurately the model distinguishes between flooded and non-flooded areas and the model's ability of capturing the flood extent based on varying input parameters.

In addition, metrics can be used to detect systematic over- or under-prediction tendencies, such as the FWR and FDR. For instance, an excess of false positives (high FWR) may indicate overly sensitive flood predictions under certain conditions, while a high FDR could suggest missed flood events.

Spatial error maps should be generated to gain insight into prediction patterns. These maps can help identify prediction errors and assess whether the model lacks important information in specific areas, such as missing hydrodynamic features or missing forcing inputs. By comparing prediction error patterns with topographic and input data, researchers can diagnose weaknesses in the model architecture or data quality, guiding future improvements.

6.5. Performance

Results showed that the deep learning model could reproduce water depth patterns with reasonable RMSE values. The RMSE for both the original dataset (0.054 m) and the case study (0.069 m) show that the model has a good accuracy. Also the CSI values for the original dataset (0.829) and the case study (0.776) are comparable to those of other works (Löwe et al., 2021; Herath et al., 2025).

A key limitation of the deep learning surrogate model is that the model can only approximate the accuracy of the underlying physics-based model. It does not improve the accuracy of the simulation, but instead aims to replicate it in less computational time. Although the surrogate model can be used for rapid assessments and scenario testing, its accuracy remains inherently bounded by the limitations of the data it was trained and validated on.

The deep learning model significantly outperforms the physics-based model SFINCS in terms of simulation speed. When both are executed on the same CPU, the deep learning model is approximately 100 times faster. This comparison excludes the additional time SFINCS requires to generate boundary conditions for its detailed simulations, further increasing the computational gain by the deep learning model. Additionally, running the deep learning model on a GPU reduces the prediction time by a factor of 8.5 compared to CPU execution. Similar speed-up factors have been found, as demonstrated by Bentivoglio et al. (2022), when replacing a numerical model by a deep learning surrogate.

Another practical consideration observed during model training is the variability in computational time between similar model configurations, even for different data folds using the same configuration. These discrepancies are largely due to differences in server resource allocation, such as the number of CPU/GPU cores or nodes assigned to a specific job. This variation can complicate one-on-one comparisons of training

efficiency unless hardware usage is controlled or standardized. Future work could benefit from standardizing computational environments or monitoring resource allocation to ensure consistent benchmarking.

Moreover, during the prediction phase, it is important that the resource allocation is tailored to the task at hand. Assigning computational resources according to the specific demands of the prediction task helps to optimize runtime and reduces unnecessary costs.

7

Conclusions

This chapter attempts to answer the research questions listed in Chapter 1, in order to answer the main aim of this research: To investigate how deep learning models compare to physics-based models on forecasting floods induced by tropical cyclones.

What data needs to be selected as input for a DL model and how does the data need to be pre-processed?
In this research, meteorological data from the spiderweb is used to extract scalar parameters such as cumulative precipitation, maximum wind speed, and offshore maximum water level. These parameters have proven to be effective as inputs for the deep learning model. However, to improve model performance in future stages, the spatial and temporal variability of these parameters should be incorporated. This would allow for a more representative and informative input structure.

Preprocessing steps such as normalization (e.g., MinMax scaling), masking of non-relevant areas (e.g., land or initially dry zones), and ensuring consistent input dimensions are essential to prepare the data for model training.

What performance metric can be used to judge the performance of different DL techniques?
To evaluate flood depth predictions, this research primarily uses Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) as regression metrics. RMSE is prioritized due to its sensitivity to outliers, which is particularly relevant in flood modeling, where underestimating extreme depths can have serious consequences.

In addition to measuring performance on a continuous scale, a threshold is applied to distinguish flooded and non-flooded areas. Classification metrics such as the Critical Success Index (CSI), Skill Score, False Wet Rate (FWR), and False Dry Rate (FDR) are used to quantify the agreement between predicted and observed flooded areas. These provide insight into how well the model captures spatial patterns of the flood.

Because the surrogate model relies on patterns present in the input data, it may miss important flood dynamics if key interactions are not captured. Therefore, combining regression and classification metrics with visual inspections of predicted versus target flood maps is essential. Visualization helps to identify systematic spatial errors (e.g., overprediction near channels or underprediction in low-lying zones) that may not be fully reflected in scalar performance scores. Combining quantitative metrics with qualitative evaluation gives a more complete understanding of model performance.

How can the model architecture be optimized to capture the processes needed to predict compound coastal flooding?
To capture the key processes driving compound coastal flooding, the model architecture was designed based on a U-Net structure with a single channel Digital Elevation Model (DEM) as spatial input. Drivers of compound coastal flooding, precipitation, wind speed, and offshore water level were included as scalar inputs representing meteorological and oceanographic forcing. These scalar values were concatenated at the deepest part of the encoder, allowing the model to integrate large-scale drivers with learned spatial features before decoding.

The results of the architectural optimization showed that the simplicity of the data did not require a complex model. Even with this minimal configuration, the model is capable of approximating both flood depth and extent predictions made by a physics-based model. Visualizations of model predictions confirmed that the spatial flood patterns were well captured and performance metrics such as RMSE, CSI, and FDR supported the model's effectiveness.

However, performance limitations in capturing local flood patterns suggest potential gains from future architectural refinements. These could include integrating spatially distributed meteorological inputs or the inclusion of scalar parameters that capture temporal characteristics. Such changes would enhance the model's ability to reflect the complex interactions between rainfall, wind, surge, and topography that characterize compound coastal flood events.

How well does the deep learning model replicate the spatial flood patterns predicted by the physics-based model?
The deep learning model demonstrates a strong ability to replicate the spatial flood patterns predicted by the physics-based model, as illustrated by low RMSE and high CSI metrics across both the original dataset and the case study. These results indicate that the model achieves good overall accuracy in predicting flood depths and extents.

However, spatial analysis maps revealed areas where the deep learning model consistently diverged from the physics-based model. In both the original dataset and the case study, the model showed difficulties in predicting the west side of the domain, with errors ranging up to 0.6 meters. Additional input features might be required to provide more information for the deep learning model to improve prediction accuracy. Future studies should focus on refining input parameters, and better separating regression and classification objectives to more accurately reflect the spatial complexity of compound coastal flooding.

How does the simulation time of DL models compare to the simulation time of the reduced-physics model?
The deep learning model greatly reduces computational time compared to the physics-based model. Its speed advantage is on the order of two magnitudes faster on CPU and even more on GPU, making it a suitable model for operational flood forecasting. Faster runtimes allow for running a larger number of simulations within a given timeframe, enabling the exploration of an increased number of potential flood scenarios and providing decision-makers with increased time to evaluate and respond.

The main research question is listed and answered:

How do deep learning models compare to physics-based models on forecasting floods induced by tropical cyclones in terms of accuracy and speed?
Deep learning models can effectively replicate the spatial flood patterns generated by physics-based models for tropical cyclone-induced events. They achieve comparable accuracy, in regression with RMSE values of 0.054 m and 0.069 m and in classification with CSI scores of 0.829 and 0.776 for the representative test dataset and case study, respectively. Spatial error maps reveal consistent errors in certain areas, particularly in the western domain. Additional input features may be needed to capture these inconsistencies. Although the surrogate model cannot exceed the accuracy of the physics-based model it is trained on, it offers a significant advantage in computational speed, approximately 100 times faster on CPU and even faster when simulated on GPU.

References

Ali, Mostafa and Simaan AbouRizk (2024). “Updating simulation model parameters using stochastic gradient descent”. In: *Automation in Construction* 166, p. 105676. DOI: https://doi.org/10.1016/j.autcon.2024.105676. URL: https://www.sciencedirect.com/science/article/pii/S0926580524004126.

Apicella, Andrea, Francesco Donnarumma, Francesco Isgrò, and Roberto Prevete (2021). “A survey on modern trainable activation functions”. In: *Neural Networks* 138, pp. 14–32. DOI: https://doi.org/10.1016/j.neunet.2021.01.026. URL: https://www.sciencedirect.com/science/article/pii/S0893608021000344.

Arribas-Bel, Dani (2023). *contextily: context geo tiles in Python*. URL: https://contextily.readthedocs.io.

Bakker, Tije M., José A.A. Antolínez, Tim W.B. Leijnse, Stuart G. Pearson, and Alessio Giardino (2022). “Estimating tropical cyclone-induced wind, waves, and surge: A general methodology based on representative tracks”. In: *Coastal Engineering* 176, p. 104154. DOI: https://doi.org/10.1016/j.coastaleng.2022.104154. URL: https://www.sciencedirect.com/science/article/pii/S0378383922000692.

Bentivoglio, Roberto, Elvin Isufi, Sebastian Nicolaas Jonkman, and Riccardo Taormina (2022). “Deep learning methods for flood mapping: a review of existing applications and future research directions”. In: *Hydrology and Earth System Sciences* 26.16. Cited by: 84; All Open Access, Gold Open Access, Green Open Access, pp. 4345–4378. DOI: 10.5194/hess-26-4345-2022. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85137807801&doi=10.5194%2fhess-26-4345-2022&partnerID=40&md5=e88829a67d4787352cddfab5eb79278f.

Camus, Paula, Fernando J. Mendez, Raul Medina, and Antonio S. Cofiño (2011). “Analysis of clustering and selection algorithms for the study of multivariate wave climate”. In: *Coastal Engineering* 58.6, pp. 453–462. DOI: https://doi.org/10.1016/j.coastaleng.2011.02.003. URL: https://www.sciencedirect.com/science/article/pii/S0378383911000354.

Chen, Liang-Chieh, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam (2018). “Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation”. In: *Computer Vision – ECCV 2018*. Cham: Springer International Publishing, pp. 833–851.

Curtis, Katherine J. and Annemarie Schneider (2011). “Understanding the demographic implications of climate change: estimates of localized population predictions under future scenarios of sea-level rise”. In: *Population and Environment* 33.1, pp. 28–54. DOI: 10.1007/s11111-011-0136-2. URL: https://doi.org/10.1007/s11111-011-0136-2.

Edmonds, Douglas A., Rebecca L. Caldwell, Eduardo S. Brondizio, and Sacha M. O. Siani (2020). “Coastal flooding will disproportionately impact people on river deltas”. In: *Nature Communications* 11.1. Cited by: 184; All Open Access, Gold Open Access, Green Open Access. DOI: 10.1038/s41467-020-18531-4. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85091724061&doi=10.1038%2fs41467-020-18531-4&partnerID=40&md5=5d63722e8c1cbdbe3ef640d489ba003b.

Emanuel, Kerry A. (2013). “Downscaling CMIP5 climate models shows increased tropical cyclone activity over the 21st century”. In: *Proceedings of the National Academy of Sciences* 110.30, pp. 12219–12224. DOI: 10.1073/pnas.1301293110. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.1301293110. URL: https://www.pnas.org/doi/abs/10.1073/pnas.1301293110.

Herath, Herath Mudiyanselage Viraj Vidura, Lucy Marshall, Abhishek Saha, Sanka Rasnayaka, and Sachith Seneviratne (2025). “Subgrid informed neural networks for high-resolution flood mapping”. In: *Journal of Hydrology* 660, p. 133329. DOI: https://doi.org/10.1016/j.jhydrol.2025.133329. URL: https://www.sciencedirect.com/science/article/pii/S0022169425006675.

Holland, Greg (2008). “A Revised Hurricane Pressure–Wind Model”. In: *Monthly Weather Review* 136.9, pp. 3432–3445. DOI: 10.1175/2008MWR2395.1. URL: https://journals.ametsoc.org/view/journals/mwre/136/9/2008mwr2395.1.xml.

Holland, Greg J., James I. Belanger, and Angela Fritz (2010). “A Revised Model for Radial Profiles of Hurricane Winds”. In: *Monthly Weather Review* 138.12, pp. 4393–4401. DOI: 10.1175/2010MWR3317.1. URL: https://journals.ametsoc.org/view/journals/mwre/138/12/2010mwr3317.1.xml.

Hu, Pan, Qiang Zhang, Peijun Shi, Bo Chen, and Jiayi Fang (2018). “Flood-induced mortality across the globe: Spatiotemporal pattern and influencing factors”. In: *Science of The Total Environment* 643, pp. 171–182. DOI: https://doi.org/10.1016/j.scitotenv.2018.06.197. URL: https://www.sciencedirect.com/science/article/pii/S0048969718322745.

Hu, R., F. Fang, C.C. Pain, and I.M. Navon (2019). “Rapid spatio-temporal flood prediction and uncertainty quantification using a deep learning method”. In: *Journal of Hydrology* 575, pp. 911–920. DOI: https://doi.org/10.1016/j.jhydrol.2019.05.087. URL: https://www.sciencedirect.com/science/article/pii/S0022169419305323.

Huang, Pin Chun, Kuo-Lin Hsu, and Kwan Lee (Feb. 2021). “Improvement of Two-Dimensional Flow-Depth Prediction Based on Neural Network Models By Preprocessing Hydrological and Geomorphological Data”. In: *Water Resources Management* 35, pp. 1–22. DOI: 10.1007/s11269-021-02776-9.

Hunter, J. D. (2007). “Matplotlib: A 2D graphics environment”. In: *Computing in Science & Engineering* 9.3, pp. 90–95. DOI: 10.1109/MCSE.2007.55.

Indolia, Sakshi, Anil Kumar Goswami, S.P. Mishra, and Pooja Asopa (2018). “Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach”. In: *Procedia Computer Science* 132. International Conference on Computational Intelligence and Data Science, pp. 679–688. DOI: https://doi.org/10.1016/j.procs.2018.05.069. URL: https://www.sciencedirect.com/science/article/pii/S1877050918308019.

Interagency Performance Evaluation Task Force (IPET) (2006). *Performance Evaluation of the New Orleans and Southeast Louisiana Hurricane Protection System: Draft Final Report, Volume VIII – Engineering and Operational Risk and Reliability Analysis*. Accessed: 2025-06-04. URL: https://usace.contentdm.oclc.org/digital/collection/p266001coll1/id/2844/.

Jamali, Behzad, Peter M. Bach, Luke Cunningham, and Ana Deletic (2019). “A Cellular Automata Fast Flood Evaluation (CA-fféé) Model”. In: *Water Resources Research* 55.6, pp. 4936–4953. DOI: https://doi.org/10.1029/2018WR023679. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2018WR023679. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2018WR023679.

Jiang, Wenjun, Xi Zhong, and Jize Zhang (2024). “Surge-NF: Neural Fields inspired peak storm surge surrogate modeling with multi-task learning and positional encoding”. In: *Coastal Engineering* 193, p. 104573. DOI: https://doi.org/10.1016/j.coastaleng.2024.104573. URL: https://www.sciencedirect.com/science/article/pii/S0378383924001212.

Jonkman, S.N. and J.K. Vrijling (2008). “Loss of life due to floods”. In: *Journal of Flood Risk Management* 1.1, pp. 43–56. DOI: https://doi.org/10.1111/j.1753-318X.2008.00006.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1753-318X.2008.00006.x.

Jordahl, Kelsey et al. (July 2020). *geopandas/geopandas: v0.8.1*. Version v0.8.1. DOI: 10.5281/zenodo.3946761. URL: https://doi.org/10.5281/zenodo.3946761.

Kingma, Diederik P (2014). “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980*.

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner (1998). “Gradient-based learning ap-plied to document recognition”. In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.

Leijnse, Tim, Maarten van Ormondt, Kees Nederhoff, and Ap van Dongeren (2021). “Modeling compound flooding in coastal systems using a computationally efficient reduced-physics solver: Including fluvial, pluvial, tidal, wind- and wave-driven processes”. In: *Coastal Engineering* 163, p. 103796. DOI: https:

//doi.org/10.1016/j.coastaleng.2020.103796. URL: https://www.sciencedirect.com/science/article/pii/S0378383920304828.

Lesser, G.R., J.A. Roelvink, J.A.T.M. van Kester, and G.S. Stelling (2004). “Development and validation of a three-dimensional morphological model”. In: *Coastal Engineering* 51.8. Coastal Morphodynamic Modeling, pp. 883–915. DOI: https://doi.org/10.1016/j.coastaleng.2004.07.014. URL: https://www.sciencedirect.com/science/article/pii/S0378383904000870.

Löwe, Roland, Julian Böhm, David Getreuer Jensen, Jorge Leandro, and Søren Højmark Rasmussen (2021). “U-FLOOD – Topographic deep learning for predicting urban pluvial flood water depth”. In: *Journal of Hydrology* 603, p. 126898. DOI: https://doi.org/10.1016/j.jhydrol.2021.126898. URL: https://www.sciencedirect.com/science/article/pii/S0022169421009483.

Luettich, Richard Albert, Joannes J Westerink, Norman W Scheffner, et al. (1992). “ADCIRC: An Advanced Three-Dimensional Circulation Model for Shelves, Coasts, and Estuaries. Report 1. Theory and Methodology of ADCIRC-2DDI and ADCIRC-3DL.” In.

McCulloch, Warren S. and Walter Pitts (1943). “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.1, pp. 115–133. DOI: 10.1007/BF02478259. URL: https://doi.org/10.1007/BF02478259.

Murphy, Allan H. (1988). “Skill Scores Based on the Mean Square Error and Their Relationships to the Correlation Coefficient”. In: *Monthly Weather Review* 116.12, pp. 2417–2424. DOI: 10.1175/1520-0493(1988)116<2417:SSBOTM>2.0.CO;2. URL: https://journals.ametsoc.org/view/journals/mwre/116/12/1520-0493_1988_116_2417_ssbotm_2_0_co_2.xml.

National Hurricane Center (2020). *Hurricane Laura (2020) Graphics Archive*. Accessed: 2024-09-11. URL: https://www.nhc.noaa.gov/archive/2020/LAURA_graphics.php.

— (2025). *Tropical Cyclone Text Product Descriptions*. https://www.nhc.noaa.gov/aboutnhcprod.shtml. Accessed: 2024-09-13.

Nederhoff, K et al. (2023). “Accounting for Uncertainties in Forecasting Tropical Cyclone-Induced Compound Flooding”. In: *EGUsphere* 2023, pp. 1–41. DOI: 10.5194/egusphere-2023-2341. URL: https://egusphere.copernicus.org/preprints/2023/egusphere-2023-2341/.

Nederhoff, K., J. Hoek, T. Leijnse, M. van Ormondt, S. Caires, and A. Giardino (2021). “Simulating synthetic tropical cyclone tracks for statistically reliable wind and pressure estimations”. In: *Natural Hazards and Earth System Sciences* 21.3, pp. 861–878. DOI: 10.5194/nhess-21-861-2021. URL: https://nhess.copernicus.org/articles/21/861/2021/.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (May 2015). “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *CoRR* abs/1505.04597. DOI: 10.48550/arXiv.1505.04597.

Roy, Chandan and Rita Kovordányi (2012). “Tropical cyclone track forecasting techniques — A review”. In: *Atmospheric Research* 104-105, pp. 40–69. DOI: https://doi.org/10.1016/j.atmosres.2011.09.012. URL: https://www.sciencedirect.com/science/article/pii/S0169809511002973.

Smith, Leslie N. (2015). “No More Pesky Learning Rate Guessing Games”. In: *CoRR* abs/1506.01186. arXiv: 1506.01186. URL: http://arxiv.org/abs/1506.01186.

Suh, Seung-Won, Hwa Lee, Hyeon Kim, and Jason Fleming (May 2015). “An efficient early warning system for typhoon storm surge based on time-varying advisories by coupled ADCIRC and SWAN”. In: *Ocean Dynamics* 65. DOI: 10.1007/s10236-015-0820-3.

Suttisinthong, N., B. Seewirote, A. Ngaopitakkul, and A. Pothisarn (2014). “Selection of Proper Activation Functions in Back-propagation Neural Network algorithm for Single-Circuit Transmission Line”. In: *International Journal of Innovative Computing, Information and Control* 8.6, pp. 4299–4318.

The Water Institute (2025). *Compound Flooding*. Accessed: 2024-09-18. URL: https://thewaterinstitute.org/projects/compound-flooding.

Wahl, Thomas, Shaleen Jain, Jens Bender, Steven D. Meyers, and Mark E. Luther (2015). “Increasing risk of compound flooding from storm surge and rainfall for major US cities”. In: *Nature Climate Change* 5.12, pp. 1093–1097. DOI: 10.1038/nclimate2736. URL: https://doi.org/10.1038/nclimate2736.

Xie, Jianbin, Xingru Feng, Tianhai Gao, Zhifeng Wang, Kai Wan, and Baoshu Yin (2024). “Application of deep learning in predicting suspended sediment concentration: A case study in Jiaozhou Bay, China”. In: *Marine Pollution Bulletin* 201, p. 116255. DOI: https://doi.org/10.1016/j.marpolbul.2024.116255. URL: https://www.sciencedirect.com/science/article/pii/S0025326X24002327.

Yazdani Abyaneh, Amir Hossein, Ali Hossein Gharari, and Vahid Pourahmadi (Nov. 2018). *Deep Neural Networks Meet CSI-Based Authentication*. DOI: 10.48550/arXiv.1812.04715.

Yessoufou, Fadel and Jinsong Zhu (2023). “Classification and regression-based convolutional neural network and long short-term memory configuration for bridge damage identification using long-term monitoring vibration data”. In: *Structural Health Monitoring* 22.6, pp. 4027–4054. DOI: 10.1177/14759217231161811.

A

U-Net python code

C:/Users/oudenes/Documents/UNet_architecture.py

```
# -*- coding: utf-8 -*-
"""
Created on Wed Jun 18 09:11:47 2025

@author: oudenes
"""

import torch
import torch.nn as nn
import torch.nn.functional as F

from UNet_architecture_helper import DownSample, DoubleConv, UpSample


class UNet(nn.Module):
    def __init__(self, in_channels=1, scalar_input_dim=2, depth=3, base_features=16, kernel_size=3):
        super().__init__()
        self.depth = depth

        # Encoder (Downsampling)
        self.encoders = nn.ModuleList()
        prev_channels = in_channels
        for i in range(depth-1):
            self.encoders.append(DownSample(prev_channels, base_features * (2 ** i), kernel_size=kernel_size, padding=kernel_size//2))
            prev_channels = base_features * (2 ** i)

        # Bottleneck
        self.bottle_neck = DoubleConv(prev_channels, prev_channels * 2, kernel_size=kernel_size, padding=kernel_size//2)

        # Scalar input projection layer (broadcast to match feature map dims)
        self.scalar_fc = nn.Linear(scalar_input_dim, prev_channels * 2)

        # Decoder (Upsampling)
        self.decoders = nn.ModuleList()
        for i in range(depth - 1, -1, -1):
            self.decoders.append(UpSample(prev_channels * 2, prev_channels, kernel_size=kernel_size, padding=kernel_size//2, stride=2))
            prev_channels //= 2

        # Final Convolution
        self.out = nn.Conv2d(in_channels=base_features, out_channels=1, kernel_size=1)

    def forward(self, dem, scalar_inputs):
        skip_connections = []
        x = dem

        # Get shape of the input
        _, _, h, w = x.shape

```

C:/Users/oudenes/Documents/UNet_architecture_helper.py

```
# -*- coding: utf-8 -*-
"""
Created on Wed Jun 18 09:13:15 2025

@author: oudenes
"""

import torch
import torch.nn as nn
import torch.nn.functional as F

class DoubleConv(nn.Module):
    def __init__(self, in_channels, out_channels, kernel_size=3, padding=1):
        super().__init__()
        self.conv_op = nn.Sequential(
            nn.Conv2d(in_channels, out_channels, kernel_size=kernel_size, padding=padding),
            nn.ReLU(inplace=True),
            nn.Conv2d(out_channels, out_channels, kernel_size=kernel_size, padding=padding),
            nn.ReLU(inplace=True)
        )

    def forward(self, x):
        return self.conv_op(x)


class DownSample(nn.Module):
    def __init__(self, in_channels, out_channels, kernel_size, padding, pad_pool=0):
        super().__init__()
        self.conv = DoubleConv(in_channels, out_channels, kernel_size=kernel_size, padding=padding)
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2, padding=pad_pool)

    def forward(self, x):
        down = self.conv(x)
        p = self.pool(down)
        return down, p


class UpSample(nn.Module):
    def __init__(self, in_channels, out_channels, kernel_size, padding, stride):
        super().__init__()
        self.up = nn.ConvTranspose2d(in_channels, in_channels//2, kernel_size=2, stride=stride)
        self.conv = DoubleConv(in_channels, out_channels, kernel_size=kernel_size, padding=padding)

    def forward(self, x1, x2):
        x1 = self.up(x1)
        x = torch.cat([x1, x2], 1)
        return self.conv(x)

```