Modelling Floods via Graph Neural Networks

with Applications to Dike-breach Floods

Roberto Bentivoglio

MODELLING FLOODS VIA GRAPH NEURAL NETWORKS

WITH APPLICATIONS TO DIKE-BREACH FLOODS

MODELLING FLOODS VIA GRAPH NEURAL NETWORKS

WITH APPLICATIONS TO DIKE-BREACH FLOODS

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus Prof.dr.ir. H. Bijl,
Chair of the Board for Doctorates
to be defended publicly on Monday 13 April 2026 at 10:00 o'clock

by

Roberto BENTIVOGLIO

This dissertation has been approved by the promotors.

Rector Magnificus,	chairperson
Prof. dr. ir. S. N. Jonkman,	Delft University of Technology, promotor
Dr. R. Taormina,	Delft University of Technology, promotor
Dr. E. Isufi,	Delft University of Technology, promotor
Independent members:	
Prof. dr. ir. M. Verlaan,	Delft University of Technology / Deltares
Prof. dr. P. Bates,	University of Bristol, United Kingdom
Prof. dr. A. H. Weerts,	Wageningen University / Deltares
Dr. ir. A. Bomers,	University of Twente
Prof. dr. Z. Kapelan,	Delft University of Technology, Reserve Member

Keywords: Flood modelling, Graph Neural Networks, Deep Learning, Dike-breach Floods

Printed by: Proefschrift-AIO

Front & Back: Flood arrival times on a synthetic topography, showing the analogy between finite volume methods and graph neural networks, by R. Bentivoglio.

ISBN 978-94-6518-285-8

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

"Journey before destination."

Brandon Sanderson

CONTENTS

LIST OF FIGURES

List of Tables

List of Acronyms and Symbols

Acronyms

AI	Artificial Intelligence
ARME	Average Relative Mass Error
AUC	Area under the receiver operating curve
BC	Boundary Condition
CFL	Courant–Friedrichs–Lewy
CNN	Convolutional Neural Network
CPU	Computational Processing Unit
CSI	Critical Success Index
DBN	Deep Belief Network
DEM	Digital Elevation Model
DL	Deep Learning
FAT	Flood Arrival Times
GAT	Graph Attention Network
GCN	Graph Convolutional Network
GNN	Graph Neural Network
GPU	Graphical Processing Unit
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MCDA	Multi-Criteria Decision Analysis
ML	Machine Learning
MLP	Multi-Layer Perceptron
MPGNN	Message-passing GNN

MPI Message Passing Interface

MSE Mean Squared Error

mSWE-GNN Multi-Scale Shallow Water Equations Graph Neural Network

NDWI Normalized Difference Water Index

PINN Physics-Informed Neural Network

PReLU Parametric Rectified Linear Unit

ReLU Rectified Linear Unit

RF Random Forest

RMSE Root Mean Squared Error

RNN Recurrent Neural Network

SVM Support Vector Machine

SWE Shallow Water Equations

SWE-GNN Shallow Water Equations Graph Neural Network

UAV Unmanned Aerial Vehicle

Symbols

$\mathscr{E}'$ Edge feature embeddings matrix

$\boldsymbol{\varepsilon}_{ij}$ Edge features

Δt Time step [s]

Δx Mesh size [m]

$\hat{\mathbf{y}}$ Predicted output

$\mathbf{A}$ Adjacency matrix

$\mathbf{A}^m$ Adjacency matrix at scale m

$\mathbf{n}_{ij}$ Outward unit normal vector between cells i, j

$\mathbf{P}^{m \to n}$ Prolongation matrix from scale m to scale n

$\mathbf{W}$ Learnable weight matrix

$\mathscr{L}$ Loss function

$\odot$ Hadamard product

Ω_i Finite volume cell i

$\Phi(\cdot)$	Encoder-processor-decoder function
$\sigma(\cdot)$	Activation function
$\mathbf{F}$	Flux Vector
$\mathbf{h}_{di}$	Embedding of dynamic node features at node i
$\mathbf{h}_{si}$	Embedding of static node features at node i
$\mathbf{u}_i^t$	Hydraulic variables at node i and time t
$\mathbf{x}$	Input variable
$\mathbf{x}_{di}$	Dynamic node features at node i
$\mathbf{x}_{si}$	Static node features at node i
$\mathbf{y}$	Output variable
$\tilde{\mathbf{J}}_{ij}$	Approximated Jacobian of numerical fluxes
a_i	Area of cell i [m^2]
e_i	Elevation at node i
g	Acceleration of gravity [m/s^2]
h	Water depth [m]
l_{ij}	Side length between cells i, j [m]
m_i	Manning's coefficient at node i
Q	Inflow discharge
q_x, q_y	Unit discharge components in x, y directions [m^2/s]
r_i	Roughness coefficient at node i
s_{0x}, s_{0y}	Bed slopes in x, y directions
s_{fx}, s_{fy}	Friction slopes in x, y directions
t	Time [s]
v	Celerity [m/s]
w_i^t	Water levels at time t and node i
G	Latent space dimension
H	Prediction horizon
O	Number of output variables
p	Number of input previous time steps

SUMMARY

Flooding is one of the most frequent and destructive natural hazards, accounting for significant human and economic losses every year. Flood hazard mapping allows to identify vulnerable areas by estimating water depth, extent, and intensity under specific scenarios. These maps are created via numerical models as they provide accurate flood simulations. However, they are computationally expensive, particularly for overland flow at high resolution. Data-driven methods based on neural networks offer a promising alternative, delivering faster predictions while maintaining high accuracy.

To provide a comprehensive perspective on deep learning for flood mapping, we first reviewed the state of the art across different applications, considering a range of flood types, spatial scales, and deep learning architectures. Our analysis shows that deep learning methods generally outperform both traditional numerical approaches and conventional machine learning in terms of speed and accuracy. However, most existing models are tailored to individual case studies, neglect the dynamic evolution of flood waves, and cannot transfer to new topographic settings and boundary conditions not seen during training. Furthermore, current approaches struggle to incorporate physical principles, represent hydraulic structures, and provide physically consistent methods for validating outputs, particularly in the context of uncertainty quantification. Finally, we highlight that dike-breach floods remain largely under-represented in the literature, despite their high uncertainty stemming from flood defence failures.

In this thesis, we introduce Graph Neural Networks (GNNs) as hydraulics-inspired surrogate models for simulating the spatio-temporal evolution of floods. While applicable to different flood types, our focus is on dike-breach floods due to their high uncertainty and particular relevance in the Netherlands and other low-lying areas. We build GNNs that are conceptually analogous to finite-volume methods used to solve the shallow water equations: finite-volume cells are treated as graph nodes, and flux exchanges are learned between adjacent cells by the model. The flood propagation in the proposed SWE-GNN model resembles hydraulics principles and enforces water to propagate only from cells with water. The model works in the same fashion as numerical solvers, auto-regressively predicting the evolution of the hydraulic states over time. By stacking multiple GNN layers, the model captures wider spatial dependencies without requiring small numerical time steps, theoretically needed for stability conditions. We also develop a multi-step-ahead loss function combined with curriculum learning that further stabilizes long-term predictions.

We propose a multi-scale GNN formulation that models flood dynamics across different spatial resolutions, enabling the capture of both local and large-scale propagation processes. Time-varying boundary conditions are incorporated through ghost cells, removing the need for separate numerical solvers to initialize simulations. To enhance generalization across unseen unstructured meshes and reduce training data demands, we enforce invariance principles, ensuring the model is independent of coordinate rotations.

This multi-scale approach proves both faster and more accurate than its single-scale counterpart. Our methods are validated on a suite of two-dimensional synthetic dike-breach simulations generated with a high-fidelity numerical solver. These datasets progressively increase in complexity by varying initial conditions, location of boundary conditions, size of the domain, computational mesh, and time-varying hydrographs used as boundary condition. Results demonstrate that the GNN generalizes well to unseen topographies, boundary configurations, and mesh configurations, without relying on inputs from numerical simulations. The models achieve a testing critical success index consistently higher than 70% in all datasets. The model also shows generalization to a real case study, dike ring 15 in the Netherlands, with only one fine-tuning simulation.

Finally, we extend the model to explicitly include hydraulic structures and to quantify uncertainty in flood hazard mapping of another real-world case study, dike ring 41 in the Netherlands. The framework is tested for large-scale uncertainty analysis with 10,000 scenarios. All simulations are completed in under 10 hours on a single GPU, corresponding to a speed-up of approximately 10,000 times with respect to the numerical solver, with over half of the scenarios maintaining plausible mass conservation. The combined scenarios are then used to produce probabilistic flood hazard maps, which assume equal likelihood of occurrence for each event. We also analyse the flood uncertainty for a given breach location and return period, showing that the model ensemble can provide better flooding estimates than the deterministic scenario.

This thesis highlights the potential of GNN-based surrogates for time-sensitive flood risk assessments under uncertainty. By demonstrating both accuracy and computational efficiency, it contributes to bridging the gap between complex hydraulic modelling and large-scale flood risk analysis. Despite being applied on dike-breach floods, the framework introduced in this thesis can readily be applied to fluvial and coastal floods without modifications. This open pathways for integrating surrogates into operational decision-making and future flood resilience planning.

SAMENVATTING

Overstromingen zijn een van de meest voorkomende en destructieve natuurrampen, die elk jaar aanzienlijke menselijke en economische verliezen veroorzaken. Met behulp van overstromingskaarten kunnen kwetsbare gebieden worden geïdentificeerd door de waterdiepte, omvang, en intensiteit in specifieke scenario's te schatten. Deze kaarten worden gemaakt met behulp van numerieke hydraulische modellen, omdat deze nauwkeurige simulaties opleveren. Ze zijn echter rekenkundig intensief, met name voor afstroming over land met hoge ruimtelijke resolutie. Datagestuurde methoden op basis van neurale netwerken bieden een veelbelovend alternatief, omdat ze snellere voorspellingen opleveren met behoud van hoge nauwkeurigheid.

Om een uitgebreid overzicht te geven van deep learning voor overstromingskaarten, hebben we eerst de stand van zaken bekeken voor verschillende toepassingen. Hierbij hebben we rekening hebben gehouden met verschillende soorten overstromingen, ruimtelijke schalen en deep learning-architecturen. Uit onze analyse blijkt dat deep learning-methoden over het algemeen beter presteren dan zowel traditionele numerieke benaderingen als conventionele machine learning wat betreft snelheid en nauwkeurigheid. De meeste bestaande modellen zijn echter afgestemd op individuele casestudy's, negeren de dynamische evolutie van overstromingsgolven en kunnen niet worden overgedragen naar nieuwe topografische omgevingen en randvoorwaarden die tijdens de training niet zijn beschouwd. Bovendien hebben de huidige benaderingen moeite om fysische principes te integreren, hydraulische infrastructuur weer te geven en fysisch consistente methoden te bieden voor het valideren van outputs, met name in de context van onzekerheidsquantificering. Ten slotte benadrukken we dat overstromingen door dijkdoorbraak nog steeds grotendeels ondervertegenwoordigd zijn in de literatuur over dit onderwerp, ondanks de grote onzekerheid die voortvloeit uit het falen van waterkeringen.

In dit proefschrift introduceren we Graph Neural Networks (GNN's) als op hydraulica geïnspireerde surrogaatmodellen voor het simuleren van de ruimtelijk-temporele evolutie van overstromingen. Hoewel ze toepasbaar zijn op verschillende soorten overstromingen, richten we ons op overstromingen door dijkdoorbraak vanwege hun grote onzekerheid en bijzondere relevantie in Nederland. We bouwen GNN's die conceptueel analoog zijn aan eindige-volumemethoden die worden gebruikt om de ondiepwatervergelijkingen op te lossen: eindige-volumecellen worden behandeld als grafiekknooppunten en fluxuitwisselingen worden geleerd tussen aangrenzende cellen door het model. De overstromingsverspreiding in het voorgestelde SWE-GNN-model is analoog aan hydraulische principes en zorgt ervoor dat water zich alleen verspreidt vanuit cellen die water bevatten. Het model werkt op dezelfde manier als numerieke modellen en voorspelt op autoregressieve wijze de evolutie van de hydraulische condities in de tijd. Door meerdere GNN-lagen te stapelen, legt het model bredere ruimtelijke afhankelijkheden vast zonder dat daarvoor kleine numerieke tijdstappen nodig zijn, die in theorie nodig zijn voor stabiliteitsvoorwaarden. We ontwikkelen ook een meerstaps-vooruit-verliesfunctie

in combinatie met curriculumleren, die langetermijnvoorspellingen verder stabiliseert.

We stellen een multischaal GNN-formulering voor die de dynamiek van overstromingen over verschillende ruimtelijke resoluties modelleert, waardoor zowel lokale als grootschalige propagatieprocessen kunnen worden vastgelegd. Tijdsafhankelijke randvoorwaarden worden geïntegreerd via spookcellen, waardoor er geen afzonderlijke numerieke oplossers nodig zijn om simulaties te initialiseren. Om de generalisatie over onbekende ongestructureerde meshes te verbeteren en de behoefte aan trainingsgegevens te verminderen, passen we invariantieprincipes toe, waardoor het model onafhankelijk is van coördinatenrotaties. Deze multischaalbenadering blijkt zowel sneller als nauwkeuriger te zijn dan zijn enkelvoudige tegenhanger. Onze methoden zijn gevalideerd op een reeks tweedimensionale dijkdoorbraaksimulaties die zijn gegenereerd met een zeer nauwkeurige numeriek model. Deze datasets worden steeds complexer door variatie in de beginvoorwaarden, de locatie van de randvoorwaarden, de grootte van het domein, het rekenmesh en de hydraulische hydrografen gebruikt als randvoorwaarden. De resultaten tonen aan dat het GNN goed generaliseert naar onbekende topografieën, grensconfiguraties en mesh-configuraties, zonder afhankelijk te zijn van input uit numerieke simulaties. De modellen behalen een kritieke succesindex die consequent hoger is dan 70% in alle datasets. Het model toont ook een generalisatie naar een echte casestudy, dijkring 15 in Nederland, met slechts één fijnafstemming simulatie.

Ten slotte breiden we het model uit om hydraulische constructies expliciet mee te nemen en om de onzekerheid in overstromingsgevaarkaarten van een andere echte casestudy, dijkring 41 in Nederland, te kwantificeren. Het raamwerk wordt getest in een grootschalige onzekerheidsanalyse met 10.000 scenario's. Alle simulaties worden in minder dan 10 uur voltooid op een enkele GPU, wat overeenkomt met een versnelling van ongeveer 10.000 keer ten opzichte van de numeriek model, waarbij meer dan de helft van de scenario's een plausibele massaconservering behoudt. De gecombineerde scenario's worden vervolgens gebruikt om probabilistische overstromingsgevaarkaarten te produceren, waarbij wordt uitgegaan van een gelijke kans op het optreden van elke gebeurtenis. We analyseren ook de onzekerheid van overstromingen voor een bepaalde breuklocatie en terugkeerperiode, waaruit blijkt dat het modelensemble betere overstromingsschattingen kan geven dan het deterministische scenario.

Dit proefschrift benadrukt het potentieel van op GNN gebaseerde surrogaten voor tijdgevoelige overstromingsrisicobeoordelingen onder onzekere omstandigheden. Door zowel nauwkeurigheid als computationele efficiëntie aan te tonen, draagt het bij aan het overbruggen van de kloof tussen complexe hydraulische modellering en grootschalige overstromingsrisicoanalyse. Hoewel het in dit proefschrift geïntroduceerde raamwerk wordt toegepast op overstromingen door dijkdoorbraken, kan het zonder aanpassingen ook worden toegepast op overstromingen door rivieren en aan de kust. De bevindingen openen wegen voor de integratie van surrogaten in operationele besluitvorming en toekomstige planning van overstromingsbestendigheid.

1

INTRODUCTION

Water is the driving force of all nature.

Leonardo da Vinci

1.1. FLOOD RISK MANAGEMENT

Flooding is among the most frequent and destructive natural hazards, causing substantial human and economic losses each year (Jonkman & Vrijling, 2008). Its impact is expected to intensify as climate change drives more frequent and severe precipitation events (Arias et al., 2021). To understand the risks to lives and infrastructure, governments are mandated to produce flood hazard maps (European Union, 2007). These maps estimate the likelihood and magnitude of flooding and are used in spatial planning decisions, for example to identify zones at high risk or regulate the placement of critical infrastructure.

The generation of flood hazard maps is a multi-step process dependent on the flood type and associated physical processes (Figure 1.1). This thesis focuses on dike-breach floods, a type of flood caused by the failure of flood defences under extreme water levels. Low-lying regions such as the Netherlands, heavily reliant on dikes, face catastrophic consequences when such defences fail. The severity is compounded not only by the intensity of the flooding itself but also by the false sense of security these defences can create, often leading to increased development in areas presumed to be safe (Di Baldassarre et al., 2018). The development of a flood hazard map begins with a comprehensive analysis of the case study area, involving the collection and integration of hydrological, topographical, and land use data. Historical flood records and the estimated probabilities of failure of existing flood defences are then used to assess the likelihood of extreme flood events, expressed in terms of return periods.

In dike-breach floods, the main driving forces are intense rainfall events, river discharges, and storm surges, each of which can result in elevated water levels in rivers, lakes, or coastal water bodies. For rainfall-induced scenarios, the first step in the modelling chain involves converting precipitation data into discharge hydrographs at the outlet of a

1

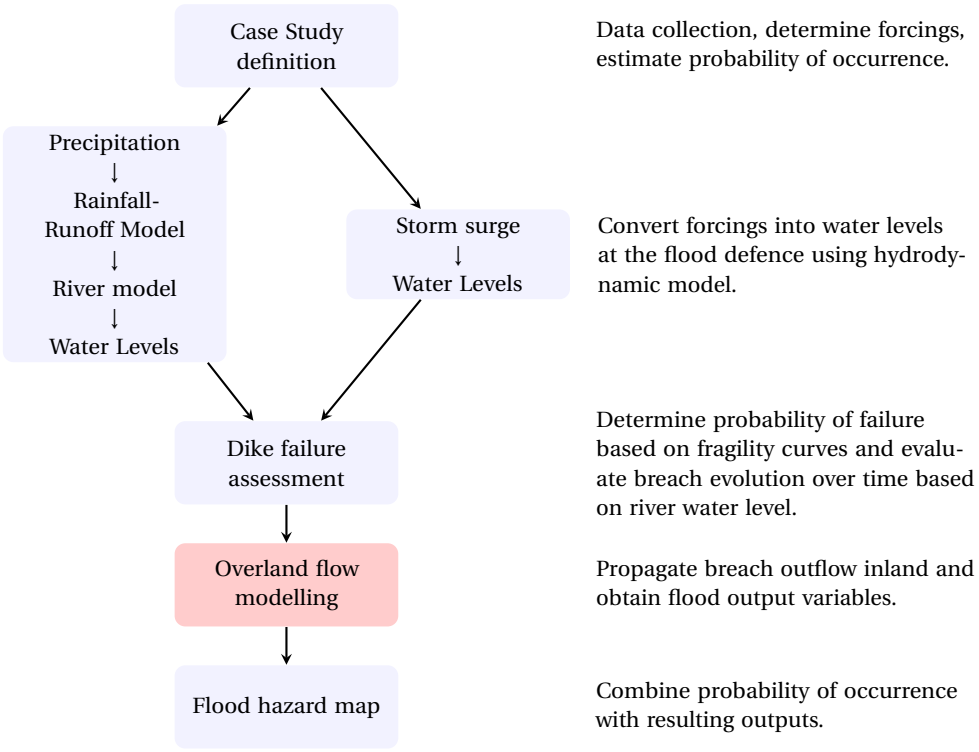


Figure 1.1: Flow chart procedure to create a dike-breach flood hazard map. The focus of this thesis is on the overland flow modelling component.

catchment using rainfall-runoff models (Beven, 2012). Alternatively, historical discharge records at a given location may be employed when available. These hydrographs serve as upstream boundary conditions for hydraulic models that simulate water level and flow velocity dynamics along the river system. In the case of storm-surge-induced dike breaches, the primary drivers are low atmospheric pressure and strong winds, which can be translated into water level time series. These are used as boundary conditions for coastal inundation models to estimate wave heights and water levels along the flood defences.

A breach occurs when the load on a flood defence is lower than its resistance. In practice, this happens when water levels exceed a critical threshold specific to a flood defence structure. This threshold is defined using fragility curves, which represent the conditional probability of structural failure as a function of hydraulic loading. The temporal evolution of the breach can be modelled using empirical , parametric, or physically based approaches (Schmitz et al., 2021; Verheij & Hydraulics, 2003; Wu & Li, 2017), which estimate both the progression of the breach and the resulting breach outflow hydrograph. Following breach initiation, an overland flow model is employed to simulate the spread of water across the floodplain (Figure 1.2). The outputs, such as water depths, velocities, and spatial extent, can be used to generate flood hazard maps by linking them to probabilities

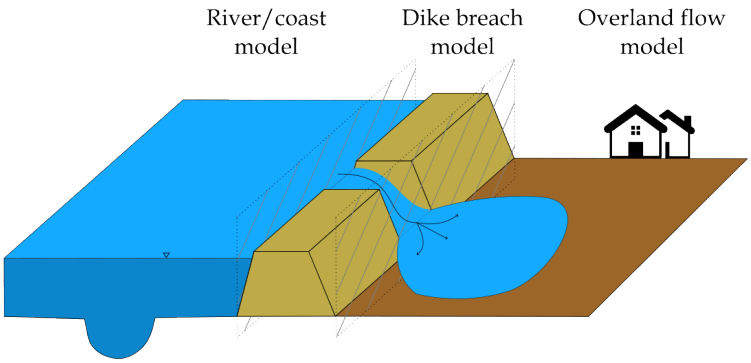


Figure 1.2: Schematics of a dike breach flood. High water levels in the river or sea cause the failure of the flood defence and lead to the flood propagation in the breached area.

of occurrence derived from statistical analysis of input conditions.

Overland flow modelling represents the most computationally intensive component of the flood hazard mapping workflow, as it typically relies on two-dimensional (2D) numerical simulations that solve the shallow water equations, a system of partial differential equations describing the motion of water in shallow domains. Although these methods are robust and physically accurate, achieving a balance between speed and accuracy remains a major challenge (Costabile et al., 2017). Moreover, multiple simulations are needed to account for different breach locations, estimated water levels, breach conditions, and other uncertain geotechnical and hydrological variables. This substantial computational demand limits the number of simulations that can be performed for a given case study, often resulting in the use of a single deterministic model that fails to capture the inherent uncertainties associated with flood events. To address this, several acceleration strategies have been explored, including parallel computing (e.g., Glenis et al., 2013; Ming et al., 2020; Zhang et al., 2014) and the adoption of simplified hydraulic models (e.g., Sridharan et al., 2021; Zhao, Balstrøm, et al., 2021). Nevertheless, parallel computing typically requires high-performance infrastructure, while simplified models may be inadequate for capturing rapidly evolving and complex flow dynamics, such as those observed in dam-break (Costabile et al., 2017; Prestininzi, 2008).

The scope of this thesis is to develop an alternative approach for overland flow modelling that achieves both speed and accuracy by leveraging deep learning techniques. The proposed data-driven surrogate model is designed to support real-time applications and probabilistic flood analyses, where large numbers of simulations or rapid computations are required.

1.2. DEEP LEARNING FOR FLOOD MAPPING: KNOWLEDGE GAPS

Deep learning (DL) refers to a class of algorithms capable of automatically learning complex feature representations directly from raw data without the need of manual feature engineering (LeCun et al., 2015). DL models are built upon a hierarchical architecture of

non-linear transformations, where each layer progressively extracts more abstract and higher-level features from the input. This multi-level representation learning enables the model to identify latent structures and patterns within the data, ultimately improving predictive accuracy and generalisation performance.

Deep learning has been applied to a variety of tasks, including flood susceptibility assessment, inundation modelling, and hazard mapping, often achieving either faster or more accurate predictions compared to traditional physics-based methods (Bentivoglio et al., 2022) (see also Chapter 3). Despite these promising developments, several key research gaps remain, which this thesis aims to address.

1. The majority of existing DL models predict maximum water depths by generalising over varying boundary conditions, such as rainfall, within a fixed spatial domain. While effective in a given case study, these models require retraining when applied to different geographical areas, thereby limiting their scalability and applicability as surrogate models.

2. Most approaches focus solely on predicting static outcomes, such as maximum water depth, and neglect the temporal dynamics of flood events. As a result, they provide limited insight into the spatio-temporal evolution of flooding, which is essential for applications like early warning systems, real-time response, and evacuation planning.

3. DL models cannot explicitly include hydraulic structures, which strongly influence the behaviour of floods and are modelled in real case studies.

4. There is a lack of research evaluating the operational viability of deep learning models for practical flood risk management tasks, such as probabilistic flood mapping.

These limitations highlight the need for more generalisable, temporally-varied, and operationally feasible DL-based solutions. In this thesis, we define **generalization** as the ability of a model to be tested on different topographies and boundary conditions than the ones it was trained on.

Although various deep learning approaches have demonstrated success in fluid dynamics applications (Fortunato et al., 2022; Kochkov et al., 2021; Pfaff et al., 2021), their direct application to flood modelling remains challenging due to fundamental differences in the nature of the problem. Conventional fluid dynamics problems typically assume a continuous presence of fluid throughout the computational domain. In contrast, flood events are characterised by highly transient and spatially variable flow conditions, often originating from discrete sources, such as dike breaches or riverbank overflows, and receding over time. Consequently, large portions of the domain may remain dry for part or even the entirety of the simulation period. This temporal and spatial intermittency introduces complexities that are not typically encountered in standard fluid dynamics scenarios. Deep learning models intended for flood prediction must therefore be designed to accommodate the presence of both wet and dry regions, as well as the evolving boundary conditions associated with the flood wave propagation. These unique characteristics limit the direct applicability of existing fluid dynamics models from the literature. The methodologies proposed in this thesis explicitly address these challenges by incorporating mechanisms to account for the dynamic nature of flooding phenomena.

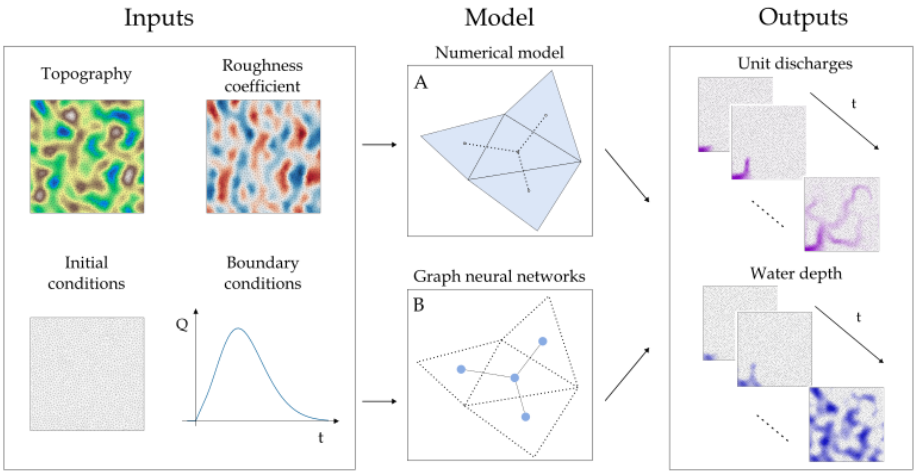


Figure 1.3: Schematics of the inputs and outputs of the flood numerical model (A) and the graph neural network models (B) developed in this thesis to simulate overland flow. The schematics highlights that both numerical and deep learning model require the same inputs (terrain elevation, roughness, initial and boundary conditions) and produce the same outputs (water depth and discharges).

1.3. RESEARCH QUESTIONS

This thesis investigates the potential of deep learning for fast and accurate spatio-temporal simulations of overland floods. We employ consolidated principles of numerical methods to improve the model generalization over multiple case studies, mesh configurations, initial conditions, and boundary conditions. The general research question is:

How can we leverage deep learning models for spatio-temporal flood prediction across varying case studies, boundary conditions, and mesh configurations in support of hazard mapping?

To address this question, we employ Graph Neural Networks (GNNs) as the core modelling framework, since they are conceptually analogous to finite-volume methods for flood modelling and naturally operate on irregular meshes. This gives them a strong relational inductive bias that enhances their transferability to diverse case studies and makes them appealing for surrogate modelling. Their flexible structure also enables to leverage analogies with numerical methods to enforce hydraulic principles and ensure physical consistency. The main research question is analysed with four key sub-questions, which are addressed in various chapters:

1. **RQ1:** What is the state-of-the-art of deep learning for flood mapping?(Chapter 3)
2. **RQ2:** How can deep learning surrogate models include hydraulic properties to support the generalization of spatio-temporal flood predictions in unseen geographical areas?(Chapter 4)

3. **RQ3:** What principles from numerical solvers can be exploited to improve the generalization of graph neural networks across boundary conditions and mesh configurations without relying on numerical model initialization?(Chapter 5)

4. **RQ4:** How can graph neural networks be used to estimate uncertainty for flood hazard mapping in case studies with hydraulic structures?(Chapter 6)

1.4. STRUCTURE OF THE THESIS

The thesis is organised into seven chapters, four of which (Chapters 3, 4, 5, 6) correspond to published or submitted journal papers, containing the literature review, core methodology for the development of the deep learning model, and application of the model.

Chapter 2 provides an overview of the categorization of floods and existing approaches to simulate them, both with numerical and conceptual models. It then presents deep learning models under the lens of inductive biases. This chapter provides a foundation to understand and motivate the design choices of the developed methods.

Chapter 3 provides an extensive review of the state-of-the-art of deep learning methods for flood mapping, spanning over different types of applications, types of floods, scales, and employed models. The research gaps found in flood modelling are then used as a base to derive the research questions of this thesis.

Chapter 4 establishes Graph Neural Networks (GNNs) as an effective architecture for emulating spatio-temporal flood simulations. It presents the SWE-GNN model, a GNN inspired by finite volume methods for flood modelling in which the learned propagation rules reflect hydraulic biases. The chapter highlights that the design of the GNN strongly influences performance and generalization to different case studies. The chapter concludes with a comparative evaluation against alternative deep learning models on synthetic benchmarks, demonstrating the improved generalisation to unseen initial conditions of the proposed approach.

Chapter 5 examines the main limitations of Chapter 4, namely the lack of generalization over boundary conditions, the dependence on regular grid structures, and the reliance on initial conditions from the original simulator. It proposes as solutions a multi-scale approach (mSWE-GNN) to improve scalability by allowing the model to operate efficiently across different spatial resolutions. It also proposes the use of ghost cells to generalize across boundary conditions and a rotation-invariant formulation to improve generalization over different mesh configurations. Finally, the developed model is applied to a real case study showcasing its effectiveness.

Chapter 6 demonstrates how the developed mSWE-GNN model enables probabilistic flood hazard mapping reducing the computational cost up to 10,000 times, while maintaining accuracy up to 40% in average relative mass error. It also integrates hydraulic structures in the model and a validation procedure to assess the reliability of the model's predictions based on principles of mass conservation.

Finally, Chapter 7 presents the main conclusions of this thesis, along with recommendations for future research, such as how to adapt the proposed methodology to different types of floods.

This thesis was developed as part of AidroLab at TU Delft, an artificial intelligence

group focused on solving water management issues with the help of deep learning techniques, mainly based on graph neural networks. Within this group, this thesis advances the use of spatio-temporal graph neural networks in hydraulic systems.

2

BACKGROUND

Real knowledge is to know the extent of one's ignorance.

Confucius

This chapter provides a comprehensive overview of the core concepts and methodologies at the base of this thesis, setting the stage for the contributions that follow. It is structured into three main sections: flood management, flood modelling, and deep learning. In the first section, we categorize floods based on their type, mapping application, and spatial scale, so that we can define the scope of the thesis within flood management. The second section focuses on the methods used to simulate floods. We begin with physically based numerical models, along with different formulations of the underlying equations of motion. Then, we review simplified modelling approaches and their assumptions. The discussion highlights the trade-offs involved in choosing between model fidelity and computational efficiency. The final section explores deep learning models under the lens of inductive biases. We describe the primary types of neural network architectures used for flood modelling, such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and motivate which structures are more appropriate for simulating floods.

This chapter is an adapted version of:

Bentivoglio, R., Isufi, E., Jonkman, S. N., & Taormina, R. (2022). Deep learning methods for flood mapping: A review of existing applications and future research directions. Hydrology and Earth System Sciences, 26, 4345–4378. https://doi.org/10.5194/hess-26-4345-2022.

and

Bentivoglio, R., Isufi, E., Jonkman, S. N., & Taormina, R. (2023). Rapid spatio-temporal flood modelling via hydraulics-based graph neural networks. Hydrology and Earth System Sciences, 27(23), 4227–4246. https://doi.org/10.5194/hess-27-4227-2023.

2.1. FLOOD RISK MANAGEMENT

Floods are defined as an overflow of water in otherwise dry land. Flood risk management aims to reduce the negative consequences of flooding on people, the economy, cultural heritage, and the environment, via the integration of structural measures (e.g., levees, retention basins, dams) and non-structural approaches (e.g., land-use planning, early warning systems, emergency preparedness, and insurance schemes) (Schanze, 2006).

As described in Chapter 1, the main tool to understand flood risk is via flood mapping. This helps to identify areas prone to flooding, quantify the intensity and probability of flood events, and assess potential damages and losses. We develop flood maps with flood simulations, which model the dynamics of flood waves under different conditions.

While there exist several categorizations of flood risk management, we focus on types of floods, mapping applications, and spatial scales. We also describe the conditioning variables that affect the uncertainty of floods.

2.1.1. TYPES OF FLOODS

We can distinguish flooding depending on how, why, and when it occurs:

- **River floods** are caused by extensive precipitation over long periods, causing the river to overflow its banks, ultimately inundating the neighbouring areas. This process is generally slow and can last for several days (Serinaldi et al., 2018).

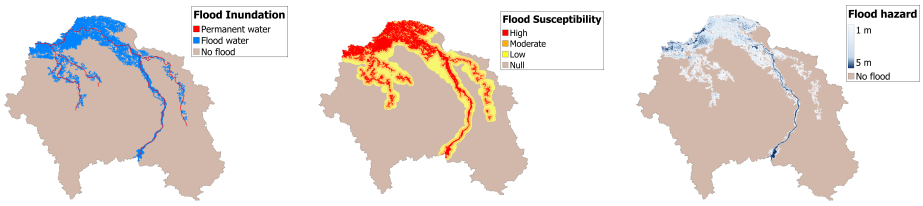
- **Flash floods** are caused by short but intense rainfall or sudden melting of snow (Sikorska et al., 2015). They are rapid and intense floods, typical of mountain and steep catchments. Flash floods are often coupled with other hazards such as debris flows (Destro et al., 2018) and landslides (Ávila et al., 2016).

- **Coastal floods** are caused by extreme meteorological conditions, which increase the water level in large bodies of water, due to a combination of low atmospheric pressure and strong winds. They occur near oceans, seas, or large lakes and we include in this category also tsunamis, although they are generated by geological phenomena such as earthquakes.

- **Urban floods** are caused by the failure of drainage from a sewer system, due to extreme precipitation, resulting in the overflow of those pipes. Depending on the city position and topography, these floods can also be affected by all the other types of floods.

- **Pluvial floods** are caused by the failure of a drainage system due to intensive precipitation. They are generally categorized as urban floods in urban environments or river floods if they also feature rainfall-driven river overflows.

- **Dam break and dike breach floods** are caused by the failure of flood protection structures, due to extreme flood events or management issues. The uncertainty in if, where, and how a defence will fail further increases the unexpectedness of these phenomena (Kamrath et al., 2006).



(a) Flood inundation (b) Flood susceptibility (c) Flood hazard

Figure 2.1: Examples of the types of flood maps analysed for a representative area: (a) shows a possible flood inundation map; (b) a flood susceptibility map; and (c) a flood hazard map, as defined in this thesis.

2.1.2. FLOOD MAPPING APPLICATIONS

We distinguish four types of flood mapping: flood susceptibility, flood inundation, flood hazard, and flood risk. In this categorization, we exclude applications which do not result in maps, such as water level forecasts. The mapping applications can be described as follows:

- **Flood inundation maps** determine the extent of a flood, during or after it has occurred (see Fig. 2.1a). Flood inundation maps represent flooded and non-flooded areas. This application is used for post-flood evacuation, protection planning, and for damage assessment. These maps can then be used also as calibration data for other applications such as flood susceptibility or flood hazard mapping. Flood images are obtained through remote-sensing techniques and processed by histogram-based models (e.g., Manjusree et al., 2012; Martinis et al., 2009), threshold models (e.g., Cian et al., 2018), and machine learning models (e.g., Hess et al., 1995; Ireland et al., 2015).

- **Flood susceptibility maps** determine the tendency to flooding of a study area based on its physical characteristics (see Fig. 2.1b). This measure is only qualitative and does not evaluate any flood variable. However, it can provide reliable information when no quantitative data is available and can be used to easily assess areas at risk at large scales. Flood susceptibility mapping is performed by considering topographical, geographical, and meteorological factors (such as altitude, slope, lithology, land use, and rainfall) and comparing their spatial distribution with past flood events. This is done with multivariate analysis (e.g., Tehrany et al., 2014; Youssef et al., 2016) and multi-criteria decision analysis (e.g., Kazakis et al., 2015; Mahmoud & Gan, 2018).

- **Flood hazard maps** measure the water depth and extent across a flooded area (see Fig. 2.1c). Hazard maps consider also different return periods of the floods and, thus, the probability of a certain event. The latter is determined through a statistical analysis based on the frequency and intensity of floods (Bobée & Rasmussen, 1995). We will refer to flood hazard also when the water depths are estimated independently of the return periods. Flood hazard can also provide a measure of the flow

velocities and of other variables derived from the combination of velocity and water depth, such as flood intensity. Flood hazard maps are compiled based on outputs by numerical hydraulic models, which simulate flood events by discretizing the governing equations and the computational domain (e.g., Teng et al., 2017).

- **Flood risk maps** estimate the expected annual flood losses by aggregating flood hazard maps over different return periods with exposure maps and vulnerability functions (de Moel et al., 2009). The expected losses can be measured both in terms of monetary losses or human lives (Jonkman & Vrijling, 2008).

2.1.3. SPATIAL SCALE

The importance of flood processes and the resolution of the flood maps varies with their spatial scale. Following de Moel et al. (2015), we distinguish between local, regional, national, and supra-national scales. The choice between scales is often subjective, but here follows a rational categorization:

- **Local** scale refers to small study areas, such as towns or a specific river stretch. If a measure of the study area is given, we consider it in this category if the area is smaller than 100km^2.
- **Regional** scale considers a specific province, watershed or large city. Study areas smaller than 100000km^2 belong to this scale.
- **National** scale refers to assessments of entire countries, for which consistent (national) data are present. To exclude small countries, the study area must be greater than 100000km^2.
- **Supra-national** scales concern assessments of an entire continent or the globe.

This thesis focuses on flood hazard maps for dike-breach floods at local and regional scales. The following section describes the existing methodologies to these products using flood models.

2.1.4. UNCERTAINTY IN FLOOD RISK MANAGEMENT

Uncertainty in flood risk management arises not only from the natural variability of the hydrological and hydraulic processes but also from limitations in data, models, and assumptions about future conditions (Domeneghetti et al., 2013). Aleatoric components are generally irreducible (we can only describe them probabilistically), whereas epistemic components are reducible with better data, calibration, and models. Based on Hall and Solomatine (2008), we can distinguish four main sources of uncertainty:

- **Hydrological and hydraulic uncertainty**: represents the variability of precipitation, snow melt, atmospheric conditions, and inflows (de Moel et al., 2014). Limited observational records, measurement errors, and the inherent randomness of extreme weather contribute to uncertainty in the magnitude and timing of floods. Mainly aleatoric, with epistemic elements from short records, measurement error, and non-stationarity.

- **Geotechnical uncertainty**: reflects the variability in structural and hydrogeological properties of the soil, especially in correspondence of earthen flood defences. This influences the likelihood of failure as well as its modality (Westerhof et al., 2023). Mainly epistemic, due to sparse/indirect characterization and scaling; may include aleatoric micro-variability.

- **Model uncertainty**: arises when representing flood processes through models, as all models rely on several assumptions which might deviate from reality. Moreover, the data employed by the models may not capture small-scale features and the parametrization of the roughness coefficient introduces further variability (Hall et al., 2011; Savage et al., 2016). Primarily epistemic (model form and parameters).

- **Socio-economic uncertainty**: relates to the estimation of potential flood impacts (De Moel et al., 2012). Future land use, infrastructure development, demographic changes, and climate change are difficult to predict, and vulnerability functions linking hazard intensity to damages vary widely. Largely epistemic, reflecting scenario assumptions, with some aleatoric-like variability.

Uncertainty can be quantified via probabilistic flood hazard maps (Apel et al., 2006). These entail running multiple scenario configurations with different values of uncertain variables. When the uncertainty is represented by a distribution, Monte Carlo sampling methods are the most used to directly quantify the probability of occurrence of a given scenario and its consequences.

2.2. FLOOD MODELLING

The behaviour of a flood inundation can be represented via flood models. These are mathematical descriptions of basic principles up to complete processes that are associated to the movement of flood water. We can differentiate flood models in two main categories: hydrodynamic (or numerical) models and simplified (or conceptual) models (Teng et al., 2017).

Numerical models are based on the discretization and solution of the physical equations that govern flow. One-dimensional (1D) models represent a channel through a mono-dimensional curvilinear axis (Cunge et al., 1980). They are fast and provide accurate results for the channel, but are less for floodplains outside the channel (Tayefi et al., 2007). Two-dimensional (2D) models represent the computational domain by unstructured meshes (e.g., Alcrudo & Garcia-Navarro, 1993) or structured meshes (rasters) (e.g., Bates & De Roo, 2000). Mesh-based finite volume methods are the most used in practice (e.g., Dottori et al., 2021). Those methods give more accurate results than 1D models, but require longer computational time (Costabile et al., 2015). There are also 1D-2D coupled models connecting a 1D domain to a 2D domain (e.g., Finaud-Guyot et al., 2011). These are used for urban floods to include the urban drainage system (Leandro et al., 2009) or in river floods to include the river channel as a 1D longitudinal element (Morales-Hernández et al., 2016). Three-dimensional (3D) models represent the domain with meshes (Lane et al., 1999) or particles (Vacondio et al., 2012). They are particularly useful for exceptional events such as tsunamis and dam breaks in which vertical accelera-

tion cannot be neglected. However, they are computationally demanding and thus less used than the other models.

Conceptual models are based on simplified hydraulic concepts and do not entail numerical simulations. They are mostly based on simple water spreading rules (Lhomme et al., 2008) or simple hydraulic equations, like Manning's formula (Song et al., 2016). However, because of the lack of simulation and physics, they generally predict only maximum water depths, disregarding any flow velocities nor their evolution in time.

Depending on the modelling approach, a trade-off must often be made between accuracy, computational efficiency, adherence to physical principles, and the range of output variables that can be predicted. Physically based numerical models, while highly accurate and grounded in fundamental equations, are typically computationally intensive. On the other hand, conceptual or simplified models offer significant improvements in speed and tractability but often at the expense of physical realism and general applicability. In this thesis, we argue that deep learning methods can offer a promising intermediate solution between these two extremes.

The following sections provide a detailed examination of these modelling approaches, focusing on their governing equations, underlying assumptions, and computational mechanisms. The deep learning models developed in this thesis focus on 2D surface flow dynamics. Consequently, the modelling background is also constrained to 2D representations of flood processes.

2.2.1. 2D NUMERICAL MODELLING

When assuming negligible vertical accelerations, floods can be modelled via the shallow-waters equations (SWE) (Vreugdenhil, 1994). These are a system of hyperbolic partial differential equations that describe the behaviour of shallow flows by enforcing mass and momentum conservation. The two-dimensional SWE can be written as

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \mathbf{F} = \mathbf{s}, \tag{2.1}$$

with

$$\mathbf{u} = \begin{pmatrix} h \\ q_x \\ q_y \end{pmatrix}, \mathbf{F} = \begin{pmatrix} q_x & q_y \\ \frac{q_x^2}{h} + \frac{gh^2}{2} & \frac{q_x q_y}{h} \\ \frac{q_x q_y}{h} & \frac{q_y^2}{h} + \frac{gh^2}{2} \end{pmatrix}, \mathbf{s} = \begin{pmatrix} 0 \\ gh(s_{0x} - s_{fx}) \\ gh(s_{0y} - s_{fy}) \end{pmatrix}, \tag{2.2}$$

where $\mathbf{u}$ represents the conserved variable vector, $\mathbf{F}$ the fluxes in the x and y directions, and $\mathbf{s}$ the source terms. Here, $h[m]$ represents the water depth, $q_x = uh[m^2/s]$ and $q_y = vh[m^2/s]$ are the averaged components of the discharge vector along the x and y coordinates, respectively, and $g[m/s^2]$ is the acceleration of gravity. The source terms in $\mathbf{s}$ depend on the contributions of bed slopes $\mathbf{s}_0$ and friction losses $\mathbf{s}_f$ along the two coordinate directions.

The SWE cannot be solved analytically unless some simplifications are enforced. Thus, they are commonly solved via spatio-temporal numerical discretizations, such as the finite volume method (e.g., Alcrudo & Garcia-Navarro, 1993). This method discretizes the spatial domain using meshes, i.e., geometrical structures composed of nodes, edges,

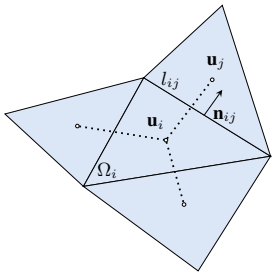


Figure 2.2: Schematic representation of an arbitrary triangular volume mesh, composed of a finite-volume cell Ω_i and its neighbouring cells. Vectors $\mathbf{u}_i$ and $\mathbf{u}_j$ represent the cells' hydraulic variables, while l_{ij} and $\mathbf{n}_{ij}$ corresponds, respectively, to the length of the mesh side and the outward unit normal vector, between cells i and j.

and faces. We consider each finite volume cell as represented by its centre of mass, where the hydraulic variables, h, q_x, and q_y, are defined (Fig. 2.2). The governing equations are then integrated over the cells, considering piece-wise constant variations, i.e., the value of the variables at a certain time instant is spatially uniform for every cell. The SWE can be discretized in several ways both in space and time (e.g., Petaccia et al., 2013; Xia et al., 2017) but we focus on a first-order explicit scheme with a cell-centred discretization of the source terms. For an arbitrary volume Ω_i and a discrete time step Δt, the SWE (eq. (2.1)) can be re-written as:

$$\mathbf{u}_i^{t+\Delta t} = \mathbf{u}_i^t + \left(\mathbf{s}_i - \sum_{j=1}^{N_i} (\mathbf{F}\cdot\mathbf{n})_{ij} \frac{l_{ij}}{a_i} \right) \Delta t \tag{2.3}$$

with $\mathbf{u}_i^t$ the hydraulic variables at time t and cell i, a_i the area of the i^{th} cell, N_i the number of neighbouring cells, l_{ij} the length of the j^{th} side of cell i, $\mathbf{s}_i$ the source terms, $\mathbf{n}_{ij} = [n_{xij}, n_{yij}]$ the outward unit normal vector in the x and y directions for side ij, and $(\mathbf{F}\cdot\mathbf{n})_{ij}$ the numerical fluxes across neighbouring cells.

Calculating the numerical fluxes across an interface of neighbouring cells results in a so-called Riemann problem, i.e., two flat surfaces separated by a jump discontinuity. This problem can be solved locally with an analytical form (Toro, 2024). However, in practice, approximations such as Roe (1981) are most commonly used because of the computational advantages. This leads to a conservation constraint at the cells interface which implies that

$$\delta(\mathbf{F}\cdot\mathbf{n})_{ij} = \tilde{\mathbf{J}}_{ij}\delta\mathbf{u}_{ij} \tag{2.4}$$

where $\tilde{\mathbf{J}}_{ij}$ is the approximation of the Jacobian of the fluxes, for example using Roe's scheme, and $\delta\mathbf{u}_{ij} = \mathbf{u}_j - \mathbf{u}_i$ indicates the difference in hydraulic variables between cells i and j (Martínez-Aranda et al., 2022).

In numerical models with explicit discretization, stability is enforced by satisfying the Courant–Friedrichs–Lewy (CFL) condition, which imposes the numerical propagation speed to be lower than the physical one (Courant et al., 1967). Considering v as

propagation speed, the Courant number C can be evaluated as

$$C = \frac{v\Delta t}{\Delta x}, \tag{2.5}$$

where Δt and Δx represent the time step and the mesh size. Assuming an Eulerian setting, in which the mesh does not change in time, this condition forces Δt to be sufficiently small, to avoid a too-fast propagation of water in space that would result in a loss of physical consistency. Small time steps imply increasing number of model iterations, which slow down numerical models over long time horizons.

Because solving the full shallow-water equations requires a lot of computational effort, research has focused on speeding up these models while preserving accuracy. This resulted in two parallel and non-excluding research directions: 1) removing complexity from the physical equations and 2) parallelizing the solution of the equations.

LOWER COMPLEXITY MODELS

The momentum terms in the shallow-water equations can be divided into several components, each responsible for a specific physical mechanism. Focusing on the x-component of the momentum in Eq. (2.2) and simplifying some terms, we obtain:

$$\underbrace{\frac{\partial q_x}{\partial t}}_{\text{Local acceleration}} + \underbrace{u\frac{\partial q_x}{\partial x} + v\frac{\partial q_x}{\partial y}}_{\text{Convective acceleration}} + \underbrace{gh\frac{\partial h}{\partial y}}_{\substack{\text{Pressure gradient}\\ \text{(inertial term)}}} = \underbrace{gh\frac{\partial (s_{0x} - s_{fx})}{\partial x}}_{\text{Source terms}} \tag{2.6}$$

where local acceleration refers to the change in the state variable over time; convective acceleration to the transport of momentum due to velocity gradients in space; inertial terms to the pressure gradients driven by water surface elevation; and source terms to the effects due to bed and friction slope. Depending on which of these terms are neglected, the full SWE can be simplified into various reduced forms, each with specific assumptions and limitations:

- **Inertial wave approximation**: this formulation assumes that convective terms are negligible relative to the other terms (Hunter et al., 2007). Under this approximation, and assuming a rectangular channel, the equations can be simplified to yield an explicit expression for the time evolution of discharge in one direction (Bates et al., 2010). This method is generally applicable in predominantly subcritical flow conditions but may yield poor results in regions with sharp spatial velocity gradients, such as urban environments (Costabile et al., 2017).

- **Diffusive wave approximation**: here, both local and convective acceleration terms are neglected, under the assumption that they contribute insignificantly compared to pressure and source terms (Hunter et al., 2005). This simplification allows for the combination of the mass and momentum conservation equations into a single diffusion-like equation, where the diffusion coefficient varies in space and time. While suitable for subcritical flows, this approximation can suffer from numerical instabilities and may require very small time steps in regions with steep gradients, such as in dam-break scenarios.

- **Kinematic wave approximation**: this most simplified form assumes that the flow is uniform and steady, thereby neglecting both acceleration and inertial terms entirely (Weinmann & Laurenson, 1979). The resulting formulation equates the bed slope with the friction slope, implying that the water surface follows the riverbed. This approximation is generally valid only in conditions where spatial and temporal variations in water depth are minimal and is typically used in large-scale hydrological models.

In conditions where these formulations hold, the computational speed-ups can go up to several times faster, based on the grid size, grid structure, and complexity of the flood wave (Hunter et al., 2005; Teng et al., 2017).

PARALLELIZATION

In explicit numerical schemes, the computations performed for each cell, at a given time step, are all independent from each other, since the inputs are all given a priori (see Eq. (2.3)). This implies that all computations can be executed in parallel. Contrarily, implicit schemes require the joint computation of the solution at each cell simultaneously, since, by definition, there are terms that depend on the solution at $t + \Delta t$ on both sides of the equation. This entails recursively solving a system of equations until a given convergence criterion. For this reason, the model does also not require the same stability conditions of explicit schemes (LeVeque et al., 2002). This makes implicit-scheme-based models more accurate and more stable but also less parallelizable.

Parallelization can be exploited in different ways depending on the type of processing unit. On computational processing units (CPUs), Open Multi-Processing (OpenMP) and Message Passing Interface (MPI) allow to parallelize programs on shared-memory or distributed-memory systems, respectively. In OpenMP, all threads operate in a common memory space, while in MPI, each process runs independently and communicates with others via explicit message passing. Contrarily, graphical processing units (GPUs) are a hardware specialized for data-parallel tasks and offer computational throughput using thousands of lightweight threads.

All three approaches have been widely reported in literature for flood modelling, reaching speed-ups of up to 2 orders of magnitude faster than their non-accelerated counterparts (Liu et al., 2018; Neal et al., 2010). Despite these methods work also for unstructured grids, most computational advantages come from the use of structured grids, which allows them to avoid explicit memory saving of the neighbourhood of each cell (Petaccia et al., 2016).

2.2.2. CONCEPTUAL FLOOD MODELS

Conceptual models employ simplified physical equations or concepts to determine the flood evolution of maximum water depths.

Storage-cell-based methods like cellular automata (Dottori & Todini, 2010; Guidolin et al., 2016) determine the spreading of water over a gridded domain using a simple set of rules. These can either employ Manning's formula to calculate the fluxes across cells or a weighted version of them. In these cases, the main limitation are the time steps, which either need to be manually calibrated or follow non-linear relations with the grid size, making the simulation potentially slower than with the full SWE for small grids.

Other approaches such as rapid flood spreading method (RFSM) avoid physical equations but instead focus on physically-based spreading principles (Lhomme et al., 2008). In RSFM, an input flood volume is spread through a domain by progressively filling interest areas, according to a simple bathtub relation, and overflowing into neighbouring areas when the corresponding flood levels exceed a given flood defence height that surrounds the area of interest. There are also approaches based on the digital elevation model (DEM) of a given area, such as height above nearest drainage (HAND) (Nobre et al., 2011), which determine the maximum flood levels by intercepting linear surface slopes with the elevation map, until a given flood volume is reached. For the conceptual models that use spreading principles the computational times are negligible as they can generally be run in the order of seconds or minutes even in standard portable computers.

The main drawback is that these models can only predict maximum water depths, without any consideration of their evolution in time. As such, we do not consider them in the rest of the thesis. Instead, we mainly focus on numerical models that solve the full SWE, which is essential to properly simulate dike breach floods. Most of this section on numerical methods is reported not only for a full understanding of the state of the art in flood modelling but mainly to highlight a link between the common practice and the methods developed in this thesis, which are strongly inspired by them.

2.3. DEEP LEARNING METHODS

Deep learning (DL) studies how neural networks learn representations from data, through multiple levels of abstraction (LeCun et al., 2015). A neural network is a non-linear compositional model formed by a hierarchical layering of parametric functions that take an input variable x and produce an estimate $\hat{y}$ of a target representation y as $\hat{y} = f(x;\theta)$, where θ are the function's parameters. The purpose of DL is to calibrate those parameters to have the best fit between predicted output and real output. The raw data x are input to the neural network and the output of each layer serves as input for the following layer, until the final layer, which coincides with the estimate $\hat{y}$. A neural network with L layers can be expressed as

$$\begin{aligned} \hat{y} &= f_L(\cdot;\theta_L) \circ f_{L-1}(\cdot;\theta_{L-1}) \circ \ldots \circ f_1(x;\theta_1), \\ x_\ell &= f_\ell(x_{\ell-1};\theta_\ell), \quad for \ \ell = 1,\ldots,L, \\ \hat{y} &\equiv x_L, \end{aligned} \tag{2.7}$$

where $f_\ell(\cdot;\theta_\ell)$ is the function at layer ℓ, $\circ$ represents the composition of functions, θ_ℓ are the trainable parameters at layer ℓ, and $\hat{y}_\ell$ is the output at layer ℓ. In a network architecture, the layers between the input and the output layer are called hidden layers, since their output is not shown. Estimating parameters θ_ℓ is typically referred to as 'learning' and it is performed by minimizing a loss function, through back-propagation (Rumelhart et al., 1986). Depending on the task, neural networks can be trained via supervised, unsupervised, or reinforcement learning. Since in flooding analysis DL has been mainly approached via supervised learning, we focus on that learning process, which will be also used to develop the models in this thesis.

Supervised deep learning models identify a mapping from input to output, given a training set of input-output pairs. For example, a training set for flood hazard mapping

may comprise a rainfall hyetograph as input x and the corresponding maximum flooded area as output y. The loss function $l(y, \hat{y})$ compares the real output y with the predicted one $\hat{y}$. This is typically a quadratic loss for regression problems, where the data is continuous (e.g., water depth), or a cross-entropy loss for classification problems, where the data is categorical (e.g., flooded and non-flooded areas). As training data, we can have observations or simulations. Observational data are derived from remote sensing, flood inventory maps, and measuring stations, while simulation data are derived from numerical solvers. Once a model is trained, its goodness of fit is analysed with a test set, composed of data that the model has not seen. If the model performs well for the test set, it is said to *generalize* or extrapolate well. The ability to generalize is one of the most important properties of DL and becomes even more important in high-dimensional inputs (Balestriero et al., 2021).

2.3.1. MULTI-LAYER PERCEPTRON

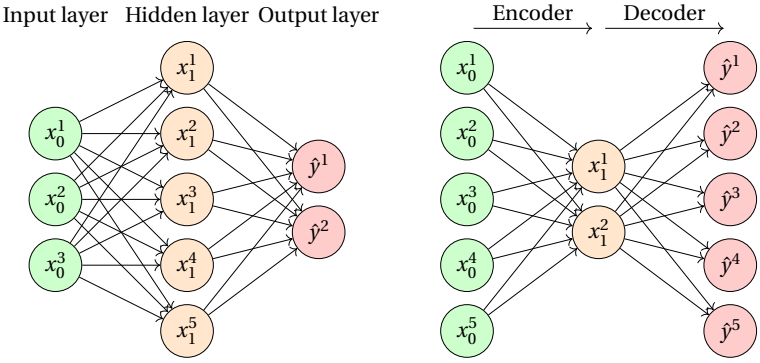


Figure 2.3: Multi-layer perceptron (MLP) composed of a sequence of three fully-connected layers (left). Every layer is connected to the following one by weights, represented by directed arrows. The values of the input, hidden, and output layers are represented, respectively, by vectors x_0, x_1, and $\hat{y}$; MLP encoder-decoder (right). The input data x_0 is encoded into a lower dimensional layer x_1,and then decoded into the output $\hat{y}$. This structure is also applicable to convolutional and recurrent layers.

Among the possible neural network layers, fully-connected ones are the most simple. In a fully-connected layer, the layer propagation rule that describes the input transformation is given by:

$$x_\ell = f_\ell(x_{\ell-1}, \theta_\ell) = \sigma(W_\ell x_{\ell-1}), \tag{2.8}$$

where x_ℓ is the output of the layer ℓ, $\sigma(\cdot)$ is a point-wise nonlinearity (e.g., ReLU, $\sigma(x) = max\{0, x\}$, or Sigmoid, $\sigma(x) = \frac{1}{1+e^{-x}}$), $x_{\ell-1}$ is the input of the layer ℓ, and the training parameter W_ℓ is a weight matrix. Multi-layer perceptrons (MLP) are composed by sequences of fully-connected layers (Fig. 2.3). The expressivity of the network increases with the dimensions of the hidden layers, as shown in Fig. 2.3. When the dimension of the hidden layers decreases and then increases, as shown in Fig. 2.3, the architecture is called encoder-decoder (ED). The idea behind this architecture is that, in high-dimensional data, only certain latent representations of the input are useful to represent the output (e.g., Taormina & Galelli, 2018). Conversely, if the input is low-dimensional, the same

architecture style can be used in the opposite way to enhance the representational power of the data (e.g., You et al., 2020).

In fully-connected layers, the values of the parameters in W are independent between them and there is no reuse of any of them. Thus, the number of learnable parameters scales with the input size, making fully-connected layers inappropriate for high-dimensionality inputs. This issue is referred to as the 'curse of dimensionality' and implies that as the dimension of the input increases, the amount of training data needed to learn representations increases exponentially (LeCun et al., 2015).

2.3.2. INDUCTIVE BIASES, INVARIANCE, AND EQUIVARIANCE

Table 2.1: Inductive biases and preferred types of data for different neural network layers (adapted from Battaglia et al. (2018)).

Layer	Data type	Inductive bias
Fully Connected	Unstructured data	-
Convolutional	Grid elements	Spatial equivariance
Recurrent	Sequences	Temporal equivariance
Graph	Networks	Permutation equivariance

To overcome the curse of dimensionality we need to exploit the structure in data. In flood analysis, data is usually structured: for example, neighbouring pixels in raster data represent spatial proximity of nearby close elements, while discharge values in a hydrograph represent temporal proximities. Neural network layers can thus be defined in a way to exploit these data structures. These assumptions create what is known as an *inductive bias*, which imposes constraints on relationships and interactions among inputs in the learning process, thus prioritizing some solutions over others (Battaglia et al., 2018) and reducing the amount of training data needed (e.g., Wang, Walters, & Yu, 2020). Inductive biases derive from the fundamental geometric principle of *symmetry* (Bronstein et al., 2021). The symmetry of a system is a transformation that leaves a certain property of said system unchanged. Symmetry results in invariance and equivariance properties. Invariance implies that transformations on the input features do not change the output (i.e., $f(g(x)) = f(x)$, $g(\cdot)$ being a generic transformation), while equivariance entails that transformations on the input features change the output via an equivalent transformation (i.e., $f(g(x)) = g'(f(x))$, $g'(\cdot)$ being a transformation equivalent to $g(\cdot)$). We explain the concept of invariance and equivariance with an example. Consider a picture with a flooded area in its top-left corner and one with the same flooded area shifted to the bottom-right corner. An invariant model would predict that there is a flooded area in both images, while an equivariant model would also reflect the change in positions of the flood, i.e., identify that the flood is in the top-left corner in one case and in the bottom-right corner in the other. In this case, invariance and equivariance are associated to a spatial translation, but the same principle applies to other transformations, such as temporal translation. Inductive biases thus lead to the reuse of parameters in different parts of

the input of each layer. For instance, *convolutional* kernels can be used on images of different dimensions and *recurrent* layers can consider time series of variable length. Fully connected layers, instead, cannot have such inductive bias capabilities, as each parameter is tied to a fixed input position and cannot generalize across varying input structures. The main characteristics for each considered layer are synthesized in Table 2.1. The input data type and the inductive biases are described for each studied layer.

2.3.3. CONVOLUTIONAL NEURAL NETWORK

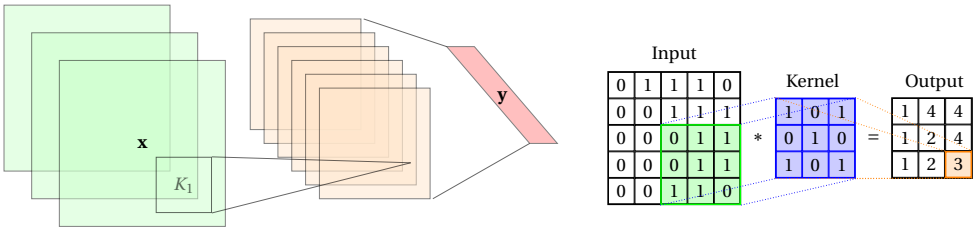


Figure 2.4: Left: Convolutional neural network (CNN) composed of a convolutional layer, and a fully-connected layer. The green squares represent an input tensor, the orange squares represent hidden layers and the red parallelogram on the right represents the output layer. The small box K_1 represents the convolutional kernel described in Eq. 2.9. The final layer depends on the task. Right: visual explanation of how convolutional kernels work. Each element of the kernel is multiplied by its matching input value. Then, all values are summed to obtain the convolved output. This process is repeated across the whole input, as the kernel shifts along it.

Convolution is an operation for which every entry of an input matrix is replaced by a spatially weighted average of its neighbouring entries, as shown in Fig. 2.4. The weights are defined by a matrix, called kernel, and are point-wise multiplied with the neighbouring entries. This procedure is then repeated, using the same kernel, for every entry in the input. Convolutional layers are a neural network layer that apply convolution on a input using trainable kernels, i.e., the kernels' weights are learned during optimization (LeCun, Bengio, et al., 1995). The propagation rule of layer ℓ of a convolutional layer is

$$x_{\ell+1} = \sigma(K_\ell * x_\ell), \tag{2.9}$$

where K_ℓ is the kernel function for the ℓ^{th} layer and $*$ is the convolution operator. Convolutional layers are mostly applied to images i.e., two dimensional spatial grids. For such inputs the kernel is a 2D matrix. Convolutional layers have an inductive bias of translational equivariance, which reflects the idea that spatially close grid elements influence each other. This results in the reuse of the same kernel across the different input parts and it implies that it matters where a pattern or object are in an image and that the model should be able to recognize it. Convolutional layers thus perform feature extraction, identifying relevant characteristics in the input. Moreover, the reuse of parameters allows inductive learning over images of different sizes or resolutions. Differently from fully-connected layers, the number of parameters in a convolutional layer depends only on the kernel size and number of kernels because of this parameter-sharing property (see Fig. 2.4). Depending on the input dimensions, we distinguish 1D convolutional layer for vector inputs, such as a rainfall hyetograph, 2D convolutional layers for matrix inputs,

such as a digital elevation model (DEM), and 3D convolutional layers for tensor inputs, such as stacked satellite images.

Convolutional neural networks (CNN) are composed of layers alternating convolution and pooling. Pooling operation replaces the output at a certain location with a summary statistic of the nearby features, thus approximating translational invariance (Bronstein et al., 2021). They extract a single feature, such as the average or maximum value in a certain neighbourhood of a point. Furthermore, pooling reduces the dimension of the input, speeding up computation. The final layers of a CNN are typically fully connected when dealing with classification or regression tasks. This layer allows to map the convolved embeddings to the number of classes or to the regressed value, respectively. Instead, if the task is to perform image segmentation or pixel-wise regression, i.e., classify specific parts of an image or predict a value for each pixel in the image, the final layers are usually composed of de-convolutional layers which perform the inverse operation of convolutional layers, in an encoder-decoder structure.

The encoder-decoder convolutional neural network is an architecture composed of two parts (Fig. 2.5). The encoder extracts high-level features from the input images, while reducing their extent, via a series of convolutional and pooling layers, while the decoder

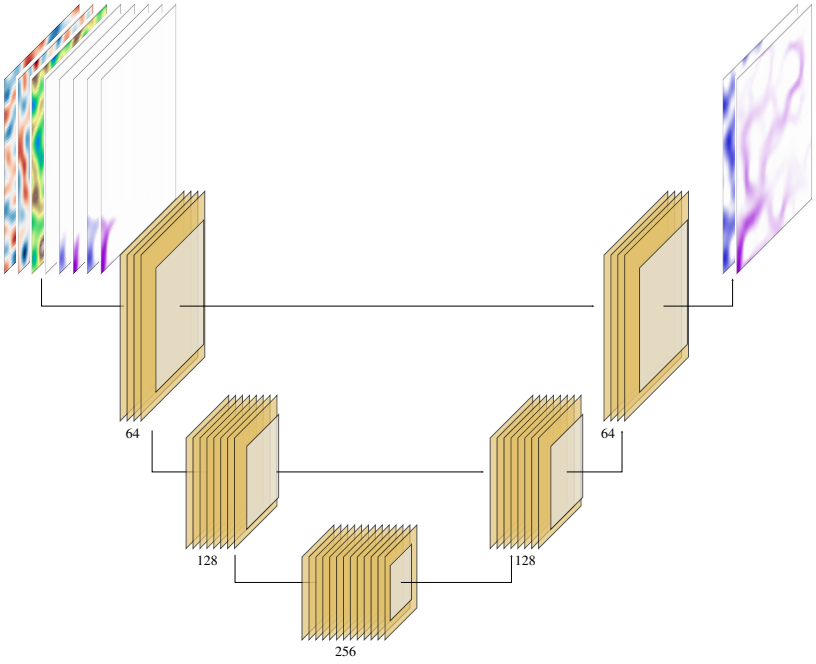


Figure 2.5: Example of U-NET architecture with first embedding dimension of 64 and three encoding blocks (Bentivoglio et al., 2023). Each block is composed of one convolutional layer, followed by a batch normalization layer, a *PReLU* activation function, another convolutional layer, and finally a pooling layer. All blocks with the same dimensions are connected by residual connections, indicated by the straight horizontal lines. The numbers below the blocks indicate the hidden features dimension.

extracts the output image from the compressed signal, via a series of de-convolutional and pooling layers. The most popular variant of this architecture is the U-Net structure, which also features residual connections between different blocks in the architecture, as shown in Fig. 2.5. These residual connections help preserve spatial information lost during downsampling and improve gradient flow during training (Goodfellow et al., 2016).

2.3.4. RECURRENT NEURAL NETWORK

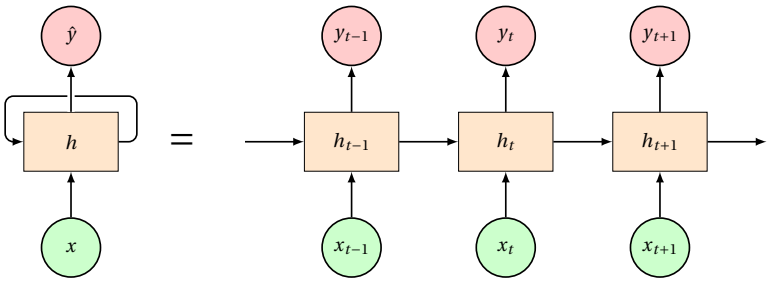


Figure 2.6: Recurrent neural network (RNN) in compact form (left) and in the unfolded form (right). The iterative structure of the RNN (left) can be unfolded in time to show how hidden states influence the solution at each time step (right). The colouring scheme indicates for each architecture the input (green), the state (orange), and the output (red).

Recurrent layers are used for processing sequential data, such as time series (Rumelhart et al., 1986). A recurrent layer can be seen as a non-linear state-space model expressing the output at time t, y_t, as a function of a former hidden state h_t and input x_t. The basic formulation for a recurrent layer is

$$\begin{aligned} h_t &= \sigma(W h_{t-1} + U x_t), \\ y_t &= \sigma(V h_t), \end{aligned} \tag{2.10}$$

where U, V, and W are trainable weight matrices. As it follows from Eq. (2.10), the hidden state encodes the temporal memory of previous time instances while the output mapping is instantaneous. These matrices are shared across time allowing the recurrent layer to exploit temporal proximities of sequential data, irrespectively of their position. This is for instance the case of discharge hydrographs (e.g., Zhou et al., 2021). Because there is an inductive bias in temporal sequences, they allow us to reuse parameters without affecting the performance.

Recurrent neural networks (RNN) are neural networks composed of recurrent layers. The iterative structure of the RNNs can be unfolded in time to show how hidden states influence the output at each time step (Fig. 2.6). However, the basic recurrent layer in (2.10) suffers from the problem of vanishing and exploding gradients (Hochreiter & Schmidhuber, 1997). This occurs due to the iterative use of the same layer which causes the weights to multiply several times when back-propagating the error, ultimately leading to “vanishing" gradients if the weights are small and “exploding” gradients if the weights are large. This constrains then the temporal memory of these networks and limits their capability to extract long-term dependencies between the past inputs and the current output.

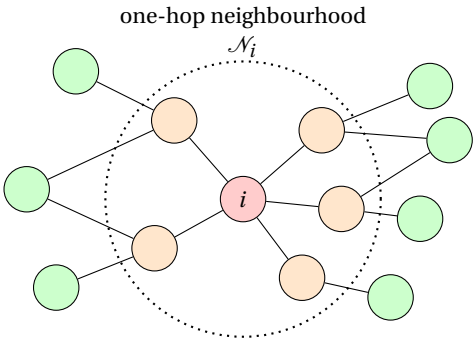


Figure 2.7: Example graph, where the dotted circle indicates the one-hop neighbourhood $\mathcal{N}_i$ (orange nodes) of the central node i (red). A graph convolutional layer aggregates the features at each neighbouring node $\mathbf{x}_j$, with $j \text{in} \mathcal{N}_i$ to obtain a new representation of the feature at the red node $\mathbf{x}'_i$.

This problem is typically solved via the use of Long Short-Term Memory (LSTM) layers (Hochreiter & Schmidhuber, 1997). This variation of recurrent layers also improves the hidden state mechanism allowing to "remember" well even information which is temporally distant. Another common variation is the Gated Recurrent Unit (GRU) (Cho, van Merrienboer, et al., 2014), which achieves comparable results with the LSTM architecture while using a simpler formulation. Same as for fully-connected and convolutional layers, recurrent layers can be used in encoder-decoder architectures. This structure can be composed of an RNN which generates a latent representation, followed by another RNN that decodes it (e.g., Cho, Van Merriënboer, et al., 2014).

The most successful applications of RNNs for flood management regard tasks related to sequences and time-series analysis, such as rainfall-runoff modelling (e.g., Kratzert, Herrnegger, et al., 2019). While RNNs are preferred over 1D-CNNs, recently the latter started gaining momentum for some time-series learning tasks (e.g., Van Den Oord et al., 2016).

2.3.5. GRAPH NEURAL NETWORKS

Graph convolution is an operation that generalizes the concept of convolution from Euclidean domains (e.g., time and space) to graph-structured data. A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a mathematical object that consists of a set of N nodes $\mathcal{V}$, which represent entities, and a set of E edges $\mathcal{E}$, which represent a connection between these entities. A graph signal $\mathbf{x} \in \mathbb{R}^N$ is a vector that assigns a value for each node of the graph.

While standard convolution works by shifting rectangular filters over a regular grid, such as the pixels in an image, graph convolution uses graph filters to produce a weighted average of a node's neighbouring signals, leveraging the graph's topology to propagate and transform information. Graph convolutional layers, or simply graph layers, are a neural network layer that apply graph convolution on a input using trainable weights in

the graph filters (Gama et al., 2020). The propagation rule of layer ℓ of a graph layer is

$$X_{\ell+1} = \sigma(\mathbf{S} X_\ell W_\ell), \tag{2.11}$$

where $X_\ell \in \mathbb{R}^{N\times F}$ is the input node feature matrix, $W_\ell \in \mathbb{R}^{F\times G}$ is a learnable filter matrix, F and G represent the number of dimensions of the input and output node feature matrices, and $\mathbf{S} \in \mathbb{R}^{N\times N}$ is the graph shift operator, i.e., a matrix that captures the connectivity of the graph and where $S_{ij} \neq 0 \text{if} i = j$ or $(i, j) \in \mathscr{E}$. The simplest form of the graph shift operator is the adjacency matrix $\mathbf{A}$ in which $a_{ij} = 1$ if an edge ij exists. The propagation rule in Eq. (2.11) can also be expressed as a local function for each node i and its neighbours $\mathcal{N}_i$ as

$$x_i^{\ell+1} = \sigma\left(\sum_{j\in\mathcal{N}_i} s_{ij} x_j^\ell w_{ij}^\ell\right), \tag{2.12}$$

where w_{ij} are learnable weights that indicate the connection strength between node i and node j and equivalent to the entries of the matrix W_ℓ.

Graph neural networks (GNNs) combine multiple graph layers and use graphs as an inductive bias to tackle the curse of dimensionality. This bias is relevant for data represented via networks and meshes, as it allows these models to generalize to unseen graphs, i.e., the same model can be applied to domains discretized by different meshes. Depending on the complexity of Eq. (2.12), we can define several classes of GNNs. These are generally grouped into convolutional, attentional, and message-passing GNNs (Bronstein et al., 2021).

Convolutional GNNs (GCN) follow Eq. (2.11) and have a graph shift operator that is fixed a priori and only depends on the connectivity of the graph. Within this category fit several examples of GNNs, such as ChebConv in which the graph shift operator $\mathbf{S}$ is the normalized Laplacian connectivity matrix $\hat{\mathbf{L}} = \left(\mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right)_{ij}$, where $\mathbf{I}$ is the identity matrix, $\mathbf{A}$ is the adjacency matrix, and $\mathbf{D}$ is the diagonal matrix (Defferrard et al., 2016).

Attentional GNNs (GAT) employ an attention-based mechanism to define the graph shift operator $\mathbf{S}$ based on their importance in relation to the target node (Velickovic et al., 2017). The ij entry of $\mathbf{S}$ reads as

$$\mathbf{s}_{ij} = \frac{exp(LeakyReLU(\mathbf{a}^T[\mathbf{W}^{(\ell)}\mathbf{x}_i^{(\ell)} || \mathbf{W}^{(\ell)}\mathbf{x}_j^{(\ell)}))}{\sum_{j\in N_i} exp(LeakyReLU(\mathbf{a}^T[\mathbf{W}^{(\ell)}\mathbf{x}_i^{(\ell)} || \mathbf{W}^{(\ell)}\mathbf{x}_j^{(\ell)}]))} \tag{2.13}$$

where $\mathbf{a} \in \mathbb{R}^{2G}$ is a learnable weight vector and $||$ denotes concatenation. In GATs, the graph shift operator is learnt based on the values of the node features and does not depend on the connectivity of the graph.

Message-passing GNNs (MPGNN) further generalize the expression of GATs and GCNs by using learnable functions instead of learnable weights (Gilmer et al., 2017). The general propagation rule is defined in the node-wise setting as

$$h_i^{(\ell+1)} = \phi\left(h_i^{(\ell)}, \sum_{j\in\mathcal{N}_i} \psi\left(\{h_j^{(\ell)}, e_{ij}\right)\right), \tag{2.14}$$

where $h_i^{(\ell)}$ and $h_j^{(\ell)}$ are the features of nodes i and j at layer ℓ, e_{ij} are the edge features between nodes i and j, and ϕ and ψ are a trainable functions (e.g., MLPs) that combine the features at node i with the features at the neighbouring nodes and edges.

In this thesis, we focus on different graph neural network architectures applied to meshes. A mesh is a structure composed of a collection of nodes, edges, and faces used to discretize a continuous domain. Meshes are commonly used for numerical simulations in many physical systems (e.g., Bomers, Schielen, & Hulscher, 2019; Ferraro et al., 2020). Their flexible definition allows to increase the resolution where needed and coarsen it otherwise, ultimately decreasing the computational time and improving efficiency (Candy, 2017). In this thesis, we establish a link between finite volume methods (Section 2.2.1) and graph neural networks, by comparing their computational structure over meshes, as further discussed in Chapter 4. We also argue that the propagation rule of the GNN is essential in correctly modelling a physical system and should be designed as to preserve relevant physical properties of the modelled system.

3

REVIEW OF DEEP LEARNING METHODS FOR FLOOD MAPPING

Research is to see what everybody else has seen, and to think what nobody else has thought.

Albert Szent-Györgyi

This chapter presents a critical review of recent studies that employ deep learning (DL) techniques for flood mapping. To conduct this review, we selected a representative set of studies based on two complementary yet interrelated dimensions. On the one hand, we focused on flood management, analysing different mapping applications, spatial scales, and types of floods. This allows to understand how DL models are being tailored to address various flood-related challenges. On the other hand, we focused on the technical aspects of the deep learning methods examining the DL architectures, the type and source of training data, and the comparative performance of these models relative to traditional or physics-based approaches. By analysing studies across these dimensions, this chapter highlights both the current capabilities and limitations of DL-based flood mapping methods. The objective of this review is to assess the current state of the art and identify key knowledge gaps. These will form the basis for the research questions addressed in the following thesis chapters.

This chapter is an adapted version of:

Bentivoglio, R., Isufi, E., Jonkman, S. N., & Taormina, R. (2022). Deep learning methods for flood mapping: A review of existing applications and future research directions. Hydrology and Earth System Sciences, 26, 4345–4378. https://doi.org/10.5194/hess-26-4345-2022.

3.1. INTRODUCTION

We retrieved papers from the Scopus database by combining the keywords “deep learning” or “neural network" with “flood” or “flooding”. The 3,338 publications obtained were then filtered to include only journal papers from January 2010 until December 2021, in the areas of *engineering*, *environmental science*, and *earth and planetary sciences*. From this reduced list of 1308 papers, we considered two major refining criteria: i) the papers should be based on the deep learning models presented in Chapter 2.3, and ii) the applications must address the spatial variability of floods (i.e., not focusing only on the temporal aspects of flood analysis). This procedure resulted in 46 reviewable papers. This list was finally extended via a snowball search that considered cited and citing works, ultimately leading to 58 eligible documents (Fig. 3.1). We find that the described methodology selected a representative subset for producing a thorough review of recent advances and developments in this field.

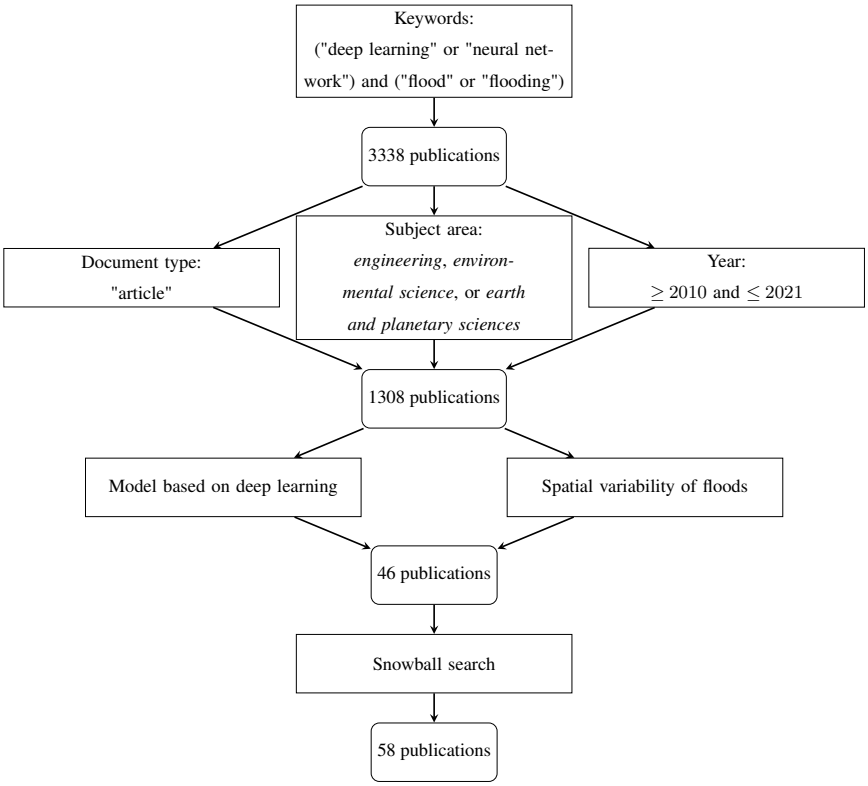


Figure 3.1: Flowchart of the methodology applied for the papers selection.

The selected papers are listed in Table 3.1 which reports major details including the flood mapping application, the type of flood, the DL model, and the spatial scale. General findings related to these three criteria are first presented in Section 3.2. Specific findings

for each application are then presented in Sections 3.3 (flood inundation), 3.4 (flood susceptibility), and 3.5 (flood hazard). These specific sections provide a more in-depth discussion on the deep learning models employed, with a focus on the architecture, the input and output data, and the performance assessment.

Table 3.1: Deep learning applications for flood mapping. References are classified in terms of flood mapping application, type of flood, deep learning (DL) model, training data, and spatial scale.

Application	Flood Type	DL Model	Training Data	Spatial Scale	Reference(s)
Inundation	River	MLP	Obs.	Regional	Li, Chen, Xu, et al. (2016) and Li et al. (2015)
		CNN	Obs.	Local	Gebrehiwot et al. (2019), Hashemi-Beni and Gebrehiwot (2021), Hou et al. (2021), Ichim and Popescu (2020), Nogueira et al. (2018), and Wieland and Martinis (2019)
				Regional	Isikdogan et al. (2017), Kang et al. (2018), Nemni et al. (2020), and Sarker et al. (2019)
	Urban	MLP	Obs.	Local	Amini (2010)
				Regional	Li, Xu, and Chen (2016)
		CNN	Obs.	Local	Peng et al. (2019)
		RNN, CNN	Obs.	Local	Dong et al. (2021)
	Coastal	CNN	Obs.	Regional	Isikdogan et al. (2017) and Liu et al. (2019)
			Sim.	Regional	Muñoz et al. (2021)
	Dam Break	MLP	Obs.	Regional	Syifa et al. (2019)
Susceptibility	River	MLP	Obs.	Regional	Ahmadlou et al. (2021), Ahmed et al. (2022), Chakrabortty, Pal, et al. (2021), Jahangir et al. (2019), Khoirunisa et al. (2021), Kia et al. (2012), Popa et al. (2019), and Saeed et al. (2021)
		CNN	Obs.	Regional	Wang, Fang, et al. (2020)
				National	Khosravi et al. (2020)
		RNN	Obs.	Regional	Fang et al. (2020)
	Flash	MLP	Obs.	Regional	Chakrabortty, Chandra Pal, et al. (2021), Costache et al. (2020), Ngo et al. (2018), Popa et al. (2019), and Tien et al. (2020)

				National	Kourgialas and Karatzas (2017)
		CNN	Obs.	Regional	Liu et al. (2021) and Panahi et al. (2021)
	Urban	MLP	Obs.	Local	Darabi et al. (2022)
				Regional	Kalantar et al. (2021)
		CNN	Obs.	Local	Zhao, Pang, Xu, et al. (2020, 2021)
				Regional	Lei et al. (2021)
Hazard	River	MLP	Sim.	Local	Chu et al. (2020), Huang et al. (2021), Jacquier et al. (2021), Lin, Leandro, Gerber, and Disse (2020), Lin, Leandro, Wu, et al. (2020), and Xie et al. (2021)
		CNN	Sim.	Local	Hosseiny (2021) and Kabir et al. (2020)
		RNN	Sim.	Regional	Kao et al. (2021) and Zhou et al. (2021)
	Flash	CNN	Sim.	Local	Yokoya et al. (2022)
	Urban	MLP	Sim.	Local	Berkhahn et al. (2019) and Chang et al. (2010)
		CNN	Sim.	Regional	Guo et al. (2020) and Löwe et al. (2021)
	Coastal	RNN	Sim.	Local	Hu et al. (2019)
	Dam Break	MLP	Sim.	Local	Jacquier et al. (2021)

MLP=Multi-layer Perceptron; CNN=Convolutional Neural Network; RNN=Recurrent Neural Network, Obs. = Observations, Sim. = Simulations

3.2. GENERAL FINDINGS

3.2.1. FLOOD MAPPING APPLICATIONS

Fig. 3.2 shows the distribution of papers for each of the applications considered: flood inundation, flood susceptibility, and flood hazard. The research community has dedicated efforts to investigate each type of application, although flood inundation and susceptibility have received the most attention. While papers on flood inundation are more evenly distributed across years, applications for flood susceptibility and, especially, flood hazard are increasing in the last few years analysed. Similar to what was observed in related fields such as hydrology (e.g., Sit et al., 2020), a strong surge in DL publications for spatial flood analysis is witnessed between 2018 and 2019. These years identify a turning point for AI in earth system sciences driven by the adoption of CNN (striped patterns in Fig. 3.2) and RNN (dotted patterns) in lieu of traditional MLP models. The late use of convolutional and recurrent models is motivated by their recent popularization and development, along with a rise in awareness of the ML advancements, contrary to fully-connected layers, that have a longer application history.

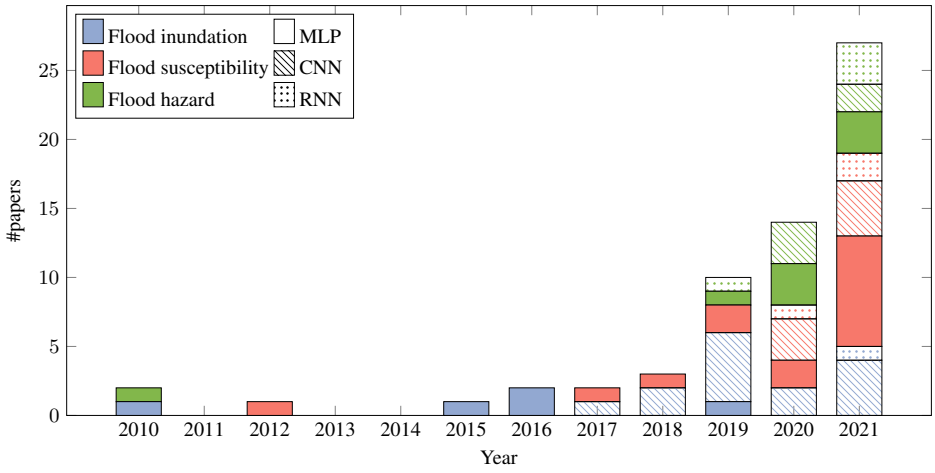


Figure 3.2: Publications by year, type of application and type of DL model. The increasing trend of the last five years (from 2017 to 2021) has been mostly driven by the applications in flood susceptibility and flood hazard.

3.2.2. FLOOD TYPES

Fig. 3.3 shows the types of flood analysed with respect to each application. River floods are the most common, with many applications in inundation and hazard mapping. This is probably because, for historical reasons, most cities in the world are built close to rivers (Kummu et al., 2011). The scientific community has dedicated significant efforts to exploring the potential of DL for urban flooding. This is difficult to model because of the complex topography and the presence of a drainage system whose dynamics need to be coupled with the overland flood (Löwe et al., 2021). Almost all papers analysing flash floods described flood susceptibility mapping applications. This is expected due to

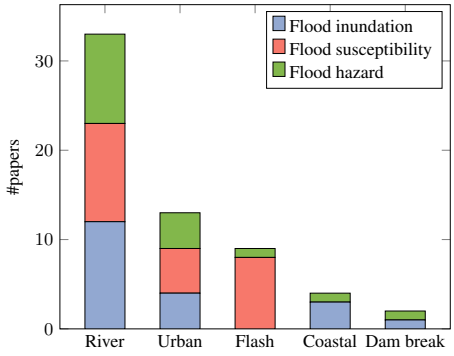
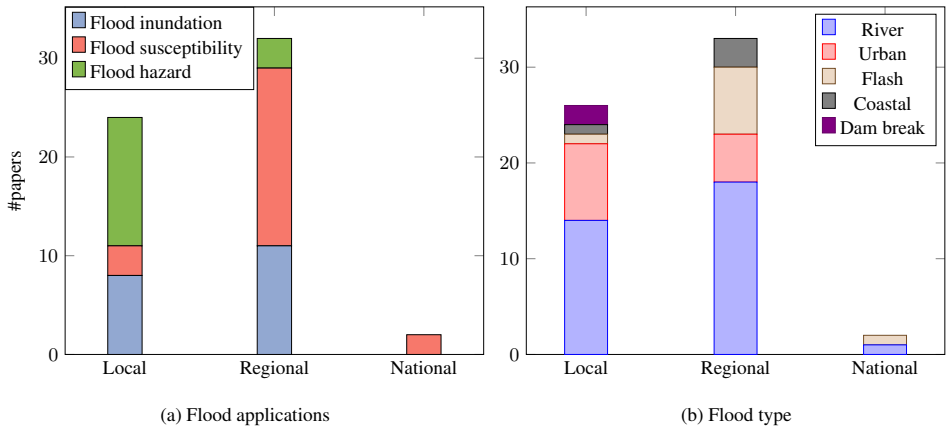


Figure 3.3: Distribution of the types of floods per flood application in the reviewed papers. River and urban floods are the most common, while flash and coastal floods have fewer occurrences.



(a) Flood applications

(b) Flood type

Figure 3.4: Distribution of the spatial scale per (a) flood application and (b) type of flood in the reviewed papers. Local and regional scales are the most used.

the short duration and the contingent nature of these phenomena, which limit remote sensing imaging and numerical simulations used in flood inundation and flood hazard mapping, respectively. Despite the importance of coastal flooding (Neumann et al., 2015), only a few papers report the use of DL for coastal flooding. While other works are available in the literature (Bowes et al., 2021; Lütjens et al., 2020, 2021), they were not considered since the employed DL models were not trained via supervised learning. Dam break floods are the least analysed type, possibly because of their relatively rare occurrence and complexity.

3.2.3. SPATIAL SCALE

As shown in Fig. 3.4, most applications consider local and regional scales. Local scale refers to towns (e.g., Berkhahn et al., 2019; Darabi et al., 2022), small catchments (e.g., Kabir et al., 2020; Lin, Leandro, Gerber, & Disse, 2020) or river reaches (e.g., Chu et al., 2020; Gebrehiwot et al., 2019). As such, they are mostly connected to urban and river floods. The cases sizes vary from very small ones, $165m^2$ (Hou et al., 2021), to small towns up to $100km^2$ (Lin, Leandro, Gerber, & Disse, 2020). Regional scale models consider a catchment (e.g., Popa et al., 2019), a province (e.g., Wang, Fang, et al., 2020) or large cities (e.g., Kalantar et al., 2021; Löwe et al., 2021). Most works focus on river floods, while some study flash, urban, and coastal floods. National scale models refer to the assessments of entire countries, with only two papers concerning such scales, respectively for Iran and Greece (Khosravi et al., 2020; Kourgialas & Karatzas, 2017). Nemni et al. (2020) and Sarker et al. (2019) consider several study areas across Africa and Asia, and Australia, respectively, but since the size of each area were smaller than $100000km^2$ they were marked as regional scale models. They also do not fit within the national scale classification since they do not encompass whole nations. Supra-national scale models assessing the entire globe or a continent have not been studied yet with deep learning models. This seems unexpected since ML techniques have already been employed at

global scales, outperforming traditional techniques, for example in the estimation of design floods along river networks (e.g., Zhao, Bates, et al., 2020). Since DL models have been shown to outperform ML models, as later outlined in this review, more models should be used at those scales in future studies.

3.2.4. DL ARCHITECTURE

Fig. 3.2 reports the architecture used for each application, showing that DL models are mainly based on fully connected and convolutional layers.

MLP networks are widely used due to their flexibility and ease of implementation. However, they are usually coupled with other techniques to reach satisfactory performances. Stochastic optimization techniques, such as genetic algorithm, firefly algorithm, and particle swarm optimization were combined with MLPs to search the optimal model's parameters (e.g., Kalantar et al., 2021; Li et al., 2015; Ngo et al., 2018). Multi-criteria decision analysis models, such as frequency ratio and analytical hierarchy process, were also coupled with MLPs to adjust the weights of each input in flood susceptibility (e.g., Costache et al., 2020; Kourgialas & Karatzas, 2017; Popa et al., 2019). Furthermore, k-means clustering was used to categorize the dataset in classes, to account for different topographical conditions; then, for each class, a MLP was trained (e.g., Chang et al., 2010; Huang et al., 2021). Combining MLPs with such methods partly compensates the lack of inductive biases, however, this lack blocks the model from employing existing structures in the data, ultimately limiting their usability. Since flooding phenomena have spatial and temporal structures, we expect MLPs to become progressively less used in this field, as hinted by the trend in Fig. 3.2.

CNNs are best suited for processing raster files and images, thanks to their spatial inductive bias. Since most data for flood analysis (e.g., elevation data, rainfall distribution fields, remote sensing image) come in this format, CNNs have been increasingly employed by the research community in the recent years. While most papers consider standard CNNs, there are a few which employ 1D-CNNs (e.g., Dong et al., 2021; Guo et al., 2020; Liu et al., 2021) and 3D-CNNs (e.g., Fang et al., 2020; Wang, Fang, et al., 2020). 1D-CNNs consider as input a hyetograph or a hydrograph of a certain event, while 3D-CNNs consider raster files stacked upon each other. Regarding the architectures, different papers for flood inundation consider an encoder-decoder structure for image segmentation and classification (e.g., Hashemi-Beni & Gebrehiwot, 2021; Liu et al., 2019; Nemni et al., 2020). For such papers, the input is a satellite image of a flood and the output is its classification in flooded and non-flooded areas. This architecture allows the models to increase its performance since it can retain high frequency details in the segmented images (Badrinarayanan et al., 2015). Guo et al. (2020) and Löwe et al. (2021) use a convolutional encoder-decoder structure for flood hazard mapping to embed a rainfall hyetograph in the latent space. In this way, they can consider both spatial and temporal data within the same framework.

RNNs have been mostly employed to model temporally-varying floods, where they can exploit best their sequential inductive bias. However, they remain the least common choice of DL architecture for spatial flood analysis. Most papers apply RNNs on a time series, such as a hyetograph or a hydrograph (e.g., Kao et al., 2021; Zhou et al., 2021). Some papers, instead, consider spatial sequentiality by reshaping the original raster data

into vectors (e.g., Fang et al., 2020; Lei et al., 2021; Panahi et al., 2021). For example, Fang et al. (2020) extract, for each pixel, its neighbouring pixels in a 3×3 window and then convert them into a vector based on spatial contiguity. However, this operation introduces arbitrariety in the sequential order chosen for arranging the input pixels, since it is independent of the underlying topography. In fact, Panahi et al. (2021) and Lei et al. (2021) show that these models underperform when compared with CNNs. Among the different RNN layers, most works consider LSTM units (Fang et al., 2020; Kao et al., 2021; Zhou et al., 2021) but simple recurrent units (Huang et al., 2021; Panahi et al., 2021) and GRUs (Dong et al., 2021) have also been employed. Some papers analysed the potential of RNNs in combination with other techniques. Kao et al. (2021) use an encoder-decoder architecture to forecast flood features based on rainfall patterns. The encoder and the decoder steps are composed of fully-connected layers, while a LSTM is present in the latent space to process rainfall data. Zhou et al. (2021) identify representative spatial locations in the study area. Then, an LSTM is trained to simulate the water levels' evolution in time at each location. A water surface is ultimately determined by interpolating the water depth at those points. Dong et al. (2021) combine 1D-CNNs and RNNs on an urban channel network. The model takes as input the channels' properties, such as their cross-sections, and rainfall and water level measures, taken from sensors in the network. This input is then given in parallel to a 1D-CNN and to a GRU whose output is then combined to predict the temporal evolution of the flood. Hu et al. (2019) deploy the LSTM model in a lower-dimensional space, obtained via proper orthogonal decomposition and singular value decomposition. The model then requires fewer data to be trained.

3.2.5. PERFORMANCE ASSESSMENT

This section discusses different approaches for assessing the performance of the DL models, i.e., how well they match the outcomes of traditional and machine learning models. Flood susceptibility and inundation models are compared with techniques such as frequency ratio (Popa et al., 2019), a type of MCDA model; the soil conservation service runoff model (Jahangir et al., 2019), a hydrologic model; and automatic threshold model (Nemni et al., 2020), a histogram-based model. They are also compared with machine learning techniques, such as support vector machines (e.g., Gebrehiwot et al., 2019; Sarker et al., 2019; Zhao, Pang, Xu, et al., 2020), random forest (e.g., Darabi et al., 2022; Zhao, Pang, Xu, et al., 2020), adaptive neuro-fuzzy inference system (Panahi et al., 2021), deep boost (e.g., Ahmed et al., 2022; Chakrabortty, Chandra Pal, et al., 2021), and radial basis function (Nogueira et al., 2018). DL models show to outperform both traditional and ML models in terms of the accuracy of the results. Flood hazard models, instead, are compared against numerical models, since they act as surrogate models. Thus, their main purpose is to increase computational speed while maintaining low prediction errors.

There are also a few papers that compared different DL models. Huang et al. (2021) compared MLPs with RNNs, while Fang et al. (2020) showed that MLPs were outperformed by the more inductive-biased approaches such as RNNs, 1D-CNNs, and 3D-CNNs. Wieland and Martinis (2019) showed that CNNs widely outperform MLPs, as expected, because of their inductive biases capabilities. Besides accuracy, the number of parameters and the data requirements are important factors when comparing DL models. A higher

number of parameters results in better performances but may also lead to overfitting, a condition where the model decreases its performance on the testing data. Hence, when deployed in similar settings such model would perform drastically worse. Moreover, data is not always available leading to possibly unfair comparisons between models with different data budgets. As such, the same model may give different outcomes depending on the considered case.

In supervised learning, we distinguish between regression and classification problems, depending on whether the target values to predict are continuous (e.g., water depth) or discrete (e.g., flooded vs non-flooded area), respectively. Depending on the task, we employ a different set of metrics to evaluate model performances.

Regression metrics are a function of the differences, or residuals, between target and predicted values. The most common metrics include the root mean squared error (RMSE), the coefficient of determination (R^2), and the mean average error (MAE). RMSE and MAE improve as they approach zero, while R^2 improves as it approaches one. In general, MAE may be preferred to RMSE since the latter is heavily influenced by the presence of extreme outliers. However, since both metrics are averaged on a domain, their comparison across different works requires careful attention.

Classification tasks can be either binary (e.g., predict flooded and non-flooded locations) or multi-categorical (e.g., classifying between permanent water bodies, buildings, and vegetated areas), according to the output number of classes. In the following discussion, we focus on the former, with concepts extending to the second case. When computing binary classification metrics, flooded areas are generally represented as positive class, while non-flooded areas as negative class. The most common metrics for flood modelling are accuracy, recall, and precision, followed by other indices such as the area under the receiver operator characteristic curve. Accuracy represents the number of correct predictions over the total. While popular and easy to implement, this metric is inappropriate for imbalanced datasets, where some categories are more represented than others. For example, if test samples feature an average 90% non-flooded area, a naïve model constantly predicting no flooding would reach 90% accuracy, despite having wrong assumptions. Furthermore, since it may be better to overestimate a flooded area than to underestimate it, one could resort to metrics such as recall that account for false negatives and thus penalize models that cannot recognize a flooded area correctly. However, when used alone, recall can lead to similar issues to those described for accuracy, e.g., yielding a perfect score for a model always predicting the entire domain as flooded. Thus, for an exhaustive understanding of the model's performance, one should also consider metrics accounting for false positives, i.e., where the model misclassifies non-flooded areas as flooded. There are several possible metrics, such as the F1 score, the Kappa score, or the Matthews correlation coefficient, each with their drawbacks and benefits (e.g., Chicco & Jurman, 2020; Delgado & Tibau, 2019; Wardhani et al., 2019). A reasonable choice is the F1 score, which is the geometric mean of recall and precision, and it thus equally considers both false negatives and false positives. Another good example is the ROC (Receiver Operating Characteristic) curve that describes how much a model can differentiate between positive and negative classes for different discrimination thresholds (Bradley, 1997). The Area under the ROC curve (AUC) is often used to synthesise the ROC as a single value. However, the AUC loses information on which parts of the dataset the model performs

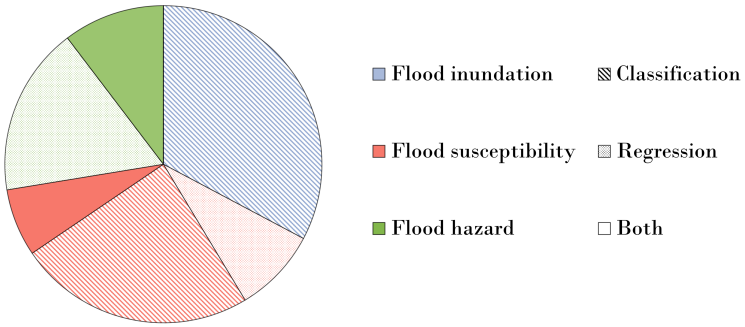


Figure 3.5: Distribution of the comparison metrics per type of application. The colours represent the different types of applications, while the patterns represent the considered metrics.

best. For this reason, one should always interpret these results carefully, especially when comparing different studies. Our purpose here is to show that, for the same case study, DL tends to outperform traditional models.

For surrogate models, the comparison is also performed in terms of their speed-up, which is determined as the ratio between the simulation time of the numerical model and the simulation time of the DL model. For a correct comparison, the training time of the DL model must be considered as well in this analysis. However, this was done only by a few papers (e.g., Guo et al., 2020; Jacquier et al., 2021; Kabir et al., 2020).

3.3. DEEP LEARNING FOR FLOOD INUNDATION

Flood inundation maps determine the extent of a flood, during or after it has occurred. We remind the reader that, in this thesis, we refer to flood inundation as the process of mapping flooded and non-flooded areas from a picture of a flood. This classification is usually binary (e.g., Nemni et al., 2020; Peng et al., 2019) but it can also be extended to include permanent water bodies (e.g., Sarker et al., 2019) (see the example Fig. 2.1a), vegetation (e.g., Ichim & Popescu, 2020), buildings (e.g., Hashemi-Beni & Gebrehiwot, 2021), and more (e.g., Muñoz et al., 2021). All the types of floods were well represented for this application but flash floods (Fig. 3.3). We attribute this to the limited frequency of observation of most remote sensing techniques.

Regarding the spatial scale, most papers focused on local and regional scales. Availability of remote sensing at wider scales is increasingly higher (e.g., Dartmouth Flood Observatory (https://floodobservatory.colorado.edu/)); however, this seems to be only partially considered. A plausible reason is the limited frequency of observation of the satellites. High temporal remote sensing imagery has a low spatial resolution. Few papers tackle this issue by increasing the resolution of the predicted flood maps, via a neural network, with a technique known as *super-resolution* (e.g., Li, Xu, & Chen, 2016; Li et al., 2015). Super-resolution enhances the quality of an input low-resolution image (Yang et al., 2019). These papers show that MLPs improve the accuracy of super-resolution mapping, with respect to other techniques such as spatial attraction models. We argue that further improvements of super-resolution could be obtained by employing CNNs, which lend

themselves naturally for such tasks, as demonstrated by applications in similar fields (Ma et al., 2019).

3.3.1. DL ARCHITECTURE

As the task of recognizing floods from a picture can be regarded as an image segmentation task, the most used deep learning models are based on CNNs. There are also a few earlier papers that use MLPs (e.g., Amini, 2010; Li, Chen, Xu, et al., 2016) because CNNs were not yet adopted by researchers of the field. Dong et al. (2021) use a combination of RNNs and 1D-CNNs to determine the temporal evolution of flooded and non-flooded nodes in an urban channel network, as described previously. In this case, the choice of recurrent and 1D-convolutional layers is well motivated due to their temporal inductive bias.

3.3.2. INPUT AND OUTPUT DATA

Satellite data is the most used input for flood inundation applications (e.g., Nogueira et al., 2018; Peng et al., 2019; Sarker et al., 2019). Other input data sources include unmanned aerial vehicles data (UAV) (e.g., Gebrehiwot et al., 2019; Ichim & Popescu, 2020), hydrographs (e.g., Hou et al., 2021) and DEMs (e.g., Hashemi-Beni & Gebrehiwot, 2021; Muñoz et al., 2021). Only Dong et al. (2021) differ from the other papers by considering sensors in place of flood pictures. Inundation maps produced by 3D numerical models are also used as target prediction (Muñoz et al., 2021). The results from the numerical model can be used as a detailed reference for the DL model. Satellite data and UAV imagery are both remote sensing data that represent a flood event seen from above. The main differences concern the scale, the resolution, and the availability. UAVs are applicable only for small areas but their resolution is higher than satellite data. UAVs can be readily used but may be unavailable in certain areas. On the other hand, satellite data is available worldwide but its frequency of observation can be limiting. Satellites can also struggle to extract information below clouded areas (e.g., Meraner et al., 2020). When combining information from different sources, the input data have different resolutions, leading to possible problems for some deep learning models, which take fixed-size inputs. One way to integrate different data resolutions is by data fusion (e.g., Muñoz et al., 2021). This process allows the creation of more consistent, accurate, and useful information than that provided by any individual data source.

3.3.3. PERFORMANCE ASSESSMENT

As defined in section 3.3, flood inundation mapping determines which cells of the flood picture are represented as flooded or not. Thus, the task is regarded as a classification problem, as confirmed by the metrics used (Fig. 3.5). The selected papers often use several metrics but for clarity we consider a single metric for each work. The metric selection depends on the employed ones and follows the considerations presented in Section 3.2.5, with preference for metrics such as F1, AUC, or recall, if available. Deep learning models have consistently shown improved performances in terms of the selected metrics (Table 3.2). Li et al. (2015) and Li, Xu, and Chen (2016) compare optimization techniques with and without MLPs for super-resolution-based flooding. They show that a DL model slightly increases the performances. This may be because the models are based on MLPs and thus neglect any spatial structure in the data, which could be considered,

Table 3.2: Performance of the deep learning and comparison with reference models for flood inundation.

Case study	Deep learning		Comparison		Reference
size [km^2]	Model	Metric	Model	Metric	
2.2	MLP	Acc = 70%	ML	Acc = 57%	Amini (2010)
225	MLP	Acc = 86.1%	SAM	Acc = 83.9%	Li, Xu, and Chen (2016)
3600	MLP+PSO	Acc = 81.6%	PSO	Acc = 79.7%	Li, Chen, Xu, et al. (2016)
5625	MLP+GA	Acc = 81%	GA	Acc = 79.3%	Li et al. (2015)
0.00016	CNN	P = 0.927	-	-	Hou et al. (2021)
0.01	CNN	Acc = 95.5%	SVM	Acc = 87.4%	Gebrehiwot et al. (2019)
0.9	CNN	IoU = 84%	SVM+RBF	IoU = 83%	Nogueira et al. (2018)
5.2	CNN	Acc = 92.4%	RG	Acc = 91.8%	Hashemi-Beni and Gebrehiwot (2021)
10	CNN	Acc = 96.4%	RF	Acc = 87.3%	Ichim and Popescu (2020)
59.3	CNN	F1 = 0.955	RF	F1 = 0.922	Peng et al. (2019)
59.3	CNN	F1 = 0.90	RF	F1 = 0.84	Wieland and Martinis (2019)
200	CNN	F1 = 0.975	-	-	Muñoz et al. (2021)
237	CNN	F1 = 0.90	NDWI	F1 = 0.70	Isikdogan et al. (2017)
10895	CNN	F1 = 0.94	M1	F1 = 0.78	Kang et al. (2018)
23300	CNN	F1 = 0.88	-	-	Liu et al. (2019)
25000	CNN	F1 = 0.92	ATM	F1 = 0.71	Nemni et al. (2020)
31450	CNN	Recall = 62.8%	SVM	Recall = 25%	Sarker et al. (2019)
4600	RNN+CNN	F1 = 0.77	CNN	F1 = 0.734	Dong et al. (2021)

Acc = Accuracy; P = Precision; ML = maximum likelihood; SAM = spatial attraction mode; PSO = particle swarm optimization; GA = genetic algorithm;SVM = support vector machine; RBF = radial basis function; RG = region growing; RF = random forest; NDWI = normalized difference water index; ATM = automatic threshold model

instead, by CNNs. Most CNN models show noticeable improvements with respect to traditional threshold methods, such as the Normalized Difference Water Index (NDWI) and automatic threshold model (ATM) (e.g., Isikdogan et al., 2017; Nemni et al., 2020; Wieland & Martinis, 2019), and with respect to machine learning models such as random forest (RF) and support vector machine (SVM). This reflects similar results obtained in image detection tasks (Badrinarayanan et al., 2015).

3.4. DEEP LEARNING FOR FLOOD SUSCEPTIBILITY

Flood susceptibility determines the tendency to flooding of a study area based on its physical characteristics and given a set of known past flood events. This is done by assigning to each location a level of susceptibility ranked from low to high (see Fig. 2.1b). The susceptibility depends on the distribution of the inputs, often called flood

conditioning factors, in function of recorded past flood events. The deep learning model then computes, for each point in the area, a score from 0 (non-flooded) to 1 (flooded). These scores are finally divided into several classes, generally using the natural (Jenks) breaks method (e.g., Fang et al., 2020; Khoirunisa et al., 2021; Wang, Fang, et al., 2020), to obtain a susceptibility map. An exception is given by Jahangir et al. (2019) and Kia et al. (2012), which train their models to predict discharge values and then use a GIS model for the mapping. In both cases, the model performs well when the recorded flood events occur in the predicted high susceptibility areas.

There exist DL-related applications for all types of flood (see Fig. 3.4b). Furthermore, Fig. 3.4a shows that most of the works are concerned with regional or wider scales (e.g., Khosravi et al., 2020; Panahi et al., 2021; Tien et al., 2020). This is expected since susceptibility mapping gives a qualitative estimate of which locations are prone to flooding. Operating on small scales may thus be limiting, both in terms of data availability and applicability for prevention strategies. The data requirements for an accurate estimate would probably be too high for a small area.

3.4.1. DL ARCHITECTURE

Most papers use MLP and CNN. Models based on MLPs consider single points or pixels as inputs (Ahmadlou et al., 2021; Khoirunisa et al., 2021; Tien et al., 2020), while CNNs consider the whole raster files (Khosravi et al., 2020; Wang, Fang, et al., 2020; Zhao, Pang, Xu, et al., 2020). Since MLPs lack inductive bias they provide less coherent results, meaning that the variation among neighboring cells can be high. This is partially solved by coupling the MLP architecture with other statistical techniques, such as frequency ratio (e.g., Costache et al., 2020; Darabi et al., 2022; Popa et al., 2019). Instead, CNNs have a spatial inductive bias, thus they inherently consider the structure of the input, providing more coherent flood maps (e.g., Khosravi et al., 2020). However, Wang, Fang, et al. (2020) and Liu et al. (2021) show that 1D-CNNs, which perform convolution on the input features for each domain's cell, are not suited for this problem, as they do not properly leverage any inductive bias. Some works showed that deep belief networks (DBN), an unsupervised variation of MLPs, could outperform standard MLPs in flood susceptibility mapping (e.g., Pham et al., 2021; Shirzadi et al., 2020).

3.4.2. INPUT AND OUTPUT DATA

The inputs for the deep learning models are several. We distinguish between five input typologies:

1. *topographical* inputs, which are derived from a digital elevation model, such as elevation, slope, and aspect;
2. *meteorological* inputs, related to the hydrological characteristics and derived from measuring stations and satellites, such as rainfall distribution and frequency;
3. *geological* inputs, related to the properties of the soil, such as lithology and soil type;
4. *geographical* inputs, related to observable surface characteristics and obtained through remote sensing, such as land use and normalized difference vegetation

index;

5. *anthropogenic* inputs, related to the presence of human-made environments, such as distance from roads.

Topographical data were the most frequent type of input. Many papers present a sensitivity analysis to determine which factors influenced the most the final results: on average, these were slope, land use, aspect, terrain curvature, and distance from the rivers (e.g., Costache et al., 2020; Fang et al., 2020; Khosravi et al., 2020; Popa et al., 2019). As there are several typologies of inputs, it is important to design an appropriate model to integrate heterogeneous environmental information.

As output data, most papers considered a flood inventory map, given by a set of flooded and non-flooded locations. The flooded locations were derived from measurements and records taken from remote sensing and stations, while non-flooded locations were taken randomly from locations with no previous flood record.

3.4.3. PERFORMANCE ASSESSMENT

In flood susceptibility analysis, both classification and regression metrics are adopted (Fig. 3.5). While classification metrics are used to identify flooded or non-flooded areas, the purpose of regression metrics is often omitted unless the reference target is a discharge hydrograph (Jahangir et al., 2019; Kia et al., 2012). Both types of metrics are used in few papers (e.g., Khosravi et al., 2020; Panahi et al., 2021). Because of the problem's setup, classification metrics are more reliable in the performance assessment. Following the considerations in Section 3.2.5, we selected as preferable metric AUC, also because of its frequent availability for flood susceptibility mapping. For all the papers with comparisons, deep learning models consistently showed improved performances with respect to the reference models with few exceptions (Table 3.3). Deep boost (DB) is a machine learning algorithm based on deep decision trees (Cortes et al., 2014) which could slightly outperform MLP in few works (Ahmed et al., 2022; Chakrabortty, Pal, et al., 2021). Combining optimization algorithms, such as particle swarm optimization, with MLPs, to improve the training, has a limited effect on the performance improvement (Kalantar et al., 2021; Ngo et al., 2018). Moreover, CNNs increase the performance with respect to traditional models more than MLPs. Fang et al. (2020) show that encoding spatial sequentiality with LSTMs work slightly better than 1D-CNNs and 3D-CNNs, however they avoid the comparison with 2D-CNNs.

3.5. DEEP LEARNING FOR FLOOD HAZARD

Flood hazard predicts the depth, velocity, and extent of floods. This application produces maps which evaluate for a certain event its maximum inundation (e.g., Berkhahn et al., 2019; Guo et al., 2020; Löwe et al., 2021) or how it evolves in time (e.g., Lin, Leandro, Gerber, & Disse, 2020; Zhou et al., 2021). While most studies consider the probability of different events, using return periods (e.g., Guo et al., 2020; Kabir et al., 2020), there are few papers which determine the water depth map for a single event (e.g., Chang et al., 2010; Hu et al., 2019). However, no papers were identified to predict the flow velocities. Since the simulation results are taken as ground-truth data for training, deep learning

Table 3.3: Performance of the deep learning and comparison with reference models for flood susceptibility.

Case study	Deep learning		Comparison		Reference
size [km^2]	Model	Metric	Model	Metric	
27	MLP+ensemble	AUC = 0.847	RF	AUC = 0.821	Darabi et al. (2022)
132	MLP	R^2 = 0.802	-	-	Khoirunisa et al. (2021)
147	MLP+PSO	AUC = 0.98	MLP	AUC = 0.96	Kalantar et al. (2021)
207	MLP	R^2 = 0.82	SCS	R^2 = 0.71	Jahangir et al. (2019)
465	MLP	AUC = 0.917	PSO	AUC = 0.929	Chakrabortty, Pal, et al. (2021)
1128	MLP	AUC = 0.901	DB	AUC = 0.917	Ahmed et al. (2022)
1465	MLP	AUC = 0.960	SVM	AUC = 0.936	Tien et al. (2020)
1510	MLP	AUC = 0.970	SVM	AUC = 0.960	Ngo et al. (2018)
2600	MLP+AHP	AUC = 0.953	MLP+FR	AUC = 0.942	Costache et al. (2020)
4673	MLP	AUC = 0.93	DB	AUC = 0.96	Chakrabortty, Pal, et al. (2021)
5264	MLP+FR	AUC = 0.97	FR	AUC = 0.937	Popa et al. (2019)
12050	MLP	AUC = 0.974	-	-	Ahmadlou et al. (2021)
132000	MLP	R^2 = 0.98	-	-	Kourgialas and Karatzas (2017)
131	CNN	AUC = 0.90	RF	AUC = 0.78	Zhao, Pang, Xu, et al. (2020)
605	CNN	AUC = 0.84	RNN	AUC = 0.82	Lei et al. (2021)
1543	CNN	AUC = 0.937	SVM	AUC = 0.883	Wang, Fang, et al. (2020)
12000	CNN	AUC = 0.832	ANFIS	AUC = 0.70	Panahi et al. (2021)
1649195	CNN	AUC = 0.75	-	-	Khosravi et al. (2020)
90016	CNN+FMV	AUC = 0.912	SVM-FMV	AUC = 0.898	Liu et al. (2021)
1543	LSTM	AUC = 0.965	3D-CNN	AUC = 0.956	Fang et al. (2020)

PSO = particle swarm optimization; SCS = soil conservation system model; SVM = support vector machine; AHP = analytic hierarchy process; RF = random forest; FR = frequency ratio; DB = deep boost; ANFIS = adaptive neuro-fuzzy inference system; FMV = fuzzy membership value

models for flood hazard mapping are used as surrogate models in place of numerical models.

The most studied types of floods are river and urban floods. As regards the spatial scale, the models are carried out at local and regional scales. This is probably due to the computational burden of performing several simulations at larger scales to train the deep learning model.

3.5.1. DL ARCHITECTURE

The deep learning models are mainly based on MLPs and RNNs. In particular, RNNs are applied when a spatial-temporal estimation of the water depths is performed. CNNs were initially discarded but are used more in recent years (e.g., Guo et al., 2020; Kabir et al., 2020; Löwe et al., 2021). Hu et al. (2019) and Jacquier et al. (2021) use a LSTM and a MLP, respectively, in combination with a reduced order modelling framework. In the first case,

the DL model is applied on the reduced space, while in the latter DL is used as surrogate for the decomposition method.

3.5.2. INPUT AND OUTPUT DATA

The inputs are hyetographs, which represent the rainfall precipitation or intensity in time (e.g., Berkhahn et al., 2019; Guo et al., 2020; Kao et al., 2021) or hydrographs, which represent the discharge in time (e.g., Chu et al., 2020; Lin, Leandro, Gerber, & Disse, 2020; Zhou et al., 2021). Other inputs such as the DEM and the roughness coefficient, also used for numerical models, are sometimes considered as additional inputs (e.g., Chang et al., 2010; Guo et al., 2020; Huang et al., 2021). Löwe et al. (2021) performed a forward selection to identify relevant topographic variables, showing that aspect and local depressions improve the model's prediction for urban floods.

The output is a water depth map. For the datasets, it is obtained via numerical models based on the 2D shallow water equations. 1D, 1D-2D, and 3D models are also used (Chang et al., 2010; Hu et al., 2019; Kao et al., 2021). The main reason why numerical models are used is to simulate events that have never occurred or have never been observed, such as floods with high return periods. Even though observed data were not employed, they could be used in future research to corroborate the transferability of such methods. When training only on the numerical models' predictions, the deep learning models' results are limited in accuracy by the numerical models' one, i.e., if the numerical model does not represent reality so will the DL model. Thus, when the model is deployed on real data, there may also be some generalization issues caused by the difference between the training and testing data. The inclusion of real measured data may thus also improve the accuracy with respect to numerical models.

3.5.3. PERFORMANCE ASSESSMENT

In flood hazard, regression metrics are used to evaluate the water depth, while classification metrics are used to evaluate the flood extent, as done for flood inundation (Fig. 3.5). While for flood susceptibility and inundation DL models were used to improve the performances, in flood hazard their main focus is to improve the speed, while still maintaining reasonably low errors with respect to the numerical predictions. This is highlighted in Table 3.4, for all papers which provide information on computational times of both numerical and deep learning models. However, the comparison of speed-up across different papers is often unrealistic since it depends on the number of performed numerical simulations and on the type of numerical model. A similar consideration persists for the error scores, as they depend on the scale of the case study and on its resolution. Moreover, the real error of models trained on numerical results depends on that of the underlying numerical simulator. Hence, the latter must be reliable to have trustworthy predictions in real scenarios. A final remark regards the loss function employed in the training of the DL models. The minimization of the squared errors does not guarantee that the solution will have physical meaning. For flood hazard mapping a possible solution is then to enforce the conservation of the mass or momentum equations by adding such terms in the loss function. This provides additional biases on the predicted solution and was shown to increase its performance in representing the numerical models (e.g., Zhang et al., 2021).

Table 3.4: Performance of the deep learning and comparison with numerical models for flood hazard.

Case study size [km^2]	Deep learning Model	Numerical Model	Comparison measure	Speed-up	Reference
0.6	MLP	2D	RMSE = 0.0013 m	1000x	Berkhahn et al. (2019)
7.7	MLP	2D	RMSE = 0.16 m	100x	Chu et al. (2020)
21	MLP+clustering	2D	RMSE < 0.3 m for 99.7% of domain	-	Huang et al. (2021)
31	MLP	1D-2D	MAE = 0.06 m	1000x	Chang et al. (2010)
92	MLP	2D	RMSE < 0.3 m for 82% of domain	-	Lin, Leandro, Gerber, and Disse (2020)
92	MLP	2D	MSE < 0.2 m^2 for 91% of domain	-	Lin, Leandro, Wu, et al. (2020)
-	MLP+ROM	2D	RE = 2.8%	33x	Jacquier et al. (2021)
10	CNN	2D	MAE = 0.0012 m	-	Hosseiny (2021)
10.5	CNN	2D	MAE < 1 m for 93% of domain	2000x	Guo et al. (2020)
14.5	CNN	2D	RMSE = 0.11 m	38x	Kabir et al. (2020)
72	CNN	2D	RMSE = 0.22 m	-	Yokoya et al. (2022)
400	CNN	2D	RMSE = 0.08 m	-	Löwe et al. (2021)
18.5	LSTM+ROM	3D	RMSE = 0.01 m	1500x	Hu et al. (2019)
271	LSTM	2D	RMSE = 0.08 m	-	Kao et al. (2021)
1479	RNN	2D	RMSE = 0.056 m	21x	Zhou et al. (2021)

ROM = reduced order modelling

3.6. KNOWLEDGE GAPS

We identified knowledge gaps regarding generalization, applications in flood management, usability, modelling limitations, and data availability. Some other minor gaps were shown in the previous section.

3.6.1. GENERALIZATION

Generalization refers to the capacity of a model to extrapolate from a training dataset into unseen testing data. This means that a DL model can correctly predict scenarios unused in its development. This property is particularly relevant because training requires data, model set-up, and time. In the context of flood modelling, there are two main generalization objectives: (i) boundary conditions, e.g., different rainfall events, and (ii) topographical changes, i.e., different case studies. However, the transference between different areas is challenging for DL models because of the difference in input and output data. In fact, except for flood inundation mapping, most reviewed papers focused on generalizing different boundary conditions, such as rainfall (e.g., Berkhahn et al., 2019; Guo et al., 2020). Instead, only a few papers tested the model on areas not considered

during training. Löwe et al. (2021) could generate flood hazard maps for unseen areas within the same study region as the training dataset, as there was little variability of inputs and outputs. Zhao, Pang, Xu, et al. (2021) instead pre-trained a model for flood susceptibility on an urban area and then used it for another similar area. They showed that pre-training improves predictions with respect to a model trained from scratch, both in cases of low and high data availability. These works show that such approaches are in their infancy and have been tested on limited datasets. A DL model which cannot generalize to new areas has to be trained every time for a new study case. Thus, it may have limited advantages over a hydraulic model, since it requires more effort, data, and time. Instead, a general DL model which can generalize to new areas could emphasize the advantages over numerical models. This concept was experimented also for rainfall-runoff modelling where DL models outperformed state-of-the-art alternatives in the prediction of ungauged basins in new study areas (Kratzert, Klotz, et al., 2019).

3.6.2. FLOOD APPLICATIONS AND USABILITY

Deep learning has proven useful for assessing flood-prone areas from the location of past events, identifying flooded areas from remote sensing images, and working as a surrogate model for numerical simulations. However, there are still several other applications within this field that could benefit from deep learning models. In particular, we address two flood management applications, *flood risk* and *real-time flood warning*. We also define two desired types of maps, *probabilistic hazard maps* and *flood arrival time maps*. Then, we discuss *dam and dike breach flood* events.

Flood risk combines the probability that a certain event occurs with the associated consequences, such as economic impacts or loss of life. The expected annual loss is a common measure obtained from flood risk assessment and depends on (i) flood hazard, given by event-specific flood characteristics, such as water depth and flow velocity, (ii) exposure, related to the elements at risk, such as buildings and critical infrastructures, and (iii) vulnerability, i.e., the inability of a system to withstand the effects of the event, given, for example, by intensity-damage curves. Flood risk maps are obtained by combining flood hazard maps with damage models. Other approaches are based on MCDA, since the exact flood magnitude and damage are uncertain (de Brito & Evers, 2016). This is done by incorporating various factors that determine flood risk, such as hazard, the performance of defences, topography, and exposure. However, MCDA is based on expert knowledge and is thus subjective. DL models solve this issue and can also yield a higher accuracy, as shown for flood susceptibility mapping. Thus, DL-based approaches could provide alternative methods for assessing flood risk. In addition to the inputs used for flood susceptibility, such as elevation and land use, flood risk mapping may require also other inputs such as population density, spatial estimates of economic value, and building types. Up to now, only Chen et al. (2021) combined DL and flood risk assessment. They showed that ML and DL approaches can estimate flood risk at regional scale, but do not compare their results against other methods, such as MCDA. One drawback of their approach is that the resulting maps were qualitative, while quantitative results should be preferable for risk assessment.

Real-time flood warning is another application that has not been widely addressed. This is needed by local authorities to inform when and where a flood may occur. While

several papers mention real-time prediction, most can be used only after the event has occurred, since they require as input the complete hyetograph or hydrograph of the event. There are a few examples based on RNNs which could forecast floods in near real-time using sensors (Kao et al., 2021) and rainfall distribution (Dong et al., 2021). However, few situations are covered and, thus, more research should focus on filling this gap. An alternative method is to predict the rainfall in real-time and then retrieve the corresponding water depth map by using a similarity measure on a large dataset of previous simulations (Chang et al., 2020). However, such a solution may be challenging because of the large storage requirements. Using DL for surrogate modelling instead showed substantial speed improvements, thus allowing for real-time simulations and forecasts. It is important, however, to provide some reliability metric for testing cases for which no ground-truth values exist. This can eventually lead to a more trust-worthy use of DL models.

Probabilistic hazard mapping captures the model uncertainty related to its inputs and outputs. As pointed out by Di Baldassarre et al. (2010), uncertainties can result in deterministic maps which are only spuriously accurate. But probabilistic maps can account for the uncertainties by assigning a probability of flooding to each domain element. This analysis is generally carried out with ensemble methods such as Monte Carlo simulations (e.g., Papaioannou et al., 2017). However, since they require a vast amount of simulations, only simpler numerical models are used. DL models could be used as surrogates to speed up computation and improve the accuracy of the simpler models.

Arrival time maps estimate the time employed by a flood to reach a certain water depth threshold. They can encode both spatial and temporal information in the same map. So, for a practitioner, they carry at one place detailed information not only on where to intervene but also when to execute mitigation measures. Despite these promises, they have been seldom used in flood management; consequently, they have also not been exploited with DL methods. Using DL for arrival map estimation may be a promising direction to identify critical infrastructure and set up corresponding evacuation plans in real-time. This application may be particularly important for exceptional flood events, such as dike breaches and dam breaks, where little forecast can be made until a failure initiates (Yakti et al., 2018).

Dam break and dike breach floods concern a relevant category of flood events that has been poorly approached with deep learning models. The motivation is probably related to the rarity of such events and the complexity of the phenomena. However, their catastrophic and unexpected effects make their modelling necessary in several situations. Moreover, the effect of flood defences' failure is often disregarded, also because location and modality of possible failures are uncertain. A common way to include the failure of structures is to investigate all possible combinations of locations and boundary conditions, but it can be constrictive both for time and storage capacities. Probabilistic hazard mapping may be a relevant application to include the uncertainty in the failure probability of the flood defence (Domeneghetti et al., 2013).

3.6.3. MODELLING LIMITATIONS

Complex interactions with the natural and built environment, such as dikes or buildings, are difficult to include in deep learning models. Kabir et al. (2020) showed that flood defences can be included if they are present in the simulations used for training and testing. However, no solution presented so far can directly include new flood defences or hydraulic structures such as bridges, canals, and culverts. While building can be included in the DEM (e.g., Löwe et al., 2021), bridges and other hydraulic structures that influence the behaviour of the floods require a more sophisticated treatment.

3.6.4. DATA AVAILABILITY

Deep learning models usually require large quantities of data to achieve good performances. While simulations can provide potentially limitless data, observed data are scarce and depend on the study area. Simulations may also encounter instability issues depending on the numerical schemes and study area. Remote sensing has provided large quantities of data since its vast development in the past decades, but satellite data is still limited by its frequency of observations and dependency on favourable meteorological conditions. Also, UAVs cannot cover wide areas at once. Precipitation and water depth data are available only in a few locations where the measuring stations are present. Thus, new data sources are needed to overcome these limitations.

Another issue, which emerges also from Section 3.2.5, is the lack of a unified framework to compare different approaches with each other. This can be achieved by creating flood-based benchmark datasets for each mapping application. For flood inundation, some datasets have been already used across different works (e.g., Bonafilia et al., 2020). However, works on both flood susceptibility and hazard mapping consider different datasets, focusing on different geographic areas or flood types. One possibility could then be to unify different case studies in a single dataset, for each application, allowing to assess the validity of a model more objectively. For flood susceptibility, case studies with the same input availability could be merged in a dataset with many flood types, scales, and geographical areas. A similar reasoning could be made for flood hazard mapping, selecting, for each case study, initial and boundary conditions for specific return periods.

3.7. SUMMARY OF THE MAIN KNOWLEDGE GAPS

This chapter presented a review of current applications of deep learning models for flood mapping. The chosen search criteria yielded a total of 58 papers published between 2010 and 2021. This review outlined several knowledge gaps, which we used to develop the research questions addressed in this thesis. These can be summarized as follows:

- **Generalization:** Most DL models are case-specific, requiring retraining for new locations. Generalization across spatial domains and boundary conditions (e.g., rainfall) remains a major challenge.
- **Modelling limitations:** DL models often fail to capture the full spatio-temporal dynamics of floods, especially in complex settings such as dike breach floods, which can be less frequent than other types of floods, but also more dangerous. Moreover, they lack inclusion of physical consistency and explicit modelling of hydraulic structures.

- **Usability:** Current DL approaches rarely include uncertainty quantification or validation against physical laws. Additionally, the absence of unified benchmark datasets and evaluation frameworks hinders the comparability and transparency of DL models in flood applications.

The research questions of this thesis (Chapter 1) aim tackle all of these gaps, by progressively increasing the complexity of the problem and of the modelled case study.

4

HYDRAULICS-BASED GRAPH NEURAL NETWORKS

All models are wrong, but some are useful.

George Box

This chapter introduces graph neural networks as an effective modelling strategy for generalizing the spatio-temporal evolution of floods over multiple case studies. We propose a graph neural network layer (SWE-GNN) inspired by finite volume methods used for numerical modelling of floods that introduces a physical (hydraulic) inductive bias. We also develop a training strategy based on curriculum learning to improve the model's stability over extended prediction horizons. We compared the SWE-GNN model with state-of-the-art DL models over three synthetic datasets of dike breach floods of increasing complexity. This chapter addresses the gaps of generalization over unseen case studies and of spatio-temporal modelling of floods, while also providing benchmark datasets. As for the rest of the thesis, we focus on modelling the overland flow caused by dike breach floods.

This chapter is an adapted version of:

Bentivoglio, R., Isufi, E., Jonkman, S. N., & Taormina, R. (2023). Rapid spatio-temporal flood modelling via hydraulics-based graph neural networks. Hydrology and Earth System Sciences, 27(23), 4227–4246. https://doi.org/10.5194/hess-27-4227-2023.

4.1. INTRODUCTION

Accurate flood models are essential for risk assessment, early warning, and preparedness for flood events. Numerical models can characterize how floods evolve in space and time, with the two-dimensional (2D) hydrodynamic models being the most popular (Teng et al., 2017). However, they are computationally expensive, making them inapplicable for real-time emergencies and uncertainty analyses. Data-driven alternatives based on deep learning provide faster numerical solvers with comparable accuracy (Bentivoglio et al., 2022). Most current works explore the generalization of boundary conditions on a fixed domain, but need retraining when applied to a new area, requiring more resources in terms of data, model preparation, and computation times.

To overcome this issue, the community is investigating the generalizability of deep learning models to different study areas. Löwe et al. (2021) proposed a CNN model to estimate the maximum water depth of pluvial urban floods. They trained their model on part of their case study and then deployed it on the unseen parts, showing consistent performances. Guo et al. (2022) accurately predicted the maximum water depth and flow velocities for river floods in different catchments in Switzerland. To incorporate the variations in catchment size and shape, they divided the domain into patches. do Lago et al. (2023) proposed a conditional generative adversarial networks that could predict the maximum water depth unseen rain events in unseen urban catchments. However, these approaches focus on a single maximum depth or velocity map, disregarding the dynamical behaviour, i.e., no information is provided on the flood conditions over space and time, which is crucial for evacuation and response to the flood.

To overcome this limitation, we propose SWE-GNN, a deep learning model merging graph neural networks (GNN) with the finite-volume methods used to solve the shallow water equations (SWE). GNNs generalize convolutional neural networks to irregular

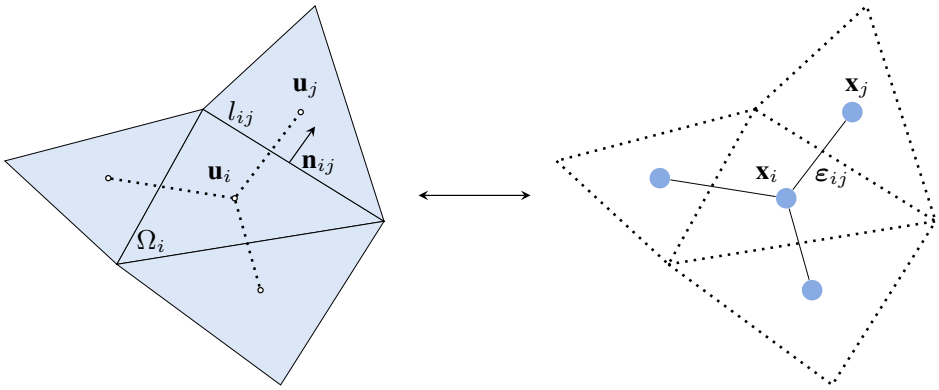


Figure 4.1: Schematic representation of an arbitrary triangular volume mesh and its dual graph. Left: a finite-volume cell Ω_i along with its neighbouring cells. Vectors $\mathbf{u}_i$ and $\mathbf{u}_j$ represent the cells' hydraulic variables, while l_{ij} and $\mathbf{n}_{ij}$ corresponds, respectively, to the length of the mesh side and the outward unit normal vector, between cells i and j. Right: the dual graph of the mesh is obtained by considering each i^{th} cell's centre as a node i, with features $\mathbf{x}_i$ and connecting neighbouring nodes, i and j, via edges ij, with features $\boldsymbol{\varepsilon}_{ij}$.

domains such as graphs and have shown promising results for fluid dynamics (e.g., Lino et al., 2021; Peng et al., 2022) and partial differential equations (e.g., Brandstetter et al., 2022; Horie & Mitsume, 2022). Hence, developing GNNs that follow the SWE equations is not only more physically interpretable but also allows better generalization abilities to unseen flood evolution, unseen breach location, and unseen topographies. In particular, we exploit the geometrical structure of the finite-volume computational mesh by using its dual graph, obtained by connecting the centres of neighbouring cells via edges (see Figure 4.1). The nodes represent finite-volume cells and edges fluxes across them. Following an explicit numerical discretization of the SWE, we formulate a novel GNN propagation rule that learns how fluxes are exchanged between cells, based on the gradient of the hydraulic variables. We set the number of GNN layers based on the time step between consecutive predictions, in agreement with the Courant–Friedrichs–Lewy conditions. The inputs of the model are the hydraulic variables at a given time, elevation, slopes, area, length, and orientation of the mesh's cells. The outputs are the hydraulic variables at the following time step, evaluated in an auto-regressive manner, i.e., the model is repeatedly applied using its predictions as inputs to produce extended simulations.

We tested our model on dike-breach flood simulations due to their time-sensitive nature and presence of uncertainties in topography and breach formation (Jonkman et al., 2008; Vorogushyn et al., 2009). Moreover, given the sensibility to floods of low-lying areas, fast surrogate models that generalize over all those uncertainties are required for probabilistic analyses. The rest of the chapter is structured as follows: Section 4.2 describes the proposed methodology. In Section 4.3, we present the dataset used for the numerical experiments. Section 4.4 shows the results obtained with the proposed model and compares it with other deep learning models. Finally, Section 4.5 discusses the results, analyses the current limitations of this approach, and proposes future research directions.

4.2. SHALLOW-WATER-EQUATIONS GRAPH NEURAL NETWORK

We develop a graph neural network in which the computations are based on the shallow-water equations. The proposed model takes as input both static and dynamic features that represent the topography of the domain and the hydraulic variables at time t, respectively. The outputs are the predicted hydraulic variables at time t+1. In the following, we detail the proposed model (Section 4.2.1) and its inputs and outputs (Section 4.2.2). Finally, we discuss the training strategy (Section 4.2.3).

4.2.1. ARCHITECTURE

SWE-GNN is an encoder-processor-decoder architecture inspired by You et al. (2020) with residual connections, that predicts auto-regressively the hydraulic variables at time t+1 as

$$\hat{\mathbf{U}}^{t+1} = \mathbf{U}^{t} + \Phi(\mathbf{X}_s, \mathbf{U}^{t-p:t}, \mathscr{E}), \tag{4.1}$$

where the output $\hat{\mathbf{U}}^{t+1}$ corresponds to the predicted hydraulic variables at time t+1, $\mathbf{U}^{t}$ are the hydraulic variables at time t, $\Phi(\cdot)$ is the GNN-based encoder-processor-decoder model that determines the evolution of the hydraulic variables for a fixed time step, $\mathbf{X}_s$ are the static node features, $\mathbf{U}^{t-p:t}$ are the dynamic node features, i.e., the hydraulic variables

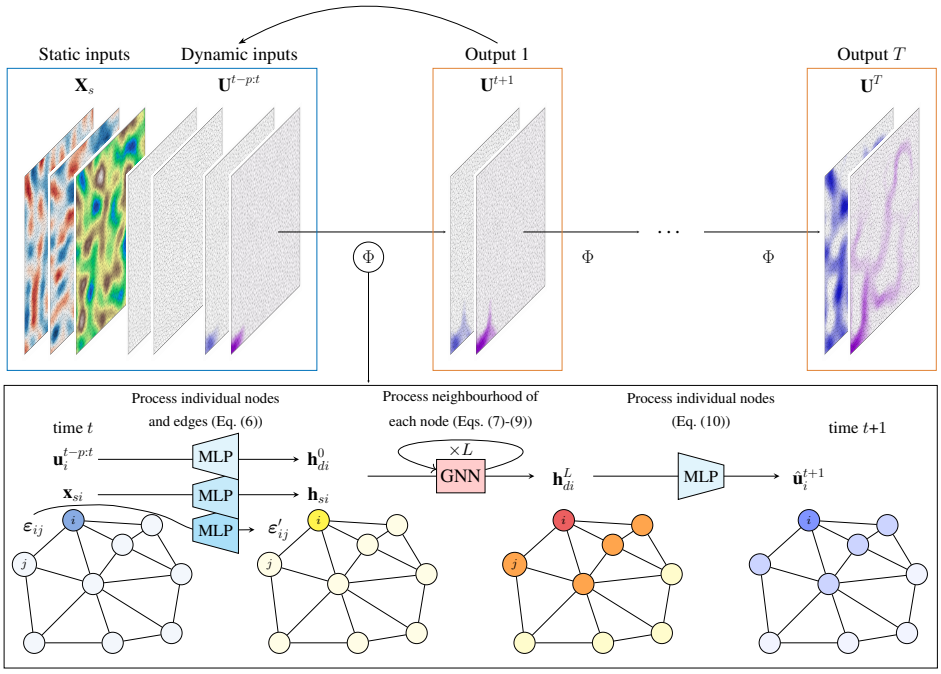


Figure 4.2: Overview of the SWE-GNN model. The model Φ takes as input the mesh discretization of the static and dynamic input (blue box) and produces an estimate of their evolution in time (orange box). The model is then repeated auto-regressively, i.e., using its predictions as inputs, to determine the spatio-temporal evolution of the flood. The encoder-processor-decoder structure of the SWE-GNN model is shown in the bottom black box. The node inputs $\mathbf{x}_{si}$ and $\mathbf{u}_i^{t-p:t}$ represent static attributes, such as elevation and slopes, and dynamic attributes, representing hydraulic variables, while the edge inputs $\boldsymbol{\varepsilon}_{ij}$ represent the mesh's geometry. The inputs are encoded into higher-dimensional embeddings $\mathbf{h}_{si}$, $\mathbf{h}_{di}^0$ (yellow nodes), and $\boldsymbol{\varepsilon}'_{ij}$ via three separate multi-layer perceptrons, shared across nodes or edges. The embeddings, whose purpose is to increase the inputs' expressivity, are used as input for the L GNN layers. The output of the GNN $\mathbf{h}_{di}^L$ (red and orange nodes) is decoded via another shared multi-layer perceptron and summed to the hydraulic variables at time t $\mathbf{u}_i^t$. The final output $\hat{\mathbf{y}}_i$ (blue nodes) represents the prediction at time $t+1$, i.e., $\hat{\mathbf{u}}_i^{t+1}$.

for time steps $t-p$ to t, and $\mathscr{E}$ are the edge features that describe the geometry of the mesh. The architecture detailed in the sequel is illustrated in Fig. 4.2.

Encoder. We employ three separate encoders for processing the static node features $\mathbf{X}_s \in \mathbb{R}^{N\times I_{Ns}}$, dynamic node features $\mathbf{X}_d \equiv \mathbf{U}^{t-p:t} \in \mathbb{R}^{N\times O(p+1)}$, and edge features $\boldsymbol{\varepsilon} \in \mathbb{R}^{E\times I_\varepsilon}$, where I_{Ns} is the number of static node features, O the number of hydraulic variables (e.g., O=3 if we consider water depth and the x and y components of the unit discharges), p the number of input previous time steps, and I_ε the number of input edge features. The encoded variables are

$$\mathbf{H}_s = \phi_s\left(\mathbf{X}_s\right), \mathbf{H}_d = \phi_d\left(\mathbf{X}_d\right), \mathscr{E}' = \phi_\varepsilon\left(\mathscr{E}\right), \tag{4.2}$$

where $\phi_s(\cdot)$ and $\phi_d(\cdot)$ are MLPs shared across all nodes that take an input $\mathbf{X} \in \mathbb{R}^{N\times I}$ and return a node matrix $\mathbf{H} \in \mathbb{R}^{N\times G}$; and $\phi_\varepsilon(\cdot)$ are MLPs shared across all edges that encode the edge features in $\mathscr{E}' \in \mathbb{R}^{E\times G}$. All MLPs have two layers, with hidden dimension G, followed by a *PReLU* activation. The encoders expand the dimensionality of the inputs to allow for higher expressivity, with the hyperparameter G being the dimension of the node embeddings. The i^{th} rows of the node matrices $\mathbf{H}_s$ and $\mathbf{H}_d$ represent the encoded feature vectors associated to node i, i.e., $\mathbf{h}_{si}$ and $\mathbf{h}_{di}$, and the k^{th} rows of the edge matrices $\mathscr{E}'$ represents the encoded feature vector associated to edge k.

Processor. We employed as processor an L-layer GNN that takes a high-dimensional representation of the static and dynamic properties of the system at time t, given by the encoders, and produces a spatio-temporally propagated high-dimensional representation of the system's evolution from time t to t+1. The propagation rule is based on the shallow water equation. In the SWE, the mass and momentum fluxes, representative of the dynamic features, evolve in space as a function of the source terms, representative of the static and dynamic features. Moreover, water can only propagate from sources of water and the velocity of propagation is influenced by the gradients of the hydraulic variables. Thus, the GNN layer $\ell = 1,\ldots,L-1$ update reads as

$$\mathbf{s}_{ij}^{(\ell+1)} = \psi\left(\mathbf{h}_{si}, \mathbf{h}_{sj}, \mathbf{h}_{di}^{(\ell)}, \mathbf{h}_{dj}^{(\ell)}, \boldsymbol{\varepsilon}'_{ij}\right) \odot \left(\mathbf{h}_{dj}^{(\ell)} - \mathbf{h}_{di}^{(\ell)}\right), \tag{4.3}$$

$$\mathbf{h}_{di}^{(\ell+1)} = \mathbf{h}_{di}^{(\ell)} + \sum_{j\in\mathcal{N}_i} \mathbf{s}_{ij}^{(\ell+1)} \mathbf{W}^{(\ell+1)}, \tag{4.4}$$

where $\psi(\cdot) : \mathbb{R}^{5G} \rightarrow \mathbb{R}^G$ is an MLP with two layers, with hidden dimension $2G$, followed by a *PReLU* activation function, $\odot$ is the Hadamard (element-wise) product, and $\mathbf{W}^{(\ell)} \in \mathbb{R}^{G\times G}$ are parameter matrices. The term $\mathbf{h}_{dj}^{(\ell)} - \mathbf{h}_{di}^{(\ell)}$ represents the gradient of the hydraulic variables and enforces water-related variables $\mathbf{h}_d$ to propagate only if at least one of the interfacing node features is non-zero, i.e., has water. The function $\psi(\cdot)$, instead, incorporates both static and dynamic inputs and provides an estimate of the source terms acting on the nodes. Thus, vector $\mathbf{s}_{ij}$ represents the fluxes exchanged across neighbouring cells and their linear combination is used as in Eq. (2.3) to determine the hydraulic variables' variation for a given cell. In this way, Eq. (4.3) resembles how fluxes are evaluated at the cell's interface in the numerical model, i.e., $\delta\mathbf{F}(\mathbf{u})_{ij} = \tilde{\mathbf{J}}_{ij}\left(\mathbf{u}_j - \mathbf{u}_i\right)$, which enforces conservation across interface discontinuities (Martínez-Aranda et al., 2022). Based on this formulation, $\mathbf{s}_{ij}$ can also be interpreted as approximate Riemann solver (Toro, 2013), where the Riemann problem at the boundary between computational

cells is approximated by function $\psi(\cdot)$, in place of equations (e.g., Roe, 1981). To reduce model instabilities, the output of $\psi(\cdot)$ is normalized along its embedding dimension, i.e., it is divided by its norm $\|\psi(\cdot)\|$. This procedure is similar to other graph normalization techniques that improve training stability (Chen et al., 2022). The contribution of each layer is linearly multiplied by $\mathbf{W}^{(\ell)}$ (Eq. (4.3)). From a numerical perspective, this is analogous to an L-order multi-time-step scheme, with L being the number of layers, where the weights are learned instead of being assigned (e.g., Dormand & Prince, 1980).

The GNN's output represents an embedding of the predicted hydraulic variables at time t+1 for a fixed time step Δt. Instead of enforcing stability by limiting Δt, as it is done in numerical models, we can obtain the same result by considering a larger portion of space, which results in increasing Δx (cfr. Eq. (2.5)). This effect can be achieved by stacking multiple GNN layers, as each layer will increase the propagation space, also called neighbourhood size. The number of GNN layers is then correlated to the space covered by the flood for a given temporal resolution. We can then write the full processor for the L GNN layers as

$$\begin{aligned}
\mathbf{h}_{di}^{(0)} &= \mathbf{h}_{di}\mathbf{W}^{(0)}, \\
\mathbf{s}_{ij}^{(\ell+1)} &= \psi\left(\mathbf{h}_{si}, \mathbf{h}_{sj}, \mathbf{h}_{di}^{(\ell)}, \mathbf{h}_{dj}^{(\ell)}, \boldsymbol{\epsilon}'_{ij}\right) \odot \left(\mathbf{h}_{dj}^{(\ell)} - \mathbf{h}_{di}^{(\ell)}\right), \\
\mathbf{h}_{di}^{(\ell+1)} &= \mathbf{h}_{di}^{(\ell)} + \sum_{j \in \mathcal{N}_i} \mathbf{s}_{ij}^{(\ell+1)} \mathbf{W}^{(\ell+1)}, \\
\mathbf{h}_{di}^{(L)} &= \sigma\left(\mathbf{h}_{di}^{(L-1)} + \sum_{j \in \mathcal{N}_i} \mathbf{s}_{ij}^{(L)} \mathbf{W}^{(L)}\right),
\end{aligned} \tag{4.5}$$

where we employ a *Tanh* activation function $\sigma(\cdot)$ at the output of the L^{th} layer to limit numerical instabilities resulting in exploding values. The embedding of the static node features $\mathbf{h}_{si}$ and of the edge features $\boldsymbol{\epsilon}'_{ij}$ do not change across layers, as the topography and discretization of the domain do not change in time.

Decoder. Symmetrically to the encoder, the decoder is composed of an MLP $\varphi(\cdot)$, shared across all the nodes, that takes as input the output of the processor $\mathbf{H}_d^{(L)} \in \mathbb{R}^{N \times G}$ and updates the hydraulic variables at the next time step, i.e., $\hat{\mathbf{U}}^{t+1} \in \mathbb{R}^{N \times O}$, via residual connections, as

$$\hat{\mathbf{U}}^{t+1} = \mathbf{U}^{t} + \varphi\left(\mathbf{H}_d^{(L)}\right). \tag{4.6}$$

The MLP $\varphi(\cdot)$ has two layers, with hidden dimension G, followed by a *PReLU* activation. Both the MLPs in the dynamic encoder and the decoder do not have the bias terms as this would result in adding non-zero values in correspondence of dry areas that would cause water to originate from any node.

4.2.2. INPUTS AND OUTPUTS

We define input features on the nodes and edges based on the SWE terms (cfr. Eq. (2.2)). We divide node features into a static component that represents fixed spatial attributes and a dynamic component that represents the hydraulic variables.

Static node features are defined as

$$\mathbf{x}_{si} = (a_i, e_i, \mathbf{s}_{0i}, m_i, w_i^t) \tag{4.7}$$

where a_i is the area of the i^{th} finite volume cell, its elevation e_i, its slopes in the x and y directions $\mathbf{s}_{0i}$, and its Manning coefficient m_i. We also included the water level at time t, w_i^t, given by the sum of elevation and water depth at time t, as node inputs, since it determines the water gradient (Liang & Marche, 2009). The reason why we include w_i^t in the static attributes instead of the dynamic ones is that this feature can be non-zero also without water, due to the elevation term, and would thus result in the same issue mentioned for the dynamic encoder and decoder.

Dynamic node features are defined as

$$\begin{aligned} \mathbf{x}_{di} &= \mathbf{u}_i^{t-p:t} = (\mathbf{u}_i^{t-p}, ..., \mathbf{u}_i^{t-1}, \mathbf{u}_i^t), \\ \mathbf{u}_i^t &= (h_i^t, |q|_i^t) \end{aligned} \tag{4.8}$$

where $\mathbf{u}_i^t$ are the hydraulic variables at time step t and $\mathbf{u}_i^{t-p:t}$ are the hydraulic variables up to p previous time steps, to leverage the information of past data and provide a temporal bias to the inputs. Contrarily to the definition of the hydraulic variables as in Eq. (2.2), we selected the modulus of the unit discharge $|q|$ as a metric of flood intensity in place of its x and y components to avoid mixing scalar and vector components and because, for practical implications, such as damage estimation, the flow direction is less relevant than its absolute value (e.g., Kreibich et al., 2009).

Edge features are defined as

$$\boldsymbol{\varepsilon}_{ij} = (\mathbf{n}_{ij}, l_{ij}), \tag{4.9}$$

where $\mathbf{n}_{ij}$ is the outward unit normal vector and l_{ij} is the cell's sides length. Thus, the edge features represent the geometrical properties of the mesh. We excluded the fluxes $\mathbf{F}_{ij}$ as additional features as they depend on the hydraulic variables $\mathbf{u}_i$ and $\mathbf{u}_j$, which are already included in the dynamic node features.

Outputs. The model outputs are the estimated water depth and unit discharge at time t+1, i.e., $\hat{\mathbf{u}}_i^{t+1} = \left(\hat{h}_i^{t+1}, |\hat{q}|_i^{t+1}\right)$, resulting in an output dimension $O = 2$. The outputs are used to update the input dynamic node features $\mathbf{x}_{di}$ for the following time step, as exemplified in Fig. 4.3. The same applies for the water level in the static attributes, i.e., $w_i^{t+1} = e_i + \hat{h}_i^{t+1}$.

4.2.3. TRAINING STRATEGY

The model learns from input-output data pairs. To stabilize the output of the SWE-GNN over time, we employ a multi-step-ahead loss function $\mathscr{L}$, that measures the accumulated error for multiple consecutive time steps, i.e.,

$$\mathscr{L} = \frac{1}{HO} \sum_{\tau=1}^{H} \sum_{o=1}^{O} \gamma_o \| \hat{\mathbf{u}}_o^{t+\tau} - \mathbf{u}_o^{t+\tau} \|_2, \tag{4.10}$$

where $\mathbf{u}_o^{t+\tau} \in \mathbb{R}^N$ are the hydraulic variables over the whole graph at time $t+\tau$, H is the prediction horizon, i.e., the number of consecutive time instants, and γ_o are coefficients used to weight the influence of each variable to the loss. For each time step τ, we evaluate the model's prediction $\hat{\mathbf{u}}^{t+\tau}$ and then use the prediction recursively as part of the new dynamic node input (see Fig. 4.3). We repeat this process for a number of time steps H and calculate the root mean squared error (RMSE) loss as the average over all steps. In this

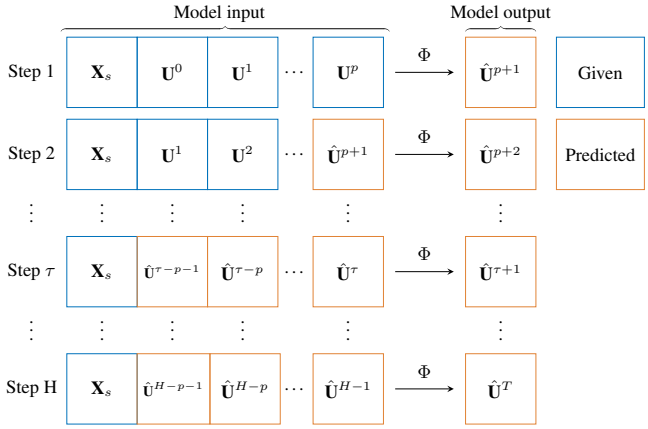


Figure 4.3: Example of auto-regressive prediction for p input previous time steps and H predicted steps ahead. The prediction at time τ are used as new inputs to predict the following time step and so on. The loss and the metrics are evaluated as the average over all steps H.

way, the model learns to correct its own predictions while also learning to predict a correct output, given a slightly wrong prediction, hence improving its robustness. After $p+1$ prediction steps, the inputs of the model are given exclusively by its predictions. During training, we limit the prediction horizon H instead of using the full temporal sequence due to memory constraints, since the back-propagation gradients must be stored for each time step.

To improve the training speed and stability, we also employed a curriculum learning strategy (Algorithm 1). This consists in progressively increasing the prediction horizon in Eq. (4.10) every fixed number of epochs up to H. The idea is to first learn the one- or few-steps-ahead predictions to fit the short-term predictions and then increase the number of steps ahead to stabilize the predictions (Wang et al., 2022).

Algorithm 1 Curriculum learning strategy

Initialize:
$H = 1$
$CurriculumSteps = 15$
$\gamma_1 = 1$ (Water depth h)
$\gamma_2 = 3$ (Unit discharge q)
for epoch = 1 to MaxEpochs **do**
$\hat{\mathbf{U}}^{t+1} = \mathbf{U}^t + \Phi(\mathbf{X}_s, \mathbf{U}^{t-p:t}, \mathscr{E})$
$\mathscr{L} = \frac{1}{HO}\sum_{\tau=1}^{H}\sum_{o=1}^{O}\gamma_o \|\hat{\mathbf{u}}_o^{t+\tau} - \mathbf{u}_o^{t+\tau}\|_2$,
Update the parameters
if epoch > CurriculumSteps*H **then**
$H = H + 1$
end if
end for

4.3. EXPERIMENTAL SETUP

4.3.1. DATASET GENERATION

We considered 130 numerical simulations of dike-breach floods ran on randomly-generated topographies over two squared domains of sizes $6.4 \times 6.4 km^2$ and $12.8 \times 12.8 km^2$ representative of flood-prone polder areas.

We generated random digital elevation models using the Perlin noise generator (Perlin, 2002) as its ups and downs reflect plausible topographies. We opted for this methodology, instead of manually selecting terrain patches, to automatize the generation process, thus allowing for an indefinite amount of randomized and unbiased training and testing samples.

We employed a high-fidelity numerical solver, Delft3D-FM, which solves the full shallow water equations using an implicit scheme on staggered grids and adaptive time steps (Deltares, 2022). We used a dry bed as the initial condition and a constant input discharge of $50 m^3/s$ as the boundary condition, equal to the maximum dike-breach discharge. We employed a single boundary condition value for all simulations as our focus is to show generalizability over different topographies and breach locations. The simulation output is a set of temporally-consecutive flood maps, with a temporal resolution of 30 minutes.

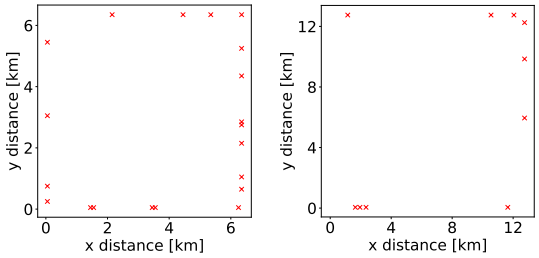
We created three datasets with different area sizes and breach locations as summarized in Table 4.1. We selected a rectangular domain discretized by regular meshes, to allow for a fairer comparison with other models that cannot work with meshes or cannot incorporate edge attributes. Furthermore, we considered a constant roughness coefficient m_i for all simulations, meaning that we use the terrain elevation and the slopes in the x and y directions as static node inputs.

1. The first dataset consists of 100 digital elevation models (DEMs) over a squared domain of 64×64 grids of length $100m$ and a simulation time of 48 hours. This dataset is used for training, validation, and testing. We used a fixed testing set of 20 simulations while the remaining 80 simulations are used for training (60) and validation (20).

2. The second dataset consists of 20 DEMs over a squared domain of 64×64 grids of length of $100m$ and a simulation time of 48 hours. The breach location changes randomly across the border with a constant discharge of $50 m^3/s$ (Fig. 4.4a). This dataset is used to test the generalizability of the model to unseen domains and breach locations.

Table 4.1: Summary of the datasets employed for training (TR), validation (VA), and testing (TE). The uncertainty accounts for the variability across the different simulations in each dataset.

Dataset and use	Number of simulations	Size (km^2)	Random breach location	Simulation duration (h)	Execution time of numerical model (s)
1 (TR,VA,TE)	100	6.4×6.4	No	48	29.5 ± 9.1
2 (TE)	20	6.4×6.4	Yes	48	32.5 ± 5.1
3 (TE)	10	12.8×12.8	Yes	120	185.5 ± 29.9



(a) Dataset 2 (b) Dataset 3

Figure 4.4: Distribution of the breach locations (red crosses) for datasets 2 and 3.

3. The third dataset consists of 10 DEMs over a squared domain of 128 × 128 grids of length of 100m. The boundary conditions are the same as for the second dataset. Since the domain area is four times larger, the total simulation time is 120 hours, to allow for the flood to cover larger parts of the domain. This dataset is used to test the generalizability of the model to larger unseen domains, unseen breach locations, and longer time horizons.

Unless otherwise mentioned, we selected a temporal resolution of Δt=1h, as a trade-off between detail and speed. When the beginning of the flood is relevant (e.g., for real-time forecasts) higher temporal resolutions are better. Contrarily, if the the final flood state is relevant, lower temporal resolutions may be better.

4.3.2. TRAINING SETUP

We trained all models via the Adam optimization algorithm (Kingma & Ba, 2014). We employed a varying learning rate with 0.005 as starting value and a fixed step decay of 90% every 7 epochs. The training was carried out for 150 epochs with early stopping. We used a maximum prediction horizon $H = 8$ steps ahead during training as a trade-off between model stability and training time, as later highlighted in Section 4.4.4. There is no normalization pre-processing step and, thus, the values of water depth and unit discharge differ in magnitude by a factor of 10. Since for application purposes discharge is less relevant than water depth (Kreibich et al., 2009), we weighted the discharge term by a factor of $\gamma_2 = 3$ (cfr. Eq. (4.10)), while leaving the weight factor for water depths as $\gamma_1 = 1$. Finally, we used $p = 1$ previous time step as input, i.e., $\mathbf{X}_d = (\mathbf{U}^{t=0}, \mathbf{U}^{t=1})$, where the solution at time $t = 0$ corresponds to dry bed conditions. The static inputs $\mathbf{X}_s$ are given by the slopes in the x and y directions, and the elevation. The corresponding inputs and outputs can be visualized in Figure 4.5.

We trained all models using the PyTorch (Version 3.10.8) (Paszke et al., 2019) and PyTorch Geometric (Version 2.2) (Fey & Lenssen, 2019). In terms of hardware, we employed an Nvidia Tesla V100S-PCIE-32GB for training and deployment ((DHPC), 2022), and an Intel(R) Core(TM) i7-8665U @1.9 GHz CPU for deployment and for the execution of the numerical model. We run the models on both GPUs and CPUs to allow for a fair comparison with the numerical models.

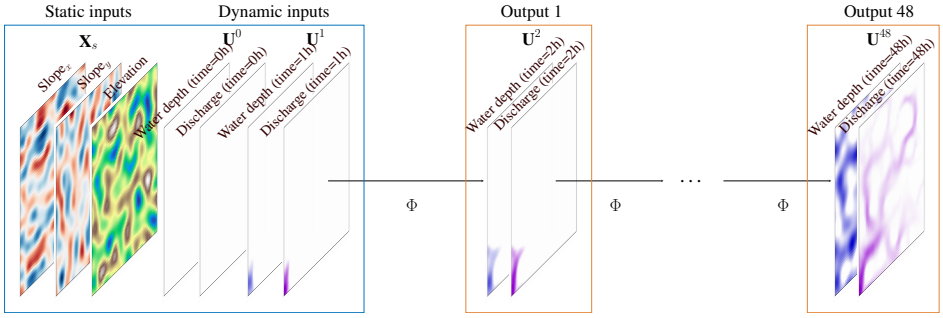


Figure 4.5: Detailed inputs and outputs used in the experiments, considering a regular mesh, $p = 1$ previous time steps and a time resolution $\Delta t = 1h$. The initial inputs are dry bed conditions, i.e., $\mathbf{U}^{t=0h}$, and the first time step of the simulation, i.e., $\mathbf{U}^{t=1h}$, given by the numerical model.

4

4.3.3. METRICS

We evaluated the performance using the multi-step-ahead RMSE (Eq. (4.10)) over the whole simulation. However, for testing, we calculated the RMSE for each hydraulic variable o independently as:

$$RMSE_o = \frac{1}{H}\sum_{\tau=1}^{H} \|\hat{\mathbf{u}}_o^\tau - \mathbf{u}_o^\tau\|_2, \tag{4.11}$$

Analogously, we evaluated the mean absolute error (MAE) for each hydraulic variable o over the whole simulation as:

$$MAE_o = \frac{1}{H}\sum_{\tau=1}^{H} \|\hat{\mathbf{u}}_o^\tau - \mathbf{u}_o^\tau\|_1, \tag{4.12}$$

The prediction horizon H depends on the total simulation time and temporal resolution, e.g., predicting 24 hours with a temporal resolution of 30 min results in $H = 48$ steps ahead. We also measured the spatio-temporal error distribution of the water depth using the critical success index (CSI) for threshold values of 0.05 m and 0.3 m, as in Löwe et al. (2021). The CSI measures the spatial accuracy of detecting a certain class (e.g., flood or no-flood) and, for a given threshold, it is evaluated as

$$CSI = \frac{TP}{TP + FP + FN} \tag{4.13}$$

where TP are the true positives, i.e., number of cells where both model and simulations predict flood, FP are the false positives, i.e., number of cells where the model wrongly predicts flood, and FN are the false negatives, i.e., number of cells where the model does not recognize a flooded area. We selected this measure as it discards true negatives, i.e., when both model and simulation predict no flood, as this condition is over-represented, especially for the initial time steps. Thus, including true negatives may give an overconfident performance estimate. We measured the computational speed-up as the ratio between the computational time required by the numerical model and the inference time of the deep learning model. Both times refer to the execution of the complete flood simulation but do not include the time required to simulate the initial time steps.

4.4. NUMERICAL RESULTS

4.4.1. COMPARISON WITH OTHER DEEP LEARNING MODELS

The proposed SWE-GNN model is compared with other deep learning methods including:

- CNN: encoder-decoder convolutional neural network, based on U-NET (Fig. 2.5). The CNN considers the node feature matrix $\mathbf{X}$ reshaped as a tensor $\mathscr{X} \in \mathbb{R}^{g \times g \times I_N}$, where g is the number of grid cells, i.e., 64 for datasets 1 and 2 and 128 for dataset 3, and I_N is the number of static and dynamic features. This baseline is used to highlight the advantages of the mesh dual graph as an inductive bias in place of an image;
- GAT: graph attention network (Velickovic et al., 2017). The weights in the propagation rule are learned, considering an attention-based weighting. This baseline is considered to show the influence of learning the propagation rule with an attention mechanism;
- GCN: graph convolutional neural network (Defferrard et al., 2016). This baseline is considered to show the influence of not learning the edge propagation rule, in place of learning it;
- SWE-GNN$_{ng}$: SWE-GNN (Eq. (4.3)) without the gradient term $\mathbf{x}_{dj} - \mathbf{x}_{di}$. This is used to show the importance of the gradient term in the graph propagation rule.

We evaluated also MLP-based models, but their performance was too poor and we do not report it. All models consider the same node features inputs $\mathbf{X} = (\mathbf{X}_s, \mathbf{X}_d)$, produce the same output $\hat{\mathbf{Y}} = \mathbf{U}^{t+1}$, produce extended simulations by using the predictions as input (as in Fig. 4.3), and use the same training strategy with the multi-step-ahead loss and curriculum learning. For the GNN-based models, we replaced the GNN in the processor, while keeping the encoder-decoder structure as in Fig. 4.2. We conducted a thorough hyperparameter search for all models, and we selected the one with the best validation loss. Table 4.2 shows the hyperparameters employed for each model. Some hyperparameters are common to all models, such as learning rate, number of maximum training steps ahead, and optimizer, while other change depending on the model, such as embedding dimensions and number or layers.

For the CNN architecture, the best model has three down- and up-scaling blocks, with 64 filters in the first encoding block. Interestingly, we achieved good results only when employing batch normalization layers, *PReLU* as an activation function, and no residual connections between time steps. All other standard combinations resulted in poor performances, which we did not report as they are outside the scope of the analysis. For the GNN-based architectures, all hyperparameter searches resulted in similar best configurations, i.e., $L = 8$ GNN layers and an embedding size G=64.

In Table 4.3, we report the testing MAE for water depth and discharges, and the CSI scores for all models. The proposed SWE-GNN model and the U-NET-based CNN perform consistently better than all other models, with no statistically significant difference in performance according to the Kolmogorow-Smirnov test (p-value less than 0.05). The CNN performs similar to the SWE-GNN because the computations on a regular grid are similar to those of a GNN. Nonetheless, there are valuable differences between the two

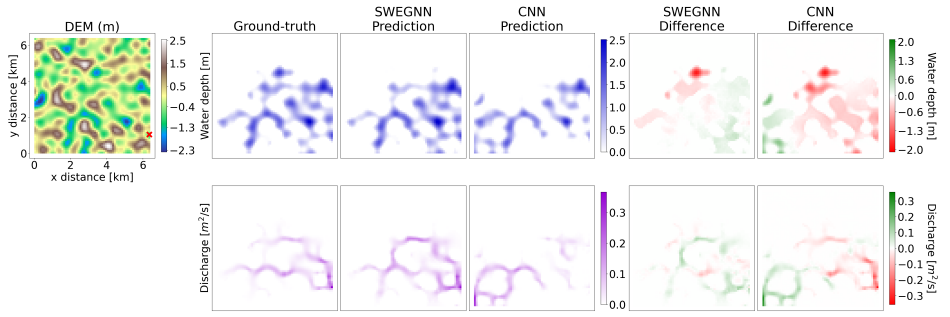
Table 4.2: Summary of the hyperparameters and related values' ranges employed for the different deep learning models. The **bold** values indicate the best configuration in terms of validation loss.

DL model	Hyperparameter name	Values' range (**best**)
All models	Initial learning rate	0.005
	Input previous time steps (p)	1
	Temporal resolution (Δt)	$1h$
	Maximum training steps ahead (H)	8
	Optimizer	Adam
GNN models	Embedding dimension (G)	8,16,32,**64**
	Number of GNN layers (L)	1,2,3,4,5,6,7,**8**,9
	Batch size	8
CNN	First embedding dimension	16,32,**64**, 128
	Number of encoding blocks	1,2,**3**,4
	Activation function	ReLU, **PReLU**, no activation
	Batch size	64

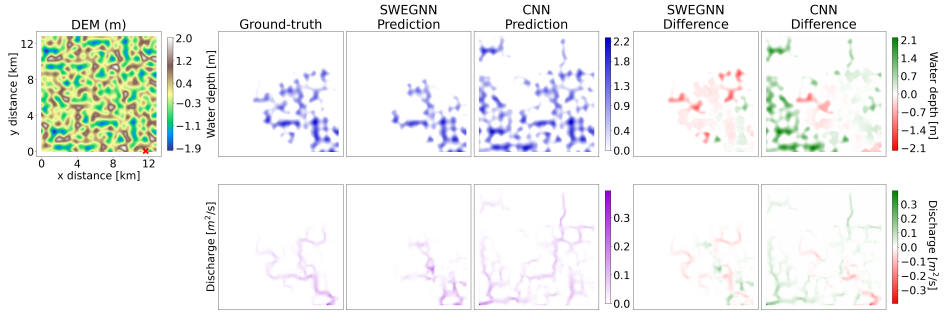
models. First, SWE-GNN is by definition more physically explainable as water can only propagate from wet cells to neighboring cells, while in the CNN there is no such physical constraint, as exemplified by Fig. 4.6. Second, as emphasized in the following section, the SWE-GNN results in improved generalization abilities. Moreover, contrarily to CNNs, GNNs can also work with irregular meshes. Regarding the other GNN-based models, we noticed that the GAT model had the worse performance, indicating that the propagation rule cannot be learned efficiently via attention mechanisms. Moreover, the GCN and the SWE-GNN$_{ng}$ achieved comparable results meaning that the gradient term gives a relevant contribution to the model as its removal results in a substantial loss in performance. We expected this behavior as, without this term, there is no computational constraint to how water propagates.

Table 4.3: Performance of the deep learning models over the test dataset 1. The provided uncertainty estimates account for the variability across the different simulations in the dataset. **Bold** results indicate the best performances, considering a statistical significance with a p-value of 0.05.

DL model	**MAE** ↓		**CSI**$_\tau$ **[%]** ↑	
	h (m) [10^{-2}]	$\|q\|$ (m^2/s) [10^{-2}]	τ=0.05 m	τ=0.3 m
CNN	**3.87 ± 1.29**	**0.42 ± 0.13**	**75.64 ± 9.40**	**73.42 ± 9.26**
GAT	9.27 ± 0.73	5.78 ± 0.11	34.50 ± 10.91	27.07 ± 8.63
GCN	6.05 ± 1.62	0.57 ± 0.11	61.14 ± 13.34	58.89 ± 11.90
SWE-GNN_ng	6.10 ± 1.56	0.63 ± 0.07	58.61 ± 11.97	57.91 ± 12.62
SWE-GNN	**3.93 ± 1.63**	**0.37 ± 0.10**	**75.85 ± 9.30**	**73.44 ± 9.28**



(a) Dataset 2



(b) Dataset 3

Figure 4.6: Comparison of the proposed SWEGNN model against the CNN, for two examples in test datasets 2 (a) and 3 (b). In each panel, the top left image represents the digital elevation model (DEM), along with a red cross in correspondence of the breach location. The following blocks represent, respectively, the ground-truth numerical results, the SWE-GNN predictions, and the CNN predictions for water depth and unit discharges, at the last time instant of the simulation (i.e., 48h for dataset 2 and 120h for dataset 3).

4.4.2. GENERALIZATION TO OTHER BREACH LOCATIONS AND LARGER AREAS

We further tested the already trained models on datasets 2 and 3, with unseen topographies, unseen breach locations, larger domain sizes, and longer simulation times, as

Table 4.4: Performance of the deep learning models over the test datasets 2 and 3, respectively composed of unseen domains with unseen breach locations and unseen domains four times bigger than the training ones, also with unseen breach locations. The provided uncertainty estimates account for the variability across different simulations. **Bold** results indicate the best performances, considering a statistical significance with a p-value of 0.05.

Test dataset	**DL model**	**MAE ↓**		**CSI$_\tau$ [%] ↑**	
		h (m) $[10^{-2}]$	$\|q\|$ (m^2/s) $[10^{-2}]$	τ=0.05 m	τ=0.3 m
2	CNN	6.50 ± 2.37	0.54 ± 0.13	51.90 ± 20.25	47.82 ± 18.42
	SWE-GNN	**4.84 ± 1.87**	**0.48 ± 0.13**	**73.62 ± 8.04**	**68.46 ± 7.13**
3	CNN	6.07 ± 1.77	**0.36 ± 0.10**	42.16 ± 15.63	40.92 ± 15.96
	SWE-GNN	**3.77 ± 1.98**	**0.31 ± 0.12**	**68.53 ± 10.18**	**64.53 ± 11.20**

described in Table 4.1. In the following, we omit the other GNN-based models, since their performance was poorer, as highlighted in Table 4.3.

Table 4.4 shows that all metrics remain comparable across the various datasets for SWE-GNN, with test MAE of approximately $0.04m$ for water depth and $0.004m^2/s$ for unit discharges, indicating that the model has learned the dynamics of the problems. The speed-up on GPU of SWE-GNN over dataset 3 further increased, with respect to the smaller areas of dataset 1 and 2, reaching values twice as higher, i.e., ranging from 100 to 600 times faster than the numerical model on the GPU. We attribute this to the deep learning models' scalability and better exploitation of the hardware for larger graphs.

In Figure 4.6, we see two examples of SWE-GNN and CNN on the test datasets 2 and 3. The SWE-GNN model predicts better the flood evolution over time for unseen breach locations, even in bigger and unseen topographies, thanks to its hydraulic-based approach. On the other hand, the CNN strongly over- or under-predicts the flood extents, unless the breach location is close to that of the training dataset, indicating that it lacks the correct inductive bias to generalize floods. For both models, the predictions remain stable even for time horizons 2.5 times longer than those in training.

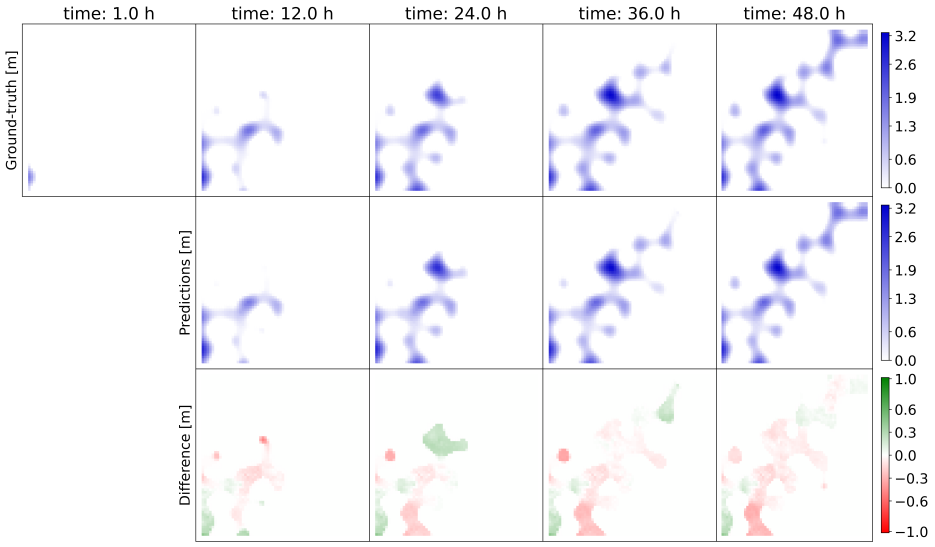
4.4.3. SWE-GNN MODEL ANALYSIS

Over the entire test part of dataset 1, the model achieves an MAE of $0.04m$ for water depth and $0.004m^2/s$ for unit discharges, with respect to maximum water depths and unit discharges respectively of $2.88m$ and $0.55m^2/s$, and average water depths and unit discharges of $0.62m$ and $0.037m^2/s$.

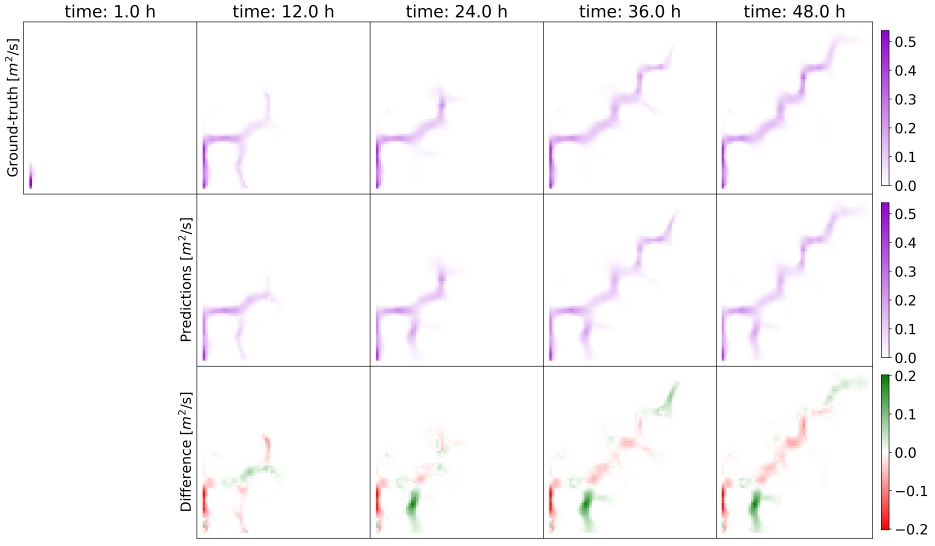
We illustrate the spatio-temporal performance of the model on a test sample in Figure 4.7. Water depth and discharges evolve accurately over time, overall matching the ground-truth numerical results. The errors are related to small over- or under-predictions, a few incorrect flow routes, and lags in the predictions resulting in delays or anticipations that are corrected by the successive model iterations. In particular, the model struggles to represent discharges in correspondence of ponding phenomena, i.e., when an area gets filled with water and then forms a temporary lake, as exemplified in the bottom-left part of the domain in Figure 4.7b. This is because of the lower contribution of the discharges to the training loss. Nonetheless, the error does not propagate over time, thanks to the multi-step-ahead loss employed during training. In fact, the model updates the solution for the entire domain at each time step. Consequently, it exploits information on newly flooded neighborhoods to recompute better values for the cells that were flooded before.

We also observe the average performance of the different metrics over time, for the whole test dataset 1, in Figure 4.8. The CSI is consistently high throughout the whole simulation, indicating that the model correctly predicts where water is located in space and time. On the other hand, both MAE and RMSE increase over time. This is partially due to the evaluation of both metrics via a spatial average, which implies that in the first time steps, where the domain is mostly dry, the error will naturally be lower. Nonetheless, the errors increase linearly or sub-linearly, implying that they are not prone to explode exponentially.

Next, we analyzed the relationship between the number of GNN layers and the temporal resolution, to validate the hypothesis that the number of layers is correlated with the time steps. Following the CFL condition, we can expand the computational domain



(a) Water depths



(b) Discharges

Figure 4.7: SWE-GNN model's predictions for water depth (a) and discharges (b). The results are displayed over time for a test topography in dataset D1_1, comparing the ground-truth output of the numerical simulation (top row) with the predictions (middle row). The difference (bottom row) is evaluated as the predicted value minus the ground-truth one; thus, positive values correspond to model over-predictions while negative values correspond to under-predictions. The legends refer to the maximum values throughout the whole simulation. The top left panel in both sub-figures represents the initial hydraulic conditions given as input to the DL model, along with the dry bed conditions at time $t = 0$.

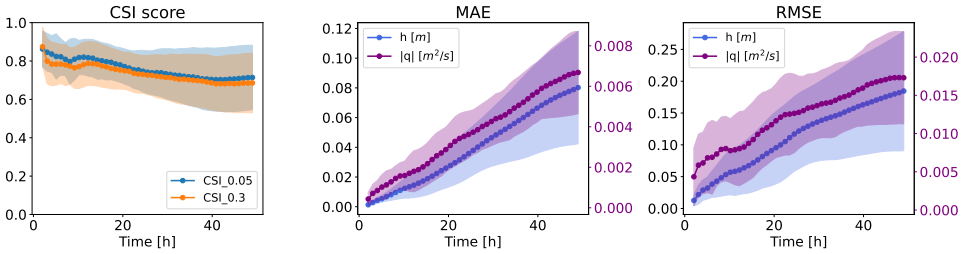


Figure 4.8: Temporal evolution of CSI scores, MAE, and RMSE for test dataset 1. The confidence bands refer to one standard deviation from the mean.

by increasing the number of GNN layers in the model instead of decreasing the time steps. We considered several models with an increasing number of GNN layers targeting temporal resolutions $\Delta t = 30, 60, 90, 120 min$. Figure 4.9 shows that lower temporal resolutions (e.g., $120 min$) require more GNN layers to reach the same performance as that of higher temporal resolutions (e.g., $30 min$). One reason why the number of layers does not increase linearly with the temporal resolution may be that the weighting matrices $\mathbf{W}_\ell$ (cfr. Eq. (4.3)) improve the expressive power of each layer, leading to fewer layers than needed otherwise.

Finally, we explored different model complexity combinations, expressed by the number of GNN layers and the latent space size, to determine a Pareto-front for validation loss and speed-up, which results in a trade-off between fast and accurate models. Figure 4.10 shows that increasing the complexity reduces both errors and speed-ups while improving the CSI, as expected. While for the GPU the number of hidden features does not influence the speed-up, the performance on the CPU depends much more on it, with bigger models being slower, implying different trade-off criteria for deployment.

We also employed the same trained models to test on dataset 3. Figure 4.11 shows that the models performs better in terms of speed with respect to the smaller areas, achieving similar CPU speedups and GPU speedups around two times higher than those in datasets 1 and 2.

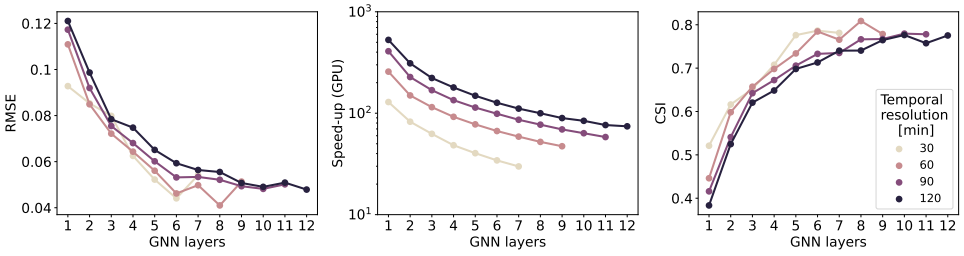


Figure 4.9: Relationship between the number of GNN layers and different temporal resolutions, in terms of validation RMSE and validation CSI. As the temporal resolution decreases and, conversely, as the time step increases, the optimal number of GNN layers, in terms of desired performance level, increases.

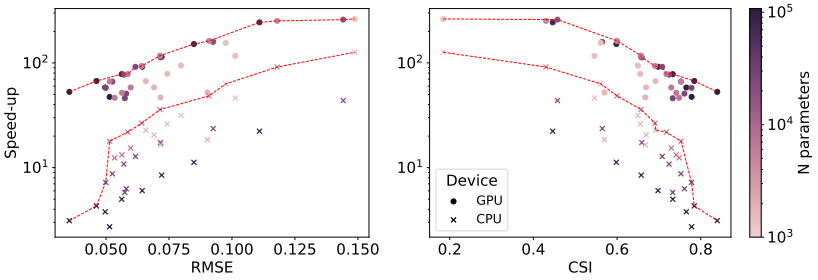


Figure 4.10: Pareto fronts (red-dotted lines) of dataset 1 in terms of speed-ups, validation RMSE, and CSI for varying number of parameters, both for CPU and GPU, for a temporal resolution Δt=1h.

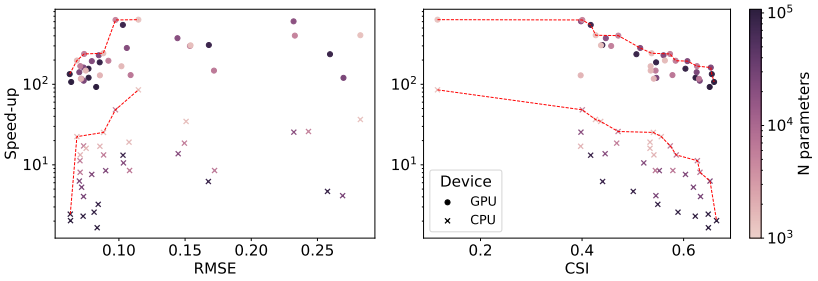
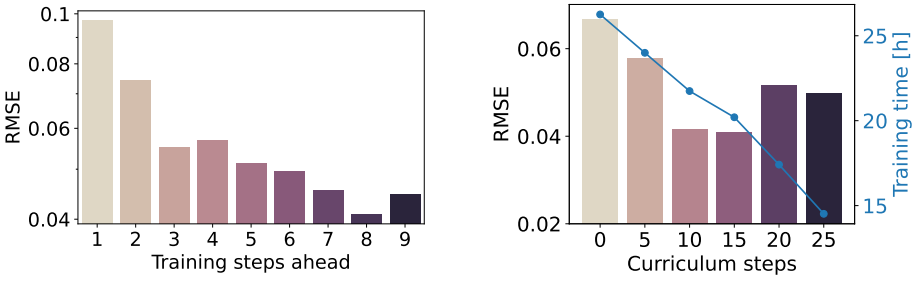


Figure 4.11: Pareto fronts on test dataset 3 (red-dotted lines) in terms of speed-ups, RMSE, and CSI for varying number of parameters for a temporal resolution Δ t=1h.

4.4.4. SENSITIVITY ANALYSIS ON THE TRAINING STRATEGY



(a) Multi-step-ahead loss

(b) Curriculum learning

Figure 4.12: Influence of: (a) the number of training steps ahead on the validation RMSE and (b) the update interval in the curriculum learning.

Finally, we performed a sensitivity analysis on the role of the multi-step-ahead function (cfr. Eq. (4.10)) and the curriculum learning (Algortihm 1) on the training performance. Sensitivity analysis is a technique that explores the effect of varying hyper-

parameters to understand their influence on the model's output. Figure 4.12a shows that increasing the number of steps ahead improves the performance. Increasing the number of steps implies higher memory requirements and longer training times. Because of the best performances and GPU availability, we selected 8 steps ahead in all experiments. However, when performing bigger hyper-parameter searches or when limited by hardware, choosing fewer steps ahead can result in an acceptable performance. Similar considerations can also be done for the CNN model.

Figure 4.12b shows that increasing the interval of curriculum steps linearly reduces the training times, while also improving the performance. The decrease in performance associated to bigger values is probably caused by the number of total training epochs, i.e., 150, which are insufficient to cover the whole prediction horizon H. Increasing the total number of epochs should increase both the performance and the training time but we avoided this analysis and chose an interval of 15 epochs for the curriculum learning strategy, as a trade-off between performance and training times. Moreover, models with curriculum steps between 0 and 15 suffered from spurious instabilities during training, that were compensated with early stopping, while models with more curriculum steps were generally more stable. This is due to sudden variations in the loss function that limit a smoother learning process.

4.5. CONCLUSION

We proposed a deep learning model for rapid flood modelling, called SWE-GNN, inspired by shallow water equations (SWE) and graph neural networks (GNN). The model takes the same inputs as a numerical model, i.e., the spatial discretization of the domain, elevation, slopes, and initial value of the hydraulic variables, and predicts their evolution in time in an auto-regressive manner. The results show that the SWE-GNN can correctly predict the evolution of water depth and discharges with mean absolute errors in time of 0.04 m and 0.004 m^2/s, respectively. It also generalizes well to previously unseen topographies with varying breach locations, bigger domains, and longer time horizons. SWE-GNN is up to two orders of magnitude times faster than the underlying numerical model. Moreover, the proposed model achieved better performances with respect to other deep learning models, in terms of water depth and unit discharge errors as well as CSI.

In line with the hypothesis, GNNs proved to be a valuable tool for spatio-temporal surrogate modelling of floods. The analogy with finite volume methods is relevant for three motivations. First, it improves the deep learning model's interpretability, as the weights in the graph propagation rule can be interpreted as an approximate Riemann solver and multiple GNN layers can be seen as intermediate steps of a multi-step method such as Runge-Kutta. Second, the analogy also provides an existing framework to include conservation laws in the model and links two fields that can benefit from each other advances. For example, multiple spatial and temporal resolutions could be jointly used, in place of a fixed one, similarly to Liu et al. (2022). Third, the methodology is applicable for any flood modelling application where the SWE hold, such as storm surges and river floods. The same reasoning can also be applied to other types of partial differential equations where finite volume methods are commonly used, such as in computational fluid dynamics. The conclusions of the chapter can be summarized as follows:

- We developed a graph neural network model where the propagation rule and the inputs are taken from the shallow water equations. The hydraulic bias improves performance and **generalization** to unseen case studies with respect to other deep learning models;
- We improve the model's stability by training it via a multi-step-ahead loss function, that results in stable predictions up to 120 hours ahead, using only the information of the first hour as initial hydraulic input;
- We show that the proposed model can surrogate numerical solvers for spatio-temporal flood modelling in unseen topographies and unseen breach locations, with two orders of magnitude speed-ups.
- We created three benchmark datasets of dike breach floods of increasing complexity, which can be used by other researchers to compare new DL models.

The current analysis was carried out under a constant breach inflow as a boundary condition. In Chapter 5, extend the analysis to time-varying boundary conditions to better represent complex real-world scenarios. This is done by employing ghost cells, elements at the domain boundaries to which we assign known values in time (LeVeque et al., 2002). This version of the SWE-GNN model cannot yet completely replace numerical models, as it requires the first time step of the flood evolution as input. This challenge is addressed in Chapter 5 by including boundary conditions in the model's inputs. Contrarily to physically-based numerical methods, the proposed model does not strictly enforce conservation laws, such as mass balance. We address this limitation in Chapter 5 by adding conservation equations in the training loss function, as is commonly done with physics-informed neural networks. Finally, while we empirically showed that the proposed model along with the multi-step-ahead loss can sufficiently overcome numerical stability conditions, we provide no theoretical guarantee that stability can be enforced for an indefinite amount of time steps.

While we tested our model on a regular grid for a fair comparison, SWE-GNN can also work on irregular meshes. In Chapter 5, we analyse the stability and performance of the model to different unstructured discretizations of the domain. In Chapter 5, we also improve the model's Pareto front by employing multi-scale methods that allow to reduce the number of message passing operations, while still maintaining the same interaction range (e.g., Fortunato et al., 2022; Lino et al., 2022). Future research could also investigate advances in GNNs with spatio-temporal models (e.g., Sabbaqi & Isufi, 2023) or generalizations to higher-order interactions (e.g., Yang et al., 2022) that may further benefit the accuracy of the model.

5

MULTI-SCALE HYDRAULIC GRAPH NEURAL NETWORKS

You don't drown by falling into water.
You only drown if you stay there.

Zig Ziglar

This chapter presents an improved version of the model presented in Chapter 4. We developed a multi-scale hydraulic graph neural network (mSWE-GNN) that models the flood at different resolutions and propagation speeds, obtaining a better Pareto front in terms of speed and accuracy than its non-multi-scale counterpart. We included time-varying boundary conditions via ghost cells, which enforce the solution at the domain's boundary and drop the need for a numerical solver for the initial conditions. To improve generalization over unseen unstructured meshes and reduce the data demand, we use invariance principles and make the inputs independent from coordinates' rotations. We further corroborate the mSWE-GNN in a realistic case study in the Netherlands and show generalization capabilities with only one fine-tuning sample. This chapter addresses the gap of joint generalization over unseen case studies, boundary conditions, and unstructured meshes for overland flow caused by dike breach floods. As for Chapter 4, we also provide a benchmark dataset of increased complexity.

This chapter is an adapted version of:

Bentivoglio, R., Isufi, E., Jonkman, S. N., & Taormina, R. (2025). Multi-scale hydraulic graph neural networks for flood modelling. Natural Hazards and Earth System Sciences, 25(1), 335–351. https://doi.org/10.5194/nhess-25-335-2025.

5.1. INTRODUCTION

Precise flood models are invaluable for evaluating risks, issuing early warnings, and improving preparedness against flood events. Two-dimensional hydrodynamic models determine the spatio-temporal evolution of floods by solving the Shallow Water Equations (SWE) (Teng et al., 2017). To address the intensive computational demands required to solve the SWE, we can resort to several strategies, such as using simplified physical models (e.g., Van den Bout et al., 2023) and high-performance clusters (e.g., Caviedes-Voullième et al., 2023). More recently, deep learning models emerged as an in-between option that can accelerate flood simulations, while maintaining high accuracy (Bentivoglio et al., 2022). Most deep learning models predict the flood evolution or its maximum depths while generalizing over different boundary conditions, such as rainfall, on a single domain. These models include transformers (Pianforini et al., 2024), convolutional neural networks (CNNs) (Berkhahn & Neuweiler, 2024; Guo et al., 2020; He et al., 2023; Kabir et al., 2020; Liao et al., 2023), graph neural networks (GNNs) (Burrichter et al., 2023), Fourier neural operators (Xu et al., 2024), and long short-term memory networks (LSTM) (Wei et al., 2024). Although these methods are effective on a given area, they must be trained again when applied to a different domain, thus hindering their use as surrogate models.

As such, research is now focusing on generalizing deep learning flood models to unseen case studies where the models were not trained on. For example, Löwe et al. (2021), Guo et al. (2022), and Cache et al. (2024) proposed CNN models to estimate the maximum water depth of pluvial floods in urban and catchment settings, respectively. do Lago et al. (2023) and do Lago et al. (2024) developed a conditional generative adversarial network to predict the maximum water depth for unseen rain events and urban catchments. In Chapter 4, we proposed a hydraulic-based graph neural network (SWE-GNN) that could predict the spatio-temporal evolution of dike-breach floods over unseen topographies. The main advantages of this model are its link with finite volume methods that make it suitable to simulate the physics on meshes and a hydraulic-based propagation rule that enforces continuity in water propagation. Moreover, compared to previous works, it can also predict the full flood's spatio-temporal evolution. However, the model cannot reproduce very different propagation speeds and needs a high number of layers when simulating large time steps, which can make the training process unstable. Moreover, this approach uses a fixed boundary condition and requires the first time step to be given by a numerical solver.

To overcome these limitations, we propose a multi-scale hydraulic graph neural network, based on the SWE-GNN. Multi-scale models combine the domain information coming from different resolutions and have shown benefits for simulating other partial differential equations (Fortunato et al., 2022; Lino et al., 2022). To drop the dependency from the numerical solver, we integrate time-varying boundary conditions via ghost cells, i.e., mesh cells that receive a known value of a given variable at the domain boundary (LeVeque et al., 2002). To improve the generalization to unseen meshes, we remove all coordinate-dependant inputs. This makes the model invariant to rotations (Bronstein et al., 2021), that is, rotations of the inputs do not affect the outputs. This helps because it prevents the direction of flooding from being biased towards a specific direction in the training data.

We validate the model on dike-breach flood simulations over non-squared domains,

discretized by irregular meshes, and with different topographies and time-varying boundary conditions. To test the applicability of this model to real world case studies, we consider a flood scenario for breaching of a levee system in the Netherlands.

5.2. METHODOLOGY

We developed a multi-scale graph neural network that combines the information at progressively coarser resolutions to propagate floods in space and time with different flow speeds (Figure 5.1). The proposed model takes as input static features that represent the topography and connectivity of the domain at different resolutions, and dynamic features that represent the hydraulic variables at time t. It then processes them via a U-shaped architecture that applies graph neural networks at different scales and combines them with down-sampling and up-sampling operators. The outputs are the predicted hydraulic variables at following time step $t+1$ at the finest available resolution. We added boundary conditions by assigning a known value of water depth or discharge to a set of cells at the domain boundary.

In the following, we detail the multi-scale mesh creation procedure (Section 5.2.1) and the model architecture (Section 5.2.2). Then, we show how to include boundary conditions (Section 5.2.3) and rotation-invariant inputs (Section 5.2.4). Finally, we describe the employed loss function (Section 5.2.5). We denoted variables x at a given scale or resolution $\mathcal{M}_m$ as $x^{\cdot,m}$, where $\cdot$ is a placeholder for other indices, and where the variable can be a scalar x, a vector $\mathbf{x}$, a matrix $\mathbf{X}$, or a tensor $\mathcal{X}$. When the superscript m is omitted, we refer to the variables at the finest scale.

5.2.1. MULTI-SCALE MESH CREATION

We designed a multi-scale model that combines meshes with progressively coarser resolutions. We employed an iterative process that requires only the boundary polygon of a selected area, without any prior knowledge of the underlying topography. First, we create a coarse mesh from a boundary polygon using the MeshKernel software (Deltares, 2024). This corresponds to the mesh in the bottleneck of the multi-scale module (Figure 5.1). Then, we refine the mesh by splitting each mesh edge in two and connecting the newly formed points via edges. Then, the mesh undergoes an iterative orthogonalization algorithm needed for the underlying numerical software Delft3D to run because of its staggered grid scheme (Deltares, 2022). For the same numerical constraints, after the orthogonalization, all elongated elements get removed, resulting in a mixture of triangular and quadrilateral elements. We define elongated elements as those whose line connecting barycentre and edge middle points is 0.1 times smaller than the other lines in the same element. We repeat these steps multiple times depending on the required scale of computations in the fine mesh. The obtained set of meshes constitutes our multi-scale mesh.

Multi-scale graph. The computational graph used in the proposed model considers as nodes the barycenters of the mesh cells, while edges connect neighbouring cells. We connect the graphs at two scales based on the spatial position of the mesh barycenters, as shown in Figure 5.2. If a fine mesh cell's centre is within a coarse mesh cell centre, then a directed inter-scale edge exists between the two nodes.

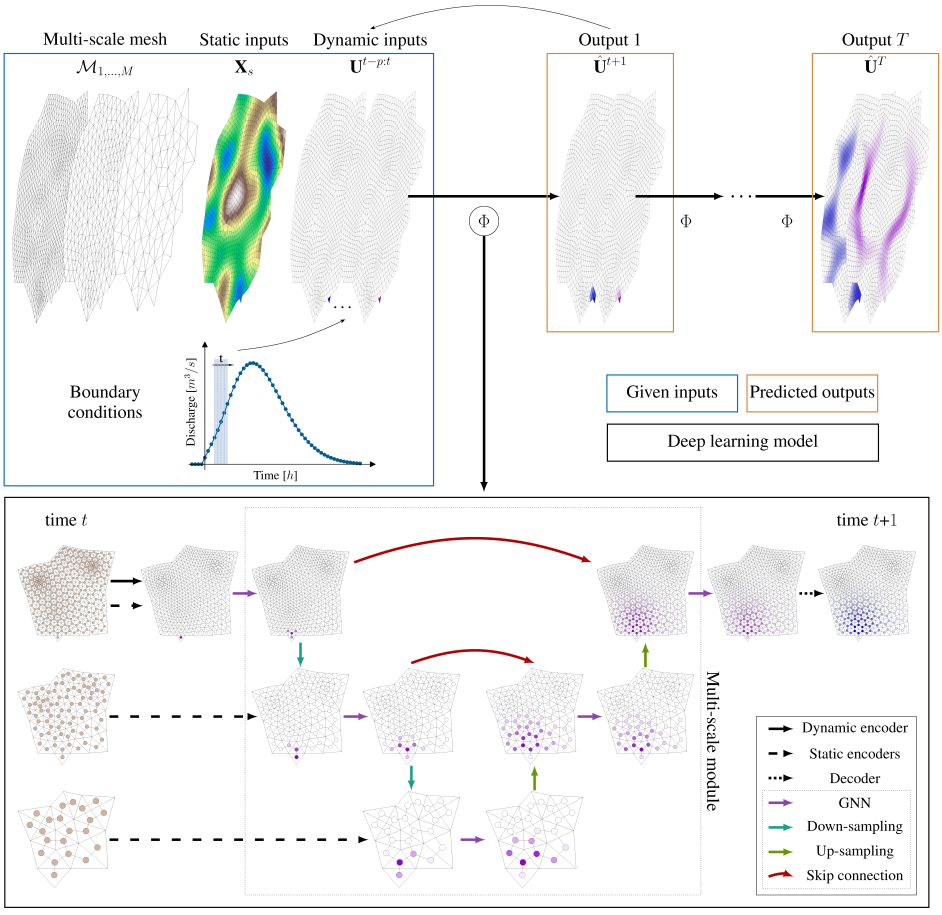


Figure 5.1: Overview of the proposed mSWE-GNN model. The model $\Phi(\cdot)$ takes as input a fine mesh and its coarser versions, along with the static and dynamic inputs defined on them (blue box, top left) and produces an estimate of the hydraulic variables in time (orange box, top right). The model is then repeated auto-regressively using its predictions as inputs (top black arrow), to determine the spatio-temporal evolution of the flood. Boundary conditions are provided at each time step by assigning a known value to a set of cells in the dynamic inputs $\mathbf{U}^{t-p:t}$. In the black box, black arrows indicate multi-layer perceptrons (MLP) present in the encoders and the decoder; blue arrows represent graph neural network layers; light green arrows down-sampling layers; dark green arrows up-sampling layers; and red arrows skip connections across different parts of the architecture.

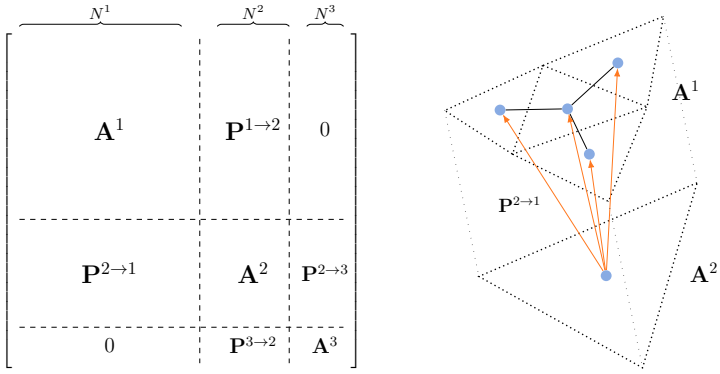


Figure 5.2: Left: adjacency matrix representation of the multi-scale graph, for a mesh with three scales. $\mathbf{A}^m \in \mathbb{R}^{N^m \times N^m}$ represents the adjacency matrix at scale m, while $\mathbf{P}^{m\to n} \in \mathbb{R}^{N^m \times N^n}$ represent the prolongation matrix from scale m to scale n. Right: example connection between a fine mesh $\mathbf{A}^1$ and a coarse mesh $\mathbf{A}^2$, where $\mathbf{P}^{2\to1}$ indicates the connectivity across the two scales.

We can describe the connectivity of the obtained multi-scale graph via a block-diagonal connectivity matrix composed by adjacency matrices $\mathbf{A}^m$ and prolongation matrices $\mathbf{P}^{m\to n}$. Adjacency matrices are squared matrices that represent the connectivity of a graph at scale m by assigning $a^m_{ij} = 1$ if edge (i, j) exists. Prolongation matrices are rectangular matrices that act like adjacency matrices, but connect one scale m to its upper or lower scale $n \in \{m+1, m-1\}$. They can also be seen as adjacency matrices for bipartite graphs whose nodes can be divided into two disjoint sets.

5.2.2. ARCHITECTURE

We develop the Multi-scale Hydraulic Graph Neural Network (mSWE-GNN) by building upon the SWE-GNN model (Chapter 4). This is an encoder-processor-decoder architecture that auto-regressively predicts the hydraulic variables at time t+1 as

$$\hat{\mathbf{U}}^{t+1} = \mathbf{U}^t + \Phi(\mathbf{X}_s, \mathbf{U}^{t-p:t}, \mathscr{E}), \tag{5.1}$$

where the output $\hat{\mathbf{U}}^{t+1}$ corresponds to the predicted hydraulic variables, $\mathbf{U}^t$ are the hydraulic variables (water depth $[m]$ and unit discharge $[m^2 s^{-1}]$) at time t, $\Phi(\cdot)$ is an encoder-processor-decoder model that determines the evolution of the hydraulic variables for a fixed time step, $\mathbf{X}_s$ are the static node features, $\mathbf{U}^{t-p:t}$ are the dynamic node features, i.e., the hydraulic variables for time steps $t-p$ to t, and $\mathscr{E}$ are the edge features that describe the geometry of the mesh. We include different mesh resolutions by defining the model $\Phi(\cdot)$ in a U-shaped architecture, inspired by Gao and Ji (2019), starting from a fine mesh to coarser ones and back to a fine mesh output. Hereafter, we describe the details of the architecture shown in Figure 5.1.

Encoder. We increase the expressivity of the inputs by employing three separate encoders for processing the static node features $\mathbf{X}_s \in \mathbb{R}^{N \times I_s}$, the dynamic node features $\mathbf{X}_d \equiv \mathbf{U}^{t-p:t} \in \mathbb{R}^{N^1 \times O(p+1)}$, and the edge features $\mathscr{E} \in \mathbb{R}^{E \times I_\varepsilon}$, with N the total number of

nodes, I_s the number of static node features, N^1 the number of nodes at the finest scale, O the number of hydraulic variables, p the number of input previous time steps, E the number of edges, and I_ε the number of input edge features. The encoded variables are defined as

$$\mathbf{H}_s = \phi_s(\mathbf{X}_s), \mathbf{H}_d = \phi_d(\mathbf{X}_d), \mathscr{E}' = \phi_\varepsilon(\mathscr{E}), \tag{5.2}$$

where $\phi_s(\cdot)$ and $\phi_d(\cdot)$ are 3-layer MLPs shared across all nodes and $\mathbf{H} \in \mathbb{R}^{N\times G}$ the encoded node features; $\phi_\varepsilon(\cdot)$ is a 3-layer MLP shared across all edges that encodes the edge features into $\mathscr{E}' \in \mathbb{R}^{E\times G}$, and G the number of features in the latent space. The encoded variables $\mathbf{H}_s$, $\mathbf{H}_d$, and $\mathscr{E}'$ represent a higher-dimensional version of the original inputs that is more expressive. We apply the shared encoders of the static features $\phi_s(\cdot)$ and $\phi_\varepsilon(\cdot)$ to all features at all scales, while the encoder of the dynamic features $\phi_d(\cdot)$ is applied only to the finest scale. The rationale behind having a shared static feature encoder for all scales is that higher-dimensional features should have a similar embedding independently of the scale, since the physical quantities are the same.

Processor. The processor propagates the encoded inputs throughout the multi-scale graph. We employ a sequence of GNN layers to propagate information at a given scale and connect two scales via down-sampling and up-sampling operators. The operations are organized in a U-shaped fashion, with a down-sampling branch from fine to coarse and a up-sampling branch from coarse to fine, as shown in Figure 5.1.

In the down-sampling branch, we start by applying L GNN layers at the encoded node and edge features $\mathbf{H}_s^1$, $\mathbf{H}_d^1$, and $\mathscr{E}'^1$ at the finest-scale mesh $\mathscr{M}_1$. Then, we apply a down-sampling operator $\downarrow: \mathscr{M}_m \rightarrow \mathscr{M}_{m+1}$ that maps the features of the finer scale $\mathscr{M}_m$ to the coarser scale $\mathscr{M}_{m+1}$. We repeat these two operations until the final coarser scale. In the up-sampling branch, we apply an up-sampling operator $\uparrow: \mathscr{M}_{m+1} \rightarrow \mathscr{M}_m$ that maps the features from the coarser scale $\mathscr{M}_{m+1}$ to the finer scale $\mathscr{M}_m$. We add skip connections to sum the output of the down-sampling GNN at scale $\mathscr{M}_m$ with the output of the up-sampling operator from scale $\mathscr{M}_{m+1}$ to $\mathscr{M}_m$. These connections facilitate information transfer and training, similarly to Ronneberger et al. (2015). Finally, we apply another set of L GNN layers to the output of the skip connections and repeat these operations until the finest scale. All GNNs, down-sampling operators, and up-sampling operators are not shared, meaning that each acts independently at one scale or across two given scales.

The GNN layers follow Bentivoglio et al. (2023) and can be expressed as

$$\mathbf{s}_{ij}^{(\ell+1)} = \psi\left(\mathbf{h}_{si}, \mathbf{h}_{sj}, \mathbf{h}_{di}^{(\ell)}, \mathbf{h}_{dj}^{(\ell)}, \boldsymbol{\varepsilon}'_{ij}\right) \odot \left(\mathbf{h}_{dj}^{(\ell)} - \mathbf{h}_{di}^{(\ell)}\right), \tag{5.3}$$

$$\mathbf{h}_{di}^{(\ell+1)} = \mathbf{h}_{di}^{(\ell)} + \sum_{j\in\mathscr{N}_i} \mathbf{s}_{ij}^{(\ell+1)} \mathbf{W}^{(\ell+1)}, \tag{5.4}$$

where $\psi(\cdot): \mathbb{R}^{5G} \rightarrow \mathbb{R}^G$ is an MLP, $\odot$ is the Hadamard (element-wise) product, $\mathbf{h}_{di}^{(\ell)}$ is the embedding of the dynamic inputs at node i and layer ℓ, $\mathbf{h}_{si}$ is the embedding of the static inputs at node i, and $\mathbf{W}^{(\ell)} \in \mathbb{R}^{G\times G}$ are learnable parameter matrices. The propagation rule in Eq. (5.3) has a hydraulic gradient-like term, $\mathbf{h}_{dj}^{(\ell)} - \mathbf{h}_{di}^{(\ell)}$, that acts as a physical constraint that allows water to propagate only from nodes which already have water. In fact, $\mathbf{h}_{di} = 0$ only if node i has both zero water depth and discharge, since the dynamic node encoder has no bias term. The predicted fluxes across nodes $\mathbf{s}_{ij}$ then combine the information from neighbouring nodes $\mathscr{N}_i$ by following the principles of numerical methods.

The down-sampling operator $\downarrow: \mathcal{M}_m \rightarrow \mathcal{M}_{m+1}$ is a mean pooling operator[1] from a fine mesh $\mathcal{M}_m$ to a coarse mesh $\mathcal{M}_{m+1}$ defined as

$$\mathbf{h}_{di}^{m+1} \leftarrow \frac{1}{|\mathcal{N}_i^{m \rightarrow m+1}|} \sum_{\mathcal{N}_i^{m \rightarrow m+1}} \mathbf{h}_{di}^{m}, \tag{5.5}$$

where $\mathcal{N}_i^{m \rightarrow m+1}$ is the set of neighbouring nodes in the finer mesh $\mathcal{M}_m$ connected vertically to the nodes in the coarser mesh $\mathcal{M}_{m+1}$ and $\mathbf{h}_{di}^{m+1} \in \mathbb{R}^G$ are the down-sampled dynamical features at node i. We used a mean pooling operation since physical features at coarser scales should resemble those at the finer scale. This approach offers a trade-off between simpler resampling methods such as nearest neighbour and more computationally intensive ones such as cubic interpolation (Maeland, 1988).

The up-sampling operator $\uparrow: \mathcal{M}_{m+1} \rightarrow \mathcal{M}_m$ is a learnable operator defined as

$$\mathbf{h}_{di}^{m} \leftarrow \sum_{\mathcal{N}_i^{m+1 \rightarrow m}} \psi^{m+1 \rightarrow m}\left(\mathbf{h}_{si}^{m}, \mathbf{h}_{si}^{m+1}, \mathbf{h}_{di}^{m}, \mathbf{h}_{di}^{m+1}\right) \odot \mathbf{h}_{di}^{m+1}, \tag{5.6}$$

where $\mathbf{h}_{di}^{m}$ are the up-sampled dynamic node features at node i in scale $\mathcal{M}_m$, $\psi^{m+1 \rightarrow m}(\cdot): \mathbb{R}^{4G} \rightarrow \mathbb{R}^G$ is an MLP, and $\mathcal{N}_i^{m+1 \rightarrow m}$ is the set of neighbouring nodes in the coarser mesh $\mathcal{M}_{m+1}$ to the nodes in the finer mesh $\mathcal{M}_m$. This expression has two important features: first, it is independent of the number of nodes in the fine scale, meaning that it works both from one-to-one node or from one-to-several nodes; second, the multiplication by $\mathbf{h}_{di}^{m+1}$ ensures that this operation only activates when a node on the coarse cell has water in it, i.e., $\mathbf{h}_{di}^{m+1} \neq 0$. Differently from the SWE-GNN layer (Eq. (5.3)), we avoid edge features, since there are none across scales, and the hydraulic gradient term since the values at one scale should be close to those at the previous scale. Thus, using a difference would result in a zero value when the features at two scales are identical.

We add skip connections to combine the outputs of the down-sampling GNNs $\mathbf{h}_{di}^{m\downarrow}$ with the outputs of the up-sampling operations $\mathbf{h}_{di}^{m\uparrow}$, before applying another GNN layer. The skip connections can be expressed as

$$\mathbf{h}_{di}^{m} \leftarrow \mathbf{h}_{di}^{m\downarrow} + \mathbf{h}_{di}^{m\uparrow}. \tag{5.7}$$

Skip connections should improve the connectivity between different parts of the architecture and combine the different propagation speeds.

The obtained mSWE-GNN architecture allows us to model the flood's propagation speed at a different scales. This is because each scale's GNN covers different portions of space based on physical nodes' distances. These separate flow speeds are combined in the architecture allowing the model to capture better their variations from one time step to another. This is particularly relevant for capturing a broader scale of dynamics with rapidly time-varying boundary conditions that change significantly the propagation speed. Moreover, this setup alleviates the requirements on the number of GNN layers at the finest scale since one layer at a coarse scale can cover the equivalent of several

[1] We also evaluated a learnable pooling operator, but the performance was lower, as highlighted in the ablation study in Sec. 5.4.3.

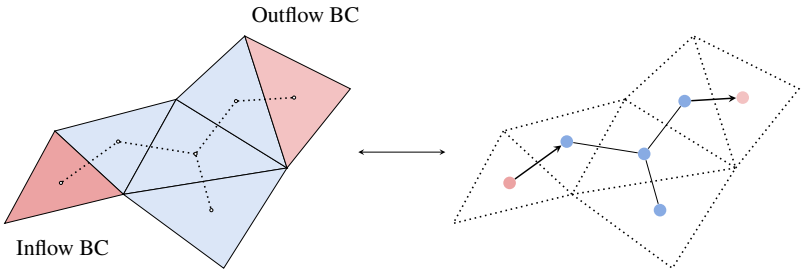


Figure 5.3: Schematic representation of an arbitrary triangular volume mesh (left) with two ghost cells for inflow and outflow boundary conditions (BC). The ghost cells (red) are added in correspondence of a boundary cell which receives a given boundary condition. In the dual graph (right), a directed edge is added from the ghost cell to the domain cell, or vice-versa, depending on whether the boundary condition is an inflow or an outflow, respectively.

layers at the finest scale. Hence, we end up with a model that is more efficient and better captures the time-varying dependencies of the flood.

Decoder. The decoder estimates the predicted hydraulic variables $\hat{\mathbf{u}}_i^{t+1} \in \mathbb{R}^O$ as a combination of the input previous time steps $\mathbf{U}_i^{t-p:t} \in \mathbb{R}^{O\times(p+1)}$ and the output of the processor at the finest scale $\mathbf{h}_{di} \in \mathbb{R}^G$. This can be expressed as

$$\hat{\mathbf{u}}_i^{t+1} = \text{ReLU}\left(\mathbf{U}_i^{t-p:t}\mathbf{w}_p + \varphi\left(\mathbf{h}_{di}\right)\right), \tag{5.8}$$

where p is the number of previous time steps, $\mathbf{w}_p \in \mathbb{R}^{p+1}$ is a learnable vector, and $\varphi(\cdot)$ is a 3-layer MLP which decodes the embeddings of the processor $\mathbf{h}_{di}$. We added a ReLU $=\max\{0, x\}$ activation function at the output of the decoder to guarantee physical values of water depths, since we know that water depth and unit discharges cannot be negative, similarly to Palmitessa et al. (2022). The learnable parameters $\mathbf{w}_p$ weigh the contribution of each input time step to the output of the model, thus acting as a 1D convolutional layer along the temporal axis.

5.2.3. BOUNDARY CONDITIONS

To include external forcings, we add boundary conditions via ghost cells, as done in numerical methods (LeVeque et al., 2002). Ghost cells are elements which belong to the computational domain but not in the physical one and act as link to external conditions. Boundary conditions related to inflows and outflows are represented via directed edges towards the real mesh and the ghost cells, respectively, as shown in Figure 5.3. The computations with directed edges in the model follow the same propagation rules as for undirected edges. Based on the forcing type, we can assign a prescribed condition for each time step of the simulation and at a specific point in the domain, to strictly enforce boundary conditions. For water levels, we impose the known value at the boundary. For discharge hydrographs, we first transform discharges $[m^3 s^{-1}]$ into unit discharges $[m^2 s^{-1}]$, by dividing the input discharge by the length of the edge across which it is passing, as in numerical methods. Wall boundaries are modelled without any ghost cell

instead of imposing reflection since this is implicitly assumed by the dual graph's structure that cannot propagate over the wall.

5.2.4. ROTATION-INVARIANT INPUTS

Most deep learning models consider coordinate-dependent features, such as the x and y components of the slopes. When applying a rotation to a domain, these values change, causing a change in the output, which is not necessarily equivalent to the applied rotation. This is a well-studied challenge in DL models (Bronstein et al., 2021) and can be solved via data augmentation, i.e., by training the model using rotated instances of the training data, or by modifying the deep learning model (e.g., Lino et al., 2022). Since the outputs of our model are scalars, we avoid using any rotation-dependant features to simplify the model and obtain a rotation-invariant model, i.e., rotations of the inputs do not affect the output. The static node features can then be expressed as $\mathbf{x}_{si} = (a_i, e_i, m_i, w_i^t)$, where a_i is the area of the i^{th} finite volume cell, e_i its elevation, m_i its Manning coefficient, and w_i^t its water level, given by the sum of the elevation and water depth at time t. To determine the values of elevation $e_{i,m}$, Manning coefficient $m_{i,m}$, and water level $w_{i,m}^t$ at the coarser scales, we perform a mean pooling operation from the finest scale to each of the coarser scales as in Eq. (5.5). As edge features, we consider $\boldsymbol{\epsilon}_{ij} = (l_{ij})$, where l_{ij} is the length of the dual edge between node i and node j. The dynamic node features are defined as $\mathbf{x}_{di} = \mathbf{u}_i^{t-p:t} = (\mathbf{u}_i^{t-p}, ..., \mathbf{u}_i^{t-1}, \mathbf{u}_i^t)$, with $\mathbf{u}_i^t = (h_i^t, |q|_i^t)$, where h_i^t is the water depth at time t and node i, and $|q|_i^t$ is the unit discharge at time t and node i.

5.2.5. LOSS FUNCTION

We employ a multi-step-ahead forecasting loss $\mathscr{L}_f$ that considers multiple model's outputs using its own predictions as inputs. This helps the model dealing with incorrect inputs and is useful to reduce accumulation of errors in time (Bentivoglio et al., 2023). It can be expressed as

$$\mathscr{L}_f = \frac{1}{HO} \sum_{\tau=1}^{H} \sum_{o=1}^{O} \gamma_o \|\hat{\mathbf{u}}_o^{t+\tau} - \mathbf{u}_o^{t+\tau}\|_2, \tag{5.9}$$

where $\mathbf{u}_o^{t+\tau} \in \mathbb{R}^N$ are the predicted hydraulic variables at time $t+\tau$, H is the prediction horizon, O the number of output hydraulic variables, and γ_o are coefficients used to weigh the influence of each hydraulic variable to the loss.

5.3. EXPERIMENTAL SETUP

5.3.1. SYNTHETIC DATASET

We created a synthetic dataset of dike-breach flood simulations, using the numerical software Delft3D (Deltares, 2022). Each simulation is discretized via an irregular mesh created from randomly generated polygons, based on ellipsoidal shapes, as described in Sec. 5.2.1. The multi-scale mesh obtained with this procedure has a total of four scales. This is an arbitrary choice selected to showcase the expressivity of the model, but different number of mesh scales would work as well, unless the coarsest scale has excessively few cells. For each mesh, we use a randomly generated digital elevation model (DEM), based on Perlin noise and combined with a small slope in a random direction, as exemplified

Table 5.1: Mean and standard deviation of elevation (above sea level), number of cells, cell area, edge length, and total flood volume for the training, validation, and testing datasets. All geometric variables refer to the properties of the finest mesh in each dataset.

Dataset	No. of simulations	Elevation [m]	Number of cells	Cell area [m^2]	Edge length [m]	Flood volume [10^6 m^3]
Train	60	-0.04 ± 0.6	$10\,018 \pm 1251$	$14\,817 \pm 5717$	182.8 ± 37.2	3.07 ± 0.66
Validation	20	-0.06 ± 0.58	$10\,029 \pm 904$	$13\,741 \pm 5125$	176.3 ± 34.9	2.9 ± 0.69
Test	20	-0.03 ± 0.53	9803 ± 1130	$13\,480 \pm 4917$	174.9 ± 33.7	3.02 ± 0.64
Test dike ring 15	10	-1.07 ± 1.17	22 881	$13\,544 \pm 5521$	174.7 ± 36.9	26.5 ± 2.54

in Figure 5.4. As boundary condition, we apply an inflow discharge hydrograph on one random border edge. The hydrograph's shape is generated based on Weibull-like probability density functions with different shape parameters (Bhunya et al., 2011). All hydrographs are right-tailed since most dike-breach hydrographs have this shape (e.g. D'Oria et al., 2022; Shustikova et al., 2020) and their peaks vary from 150 to 300 $m^3 s^{-1}$, as shown in Figure 5.6, in line with realistic breach inflows. For the Manning's roughness coefficient m, we used a spatially uniform value of 0.023 $m^{-1/3} s$, which is kept the same throughout all simulations. The dataset comprises 100 simulations, 60 used for training, 20 for validation, and 20 for testing. Each simulation has as output a temporal resolution of two hours for a total simulation time of 96 hours, or 48 steps ahead. The datasets' statistics in terms of elevation (above sea level), number of cells, cell area, edge length, and total flood volume are reported in Table 5.1. Compared to the dataset in Bentivoglio et al. (2023), this has more complexity, both in terms of mesh structure and discharge conditions.

5.3.2. CASE STUDY: DIKE RING 15

We assess the transferability of the trained model by applying it to dike ring 15 Lopiker en Krimpenerwaard in the Netherlands, which surrounds and protects the area between Rotterdam and Utrecht (Figure 5.5). This area is prone to flooding and protected entirely by a system of levees. This case study has an area of 31400 ha, with a total population of 201,500 inhabitants and an expected flood damage per event of 5.1 billion euros (Boon & Witteveen+Bos, 2011). We chose this area because, depending on the location of the

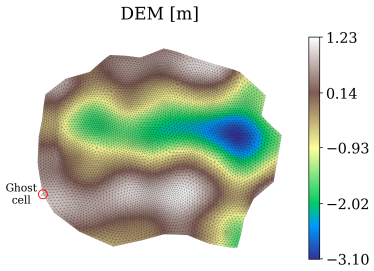


Figure 5.4: Example mesh with the corresponding digital elevation model (DEM) for one simulation in the synthetic dataset.

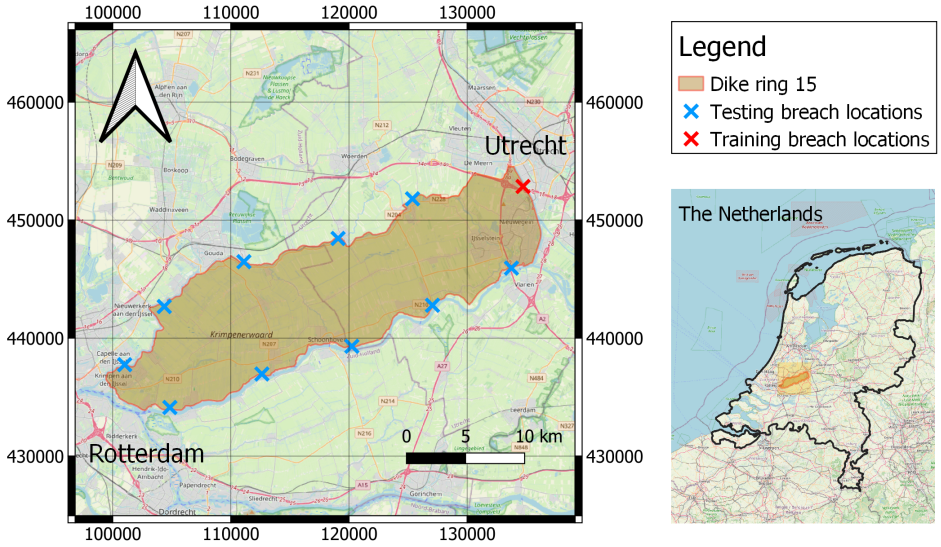


Figure 5.5: Dike ring 15, in the Netherlands (coordinate system EPSG:28992 - Amersfoort / RD New). The crosses indicate the location of the dike breaches used for training and testing. The maps are taken from ©OpenStreetMap contributors 2024. Distributed under the Open Data Commons Open Database License (ODbL) v1.0.

breach, the basin has a bathtub or sloped response, meaning that water fills up the domain evenly or has a preferential drainage direction, respectively (Rijkswaterstaat, 2016). We simplified the hydraulic components not represented in the training dataset. Specifically, we removed all water bodies and every infrastructure that is not directly included in the DEM. Moreover, we assumed constant roughness coefficients throughout the whole area.

As boundary conditions, we created a set of inflow discharges that follow a different distribution from the training ones. This has an initial rise, following an hypothetical widening of the breach, and a decreasing limb in time that ends with non-zero discharge. We also increased the peak discharge to match realistic values for the considered case study, with values between 700 and 1000 $m^3 s^{-1}$, corresponding to inflows of a fully developed breach, which could be more than $100m$ wide. This results in an increase of the total flood volumes by approximately nine times with respect to the synthetic dataset. For the breaching locations, we selected 11 approximately equidistant spots along the contour of the dike ring (see Figure 5.5). This allows us to capture a comprehensive hydraulic responses of the basin.

The selected case study is more than twice as big as the synthetic datasets and has different elevation patterns, leaving more space to develop different flood dynamics. Hence, we decided to test the model also with a fine-tuning step, employing a single simulation for training and validation. In the experiments, we analyse the effect of adding this fine-tuning step after training the model on the synthetic dataset.

5.3.3. NORMALIZATION

The static attributes (node and edge features) are determined at all scales when creating the dataset. Since the values of areas a and edge lengths l change significantly across scales, we standardize those features separately for each scale. Specifically, we collect all training instances of a given variable x at mesh scale $\mathcal{M}_m$ and determine their mean μ and variance σ. The normalized variables are then obtained as $\hat{x} = \frac{x-\mu}{\sigma}$, where $\hat{x}$ indicates the standardized variable. The remaining variables are not processed by any normalization procedure.

5.3.4. TRAINING SETUP

We trained all models with PyTorch (Version 2.0.1) (Paszke et al., 2019) and PyTorch Geometric (Version 2.4) (Fey & Lenssen, 2019) libraries, using the Adam optimization algorithm (Kingma & Ba, 2014). We performed several preliminary trials to identify a set of suitable training hyperparameters for the experiments; see Table 5.2. We used a learning rate scheduler with a fixed step decay of 0.7, every 20 epochs, starting from 0.003. The training was carried out for 200 epochs with early stopping, using 16-bit mixed-precision to decrease the computational burden. During training, we clipped the gradients with a value higher than 1, to improve training stability, and employed a curriculum learning strategy as in Bentivoglio et al. (2023), with a maximum training prediction horizon $H = 6$ steps ahead (Eq. (5.9)). We used $p = 2$ previous time steps as dynamic inputs, i.e., $\mathbf{X}_d = (\mathbf{U}^{t-2}, \mathbf{U}^{t-1}, \mathbf{U}^{t})$. The coefficients used in the loss function (Eq. (5.9)) are $\gamma_1 = 1$ for the weight of the water depths, and $\gamma_2 = 7$ for the weight of the unit discharge. We used these values to weight more water depths, which values are generally more than 10 times larger than the discharge ones, as we deem them more important.

In terms of hardware, we employed an NVIDIA A100 80GB PCIe ((DHPC), 2022) for training and deployment of the deep learning models, and an Intel(R) Core(TM) i7-8665U @1.9 GHz CPU for the execution of the numerical model. Note that the numerical model cannot run on GPUs, but we used the available OpenMP option to parallelize the

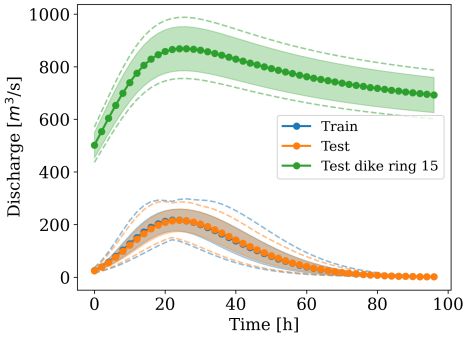


Figure 5.6: Distribution and shape of the hydrographs used as inputs for the training (blue), synthetic test (orange), and dike ring 15 test (green) simulations. The shaded region indicates one standard deviation away from the mean, at each time step. The dotted lines represent the envelopes of the minimum and maximum discharges at each time step.

computations on 8 CPU threads.

5.3.5. METRICS

We evaluated the models' performance using a multi-step-ahead mean absolute error (MAE) for each hydraulic variable $\hat{\mathbf{u}}_o^\tau$ over the full simulation, expressed as:

$$MAE_o = \frac{1}{H}\sum_{\tau=1}^{H} \|\hat{\mathbf{u}}_o^\tau - \mathbf{u}_o^\tau\|_1, \tag{5.10}$$

with H being the prediction horizon. Note that while the training loss in Eq. (5.9), is evaluated over a limited number of time steps, the validation loss function in Eq. (5.10) is evaluated on the full simulation, to mimic the testing conditions.

We also measured the spatio-temporal error distribution of the water depth using the critical success index (CSI) for threshold values of 0.05 m and 0.3 m as in Bentivoglio et al. (2023). The CSI measures the spatial accuracy of detecting a certain class (e.g., flood or no-flood) and for a given threshold it is evaluated as

$$CSI = \frac{TP}{TP + FP + FN} \tag{5.11}$$

where TP are the true positives, i.e., the number of cells where both numerical and deep learning models predict water depth above a given threshold, FP are the false positives, i.e., the number of cells where the deep learning model wrongly predicts water depth above a given threshold, and FN are the false negatives, i.e., the number of cells where the deep learning model does not predict water depth above a given threshold. We measured the computational speed-up as the ratio between the computational time required by the numerical model and the inference time of the deep learning model. We did not consider the computational time to create the meshes, since they are needed for both methods. Unless otherwise mentioned, the deep learning model is run in parallel over all testing simulations, differently from the numerical model (Section 5.4.5). This choice is reasonable since we can use this model for probabilistic forecasts, where multiple simulations may be run in parallel.

5.4. RESULTS

5.4.1. COMPARISON WITH SWE-GNN

To highlight the improvements given by multi-scale modelling, we compared the mSWE-GNN model with an enhanced SWE-GNN model that includes ghost cells, rotation-invariant inputs, and the 1D CNN in the decoder, but lacks the multi-scale component. We did not compare with the standard SWE-GNN since it would not be able to run without a numerical input. We also did not compare against other baselines as the SWE-GNN performs better than them (Bentivoglio et al., 2023), so we assumed the same holds for the enhanced version. Both models underwent a hyperparameter search procedure based on the number of GNN layers and the number of hidden features, as reported in Table 5.2. Since the amount of hyperparameters is high, some values were taken based on similar studies in literature (e.g., Bentivoglio et al., 2023).

Table 5.2: Summary of the hyperparameters and related values' ranges employed for the different deep learning models. The **bold** values indicate the best configuration in terms of validation loss.

DL model	Hyperparameter name	Values' range (**best**)
All models	Initial learning rate	0.003
	Input previous time steps (p)	2
	Maximum training steps ahead (H)	6
	Optimizer	Adam
	Batch size	12
SWE-GNN	Embedding dimension (G)	16,32,50,**64**
	Number of GNN layers (L)	10, 12, 14, **16**, 18
mSWE-GNN	Embedding dimension (G)	16,32,50,**64**
	Number of GNN layers (L)	2,3,**4**,5

This resulted in a set of models with different performances in terms of accuracy and speed as reported in Figure 5.7. The results show that the multi-scale structure helps the model to better capture flow variations across time, resulting in a better Pareto front for validation losses and CSI. The mSWE-GNN has on average more parameters than the SWE-GNN because it has several GNNs (two per each scale, except one for the bottleneck) which makes it by default bigger. Despite this, the mSWE-GNN is comparatively faster, with speed-ups of up to 1200 times, since at the finest scale it has fewer layers than the SWE-GNN. This reduces substantially the computations since the finest scale is the one with most nodes and edges. Moreover, the training process resulted also more stable in the mSWE-GNN, probably due to the lower number of GNN layers.

For the remaining analyses, we selected the mSWE-GNN model with the best performance, which consists of 4 GNN layers for each scale, a hidden feature dimension of 64, and around 811'000 learnable parameters. Despite the limited amount of training samples and the amount of variability in simulated conditions, the model captures the

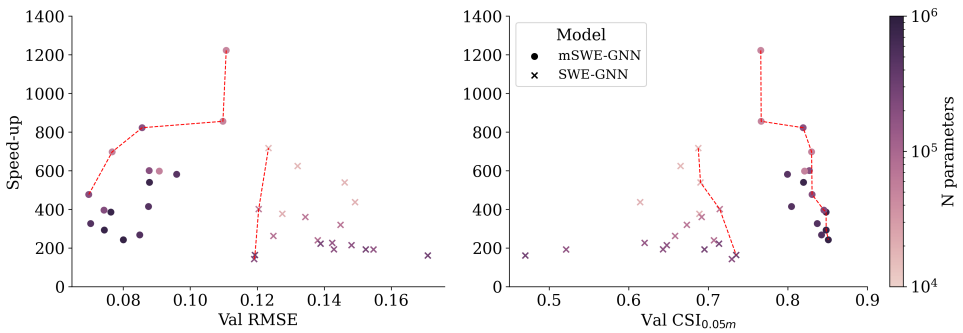


Figure 5.7: Pareto front of the mSWE-GNN and SWE-GNN models for speed-ups, validation RMSE (left), and validation CSI with 0.05 m water depth threshold (right). The models' size varies with number of hidden features and number of GNN layers.

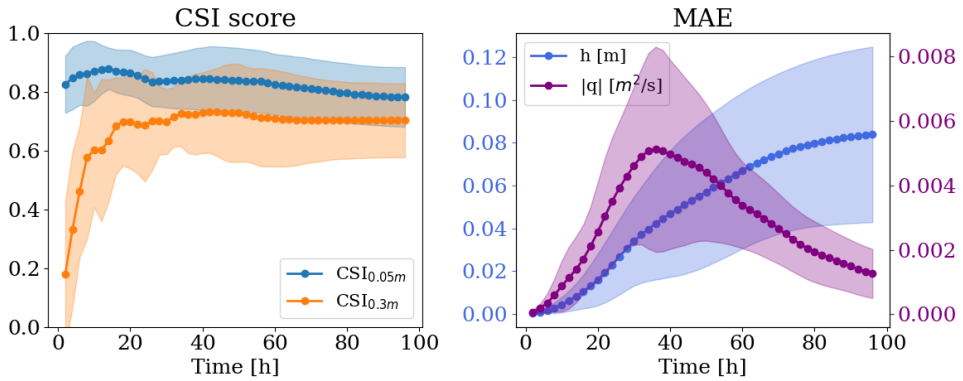


Figure 5.8: Temporal evolution of CSI scores (left) and MAE of water depth h and unit discharge q (right) for the test dataset. The confidence bands refer to 1 standard deviation from the mean.

flow patterns. Figure 5.8 reports the evolution of the critical success index (CSI) for the water depth thresholds of 0.05 m and 0.3 m and the mean absolute errors (MAE) for water depth and unit discharge for the test dataset. The $CSI_{0.05m}$ stays constantly high for all simulations. On the other hand, the $CSI_{0.3m}$ starts low: this is due to an initial scarcity of water depths higher than $0.3m$, which skews the performance to lower values. The MAE of unit discharge seems correlated with the input breach discharge values, meaning that the biggest errors occur at the hydrograph peak and the smallest close to the tail. Indeed, the highest errors are generally located near the breach location, where the most rapid processes occur. Thus, when the inflow discharge decreases, so does the error. The MAE of water depth instead rises with time as also reported in Bentivoglio et al. (2023). The main reason for the increase in water depth MAE over time is that as the flood progresses, it covers a greater spatial extent, increasing the number of cells where prediction errors can occur. In this case, however, the errors plateau at the end of the inflow hydrograph, indicating that water flow is stopping.

We analysed the evolution in space and time of the unit discharges for one test simulation in the synthetic dataset, to highlight that the model is now able to correctly model the filling and emptying dynamics. Figure 5.9 shows that discharges are modelled very well by the model, both in the ascending and the descending phases of the input hydrograph. This is in line with the hypothesis of Bentivoglio et al. (2023) according to which the model is able to capture these draining and decreases in discharges when presented sufficient samples of it in the training dataset.

5.4.2. TRANSFER LEARNING TO REALISTIC CASE STUDY

After training the model with the synthetic dataset, we tested it on dike ring 15, for different breach locations with varying discharges. The zero-shot testing of the model without any fine tuning resulted in modest performances in Table 5.3. We attribute this mismatch to the difference in total flood volume but also the domain size and the different elevation patterns when compared to the training ones (see Table 5.1). Moreover, this implies different hydraulic dynamics, such as the presence of sloped basin which

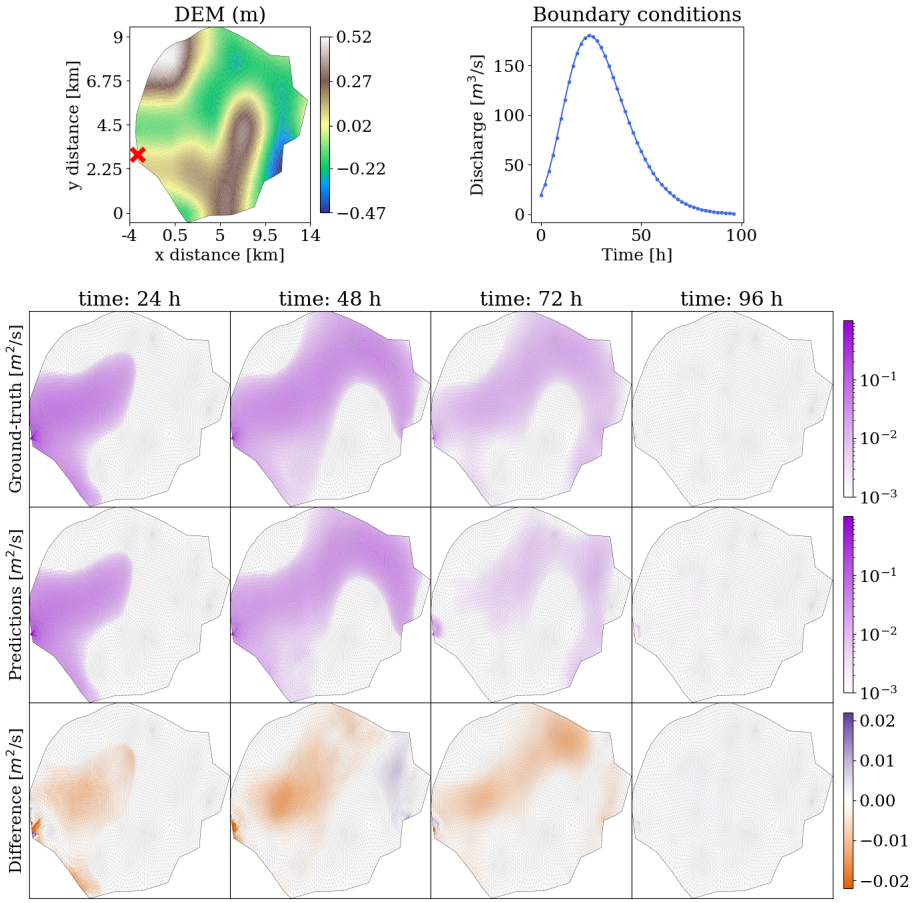


Figure 5.9: mSWE-GNN's predictions for unit discharges a test simulation from the synthetic dataset. The evolution over time for ground-truth output of the numerical simulation (top row) with the predictions (middle row) are represented using a logarithmic scale to better appreciate the values' distribution. The difference (bottom row) is evaluated as the predicted value minus the ground-truth one and is kept with a standard scale to highlight the use of the logarithmic scale; positive values correspond to model over-predictions while negative values correspond to under-predictions.

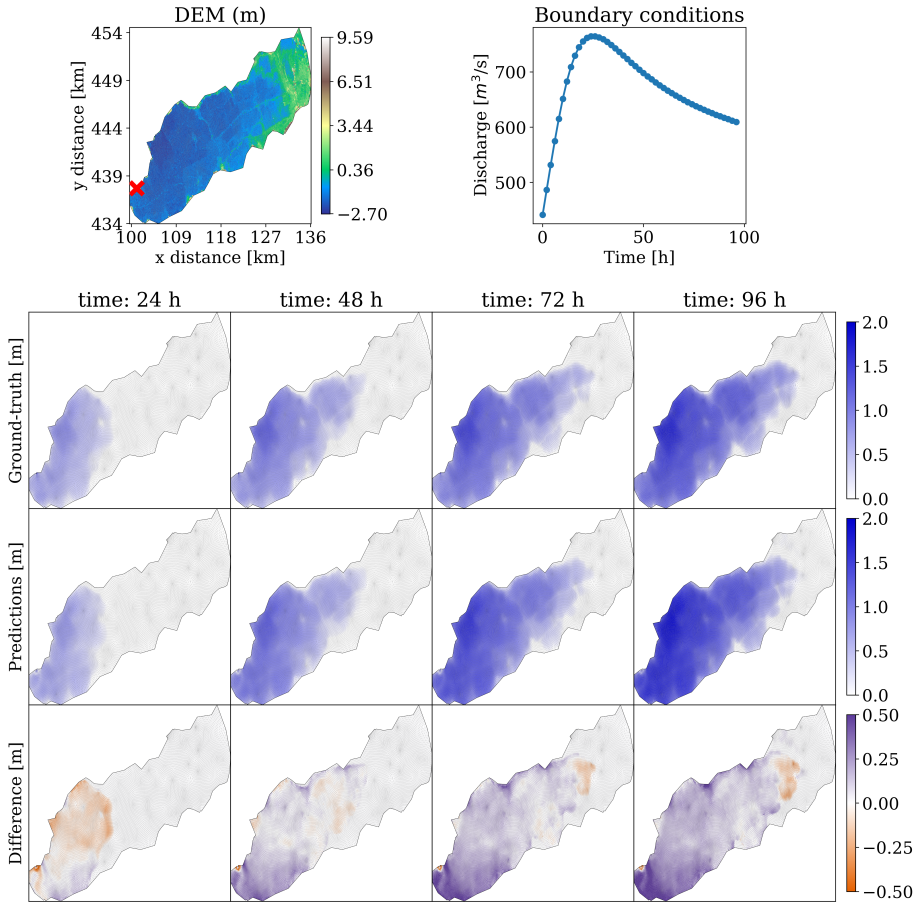


Figure 5.10: mSWE-GNN's predictions for water depth on a testing simulation for dike ring 15. The topography is presented in the top left plot, the discharge hydrograph in the top right, and below the evolution over time for ground-truth output of the numerical simulation (top row) with the predictions (middle row). The difference (bottom row) is evaluated as the predicted value minus the ground-truth one; thus, positive values correspond to model over-predictions while negative values correspond to under-predictions. All plots represent values only on the finest mesh.

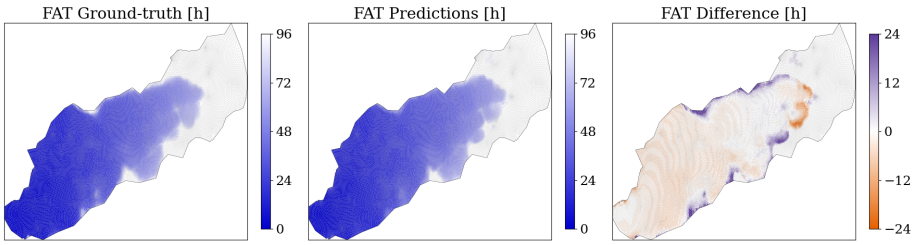


Figure 5.11: Flood arrival times (FAT) for a water depth threshold of 0.05 m for a test case from dike ring 15, given for the numerical simulation (left), the predictions (center), and the difference (right). Darker colors in the first two maps indicate a faster arrival of the water, while white cells indicate absence of water. In the difference map, positive values indicate that the model estimates later arrival times than the numerical simulation, while negative values indicate that the model predicts earlier arrival times. All plots represent values only on the finest mesh.

accumulates water in a downstream area (in the bottom left of the domain) without further propagation, which are not sufficiently represented in the training domain.

We then performed a fine-tuning step consisting of training again the previously-trained model with one extra simulation from the new case study. We trained and validated on the same simulation since we wanted to minimize the amount of data needed to fine tune the model. While in principle this might lead to overfitting, it was not the case here. This is probably due to the inductive biases of the model which constrain the model to learning only local dynamics. Additionally, the training process considers only a limited number of predicted steps ahead, while the full simulation has many more. Consequently, the model is forced to learn different dynamics in time rather than overfitting on a single temporal pattern, even if we are training on a single simulation. Table 5.3 shows that adding just one simulation improves the testing performance on the rest of the dike ring by 158% and 62% in MAE for water depth and discharge, respectively, and 38% and 78% for $\text{CSI}_{0.05m}$ and $\text{CSI}_{0.3m}$.

Figure 5.10 shows the model performance for the prediction in time of water depth for one test case. Water depths are overall well predicted in the domain, including water accumulation in the western part of the area. While the absolute values of the difference may be relatively high in these areas, they do not matter as much for practical purposes since those locations are either way flooded with a high water depth, thus the associated damages will be equivalent.

Figure 5.11 shows that the spatio-temporal evolution of the predicted flood is in line

Table 5.3: Effect of fine-tuning the mSWE-GNN model on dike ring 15. The provided uncertainty estimates account for the variability across different simulations. All metrics refer only to the finest mesh.

Fine-tuning	**MAE** ↓		**CSI$_\tau$ [%]** ↑	
	h [$10^{-2}m$]	$\lvert q\rvert$ [$10^{-2}m^2s^{-1}$]	τ=0.05 m	τ=0.3 m
No	31.09 ± 5.42	3.37 ± 1.24	63.36 ± 19.54	46.06 ± 18.62
Yes	12.07 ± 4.19	2.08 ± 0.82	87.68 ± 10.3	81.82 ± 16.07

with the corresponding numerical simulations, as indicated by the low errors of flood arrival times (FAT) for the critical threshold of 0.05 m of water depth. FAT indicate the arrival time of water with a given depth threshold, for each cell in the domain. Most errors are located at the wave front during the end of the simulation, as previously mentioned, or in false positive areas that are not flooded in the numerical model.

Figure 5.12 indicates that the model performance is consistently high for all testing breach locations of the dike ring 15 dataset, as suggested by the high $CSI_{0.05m}$ values, which are always above 0.8. One reason why the model performs so well is that the final flood map tends to converge to the downslope accumulation area in the bottom left area of the domain. This also proves that the model can correctly model the response dynamics of the system, independently of where the breaching starts.

The good performance of the mSWE-GNN model is accompanied by a substantial speed-up of the underlying model. When testing, the model has a speed-up of more than 700 times with respect to the original simulations, as highlighted in Table 5.4. This indicates a good scaling with the size of the domain, with higher speed-ups for bigger domains. One other possible explanation is related to the simulated discharges. Numerical simulations of slow flows are generally more stable and faster to compute than those with high Froude numbers, which are more present in dike ring 15. Contrarily to numerical models, the mSWE-GNN has no such stability constraints, which make it unbound by the same limitations and thus faster.

We reported the execution run times of the numerical and trained mSWE-GNN models for both testing datasets. Since the deep learning model is run in parallel, the prediction times per simulation are averaged out through all simulations. We measure the run time variability by running the model 10 times and reporting the corresponding mean and standard deviation. For both dataset, the model achieves a great speed-up, of more than two orders of magnitude, which could further increase when selecting a smaller model from the Pareto front in Figure 5.7.

In terms of training times, the SWE-GNN model took between 5 to 30 hours while the mSWE-GNN 2 to 15 hours, depending on the model complexity. The fine-tuning process, with the selected mSWE-GNN model in Section 5.4.1, took around 20 minutes. The fine-tuning time can be reduced to 5 minutes by decreasing the number of epochs, while still obtaining comparable performance. If we evaluate the speed-up on dike ring 15 including also the time to run the fine-tuning numerical simulation and the time to train

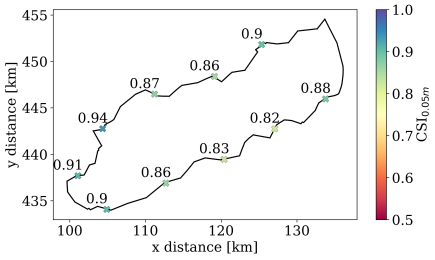


Figure 5.12: Performance in terms of CSI for a water depth of 0.05 m for all testing breach locations in dike ring 15, for the fine-tuned mSWE-GNN.

it, we still achieve a speed-up of 4 to 8 times, depending on the number of fine-tuning epochs.

5.4.3. ABLATION STUDY

Finally, we performed an ablation study to determine the role of the different components in the mSWE-GNN (Table 5.5), such as the multi-scale module, the convolutional decoder, and the rotation invariant inputs. We also reported the performance of the best SWE-GNN model from Section 5.4.1. The results reported in Table 5.5 show that all of the added or removed components contribute to the performance on the test dataset. The speed-up was consistent throughout all mSWE-GNN configurations and we report it in Table 5.4.

Multi-scale module. We analysed the effects of using a learnable down-sampling operator in place of a mean pooling in Eq. (5.5) and removing skip connections in Eq. (5.7). For the learnable down-sampling operator, we used a 3-layer MLP shared across each intra-scale edge that takes as inputs the dynamic node feature at nodes i in $\mathcal{M}_m$ and node j in $\mathcal{M}_{m+1}$, similarly to Eq. (5.6).

Using a learned down-sampling operator results in a lower performance. We argue this is caused by the unnecessary complexity of the operation and due to the common mean aggregation term, which is needed to make the model work with a flexible number of nodes, that cancels the expressivity of the MLP.

Removing skip connections does not influence as much the performance. This indicates that most of the computations are performed after the architecture bottleneck, while the down-going branch is responsible for smaller details that are not captured in the up-going branch. This means that the number of layers in the down-going branch can probably be reduced, while keeping good performance with less model complexity.

Decoder. We compared the convolutional decoder (Eq. (5.8)) with a residual connection which simply sums the output of the previous time step to the output of the decoder's MLP before applying a *ReLU* activation, i.e., $\hat{\mathbf{U}}^{t+1} = \sigma\left(\mathbf{U}^t + \varphi\left(\mathbf{H}_d^{(L)}\right)\right)$. Using the 1D-CNN in the decoder results in better testing and validation metrics, meaning that different time steps contribute unevenly to the final model output. This allows the model to better capture variations in time, especially due to rapid variations in boundary conditions.

Rotation invariant inputs. We added the x and y components of the slope and orientation of mesh edges as static inputs to show that including rotation-dependent inputs worsens generalization (Table 5.5). The reason for this is that all simulations are quite different one from the other in terms of breach location and orientation of the meshes. Consequently, a model with rotation-dependent inputs would require much more training data to generalize well to all spatial configurations.

Table 5.4: Run times of the numerical model and the selected mSWE-GNN model for the two testing datasets and their respective speed-ups.

Dataset	Numerical model [s]	mSWE-GNN [s]	Speed-up [-]
Test	80.8 ± 15.4	0.33 ± 0.10	250 ± 25
Test dike ring 15	611 ±211	0.81 ± 0.23	750 ± 50

Table 5.5: Ablation study on the removal or addition of individual architectural and training components, for the synthetic testing dataset. These are: using a learnable pooling for the down-sampling operator, removing skip connections in Eq. (5.7), removing the 1D-CNN in Eq. (5.8), and using rotation-dependent inputs. The best results are reported in **bold**.

	DL model	MAE ↓		CSI_τ [%] ↑	
		h $[10^{-2}m]$	$\|q\|$ $[10^{-2}m^2s^{-1}]$	τ=0.05 m	τ=0.3 m
	SWE-GNN	9.52 ± 5.03	0.42 ± 0.16	68.70 ± 18.90	51.70± 22.10
	mSWE-GNN	**4.84 ± 2.30**	**0.27 ± 0.13**	**84.02 ± 9.18**	**69.56 ± 17.25**
mSWE-GNN	with learnable pooling	5.72 ± 3.09	0.32 ± 0.13	81.23 ± 12.23	63.67 ± 19.66
mSWE-GNN	w/o skip connections	5.22 ± 2.22	0.32 ± 0.15	82.44 ± 10.82	66.81 ± 17.31
mSWE-GNN	w/o 1D-CNN	5.57 ± 2.50	0.32 ± 0.14	80.75 ± 10.83	65.03 ± 19.21
mSWE-GNN	w/o rotation invariant inputs	6.07 ± 2.27	0.34 ± 0.15	79.93 ± 10.18	62.89 ± 18.28

5.4.4. MASS CONSERVATION

We proposed a regularization term $\mathscr{L}_c$ that enforces a global mass conservation per each time step. This reads as

$$\mathscr{L}_c = \left| \sum_{i=1}^{N} a_i \Delta \hat{h}_i - Q\Delta t \right| \tag{5.12}$$

where N is the number of nodes in the output mesh, $\Delta \hat{h}_i$ is the variation in predicted water depth at node i, $a_i \Delta \hat{h}_i$ is the variation in predicted volume at node i, Q is the inflow discharge, and Δt is the time interval between t and $t+1$, in which the discharge is assumed to be constant. This enforces the total amount of volume entering the domain $Q\Delta t$ to be redistributed in the domain so that the volume is conserved.

We carried out supplementary experiments to explore the benefit of adding this term to the forecasting loss in Eq. (5.9). The combined loss $\mathscr{L}$ can be expressed as

$$\mathscr{L} = \mathscr{L}_f + \alpha * \mathscr{L}_c, \tag{5.13}$$

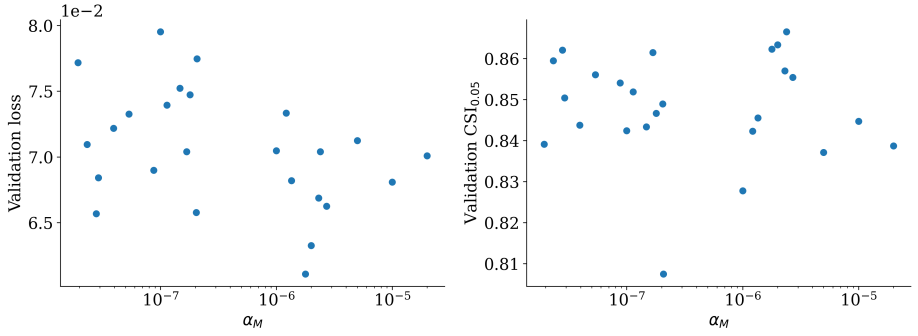


Figure 5.13: Performance in terms of validation loss and $CSI_{0.05m}$, for varying values of the mass conservation weight α_M.

where α is weighs the contribution of the mass conservation term. For the mass conservation weight α in Eq. (5.12), we uniformly sampled values in an interval from 10^{-8} to $5 * 10^{-5}$, using a logarithmic distribution. The reason why these values are small is due to the flood volumes being more than 10^6 times higher than water depths, as reported in Table 5.1.

Figure 5.13 shows that the validation loss and CSI are slightly negatively correlated with α, meaning that losses tend to improve and classification worsen. The reason why losses slightly improve might be because the added loss term depends only on the predicted water depth, so it enforces that value to be more precise. However, the conservation loss acts globally for each time step, instead of locally. So, the model cannot correctly improve the spreading of the flood but only the absolute values of total water depth. From these plots, we cannot extract any meaningful conclusion since there is no statistical significance, as highlighted by p-values of 0.42 and 0.48, respectively. Moreover, the performance in the testing dataset follows an opposite trend, further indicating that inclusion of this term is not consistently better.

This in part contradicts the idea of physics-informed neural networks (PINN), according to which adding a physical loss term improves performance (Raissi et al., 2019). One motivation is that the loss we employ does not rely on auto-differentiation in the same way that PINNs do. We also evaluate it globally, rather than at individual points as in PINNs. Implementing a PINN loss would require adjustments to the model's inputs and outputs to allow auto-differentiation to estimate the derivatives of the predicted target variables. Moreover, PINNs are typically designed to solve a given physical problem for a single set of boundary and initial conditions, thus limiting the model's capacity to generalize across varying conditions, which is a prerogative of our work. Although we did not adopt this approach here, it could be explored in future studies. Notably, our loss term is independent of ground-truth data, making it a possible self-supervised loss that could be explored in future works.

5.4.5. PARALLEL SIMULATIONS

We refactored the code to compute all testing simulations in parallel, instead of in series, by using batches. To analyse the speed-ups provided by parallel execution, we selected all models in the Pareto front of the mSWE-GNN in Figure 5.7, which have different number of parameters and number of GNN layers per scale. We then ran the models using an increasing number of simulations in parallel, indicated by the batch size.

Figure 5.14 indicates that the speed-up almost doubles with the batch size, independently of the size of the model. It also highlights that the main computational effort comes from an increase number of GNN layers, rather than just the total number of model parameters, as also reported in Bentivoglio et al. (2023). When running 20 simulations, i.e., the full testing dataset, in parallel instead of in series, we provide a further speed-up of 4.5 times on average across different models' sizes. Further speed-ups may be achievable by optimizing the code; for example just-in-time(JIT)-compiling of the PyTorch code into optimized kernels can further accelerate the execution of the model by two or three times (Paszke et al., 2019). In a similar fashion, Intelligent Processing Units (IPUs), which are a novel processing unit that has faster inference on graphs, can further speed-up the model by two to four times (Knowles, 2021).

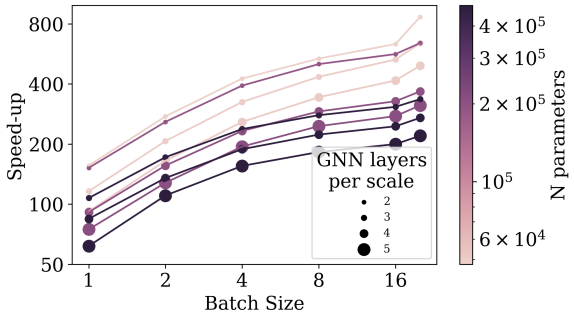


Figure 5.14: Speed-ups of mSWE-GNN for the synthetic test dataset, considering varying batch sizes, i.e., how many simulations are run in parallel. The results are reported for all Pareto front models from Figure 5.7. Both axes are in log(2) scale.

5.5. DISCUSSION

We proposed a multi-scale graph neural network model (mSWE-GNN) that can generalize flood simulations to unseen irregular meshes, topographies, and time-varying boundary conditions, with speed-ups up to 700 times compared to the underlying numerical model. The mSWE-GNN generalizes well to realistic case studies with as little as one fine-tuning simulation. We expect the model to further improve performance and reduce risk of overfitting by increasing the number of fine-tuning simulations. This result is in line with a similar finding for pluvial flooding where one fine-tuning simulation was enough to help generalization to diverse case studies (Cache et al., 2024). Since the model can generalize well with as little as 60 training simulations, we believe that training the model on a substantially larger amount of data might even remove the need of fine-tuning, although this could still be needed for more complex domains.

One key to the model's success are the different scales, which enable learning varying speeds of flood propagation and capturing the hydraulic processes, contrary to the SWE-GNN, which learns a more limited range of speeds. The multi-scale nature of the model allows optimizing computations for areas where fine details are relevant only in small portions of the domain. In the same way, scales can be used to better include the presence of 1D structures in the domain such as channels and elevated elements. In Chapter 6, we include them in the coarser meshes by using slimmer cells that overlap with the channel, as done in numerical models (Bomers, Mathias, et al., 2019). On the other hand, structures that markedly influence the flow propagation, like levees, can simply be omitted in the coarser meshes, by leaving holes in correspondence of them. This artificially blocks the possibly faster flow propagation of coarser scales. Once over-topping of said levee occurs at the fine scale, then the faster propagation can begin anew in the coarser scales. Alternatively, we can model a characteristic feature, such as the elevated element's height as an edge feature, to explicitly include it in the learning process, as we do in Chapter 6. We improved the model generalization to unseen meshes by considering rotation-invariant inputs. This was possible because we considered scalar outputs, since we deemed the intensity of the flood more important than also knowing its direction for practical uses (Kreibich et al., 2009).

While the current model framework can work for dike-breach floods, we did not evaluate it for other types of floods. For river and coastal floods, the model should work without any changes since the inputs are of the same type as dike breach floods, e.g., upstream discharge hydrograph or sea water levels. On the other hand, pluvial floods require precipitation as a further input. Assuming rainfall as a spatially distributed variable, it could be added as a dynamic forcing; this could work in a similar way as for static features, but changing at each time step, independently of the predicted output. For urban floods, the drainage system should also be included. This could be done, as in numerical methods, by coupling the overland flow, predicted by the mSWE-GNN, with a 1D model for the sewers, possibly with another learned GNN as in Garzón et al. (2024).

Regarding the process of mesh creation, we constructed the coarse-scale meshes based on the boundary polygon of the considered areas. However, this requires the user to create a mesh with a top-down approach and limits the use of an existing fine-scale mesh. This could be solved by using a different multi-scale mesh creation approach. For example, Lino et al. (2022) used a sampling strategy based on a regular partitioning of the domain, which allows the coarse meshes to have similar edge lengths, independently from the fine mesh. Alternatively, we could use the same mesh creation procedure to only generate the coarse scale meshes and use existing detailed meshes in the fine scale. The latter may be problematic when fine structures are present that markedly alter the flow of the flood, making the automatic mesh generation procedure challenging.

The boundary condition insertion technically works also for given water levels at the boundary, but we did not analyse it. Moreover, we did not analyse the performance for multiple concurring boundary conditions, despite the model can already accommodate them. We employed a constant and spatially uniform roughness coefficient, meaning that we did not assess how the model generalizes to different values and spatial distributions. This might lead to different dynamics that, following the same reasoning as for the different speeds of propagation, the model should still be able to capture.

To simplify the hyperparameter selection process, we also selected an equal number of GNN layers for all scales. Instead, we could further optimize the Pareto front by changing the number of layers at each scale independently. Additionally, we did not compare with other recent developments in deep learning models, such as Fourier Neural Operators (Li et al., 2020) or Neural fields (Yin et al., 2023), since they either do not generalize across different irregular meshes or their application to flooding would not be trivial. We remark that most of the speed-ups come from the use of a GPU, as all processes are parallelizable. This is a well-known benefit of deep learning models and the mSWE-GNN enjoys it.

For practical applications, there are still several components that must be included to match numerical models for real case studies. Future studies should investigate the inclusion of time-varying breach growth models or components such as existing water bodies and linear elements, such as roads and secondary dikes. In Chapter 6, we use the proposed model to create a probabilistic framework to assess many different flood scenarios and uncertainties in boundary conditions and breaching conditions (Vorogushyn et al., 2010).

5.6. CONCLUSION

We proposed a multi-scale hydraulic graph neural network, called mSWE-GNN, that models flood propagation in space and time across multiple resolutions. The model takes as input static attributes, such as topography, and dynamic attributes, such as water depth and unit discharge at time t, and predicts their evolution at the following time step $t+1$. This is done via a U-shaped architecture that applies graph neural networks at different scales and combines them with down-sampling and up-sampling operators. This captures a broader range of dynamics by jointly modelling the flood propagation speeds at different scales. We included time-varying boundary conditions via ghost cells. We also improved the generalization to unseen meshes by using rotation-independent inputs.

The main conclusions of this chapter can be summarized as follows:

- We develop a multi-scale approach which improves the simulations both in speed and accuracy, with speed-ups of up to 1000 times and mean absolute errors of 0.05 m and 0.003 $m^2 s^{-1}$ for water depth and unit discharges, respectively;
- We include time-varying boundary conditions via the use of ghost cells to remove the dependency from the numerical models and we improve generalization to unseen meshes by making the model's inputs invariant to rotations;
- We show that the model generalizes well to a realistic case study with bigger area and wider range of boundary conditions than the training ones, with only one fine-tuning simulation.

The results of this chapter show that the proposed mSWE-GNN model is an effective surrogate for spatio-temporal flood simulations with high generalization capabilities. In the following chapter (Chapter 6), we employ the mSWE-GNN model to describe the flood uncertainties of dike breach floods in a real case study. To complement the complexities of the case study, we also include hydraulic structures by modifying the inputs of the model.

6

PROBABILISTIC FLOOD MODELLING

"Creativity requires the courage to let go of certainties."

Erich Fromm

This chapter presents an application of the multi-scale hydraulic graph neural networks (mSWE-GNN) model developed in Chapter 5. We use the mSWE-GNN to generate probabilistic dike-breach flood hazard maps for a real case study. To better capture the complexities of realistic case studies, we extend the model's input to include hydraulic structures such as canals, underpasses, and elevated elements. Additionally, we introduce a mass-conservation-based validation assessment to evaluate the physical consistency of the model's predictions. We apply this framework to quantify the uncertainty in dike-breach flooding within dike ring 41 in the Netherlands, a low-lying area surrounded by river dikes and located along the Maas and Rhine rivers. The sources of uncertainty include location of potential breaches and breach outflow hydrographs which are influenced by river water levels and dike parameters. Thanks to the computational speed-ups of around 10,000 times, compared with the reference numerical models, the approach allows for the efficient generation of probabilistic flood hazard maps. This method provides an effective tool for preliminary flood hazard assessments and supports the prioritization of scenarios for more detailed physical simulations. This chapter address the gaps of operational use, uncertainty quantification, and inclusion of hydraulic structures.

This chapter is an adapted version of:

Bentivoglio, R., Jonkman, S. N., Isufi, E., and Taormina, R., 2025. Probabilistic Flood Hazard Mapping for Dike-Breach Floods via Graph Neural Networks, EGUsphere 2025 (2025): 1–29. https://doi.org/10.5194/egusphere-2025-5582

6.1. INTRODUCTION

Probabilistic flood hazard maps quantify the likelihood of different flooding scenarios, based on their uncertainty. Unlike deterministic approaches, which compute a single estimate of water depth, extent, and intensity for a specific return period (Dottori et al., 2021), probabilistic maps explicitly account for uncertainties in flood drivers and system responses. These uncertainties may stem from factors such as river discharge hydrographs (Savage et al., 2016), maximum water levels (de Moel et al., 2014), roughness coefficients (Hall et al., 2011; Savage et al., 2016), or flood duration (de Moel et al., 2014). Further uncertainties can appear when quantifying the statistical fit of a model with limited data and treating metrics as deterministic (Huang & Merwade, 2024). As a result, probabilistic approaches avoid underestimating risks that happen with deterministic methods (Hall & Solomatine, 2008; Savage et al., 2016).

Building probabilistic hazard maps remains challenging as the number of uncertain variables can be large, particularly for dike breaching, where additional geotechnical properties must be considered. Uncertainties include breach location (D'Oria & Maranzoni, 2019; Westerhof et al., 2023), breach width (de Moel et al., 2014; Mazzoleni et al., 2014), breach development time (Apel et al., 2006; Ferrari et al., 2020), failure time (D'Oria & Maranzoni, 2019), and failure mechanism (D'Oria & Maranzoni, 2019; Mazzoleni et al., 2014). Estimating output uncertainty may require up to hundreds of thousands of simulations, making standard numerical flood models computationally prohibitive, unless using large high-performance clusters (Gibbons et al., 2020). Simplified hydraulic models have been explored to reduce costs, but they often provide low accuracy or limited outputs, such as only final water levels (Apel et al., 2006; de Moel et al., 2014).

Recent years have seen a rapid expansion of deep learning (DL) surrogate models as fast and accurate alternatives to traditional numerical models (Bentivoglio et al., 2022). Most studies focus on predicting maximum water depth maps (e.g., Gao et al., 2024; Guo et al., 2020; Liao et al., 2023) or predicting the full spatio-temporal evolution of floods (e.g., Burrichter et al., 2023; Cao et al., 2024; Pianforini et al., 2025; Song et al., 2025) while generalizing on different boundary conditions, such as rainfall or river discharges. In terms of types of floods, most works investigate pluvial floods, mainly driven by rainfall (Shao et al., 2024; Wang et al., 2024), while few others cover coastal (Xu & Gao, 2024), river (Pianforini et al., 2025), and dike-breach floods (Wei et al., 2024). Despite achieving good accuracy and speed, these models focus on a single domain, meaning that they require re-training in unseen case studies or even placement of localised boundary conditions (contrarily to, for example, spatially distributed rainfall), ultimately, limiting their practical use. To address this limitation, several studies have addressed the transferability of DL models to unseen case studies and boundary conditions, with convolutional-based models (Cache et al., 2024; do Lago et al., 2023; Guo et al., 2022) and graph-based ones (Bentivoglio et al., 2023; Bentivoglio et al., 2025; Kazadi et al., 2024). In particular, graph-based models showed high transferability to boundary conditions and locations and a stronger link with physics.

However, DL models are typically validated on a limited range of simulations, leaving their reliability in truly unseen scenarios uncertain unless additional reference simulations are available. They also cannot accommodate hydraulic structures (e.g., canals, elevated roads, underpasses), whose complex geometries strongly affect flow. To achieve

probabilistic flood hazard maps in realistic settings, DL models require two properties. First, they need to account for hydraulic structures rather than only via digital elevation models. This is challenging because these small-scale structures, though minor in size, can significantly alter flood behaviour. Second, they require a validation procedure that works without ground-truth data as the latter is often unavailable.

Tackling the above challenges, this work advances probabilistic flood modelling by using a deep learning surrogate for real-world probabilistic dike-breach flood hazard mapping. First, we integrate hydraulic structures such as canals and elevated elements into deep learning models as inputs, overcoming a key limitation of existing surrogates. Specifically, we consider the mSWE-GNN from Bentivoglio et al. (2025), as it supports time-varying boundary conditions, ensures physical consistency, and is the only model achieving demonstrable generalizability to unseen boundary conditions as well as unseen boundary locations, a key requirement for probabilistic flood mapping. The graph nature of the model allows integrating hydraulic structures by explicitly representing them as additional edge or node features and by adapting the computational mesh to them, enabling the network to learn how such structures influence flow propagation.

Second, we introduce an average relative mass error (ARME) metric based on mass conservation to assess the validity of surrogate predictions, particularly under unseen scenarios where reference solutions are unavailable. Third, we validate our approach on a large-scale realistic low-lying area in the Netherlands protected by flood defences and with a wide coverage of hydraulic structures, with approximately 180,000 computational cells. We considered uncertainty in breach outflow hydrograph and breach location, influenced by both river water levels and dike strength.

6.2. METHODOLOGY

We designed a graph-based surrogate model for dike breach flood modelling that includes hydraulic structures as inputs (Figure 6.1a). We trained and tested our model on a dataset of numerical simulations, auto-regressively predicting water depths and unit discharges over time. We analysed the uncertainty in flood hazard mapping considering an ensemble of breach locations and outflow discharges. To improve prediction reliability when no ground-truth simulations exist, we introduced a verification procedure based on mass conservation (Figure 6.1e). Using the plausible simulations we then estimated probabilistic hazard maps.

This section first describes the surrogate flood model and its adaptation to include hydraulic structures and water bodies (Section 6.2.1). Next, we introduce the mass-conservation-based metric (Section 6.2.2). Finally, we detail the procedure to create probabilistic flood maps that account for input uncertainties (Section 6.2.3).

6.2.1. MULTI-SCALE HYDRAULIC GRAPH NEURAL NETWORKS

MODEL

The multi-scale hydraulic graph neural network (mSWE-GNN) is a graph-based deep learning architecture that models the two-dimensional spatio-temporal evolution of floods (Bentivoglio et al., 2025). It treats the cells of a computational mesh as nodes in a graph and connects neighbouring cells with edges. It learns flood spreading by

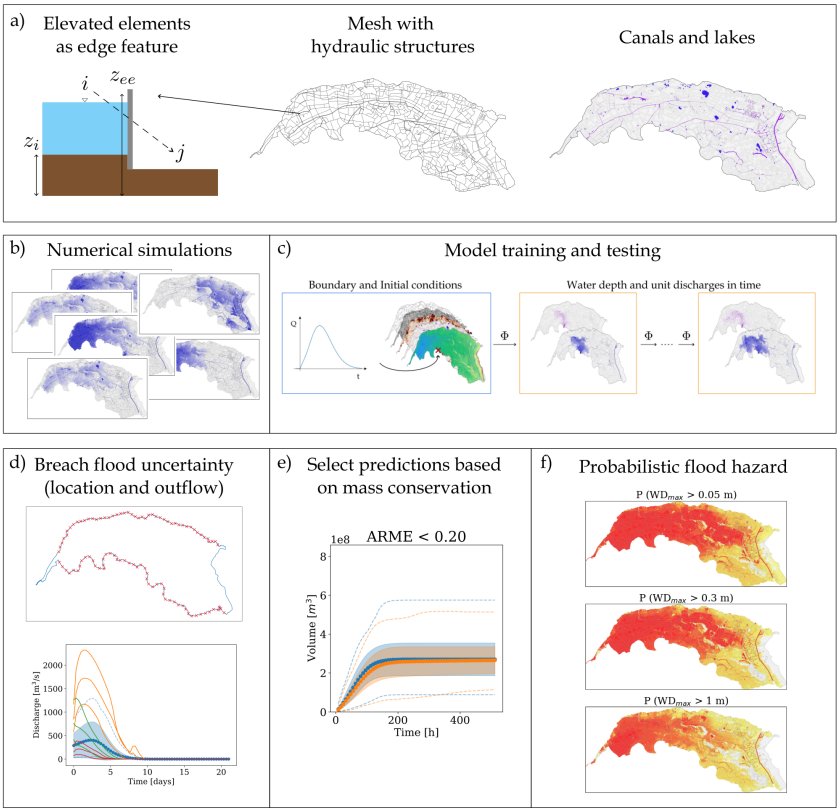


Figure 6.1: Proposed methodology for probabilistic dike-breach flood hazard mapping. **a**) Hydraulic structures such as elevated elements and canals are inputs. The difference between elevated elements (z_{ee}) and the terrain (z_i) is treated as an additional edge feature. **b**) The model is trained and tested on a dataset of numerical simulations. **c**) The network Φ receives node features (topography, roughness, water bodies, and initial hydraulic states) and edge features (mesh connectivity and elevation differences at hydraulic structures). It predicts water depths and unit discharges at the next time step, repeating this process auto-regressively using the predicted outputs as a new initial condition to simulate the full spatio-temporal flood dynamics. Boundary conditions are enforced through ghost cells. The architecture operates across multiple mesh resolutions, with node and edge features defined at each scale. A zoom-out detail is shown in Figure 6.2. **d**) Flood uncertainty, represented via 10,000 combinations of breach locations and outflow discharges. **e**) Plausible simulations are selected based on the average relative mass error computed against the input inflows. **f**) Selected simulations provide conditional probabilities of flooding, assuming the same probability of occurrence for each scenario.

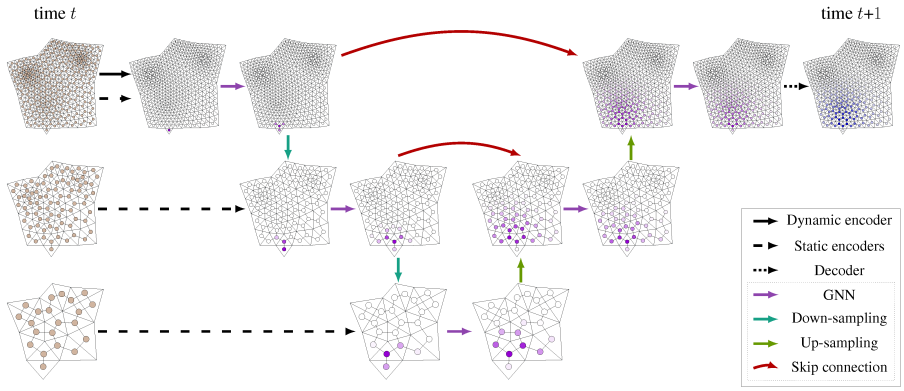


Figure 6.2: The mSWE-GNN model (Bentivoglio et al., 2025). The inputs are node and edge features defined on multiple mesh resolutions at time t and predicts water depths and discharges at the next time step $t+1$. The multi-scale architecture consists of three encoders, which create high-dimensional embeddings of the inputs, a U-Net-like processor, which consists of a sequence of graph neural network layers followed by down-sampling and up-sampling operators, and a decoder, which converts node embeddings into hydraulic variables.

combining local flow propagation with a series of graph neural network (GNN) layers at different spatial resolutions (Figure 6.2). Each GNN computation is based on the finite volume approximation of the shallow water equations, enforcing a physical bias in the propagation rule (Bentivoglio et al., 2023). The model takes static features representing topography, terrain roughness, and domain connectivity at different resolutions, and dynamic features representing hydraulic variables at time t. It processes these inputs with a U-shaped architecture that applies GNNs at multiple scales and combines them through skip connections. The model predicts the hydraulic variables, water depth h $[m]$ and the absolute value of unit discharge $|q|$ $[m^2/\mathrm{s}]$, at the next time step $t+1$, at the finest available resolution. Ghost cells at the domain boundary enforce known boundary conditions.

The mSWE-GNN uses an explicit numerical scheme to auto-regressively predict hydraulic variables at time $t+1$ as

$$\hat{\mathbf{U}}^{t+1} = \sigma\left(\mathbf{U}^{t} + \Phi(\mathbf{X}_s, \mathbf{U}^{t-p:t}, \mathscr{E})\right), \tag{6.1}$$

where $\hat{\mathbf{U}}^{t+1}$ is the predicted hydraulic variables, $\mathbf{U}^{t}$ are the hydraulic variables (water depth $[m]$ and unit discharge $[m^2\ \mathrm{s}^{-1}]$) at time t, $\Phi(\cdot)$ is the model for a fixed time step, $\mathbf{X}_s$ are static node features, $\mathbf{U}^{t-p:t}$ are dynamic node features for time steps $t-p$ to t, with p indicating the number of previous times steps given as input, σ is a rectified linear unit (ReLU) used for guaranteeing positive hydraulic variables, and $\mathscr{E}$ are edge features. The node features are divided into static and dynamic to isolate the hydraulic variables so that cells without any water will have dynamic node features equal to zero: this concept is used in the SWE-GNN layers to preserve physical consistency in water propagation (Bentivoglio et al., 2023). Node and edge features also neglect coordinates and orientation-dependant values to ensure translational and rotational invariance.

INCLUDING HYDRAULIC STRUCTURES

We model hydraulic structures, such as canals, elevated roads, and underpasses by modifying the computational meshes, edge features, and node features. These modifications provide more physic inductive bias to the model but do not affect the propagation rule in Eq. (6.1).

Canals: We create longitudinal polygonal elements in the coarse mesh resolutions to represent canal segments (see Figure 6.1a). This helps the model recognize their distinct propagation speeds, similarly to how 1D elements work in numerical models. The longitudinal elements are not needed in the finest scale if the mesh cells are already small enough to correctly separate canals from the terrain. We add binary node features to indicate the presence of a canal, using one-hot encoded vectors that are one for canal cells and zero otherwise.

Water bodies: We create polygonal elements in the coarse mesh resolutions to represent water bodies, such as ponds and lakes (see Figure 6.7a,b,c). This helps the model recognize that they are not source points. Similarly to canals, the polygonal elements are not needed in the finest mesh if it is detailed enough to separate water bodies from dry terrain. As for canals, we add binary node features to indicate the presence of a lake, using one-hot encoded vectors that are one for lake cells and zero otherwise.

Elevated roads: We model elevated roads and similar one-dimensional elements with a marked elevation difference from the surrounding topography via edge features. We identify graph edges that intersect these structures using geospatial intersection and assign each directed edge (from node i to node j) a value equal to the height difference from the source node i to the elevated element z_{ee}, i.e., $z_{ee} - z_i$, as shown in Figure 6.3. This feature increases with elevation difference, guiding the model to recognize that water can only cross the structure once the water level surpasses its height, and thus it is more difficult for flow to occur from that side. We also modify the coarsest mesh to create polygonal elements that follow the shape of elevated roads and encompass areas partially or fully enclosed by elevated features, similarly to Lhomme et al. (2008). This helps the model recognize their presence and understand that they may block water flow locally.

Underpasses: Underpasses are represented by lowering the elevation at the underpass location, effectively creating a hole that allows water to flow beneath elevated roads. This ensures the model can learn that water is not fully blocked by elevated structures.

MODEL INPUTS

The inputs consist of static and dynamic node features and edge features extracted from the mesh and its variables.

The **static node feature** for the i^{th} finite volume are $\mathbf{x}_{si} = (a_i, e_i, r_i, w_i^t, c_i, l_i)$, where a_i is its area, e_i its elevation, r_i its roughness coefficient, w_i^t its water level, given by the sum of the elevation and water depth at time t, and c_i and l_i are binary masks that indicate whether a node represents a canal or a lake, respectively. Following Bentivoglio et al. (2023), we treat the water level w_i^t as a static rather than a dynamic feature. Although it is updated over time, it retains non-zero values even in dry cells due to the elevation term, meaning that including it among the dynamic inputs would violate the hydraulic-preservation criterion of the SWE-GNN layer.

The **dynamic node features** $\mathbf{x}_{di} = \mathbf{u}_i^{t-p:t} = (\mathbf{u}_i^{t-p}, ..., \mathbf{u}_i^{t-1}, \mathbf{u}_i^t)$, represent the initial and

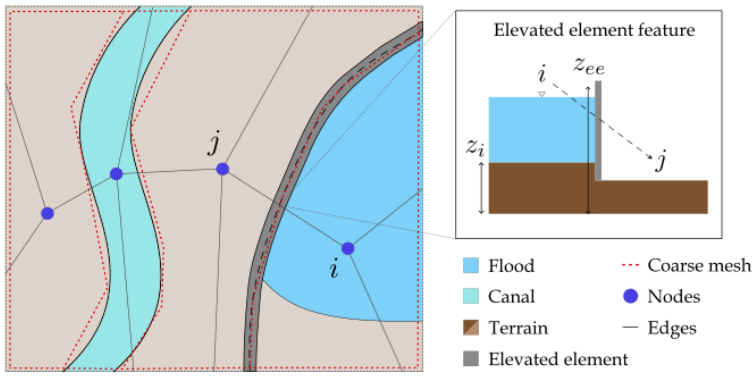


Figure 6.3: Left: Example coarse mesh creation process around canals and elevated roads. The mesh elements are adapted to fit the shape of these hydraulic objects. Right: schematization of how elevated elements are added as edge features. For each edge (i, j) that intersects an elevated element, we determine the feature as the difference in elevation from that of the element (z_{ee}) to that of the source node i (z_i).

previous states of the hydraulic variables, where $\mathbf{u}_i^t = (h_i^t, |q|_i^t)$ and h refers to water depths $[m]$ and $|q|$ to the unit discharges $[m^2 s^{-1}]$, respectively.

The **edge features** are $\boldsymbol{\varepsilon}_{ij} = (l_{ij}, z_{ie})$, where l_{ij} is the distance between the centres of nodes i and j and z_{ie} is the elevation difference between the elevated element z_{ee} and that of the source node z_i, that is $z_{ee} - z_i$, which is set to zero in case of no structure.

As in Bentivoglio et al. (2025), all inputs are independent of the coordinate values and mesh orientation as it makes the model more generalizable and less prone to overfitting on a specific case study. Similarly, edge features depend only on the connectivity between two nodes and not on their orientation.

TRAINING

We train the mSWE-GNN end-to-end using flood simulations as training data. We apply a multi-step-ahead loss and a curriculum learning strategy to minimize error accumulation over time (Bentivoglio et al., 2023):

$$\mathscr{L} = \frac{1}{HO} \sum_{\tau=1}^{H} \sum_{o=1}^{O} \gamma_o ||\hat{\mathbf{u}}_o^{t+\tau} - \mathbf{u}_o^{t+\tau}||^2, \tag{6.2}$$

where $\hat{\mathbf{u}}_o^{t+\tau}$ are the predicted hydraulic variables at time $t+\tau$, H is the prediction horizon, O is the number of output hydraulic variables, and γ_o are coefficients that weigh each variable's influence on the loss.

6.2.2. MASS VALIDATION

We introduce a validation method based on mass conservation to evaluate the model's outputs when no ground-truth exists. We compare the time evolution of water volumes predicted by the model with the ones derived from the inflow discharge hydrograph used as a boundary condition. This comparison provides a physically interpretable criterion to

identify outputs that deviate from expected hydraulic behaviour, increasing trust in the model's predictions.

We calculate ground truth flood volumes V_t $[m^3]$ at time t as the cumulative mean discharge entering the entire domain up to time t:

$$V_t = \sum_{\tau=0}^{t} \frac{(Q_{\tau+1} + Q_\tau)}{2} \Delta\tau \tag{6.3}$$

where Q_τ $[m^3/s]$ is the inflow discharge at time τ and $\Delta\tau$ is the time interval between τ and $\tau+1$ $[s]$. We compute predicted flood volumes $\hat{V}_t$ $[m^3]$ at time t as the sum of the water volumes in each cell minus any initial water volume at time $t=0$:

$$\hat{V}_t = \sum_{i=1}^{N} a_i \hat{h}_i^t - V_0, \tag{6.4}$$

where N is the number of nodes in the output mesh, $\hat{h}_i^t$ $[m]$ is the predicted water depth at node i and time t, $a_i \hat{h}_i^t$ $[m^3]$ is the predicted volume at node i and time t, and V_0 $[m^3]$ is the initial water volume before the flood begins.

We measure the discrepancy between true and predicted volumes over time using an average relative mass error (ARME), defined as:

$$\text{ARME} = \frac{1}{T} \sum_{t=1}^{T} \left| \frac{\hat{V}_t - V_t}{V_t} \right|, \tag{6.5}$$

which measures the average relative error in flood volume over time, similarly to how mass conservation is determined in numerical models (e.g., Brufau et al., 2002). We assume that volumes are equivalent to mass, since the density of water is constant. The ARME provides a single interpretable value that reflects physical plausibility rather than just statistical fit. It penalizes deviations that would create or lose water artificially, while tolerating minor fluctuations that do not affect overall flood dynamics. ARME values near zero indicate better agreement between prediction and ground truth; larger values show greater discrepancies. A value of 1.0 means an average 100% relative deviation in predicted volumes.

We focus on total flood volumes as a validation metric because we can measure it from the model inputs, as in numerical models. Flood volume is also linked to damage and casualties (den Heijer & Kok, 2023). We avoid other curve comparison metrics, such as the coefficient of determination (R^2), because they are too sensitive to localized discrepancies and may reject plausible, though imperfect, outputs. One may also consider this term as a regularization in the loss function during training to penalize towards mass conservation, but in Bentivoglio et al. (2025) it has been shown that mass conservation does not necessarily improve the performance. Thus, we use the ARME solely for validation where no ground truth exists.

6.2.3. PROBABILISTIC DIKE BREACH FLOOD MODELLING

We create probabilistic flood maps by running multiple simulations with different inputs and quantifying the uncertainty based on the likelihood of each output. Uncertainties

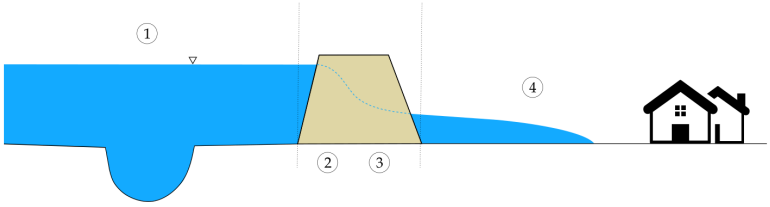


Figure 6.4: Schematics of a dike breach flood model, which includes 1) river flow simulation, 2) identification of failure mechanisms, 3) dike breach modelling, and 4) flood wave propagation.

arise at multiple stages, each linked to specific processes or variables. Although the entire process can be represented as a single integrated model, it is often more practical to use a sequence of distinct steps (see Figure 6.4). A typical workflow for analysing such flood events includes:

1. **River flow simulation:** Develop and run a one-dimensional hydrodynamic model of the river system to estimate water levels along the dike over time.

2. **Identification of dike failure mechanisms:** Use historical records and geotechnical data to determine the most likely dike failure modes and locations.

3. **Dike breach modelling:** Simulate dike breach evolution based on water levels from the one-dimensional model, identified failure mechanisms, and dike structural characteristics. Derive breach development and outflow hydrograph from the water level difference across the dike (e.g., Verheij & Hydraulics, 2003).

4. **Flood wave propagation:** Simulate the spatio-temporal spreading of floodwater across the inundation area using a two-dimensional overland flow model.

The resulting outputs, such as water depths, velocities, and extent, can then be used to generate flood hazard maps by linking them to probabilities of occurrence derived from statistical analysis of input conditions.

The mSWE-GNN emulates only the final two-dimensional flood wave propagation. We represent all earlier-stage uncertainties by varying two key breach parameters: (i) the outflow discharge hydrograph through the breach and (ii) the breach location. The outflow is determined by the breach geometry, hydraulic boundary conditions, breach initiation and development processes. The location depends on the geotechnical properties of the dike, which determine the dominant failure mechanism and its associated fragility curve. Instead of modelling each factor individually, we consider the hydrographs and breach locations known and we use them as boundary conditions for the mSWE-GNN.

We generate probabilistic flood hazard maps by running ensembles of scenarios with varying boundary conditions. The outputs are aggregated into spatial probability fields, where each computational cell encodes the likelihood distribution of a flood variable (e.g., maximum water depth or flood arrival time). We summarize this uncertainty through a set of quantiles of these distributions. The reported likelihoods are conditioned on the

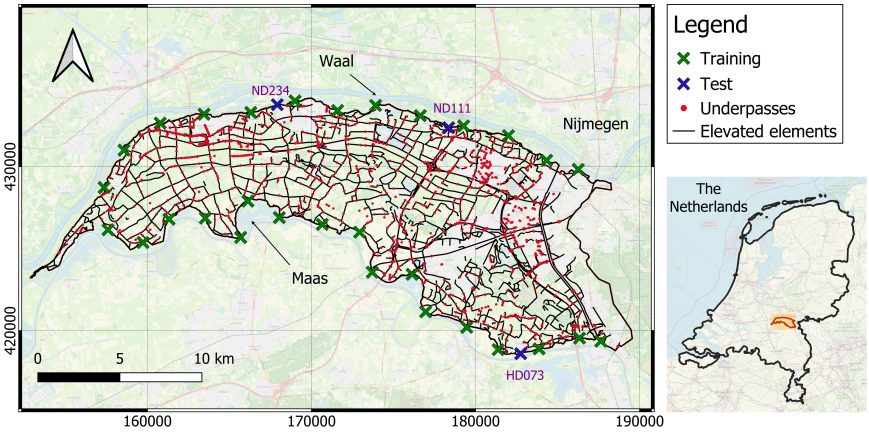


Figure 6.5: Dike ring 41, in the Netherlands (coordinate system EPSG:28992 - Amersfoort / RD New). The crosses indicate the location of the dike breaches used for training and testing. The labels ND111, ND234, and HD073 indicate the three locations in the testing dataset. The maps are taken from ©OpenStreetMap contributors 2025. Distributed under the Open Data Commons Open Database License (ODbL) v1.0.

specified boundary conditions and are therefore independent of the absolute probability of dike failure or the flood return period (that is, they are conditional on a given set of breaches and failure locations). Consequently, each scenario is assumed to occur with equal probability. To obtain unconditional occurrence probabilities, which account for the fact that different scenarios may have different likelihoods, the conditional values can be weighted by estimates of defence fragility and hydrological frequency. The integration of these defence failure probabilities and return period estimates is treated as a separate step outside the present framework and is not included in this chapter.

6.3. EXPERIMENTAL SETUP

6.3.1. CASE STUDY

We selected dike ring 41, “Land van Maas en Waal", in the Netherlands, representative for low-lying protected areas along rivers, as a case study for probabilistic flood mapping. This area is surrounded by the Meuse and Waal rivers and contains a high density of hydraulic structures. It covers 27,900 ha, supports a population of 251,900, and previous studies estimate an expected flood damage per event of 5.9 billion euros (Rijkswaterstaat, 2016). The same study identifies piping and overtopping as the main failure mechanisms, with fragility curves that vary by dike segment. The dynamics of the flood change significantly with breach location and outflow hydrograph, due to the basin slope and the presence of many elevated elements and canals (Figure 6.5).

For training and validation data, we used, respectively, 30 and 10 numerical simulations performed with Delft3D (Deltares, 2022), each with a different breach location, selected to be approximately equidistant along the dike ring boundary, and a different dike outflow hydrograph over time as boundary conditions. We determined these hydrographs as in Bentivoglio et al. (2025), using synthetic hydrographs with peak discharges from

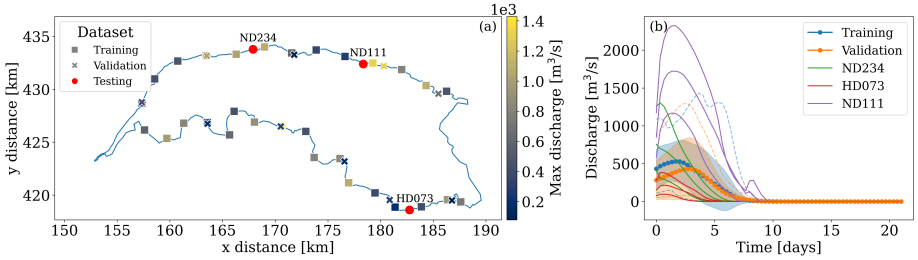


Figure 6.6: (a) Spatial distribution of the training, validation, and testing breach locations and associated maximum breach discharge. The north and south side of the domain are surrounded by the Meuse and Waal rivers. (b) Training, validation, and testing discharge hydrographs used as boundary conditions for the simulations. The shaded regions indicate one standard deviation away from the mean, at each time step, while the dotted lines represent the envelopes of the minimum and maximum discharges at each time step. The labels ND111, ND234, and HD073 indicate the three testing locations, with increasing discharges for increasing return period.

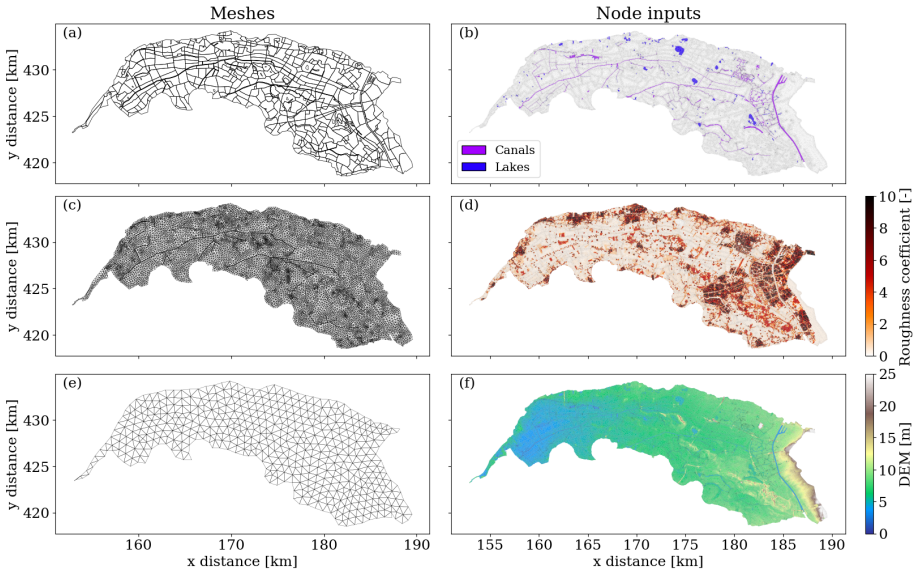


Figure 6.7: Part of the meshes and static node features used as inputs of dike ring 41. (a) and (c) represent the two coarse scales meshes (scale 3 and 2) employed in the experiments, in which the mesh polygons follow the presence of relevant hydraulic structures; (e) corresponds to an example of coarse mesh (scale 4) used in the ablation study obtained without including relevant geometrical boundaries; (b) shows the location of the canals and ponds/lakes; (d) represents the distribution of the White-Colebrook roughness coefficient, with higher values indicating urban areas; (f) shows the digital elevation model (DEM) of the area.

100 to 1,500 m^3/s (Figure 6.6). All simulations use the same computational mesh, which consists of approximately 180,000 mesh faces and 300,000 mesh edges (see Table 6.1). For testing, we considered nine scenarios: three breach locations and three return periods of estimated river water levels (100, 1000, and 10000 years), obtained from the Dutch national flood hazard maps (VNK) using Delft3D (Rijkswaterstaat, 2016). We selected the testing locations based on the availability of pre-computed spatio-temporal simulations and to cover a wide range of flood events, including some much larger than those in training (see Figure 6.6). Each test simulation assumes instantaneous breach formation, with breach development based on the Verheij-Van der Knaap equations (Verheij & Hydraulics, 2003). Test boundary conditions are river water levels over time, determined for the Meuse and Waal rivers using the GRADE method (Hegnauer et al., 2014). Each scenario, for all datasets, uses a simulation time of 21 days and an output temporal resolution of eight hours, for a total of 64 time steps, matching the VNK simulation characteristics. All numerical simulations are run on a AMD Ryzen 7 5700X 8-Core Processor (3.40 GHz) CPU, using four OpenMP threads.

After training and testing, we analysed the spatial sensitivity of the model to different boundary conditions by further testing it on 100 different breach locations, each with 100 different discharge hydrographs, generated as in the training data. Each combination has the same probability of occurrence. We used this sensitivity analysis to i) quantify the variability of the ARME under different scenarios and ii) determine a suitable threshold for the ARME to select plausible simulations.

We then obtained probabilistic flood hazard maps in flood arrival times and maximum water depths for a given test breach location and return period. For this analysis, we used the simplified method in Besseling et al. (2025) to compute the discharge hydrographs from the river water levels. We estimated the uncertainty in breach outflow by repeatedly sampling the probability of failure of a dike segment, from the dike's fragility curves for piping and overtopping, assuming different water levels over time as hydraulic loading. We performed this analysis only on test simulations, since this simplified method requires a ground-truth numerical simulation for calibration and cannot generalize to unseen locations.

6.3.2. TRAINING SETUP

We trained all models with PyTorch (Version 2.5) (Paszke et al., 2019) and PyTorch Geometric (Version 2.6) (Fey & Lenssen, 2019), using the Adam optimizer (Kingma & Ba, 2014). Based on previous studies, we used a learning rate scheduler with a fixed step decay of 0.7 every 15 epochs, starting from 0.003. Training ran for 100 epochs with early stopping and used 16-bit mixed-precision to reduce computational load. During training, we clipped gradients above two to improve stability and used a curriculum learning strategy as in Bentivoglio et al. (2023), with a maximum training prediction horizon $H = 5$ steps ahead (Eq. (6.2)). Although training used fixed time windows, we evaluated all validation and testing simulations over the full simulation time, without requiring any numerical solution as input. We used $p = 2$ previous time steps as dynamic inputs, i.e., $\mathbf{X}_d = (\mathbf{U}^{t-2}, \mathbf{U}^{t-1}, \mathbf{U}^{t})$. The loss function coefficients (Eq. (6.2)) were $\gamma_1 = 1$ for water depths and $\gamma_2 = 7$ for unit discharge, as in Bentivoglio et al. (2025), to give a more balanced weight to water depths, which are generally more than ten times larger than discharge values. All experiments ran

on AMD INSTINCT MI200 GPUs with 64GB RAM, provided by the LUMI cluster (EuroHPC, 2025).

We trained the mSWE-GNN with a hidden feature size of 64, three mesh scales (details found in Table 6.1), and a heterogeneous distribution of GNN layers across these scales. Specifically, we used two layers for the finest scale after U-Net pooling, five and three layers for the middle scale (before and after the U-Net bottleneck), and six layers for the coarsest scale. Preliminary experiments showed that adding more layers, especially before the U-Net bottleneck, increased model complexity and training time without improving performance. This is because the coarsest scale already captures large-scale flow dynamics, while the finer scales refine local details. Because of the high dimensionality of the search space and long training times (estimated in about ten hours on eight AMD INSTINCT MI200 GPUs, 64GB), we did not conduct an extensive hyperparameter analysis.

6.3.3. METRICS

We measured model performance using four metrics:

- **Regression**: we used a multi-step-ahead mean absolute error (MAE) for each hydraulic variable $\hat{\mathbf{u}}_o^\tau$ over the full simulation, expressed as MAE = $\frac{1}{H}\sum_{\tau=1}^{H}\|\hat{\mathbf{u}}^\tau - \mathbf{u}^\tau\|_1$, with H being the full simulation duration.

- **Classification**: we used the critical success index (CSI), which measures spatial accuracy in detecting a class (e.g., flood or no-flood) for a given threshold. CSI is evaluated as CSI = $\frac{TP}{TP+FP+FN}$, where TP are true positives (cells where both numerical and deep learning models predict water depth above threshold), FP are false positives (cells where the deep learning model wrongly predicts water depth above threshold), and FN are false negatives (cells where the deep learning model does not predict water depth above threshold). We computed CSI, averaged over time, for water depth thresholds of 0.05m and 0.3m, as in Bentivoglio et al. (2023).

- **Speed-up**: we measured computational speed-up as the ratio of numerical model computation time to deep learning model inference time. Both times exclude mesh creation, data pre-processing, and post-processing.

- **Plausibility**: we used the average relative mass error (ARME) as a validation metric

Table 6.1: Summary of mesh characteristics for the different meshes used in the mSWE-GNN, for both the baseline case with meshes adapted to hydraulics structures and without adaptation (Sec. 6.4.4).

Mesh level	# Faces	# Edges	Face area (m^2)	Edge length (m)
scale 1 (finest)	178124	298729	1490 ± 1247	46.74 ± 16.81
scale 2 (adapted mesh)	19869	14761	13470 ± 6986	178.52 ± 54.77
scale 3 (adapted mesh)	634	1486	422151 ± 378531	3021.87 ± 1596.29
scale 2 (not adapted mesh)	15563	11532	17196 ± 2791	202.20 ± 27.17
scale 3 (not adapted mesh)	2630	3833	101752 ± 18892	489.96 ± 67.13
scale 4 (not adapted mesh)	866	1237	308615 ± 64612	854.12 ± 126.44

to assess the plausibility of model predictions in scenarios without ground-truth simulations, as described in Section 6.2.2.

6.4. RESULTS AND DISCUSSION

6.4.1. MODEL INVESTIGATION

Table 6.2 shows the model consistently achieves high CSI values, indicating effective prediction of the spatio-temporal evolution of floods across breach locations and return periods. On the test dataset, the model attains an average CSI of 73.6% for the 5 cm threshold and 71.1% for the 30 cm threshold, with mean absolute errors of 64.5 cm for water depth and 1.31 m^2/s for unit discharge. The model performs best in the central range of flood volumes, where training data is concentrated, with lower errors and higher CSI values. The difference between the performance on the training and test datasets is driven by the large variability in total flood volumes, with testing volumes greatly exceeding those in training (see Figure 6.6). The model also tends to over-predict flood extent for the smallest events, since the loss function penalizes larger floods more than smaller ones. Mean absolute errors in water depth tend to increase with the return period, but those for unit discharge remain relatively stable. This is because the largest values of unit discharge occur closest to the breach and then decrease over time with the inflow hydrograph, while water depths accumulate over time, leading to larger errors for more extreme floods.

Among the three test locations, HD073 shows the lowest MAE values, due to less intense flooding. CSIs remain high except for the 100-year return period, where the breach is upstream in a sloped area and small discharge variations cause large changes in inundation. Location ND234 achieves the best overall performance, as it is located downstream and floods consistently during large events, resembling training patterns. ND111 has the highest MAEs, with all simulations producing flood volumes much greater than those in training (Figure 6.6). Despite this, the model captures the spatio-temporal variability of floods, as shown by high CSI values. At ND111, all return periods nearly fill the domain with water, resulting in very high water depths, up to an average of 3.22m covering 90% of the domain. These results highlight that model accuracy depends on both the location and magnitude of flood events. The model performs best when test scenarios resemble training conditions, while extreme or atypical events lead to higher errors but still preserve spatial flood patterns.

6.4.2. MASS BALANCE ERROR

We tested the variability of the ARME on a set of 10,000 flood scenarios. This consisted of 100 equidistant breach locations along the breachable dike perimeter, each with 100 unique breach discharge hydrographs, generated as in training and validation. We designed this range to be purposefully much wider than for a practical scenario, to assess a more complete response of the model to different boundary conditions and locations. We define as “plausible" any simulation whose ARME is below a selected threshold.

We analysed how the ARME threshold affects the distribution of predicted flood volumes over time in Figure 6.8. The number of plausible simulations increases with the threshold, as expected, with approximately 50% of the simulations having an ARME

Table 6.2: Training, validation, and testing metrics for the mSWE-GNN on dike ring 41, reporting mean and standard deviation for the mean absolute error (MAE) for water depth and unit discharge and critical success index for a water depth threshold τ (CSI_τ). These metrics are reported also for the three test locations HD073, ND234, and ND111, for the different return periods (RP). Arrows indicate whether higher (↑) or lower (↓) values are better.

Dataset	**ID**	**RP**	**Total volume**	**MAE ↓**		**CSI_τ [%] ↑**	
		[yrs]	[10^6 m^3]	h [10^{-2} m]	$\|q\|$ [10^{-2} m^2 s^{-1}]	τ=0.05 m	τ=0.3 m
Train	-	-	220.1 ± 131.4	29.56 ± 17.23	1.46 ± 1.00	81.74 ± 9.34	80.40 ± 11.39
Val	-	-	184.7 ± 175.5	23.98 ± 12.64	1.41 ± 1.72	73.80 ± 13.54	71.54 ± 14.32
Test	-	-	323.4 ± 349.2	64.84 ± 81.38	1.31 ± 1.14	73.60 ± 14.90	71.15 ± 14.21
Test	HD073	10000	89.70	26.32	0.99	75.21	71.78
		1000	48.86	15.0	0.49	66.17	66.86
		100	15.99	11.15	0.14	50.12	54.05
	ND234	10000	299.45	30.65	0.89	88.49	85.75
		1000	170.48	13.75	0.66	83.99	84.7
		100	62.21	31.09	0.42	66.94	63.94
	ND111	10000	974.55	247.81	3.25	69.53	61.43
		1000	734.98	152.26	2.85	74.75	67.45
		100	395.71	55.51	2.13	87.15	84.39

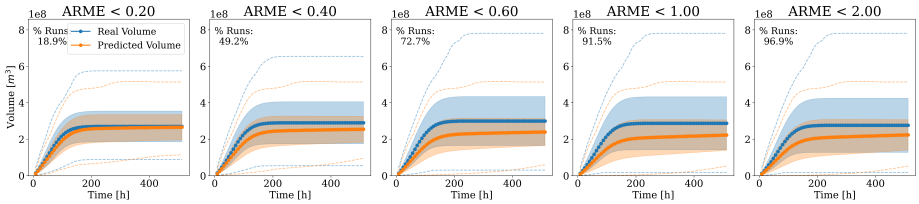


Figure 6.8: Comparison of real and predicted flood volumes over time for multiple ARME thresholds. The shaded regions indicate one standard deviation away from the mean, at each time step, while the dotted lines represent the envelopes of the minimum and maximum volumes at each time step. Increasing the threshold increases the number of plausible simulations but also the discrepancy, both in terms of mean and standard deviation, from the true volumes.

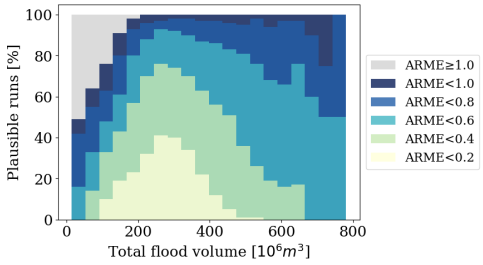


Figure 6.9: Percentage of plausible simulations as a function of the total flooding volume, for different ARME thresholds.

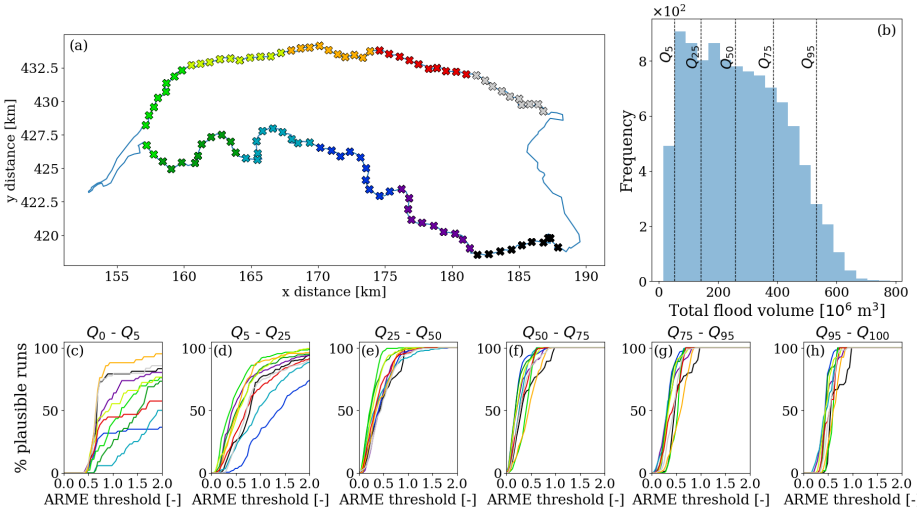


Figure 6.10: Percentage of plausible simulations per breach location, total flood volume ranges, and ARME thresholds. a) Distribution of the testing breach locations. The colours represent different breach location areas, grouped by distance in 10 different zones. b) Frequency of total flood volumes and the corresponding quantiles at 5%, 25%, 50%, 75%, and 95%, determined from the theoretical total flood volumes in each testing simulation. c-h) Percentage of plausible simulations for different volume quartile ranges, for increasing values of the ARME threshold.

6

< 0.4. For higher thresholds, model predictions tend to underestimate volumes, with flood volumes capping at approximately 500 million m^3, further explaining the lower test performance for the largest floods. We can derive a similar conclusion from Figure 6.9, which shows the percentage of plausible simulations for different total flood volumes and ARME thresholds. Most predictions in terms of mass conservation have a range of total volumes between 200 millions and 400 millions m^3 of water. This reflects well the distribution of training simulations with a stronger bias towards higher volumes, because of the training loss function that focuses more on the highest water depths.

Figure 6.10 shows how the percentage of plausible simulations changes with the ARME threshold for different breach location areas and total flood volume quantiles. Steeper curves closer to zero indicate better model performance, as more simulations have low ARME. The central volume ranges (from approximately 150 mil m^3 to 400 mil m^3) yield the best performance, with most simulations having a low ARME. Figure 6.9 also confirms the same finding, aligning with the test dataset's optimal prediction range. Across all volume ranges, most breach locations show similar trends, meaning that the model's performance is consistent independently of the breach location. Simulations with breaches in the western downstream area (green colours) perform best across most volume ranges, with pronounced responses to ARME threshold changes as floods starting here tend to fill the area like a bathtub, creating a recognizable flow pattern.

In contrast, the highest and lowest volume quantiles, which fall outside the training range, behave worse. For the largest floods, most simulations show a sharp increase in ARME between 0.5 and 1, indicating consistent underestimation of volumes; however,

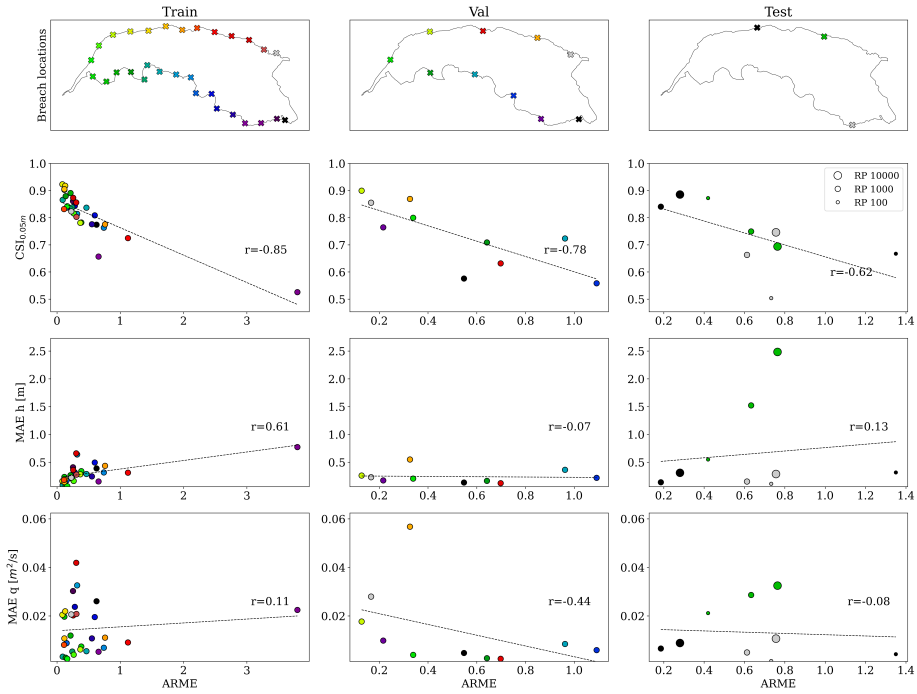


Figure 6.11: Distributions of ARME values for the training, validation, and testing datasets, as a function of CSI at 0.05m and MAEs for water depth (h) and unit discharges (m^2/s). The colours match the location of the breach, while the size, for the testing dataset, indicates the return period.

even for very high discharges, the model has no instabilities, as indicated by the lack of exponentially increasing flood volumes curves (Figure 6.8). In the lowest volume quantiles, curves have the highest variability, as small volume errors cause larger ARME changes and lead to a bigger spread in performance. For this range, upstream locations (black, grey, and purple markers) perform better, suggesting the model understands better flow patterns for small floods when the terrain is sloped. The central-south area, dividing the western downstream bathtub region from the eastern upstream sloped one, shows more frequent errors, likely because small errors in flood routing cause larger inconsistencies, also due to the presence of several canals and elevated elements.

We evaluated the correlation between CSI and MAE with the ARME for the training, validation, and testing datasets, for which ground-truth data exists (Figure 6.11). Across datasets, ARME and CSI exhibit a consistent negative dependence with correlation coefficients around -0.7, indicating that low ARME values are associated with high CSI values (Table 6.3). Contrarily, we found little correlation with the mean absolute errors for water depths (r $\approx$ 0.6 only in training) and no correlation with the discharge mean absolute errors. This means that simulations with a low ARME predict well the spatio-temporal evolution of the flood and can be used as a valid proxy to determine the plausibility of testing simulations, but are not always reliable in terms of hydraulic variables' values.

Whereas numerical models require strict mass balance with an ARME close to zero,

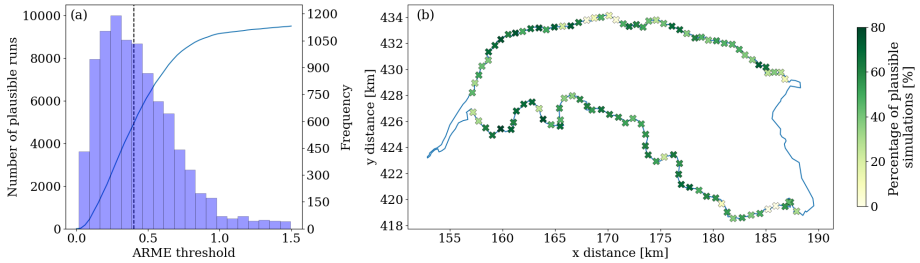


Figure 6.12: a) Cumulative number of plausible simulations and frequency different ARME thresholds. b) Spatial distribution of the percentage of plausible simulations, assuming a ARME threshold of 0.4.

our experiments tolerate higher variability since we do not explicitly enforce mass conservation and an ARME<0.4 always correlate with high CSI (>0.75), which means that the spatio-temporal dynamics are well represented (Figure 6.11). For this reason, we selected as a reasonable threshold an ARME = 0.4, though lower values can also be employed based on the desired level of model performance. For this threshold of ARME < 0.4, we analysed the spatial distribution of the percentage of plausible simulations per breach location (Figure 6.12b). Similarly to what we found in the general analysis, the highest percentage of plausible simulations is associated with the western breaches, thanks to the bathtub accumulation pattern. There are also spurious locations for which no or few simulations provided had an ARME < 0.4. For the two locations in the south-east border, this seems to be correlated with the presence of an area that rarely gets flooded due to the locally higher topography close to the breaches. This causes the model to severely underestimate the intensity of flooding around that area if there is a local breach and, consequently, in

Table 6.3: Pearson and Spearman correlation coefficients (with 95% confidence intervals) between ARME and each metric across datasets. Correlations higher than 0.6 are marked in **bold**.

Dataset	Metric	Pearson r [95% CI]	Spearman ρ [95% CI]
Train	$CSI_{0.05m}$	**−0.851** [−0.924, −0.717]	**−0.871** [−0.935, −0.753]
	$CSI_{0.3m}$	**−0.838** [−0.918, −0.695]	**−0.851** [−0.924, −0.716]
	MAE_h	**0.611** [0.339, 0.789]	**0.663** [0.414, 0.820]
	MAE_q	0.110 [−0.243, 0.436]	0.104 [−0.248, 0.432]
Validation	$CSI_{0.05m}$	**−0.781** [−0.931, −0.404]	**−0.830** [−0.948, −0.515]
	$CSI_{0.3m}$	**−0.684** [−0.897, −0.213]	**−0.733** [−0.915, −0.306]
	MAE_h	−0.059 [−0.591, +0.509]	−0.261 [−0.710, 0.339]
	MAE_q	−0.437 [−0.796, +0.151]	−0.612 [−0.870, −0.092]
Test	$CSI_{0.05m}$	**−0.604** [−0.875, −0.046]	**−0.700** [−0.909, −0.211]
	$CSI_{0.3m}$	**−0.709** [−0.912, −0.227]	**−0.700** [−0.909, −0.210]
	MAE_h	0.086 [−0.513, 0.629]	0.333 [−0.297, 0.762]
	MAE_q	−0.093 [−0.633, 0.508]	−0.050 [−0.606, 0.539]

the rest of the domain.

6.4.3. PROBABILISTIC FLOOD MAPPING

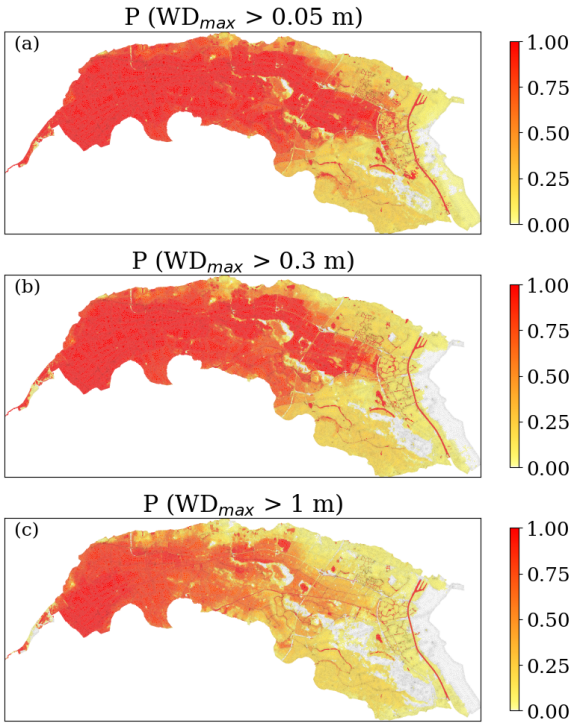


Figure 6.13: Predicted probabilities of maximum water depths exceeding different thresholds. They are determined using only the selected simulations among the tested 10000 configurations that had a ARME < 0.4, assuming that each simulation has the same likelihood of occurrence.

LARGE-SCALE UNCERTAINTY

We exemplified the model in a probabilistic setting by computing the conditional probability of exceeding a certain maximum water depth for the large-scale uncertainty analysis in breach location and outflow as described in Section 6.4.2. Figure 6.13 shows the likelihood on having a maximum water depth higher than $0.05m$, $0.3m$, and $1m$. This probability is conditional to a breach occurring and is independent of the return period, meaning that all events have the same probability of occurrence. It is also computed over all 10,000 testing simulations with an ARME < 0.4. Selecting only plausible results skews the probability distributions to be less spread out with respect to the complete set, giving a clearer distribution estimate. This analysis confirms that the western downstream area of dike ring 41 is most likely to flood, due to its bathtub-like accumulation. The eastern upstream part is less likely to flood, requiring either a breach in that area or a large event.

The model predicted all 10,000 simulations in about ten hours, the same average time

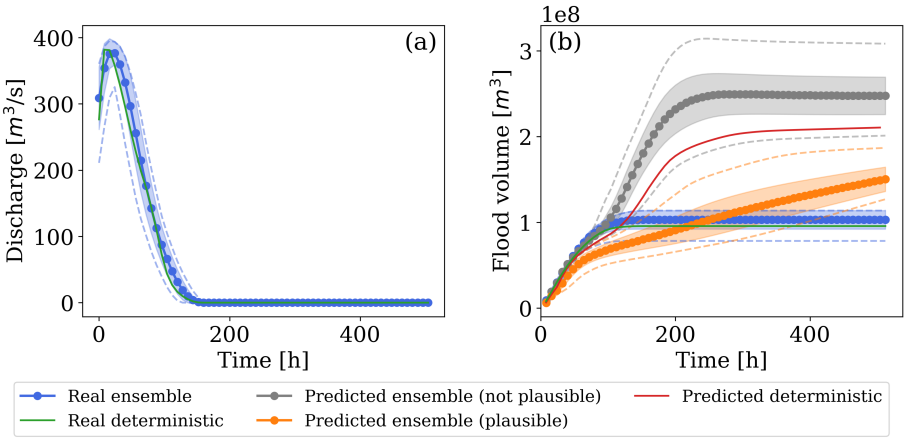


Figure 6.14: Range of discharges and volumes over time for the deterministic and ensemble cases, for true values and predicted ones, plausible and not. The shaded regions indicate one standard deviation away from the mean, at each time step, while the dotted lines represent the envelopes of the minimum and maximum volumes at each time step.

the numerical model needs for a single numerical simulation, corresponding to speed-up of 10,000 times. Even when considering that only 50% of the simulations were plausible the approach is highly efficient, highlighting its potential for large-scale probabilistic flood mapping and rapid scenario analysis.

COMPARISON OF ENSEMBLE AND DETERMINISTIC SCENARIOS

We quantified the uncertainty in flood hazard predictions in a test scenario by analysing the variability in maximum water depths and flood arrival times for different boundary conditions. We assumed a threshold of $0.05m$ for identifying when a cell got flooded. We determined the breach outflow hydrograph from river water levels using the calibrated conceptual method in Besseling et al. (2025). This approach models breach evolution and outflow as either free flow or submerged flow, depending on whether the breach is unconfined or confined. The outflow is calibrated via a parameter fitted using a ground-truth reference outflow from a numerical simulation. The outflow hydrographs served as boundary condition for the trained mSWE-GNN model. We compared the ensemble results against the single-scenario prediction and its corresponding ground-truth simulation. We refer to this single prediction as the 'deterministic' result.

We selected breach location HD073 and a return period of 10,000 years as a representative test. Using only non-identical hydrographs, sampled from the fragility curves, we obtained 206 different boundary conditions. Considering only simulations with ARME below 0.4, 25% of the ensemble produced flood volumes over time close to the ground truth (Figure 6.14). While the deterministic prediction tends to overestimate the final flood volumes, plausible ensemble members provide a more accurate representation of the flood dynamics, both in terms of maximum water depths (Figure 6.15) and flood arrival times (Figure 6.16). The 50^{th} percentile of the ensemble predicts lower water depths and slower propagation compared to the deterministic case, aligning more closely

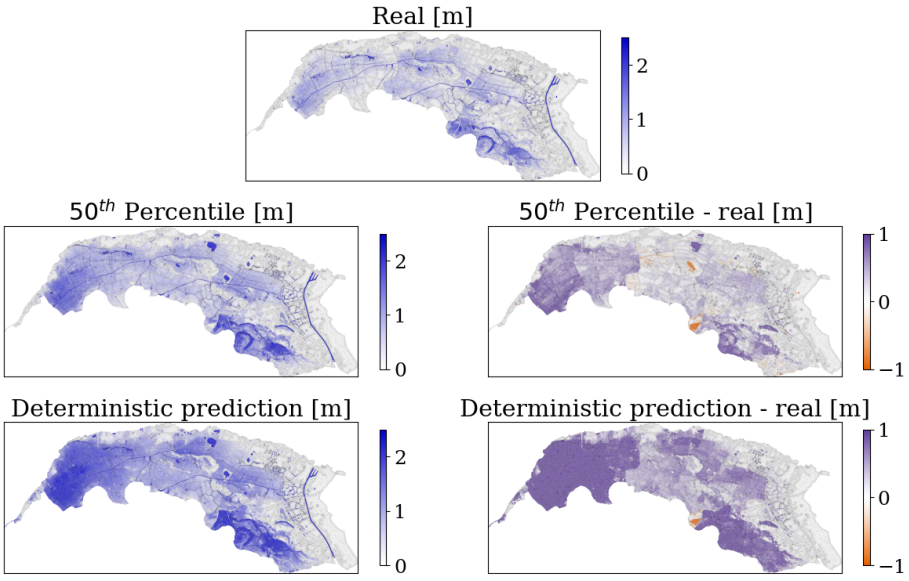


Figure 6.15: mSWE-GNN ensemble predictions of maximum water depths $[m]$ for the 50^{th} percentile compared to the ground-truth numerical simulation for breach HD073 and a return period of 10,000 years. The plots on the right side represent the difference between the mSWE-GNN predictions and the numerical model's.

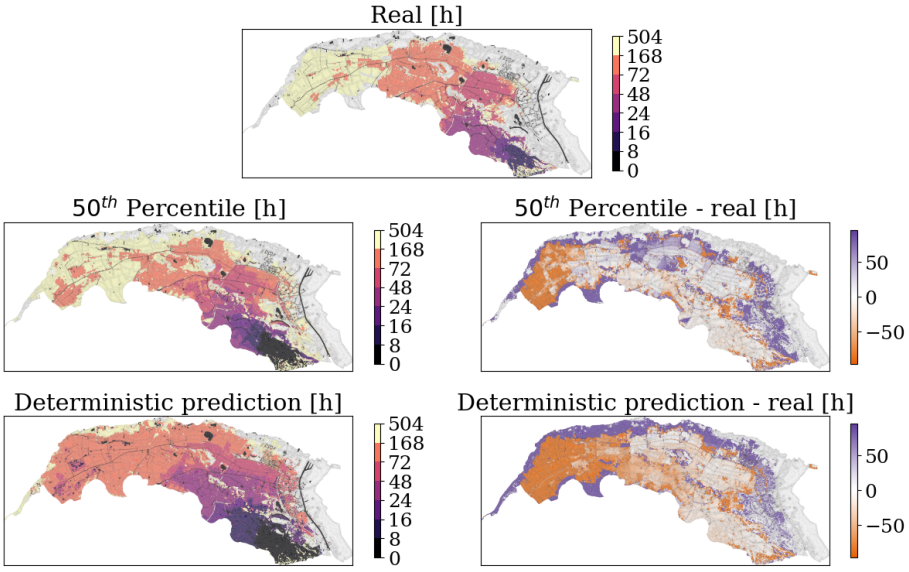


Figure 6.16: mSWE-GNN ensemble predictions of flood arrival times $[h]$for the 50^{th} percentile compared to the ground-truth numerical simulation for breach HD073 and a return period of 10,000 years. The plots on the right side represent the difference between the mSWE-GNN predictions and the numerical model's.

Table 6.4: Ablation study for the inclusion of hydraulic structures in the model as node, edge, or mesh features. w/o = without.

DL model	Dataset	MAE ↓		CSI$_\tau$ [%] ↑	
		h [$10^{-2}m$]	$\|q\|$ [$10^{-2}m^2s^{-1}$]	τ=0.05 m	τ=0.3 m
base model	val	**23.98 ± 12.64**	1.41 ± 1.72	**73.80 ± 13.54**	**71.54 ± 14.32**
	test	**64.84 ± 81.38**	1.31 ± 1.14	**73.60 ± 14.90**	**71.15 ± 14.21**
w/o extra	val	27.45 ± 19.74	**0.92 ± 0.94**	72.31 ± 15.20	70.11 ± 15.46
node features	test	70.83 ± 85.70	**0.97 ± 0.87**	**73.58 ± 14.24**	70.15 ± 13.70
w/o extra	val	36.71 ± 31.31	1.09 ± 1.17	69.14 ± 14.22	65.65 ± 15.60
edge features	test	68.99 ± 82.90	0.99 ± 0.88	73.27 ± 9.89	67.92 ± 10.26
only adapted	val	34.43 ± 22.04	1.22 ± 1.64	70.31 ± 16.03	67.97 ± 16.14
coarse meshes	test	71.32 ± 77.96	1.28 ± 1.00	70.42 ± 19.01	67.72 ± 18.26
w/o adapted	val	36.45 ± 17.69	1.21 ± 1.16	68.22 ± 18.85	65.92 ± 19.16
coarse meshes	test	69.06 ± 70.22	1.41 ± 1.01	68.72 ± 19.94	66.20 ± 19.12
no hydraulic	val	46.92 ± 28.70	1.12 ± 1.18	63.82 ± 17.66	60.00 ± 18.35
structures	test	77.91 ± 79.46	1.20 ± 0.95	67.06 ± 18.91	62.53 ± 18.02
base model	val	60.48 ± 26.82	2.33 ± 1.41	60.75 ± 18.70	58.43 ± 20.36
($z_{ee} - z_i = 0$)	test	85.30 ± 41.87	2.88 ± 1.42	63.49 ± 21.88	61.31 ± 21.32

with ground-truth observations. Although these predictions do not exactly correspond to the ground-truth simulation, as they assume a smaller outflow discharge, they demonstrate that the model's performance can improve with slight adjustments to boundary conditions, when selecting simulations with smaller ARME.

The predicted spread of the ensemble is larger than the theoretical one calculated from the boundary conditions (Figure 6.14b), primarily because the chosen location is upstream of multiple hydraulic structures, making it especially sensitive to minor variations in outflow discharges, as also reflected in the elevated training error for one of the upstream breaches (Figure 6.11, purple marker). This sensitivity is also amplified by the limited amount of training data, which makes the model's response more variable in these conditions. In contrast, predicted uncertainty is lower for smaller events and downstream locations, where the influence of small variations in boundary conditions is reduced, leading to more stable and consistent predictions.

6.4.4. ABLATION STUDY: INCLUDING HYDRAULIC STRUCTURES

We analysed the impact of hydraulic structures as node and edge features, and coarse mesh, as discussed in Section 6.2.1. For the case without adapted coarse meshes, we used four scales (details found in Table 6.1) instead of three to compensate for the lack of elongated cells. As in the base architectures, we distributed GNN layers heterogeneously across scales: two layers for the finest scale after U-Net pooling, three and four layers for scale one (before and after the U-Net bottleneck), four and two layers for the coarser scale, and six layers for the coarsest scale.

Table 6.4 shows that all proposed changes improved most metrics (row 1, base model). Using coarse meshes that fit existing hydraulic structures produced the largest improvement (cfr. row 5, w/o adapted coarse meshes), likely due to better treatment of canals, which convey water quickly in a specific direction. In GNN models, the number of layers should match the speed of water propagation within a given time range, set by the model's time step. Adding longitudinal mesh elements allows water to move efficiently without requiring many GNN layers. However, adapting the meshes alone, without including edge features for hydraulic structures, led to worse results, especially in testing (row 4, only adapted coarse meshes). This suggests that while mesh adaptation helps, it is insufficient alone; the model also needs explicit information about hydraulic structures to learn their effects on flood propagation.

Adding elevated elements as an edge feature improved results mainly in validation (cfr. row 3, w/o extra edge features). Its effect on the test dataset was less pronounced, likely because very large floods overtop elevated elements easily. To further validate the importance of elevated elements as edge features, we also tested the base model after setting these values to zero (row 7, base model $z_{ee} - z_i = 0$). The marked drop in performance confirms that the model has effectively learned to account for the influence of these structures on flow dynamics.

Marking lakes and canals with a binary node feature had the least impact (row 2, w/o extra node features), probably because the model already learns this information from the water variable vectors. However, removing these features improved the MAE for unit discharge for both validation and testing datasets. This suggests that the model can infer the presence of these structures from the flow patterns, and explicitly providing this information may not be necessary.

6.5. CONCLUSIONS

This study demonstrated the feasibility of using deep learning surrogates for probabilistic flood hazard mapping in real case studies. To this end, we included hydraulic structures such as canals, underpasses, and elevated roads as model inputs and introduced the average relative mass error (ARME) as a physically-based plausibility metric to validate the model in absence of ground-truth data. With this framework, we generated probabilistic flood maps that quantify uncertainty in maximum water depths and flood arrival times for varying breach locations and discharge hydrographs.

We found a strong correlation between the ARME and the critical success index (CSI), confirming the ARME's value as an indicator of simulation reliability when reference data are unavailable. In a large-scale analysis of 10,000 scenarios, approximately half of them produced physically consistent outcomes, predominantly within mid-range breach outflows, suggesting that model generalization remains most reliable within trained discharge regimes. Beyond quantifying uncertainty, the framework offers a practical tool for identifying locations most sensitive to input variations and for prioritizing expensive numerical simulations, thereby improving the efficiency of flood risk assessment.

While the current study focused on a single case study and simplified breach representation, future research should validate the framework to multiple interacting breaches, integrate dynamic river and breach evolution models, and validate across diverse case studies. Moreover, future studies could include other sources of uncertainties, such

as river water levels or floodplain roughness (Hall & Solomatine, 2008). Future studies should also include the probabilities of dike failure by comparing the expected hydraulic loads and the geotechincal properties of dike segments to obtain a complete probabilistic map (e.g., Jongejan & Maaskant, 2015). Moreover, future works could explore the use of mixture-of-experts models to improve the model performance across a wider range of boundary conditions, for example by combining or selecting the output of different models, each trained with a smaller range of conditions.

Overall, this framework advances the deployment of rapid surrogate models for probabilistic flood analysis and flood hazard mapping. It enables real-time scenario assessment, quantifies flood variability for different locations and boundary conditions, and helps identify the most critical breach and discharge combinations. These capabilities support more effective and efficient flood risk management and represent a significant step toward operationalizing deep learning surrogates in flood risk assessment.

7

Conclusions and Recommendations

A river cuts through rock, not because of its power, but because of its persistence.

James N. Watkins

7.1. Conclusions

This thesis establishes graph neural networks (GNNs) as a surrogate model to simulate the spatio-temporal evolution of floods. We positioned its contributions via a thorough review of the literature of deep learning (DL) methods for flood mapping and identified three main research gaps related to generalization of DL models, their modelling limitations, and operational issues. We define *generalization* as the capability of a DL model to accurately predict the spatio-temporal evolution of floods under topographic settings and boundary conditions that were not included in its training. Based on the identified gaps, we formulated a series of research questions and, through their investigation, demonstrated that graph neural networks provide a robust framework for real-time flood uncertainty quantification in practical case studies. The aim of the developed surrogate model is to replace the two-dimensional overland flow component with hydraulic models required to produce flood hazard maps.

We first proposed a GNN model (SWE-GNN) based on the finite volumes method to solve the shallow water equations (SWE). In this model, the propagation rule follows hydraulic principles, which allow it to preserve some physics by constraining water to spread only from cells already containing water. The number of GNN layers is proportional to the time resolution and the flood propagation's speed. Combined with an innovative training strategy, this model outperformed other DL models in terms of performance and generalization to unseen case studies. We then developed an improved multi-scale version (mSWE-GNN) that could combine the flood propagation at different resolution to achieve a better Pareto front in terms of accuracy and speed, compared to the non-multi-scale

counterpart. The model could generalize to unseen time-varying boundary conditions, thanks to the use of ghost cells, and to unseen mesh configurations, thanks to the use of rotation-invariant inputs. Finally, we included hydraulic structures as inputs and we estimated uncertainty in flood hazard mapping of a real case study in the Netherlands, showcasing its accuracy and speed for practical applications. In this chapter, we summarize the main conclusions of this dissertation, based on the proposed research questions, highlight possible future research directions, and reflect on the impact of the proposed models on a broader setting. Throughout the thesis, a series of benchmark datasets was made available, which can openly accessed via Zenodo[1,2]. Similarly, all models and codes developed in this thesis are stored in GitHub repositories[3,4].

7.2. ANSWERS TO THE RESEARCH QUESTIONS

We address the main research question as well as each research sub-question defined in Chapter 1 with its corresponding conclusions.

Main RQ: How can we leverage deep learning models for spatio-temporal flood prediction across varying case studies, boundary conditions, and mesh configurations in support of hazard mapping?

Deep learning models can support spatio-temporal flood prediction across diverse case studies, boundary conditions, and mesh configurations by combining physical principles with graph-based architectures. Hydraulic-based graph neural networks embed hydraulic knowledge, ensuring plausible predictions and enabling generalization to unseen locations including real case studies, while achieving speed-ups up to 10,000 times compared with numerical models. Multi-scale designs, ghost cells, and rotation-invariant features further allow adaptation to new boundary conditions and unstructured meshes without relying on numerical model outputs. Large-scale ensemble simulations provide fast uncertainty estimates, validated against conservation laws, enabling the generation of accurate, physically consistent, and computationally efficient flood hazard maps across varying scenarios, especially in the context of dike-breach floods.

RQ1: What is the state-of-the-art of deep learning for flood mapping?(Chapter 3)

Deep learning models outperform traditional numerical approaches and other machine learning techniques in terms of speed and accuracy, depending on the mapping application. Convolutional neural networks are the dominant state-of-the-art architecture for flood mapping thanks to their spatial inductive bias. However, their structure makes them more suitable for flood mapping from satellite images than for surrogating numerical simulations, because they cannot easily adapt to flexible meshes. On the other hand, graph neural networks, which we develop and analyse in this thesis have that potential.

[1] https://dx.doi.org/10.5281/zenodo.7764418
[2] https://dx.doi.org/10.5281/zenodo.13326595
[3] https://github.com/RBTV1/SWE-GNN-paper-repository-
[4] https://github.com/RBTV1/mSWE-GNN

RQ2: How can deep learning surrogate models include hydraulic properties to support the generalization of spatio-temporal flood predictions in unseen geographical areas?(Chapter 4)

Deep learning surrogates based on graph neural networks (GNNs) allow to include hydraulic constraints in the layer propagation rule. By leveraging the analogy with finite volume methods for shallow water equations, a novel GNN layer (SWE-GNN) was developed that introduces a physical bias, ensuring that water spreads only from cells already containing water. The optimal number of GNN layers is proportional to the mesh size and output time resolution. Long-term prediction instabilities are mitigated through a multi-step loss function combined with a curriculum learning strategy. This approach is tested on three synthetic datasets of increasing difficulty showing that it enhances generalization to unseen flood locations compared to other deep learning models.

RQ3: What principles from numerical solvers can be exploited to improve the generalization of graph neural networks across boundary conditions and mesh configurations without relying on numerical model initialization?(Chapter 5)

Multi-scale methods with ghost cells and rotation-invariant inputs enable generalization of graph neural networks to new boundary conditions and mesh configurations without requiring initialization from numerical models. By applying GNNs at multiple spatial resolutions and combining them via a U-Net style architecture, the model achieves computational efficiency and scalability. Ghost cells handle time-varying boundary conditions, while rotation-invariant node features allow adaptation across unstructured meshes. This design allows the network to generalize to entirely new case studies achieving both speed and accuracy. The model also generalizes from synthetic simulations to a real case study - dike ring 15 in the Netherlands - achieving a mean critical success index (CSI) of 0.87 with a single fine-tuning simulation.

RQ4: How can graph neural networks be used to estimate uncertainty for flood hazard mapping in case studies with hydraulic structures?(Chapter 6)

GNNs can estimate uncertainty in flood hazard mapping thanks to a large ensemble of simulations carried out up to 10,000 times faster than with physics-based models. The outputs should also be validated against a mass conservation criterion to ensure physical plausibility, especially in conditions where no ground-truth data is available. Hydraulic structures can be encoded in the model via modifications in the node, edge, or mesh characteristics taken as input. This approach can be used in realistic case studies to generate probabilistic flood hazard maps that consider uncertainty in breach location and breach outflow discharges for dike-breach flooding. Testing on a real case study (dike ring 41, the Netherlands) shows that the model can achieve a mean CSI of 0.74 across a wide range of scenarios.

7.3. FUTURE RESEARCH DIRECTIONS

1. **Other types of floods**. While this thesis focused on dike-breach floods, the same methods developed here can be applied with little modifications to other types

of floods. For river floods, with or without defences, the model can work in the same way since inputs and outputs are generally identical. For dam breaks, the model does also not require any changes, but might benefit from high resolution in the output temporal frequency, due to the rapidly-evolving flows. For pluvial floods, precipitation data must be added as a node feature both using previous values and forecasted ones, similarly to how this is done in distributed rainfall-runoff modelling. For urban floods, 1D urban drainage systems can be combined to 2D surface floods via links between the two computational graphs (e.g., Finaud-Guyot et al., 2011). For coastal floods and tsunamis, water levels can be imposed as boundary conditions, while additional forcings such as wind could be added as node features.

2. **Large-scale flood warning**. Several countries employ real-time early warning systems and flood mapping at a national and continental scales. These results are currently obtained either with hydrological models or only performed at low frequency for higher resolution maps. Employing fast surrogates can allow countries to obtain both fast and accurate results at the same time, similarly to how meteorological agencies such as the European Centre for Meteorological Weather Forecast (ECMWF) already employ DL-based alternatives to traditional numerical models (Lang et al., 2024). Similarly to what we report in this thesis, these models could be used for real-time forecasting of floods and for uncertainty analysis.

3. **Foundational surrogate flood models**. This thesis focused on low-lying areas as case studies, since they are commonly protected by flood defences. Future studies should also verify if the same model works under different topographical conditions, such as mountainous areas and larger river systems. A collection of homogenized simulations dataset can then be the starting point to develop a foundational model for floods, i.e., a single model that works under very different flood conditions, in terms of timings, dynamics, and spatial characteristics, similarly to what was done in hydrology (Addor et al., 2017). The data efficiency of the models developed in this thesis suggests that only a relatively small number of simulations may be needed for such a large and diverse dataset. However, building a truly foundational model ultimately requires integration with real-world data, which are often scarce, heterogeneous, and cannot capture the complete spatio-temporal evolution of flood events (see next point). Simulations therefore remain a necessary preliminary step toward this goal.

4. **Data integration**. Surrogate models are, by definition, trained on numerical simulations. However, for many case studies there are also observations that can be used to fine-tune these models. These can be collected from measuring stations, water marks, satellite images, and social-crowded images. A continuous integration of such data can potentially improve the accuracy of surrogate models to be better than state-of-the art numerical methods, as was shown, for example, in weather forecasting (Lam et al., 2023). This step is also essential in developing true foundational models that are not exclusively dependent on simulations.

5. **Combination with morphological models**. Morphological models determine the

joint evolution of a flood over time and the river bathymetry. Because of the coupling of equations that are highly unstable, these systems require even more computational resources than hydraulic flood models (Williams et al., 2016). Since the numerical structure of these problems is very similar to that of water flows alone, the models developed in this thesis could provide a promising avenue to supplement the field of morphodynamics with faster models.

6. **Hybrid models**. Preserving physical guarantees can only be done when constraining the model to respect physical equations. A possible future research direction consists in combining numerical models with machine learning by making the codes differentiable (Kochkov et al., 2024). In this way, the model preserves physical meaningfulness but also allows to automatically calibrate the model on observations, for example by adjusting the roughness coefficient.

7. **Other DL models**. This thesis focused on graph neural networks because of their relational inductive bias and computational similarity with finite volume methods, which allowed to integrate physical principles from numerical methods in hydraulics. GNNs exploit the computational mesh as a structural bias for the hydraulic computations. Advances in GNNs with spatio-temporal models (e.g., Sabbaqi & Isufi, 2023) or generalizations to higher-order interactions (e.g., Yang et al., 2022) may further benefit this bias and achieve better performance. Moreover, new promising models, such as neural operators (Li et al., 2020), neural fields (Xu et al., 2021), and diffusion models (Chen et al., 2023), still have to make their way into flood modelling. This is likely because they are generally developed for fluid dynamics problems, for which the medium is generally continuous over the full computational domain, contrarily to floods in which many areas might remain or become dry during a flood event.

8. **Flood hazard mapping**. The models developed in this thesis focus on simulating overland flow, to obtain deterministic and probabilistic flood hazard maps. To obtain the true probabilities of occurrence, we need to weight the conditional probabilities (Chapter 6) by the failure probability and return period. This provides a more accurate measure of the probability of flooding, which can further be included in flood risk mapping via the addition of vulnerability curves and exposure maps.

7.4. BROADER PERSPECTIVE

Deep learning is still at an early stage in flood management, but it could become as transformative as numerical models. Rather than replacing existing tools, it can complement them, especially where physics-based methods face limitations. Promising uses include uncertainty quantification and real-time forecasting, which can leverage the speed and accuracy of these approaches. Deep learning models can also serve as a rapid preliminary assessment tool that flags scenarios for more detailed simulation with physics-based models. This workflow already resembles how conceptual models are used in hydrology. Still, operational deployment should proceed gradually and be backed by rigorous benchmarking to build trust through consistent and demonstrable value.

Interpretability, robustness, and uncertainty quantification remain critical challenges to overcome before deep learning can support high-stakes decisions in disaster response or infrastructure planning. To this end, it is essential to have a transparent evaluation with standardized datasets and shared benchmarks. Adoption will also require institutional and cultural changes, including closer collaboration among AI researchers, domain experts, policymakers, and communities. Ultimately, this thesis contributes to that shift by advancing more data-efficient, generalizable, and physically grounded models, paving the way for trustworthy and scalable AI tools in flood management.

ACKNOWLEDGEMENTS

It's difficult to effectively summarize a journey of 5 years. I started this PhD looking for a challenge and I got more than I was expecting. Navigating through the PhD wouldn't have been the same without the help and presence of so many people next to me. I could have spent so many more kind words to each of you, but I'll be quite short for the sake of length.

First of all, I want to express my sincere gratitude to my supervisors Riccardo Taormina, Elvin Isufi, and Bas Jonkman for supporting me from the beginning to the end of this PhD. I'm extremely grateful to Riccardo not only for trusting in my capabilities from the very beginning and selecting me for this position 5 years ago, but also for the continuous support, encouragement, involvement in opportunities, and help that I've received since. A huge thanks also to Elvin, whose thorough, extensive, and repeated rounds of comments and questions, which, despite making it also frustrating at times, made me grow professionally more than from any other source. I wouldn't be the researcher I am now without this. Thank you also for trying to always organize events and group activities to bond across PhDs. Thanks to Bas for always replying promptly, being available, and providing guidance, despite having a full agenda.

Throughout the PhD I've managed to collaborate with several other people and for that I want to thank them. Thanks to Ruben Dahm for facilitating the interactions with Deltares and allowing me to spend a year partially visiting there. Thanks to Bob Maaskant from HKV for providing me with several simulations and Delft3D models that made the last part of my PhD possible. Thanks to Leon Besseling for the fruitful discussions, random concert encounters, and for helping with dike-breach modelling. Thanks to Marcio Giacomoni and Cesar do Lago for interesting conversations at the beginning of my PhD that also led to a nice paper together. Thanks to Bulat for hating on GNNs together with me and Alex and managing nonetheless to make them work also for water networks.

Thanks to all the friends and colleagues at Water Management I had the pleasure to spend time with at TU in these past 5 years. First of all, a special thanks to my paranymph Alex, for sharing most of these long 5 years together at Aidrolab and struggling at the same time with GNNs. I couldn't have wished for a better friend to share this journey with. Another special thank to João for being a great colleague, roommate, and friend. I also want to especially thank Aashna, Andrés, Anto, Job, Katja, and Luuk for the long chats and activities both in and outside of the office. For the lunches and many office activities, thanks to André, Andrea, Anik, Alejandra, Arianna, Bas, Carina, Carolina, Cole, Jana, Jerom, Julia (Brazil), Julia (Norway), Katherine, Kostantinos, Iosif, Laura, Mario, Nathalie, Omar, Qin, Roos, Sander, Sid, Simon, Sofia, Tianlong, Ties, Tuğba, Valeria, Veerle, Yana, and Yoeri (It's a long list so I hope to not forget anyone). Being surrounded by some many good people made it easier to get through the PhD.

In this PhD, I had the chance to also interact with a lot of people at Computer Science (EEMCS). For the several talks about the supervision of Elvin and for more fundamental

chats, thanks to Andrea, Bishwadeep, Chengen, Maosheng, Mohamed, Mohammad, and Ting. I also managed to spend quite some time with teaching and education, and for that I'm very grateful. In this regard, thanks to Anne, Joep, Leon, Robert, Tom, and Yuri as well as the rest of the MUDE and DSAIE teams for making the experience even nicer. I spent quite some time with the faculty (and department) PhD councils in the last years. I'm glad I could help trying to improve the conditions for PhDs with people like Angeliki, Entela, Enxhi, Til, Nirvana, Weilun, and all the other members that I had the chance of meeting at the beginning and the end of my period there.

Thanks to Steven, Anja, and Yvo for being among the first friends I could make when arriving in Delft and for the many times out even after that. Thanks to Zé and Stephan for the many fun days and nights spent playing board games. Thanks also to the whole Hitmanics team, with which I could detach from work to play some baseball. In particular thanks to Amedeo, Dylan, Eloi, Keiya, Geert, Marius, Martijn, Mauryze, Mees, Raf, Teddy, Tom, and Vincent for the many games and trainings together.

Grazie ai miei fantastici genitori per il supporto (sia economico che morale) che ho sempre ricevuto, sia in casa che fuori. Uno strano grazie anche a Claudio, per avermi reso così competitivo in ogni cosa; alla fine ha aiutato a potercela fare fino a qua.

Por último, pero no menos importante, infinitas gracias a mi paraninfa y futura mujer Patricia por apoyarme siempre, en las buenas y en las malas. Tu presencia hace mi vida mejor y tengo mucha suerte por tenerte a mi lado.

CURRICULUM VITÆ

Roberto BENTIVOGLIO

14-12-1996 Born in Piacenza, Italy.

EDUCATION

2020–2026 Ph.D. Civil Engineering

Technical University Delft, Delft, NL

Thesis: Modelling floods via graph neural networks with applications to dike-breach floods

Promotors: Prof. S.N. Jonkman, Dr. R. Taormina, Dr. E. Isufi

2018–2020 Masters of Science in Civil Engineering for Mitigation of Risk from Natural Hazards (Grade: 110/110 cum Laude)

University of Pavia, Pavia, IT; University School for Advanced Studies, Pavia, IT

2015–2018 Bachelor of Science in Civil and Environmental Engineering (Grade: 110/110 cum Laude)

University of Pavia, Pavia, IT

2010–2015 High school

Liceo M. Gioia, Piacenza, IT

List of Publications

Journal Publications

- **Bentivoglio, R.**, Jonkman, S. N., Isufi, E., and Taormina, R., 2025. *Probabilistic Flood Hazard Mapping for Dike-Breach Floods via Graph Neural Networks*, EGUsphere 2025 (2025): 1–29.
- **Bentivoglio, R.**, Isufi, E., Jonkman, S. N., and Taormina, R., 2025. *Multi-scale hydraulic graph neural networks for flood modelling*, Natural Hazards and Earth System Sciences, 25(1), pp.335-351.
- **Bentivoglio, R.**, Isufi, E., Jonkman, S.N. and Taormina, R., 2023. *Rapid spatio-temporal flood modelling via hydraulics-based graph neural networks*. Hydrology and Earth System Sciences, 27(23), pp.4227-4246.
- **Bentivoglio, R.**, Isufi, E., Jonkman, S.N. and Taormina, R., 2022. *Deep learning methods for flood mapping: a review of existing applications and future research directions*, Hydrology and Earth System Sciences, 26, 4345–4378, 2022.
- Kerimov, B., **Bentivoglio, R.**, Garzón, A., Isufi, E., Tscheikner-Gratl, F., Steffelbauer, D.B. and Taormina, R., 2023. *Assessing the performances and transferability of graph neural network metamodels for water distribution systems*. Journal of Hydroinformatics, 25(6), pp.2223-2234.
- do Lago, C.A., Giacomoni, M.H., **Bentivoglio, R.**, Taormina, R., Junior, M.N.G. and Mendiondo, E.M., 2023. *Generalizing rapid flood predictions to unseen urban catchments with conditional generative adversarial networks*. Journal of Hydrology, 618, p.129276.

International Conference Publications

- Sørensen, A.J., **Bentivoglio, R.**, Mikkelsen, P.S., Taormina, R., and Löwe, R., 2025. *Neural networks for the simulation of pluvial urban flooding*. In 13th Urban Drainage Modelling Conference. Innsbruck, Austria.

BIBLIOGRAPHY

Addor, N., Newman, A. J., Mizukami, N., & Clark, M. P. (2017). The camels data set: Catchment attributes and meteorology for large-sample studies. *Hydrology and Earth System Sciences*, *21*(10), 5293–5313. https://doi.org/10.5194/hess-21-5293-2017

Ahmadlou, M., Arora, A., Thi, N., & Linh, T. (2021). Flood susceptibility mapping and assessment using a novel deep learning model combining multilayer perceptron and autoencoder neural networks. *Journal of flood risk management*, (October 2020), 1–22. https://doi.org/10.1111/jfr3.12683

Ahmed, N., Hoque, M. A.-A., Arabameri, A., Pal, S. C., Chakrabortty, R., & Jui, J. (2022). Flood susceptibility mapping in brahmaputra floodplain of bangladesh using deep boost, deep learning neural network, and artificial neural network. *Geocarto International*, *37*(25), 8770–8791. https://doi.org/10.1080/10106049.2021.2005698

Alcrudo, F., & Garcia-Navarro, P. (1993). A high-resolution Godunov-type scheme in finite volumes for the 2D shallow-water equations. *International Journal for Numerical Methods in Fluids*, *16*(6), 489–505. https://doi.org/10.1002/fld.1650160604

Amini, J. (2010). A method for generating floodplain maps using ikonos images and dems. *International Journal of Remote Sensing*, *31*(9), 2441–2456. https://doi.org/10.1080/01431160902929230

Apel, H., Thieken, A. H., Merz, B., & Blöschl, G. (2006). A probabilistic modelling system for assessing flood risks. *Natural hazards*, *38*, 79–100. https://doi.org/10.1007/s11069-005-8603-7

Arias, P., Bellouin, N., Coppola, E., Jones, R., Krinner, G., Marotzke, J., Naik, V., Palmer, M., Plattner, G.-K., Rogelj, J., Rojas, M., Sillmann, J., Storelvmo, T., Thorne, P., Trewin, B., Achuta Rao, K., Adhikary, B., Allan, R., Armour, K., … Zickfeld, K. (2021). *Technical summary* (V. Masson-Delmotte, P. Zhai, A. Pirani, S. Connors, C. Péan, S. Berger, N. Caud, Y. Chen, L. Goldfarb, M. Gomis, M. Huang, K. Leitzell, E. Lonnoy, J. Matthews, T. Maycock, T. Waterfield, O. Yelekçi, R. Yu, & B. Zhou, Eds.). Cambridge University Press. https://doi.org/10.1017/9781009157896.002

Ávila, A., Justino, F., Wilson, A., Bromwich, D., & Amorim, M. (2016). Recent precipitation trends, flash floods and landslides in southern Brazil. *Environmental Research Letters*, *11*(11). https://doi.org/10.1088/1748-9326/11/11/114029

Badrinarayanan, V., Handa, A., & Cipolla, R. (2015). Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. *arXiv preprint*. https://doi.org/10.48550/arXiv.1505.07293

Balestriero, R., Pesenti, J., & LeCun, Y. (2021). Learning in high dimension always amounts to extrapolation. *arXiv preprint*. https://doi.org/10.48550/arXiv.2110.09485

Bates, P. D., & De Roo, A. P. (2000). A simple raster-based model for flood inundation simulation. *Journal of Hydrology*, *236*(1-2), 54–77. https://doi.org/10.1016/S0022-1694(00)00278-X

Bates, P. D., Horritt, M. S., & Fewtrell, T. J. (2010). A simple inertial formulation of the shallow water equations for efficient two-dimensional flood inundation modelling. *Journal of hydrology*, *387*(1-2), 33–45. https://doi.org/10.1016/j.jhydrol.2010.03.027

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. (2018). Relational inductive biases, deep learning, and graph networks. *arXiv*, 1–40. https://doi.org/t10.48550/arXiv.1806.01261

Bentivoglio, R., Isufi, E., Jonkman, S. N., & Taormina, R. (2022). Deep learning methods for flood mapping: A review of existing applications and future research directions. *Hydrology and Earth System Sciences*, *26*, 4345–4378. https://doi.org/10.5194/hess-26-4345-2022

Bentivoglio, R., Isufi, E., Jonkman, S. N., & Taormina, R. (2023). Rapid spatio-temporal flood modelling via hydraulics-based graph neural networks. *Hydrology and Earth System Sciences*, *27*(23), 4227–4246. https://doi.org/10.5194/hess-27-4227-2023

Bentivoglio, R., Isufi, E., Jonkman, S. N., & Taormina, R. (2025). Multi-scale hydraulic graph neural networks for flood modelling. *Natural Hazards and Earth System Sciences*, *25*(1), 335–351. https://doi.org/10.5194/nhess-25-335-2025

Berkhahn, S., Fuchs, L., & Neuweiler, I. (2019). An ensemble neural network model for real-time prediction of urban floods. *Journal of Hydrology*, *575*(May), 743–754. https://doi.org/10.1016/j.jhydrol.2019.05.066

Berkhahn, S., & Neuweiler, I. (2024). Data driven real-time prediction of urban floods with spatial and temporal distribution. *Journal of Hydrology X*, *22*, 100167. https://doi.org/10.1016/j.hydroa.2023.100167

Besseling, L. S., Bomers, A., Warmink, J. J., & Hulscher, S. J. (2025). A conceptual model to quantify probabilistic dike breach outflow. *Natural Hazards*, 1–29. https://doi.org/10.1007/s11069-025-07500-z

Beven, K. J. (2012). *Rainfall-runoff modelling: The primer*. John Wiley & Sons.

Bhunya, P., Panda, S., & Goel, M. (2011). Synthetic unit hydrograph methods: A critical review. *The Open Hydrology Journal*, *5*(1). https://doi.org/10.2174/1874378101105010001

Bobée, B., & Rasmussen, P. F. (1995). Recent advances in flood frequency analysis. *Reviews of Geophysics*, *33*(S2), 1111–1116. https://doi.org/10.1029/95RG00287

Bomers, A., Schielen, R. M., & Hulscher, S. J. (2019). The influence of grid shape and grid size on hydraulic river modelling performance. *Environmental Fluid Mechanics*, *19*(5), 1273–1294. https://doi.org/10.1007/s10652-019-09670-4

Bomers, A., Mathias, R., Schielen, J., & Hulscher, S. J. M. H. (2019). The influence of grid shape and grid size on hydraulic river modelling performance. *Environmental fluid mechanics*, *19*, 1273–1294. https://doi.org/10.1007/s10652-019-09670-4

Bonafilia, D., Tellman, B., Anderson, T., & Issenberg, E. (2020). Sen1floods11: A georeferenced dataset to train and test deep learning flood algorithms for sentinel-1. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Boon, M. J. J., & Witteveen+Bos. (2011). Veiligheid nederland in kaart 2 overstromingsrisico dijkring 15: Lopiker- en krimpenerwaard. https://www.helpdeskwater.nl/onderwerpen/waterveiligheid/programma-projecten/veiligheid-nederland/

Bowes, B. D., Tavakoli, A., Wang, C., Heydarian, A., Behl, M., Beling, P. A., & Goodall, J. L. (2021). Flood mitigation in coastal urban catchments using real-time stormwater infrastructure control and reinforcement learning. *Journal of Hydroinformatics*, *23*(3), 529–547. https://doi.org/10.2166/hydro.2020.080

Bradley, A. P. (1997). The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, *30*(7), 1145–1159. https://doi.org/10.1016/S0031-3203(96)00142-2

Brandstetter, J., Worrall, D., & Welling, M. (2022). Message passing neural pde solvers. *arXiv preprint*. https://doi.org/10.48550/arXiv.2202.03376

Bronstein, M. M., Bruna, J., Cohen, T., & Veličković, P. (2021). Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint*. https://doi.org/10.48550/arXiv.2104.13478

Brufau, P., Vázquez-Cendón, M., & García-Navarro, P. (2002). A numerical model for the flooding and drying of irregular domains. *International Journal for Numerical Methods in Fluids*, *39*(3), 247–275. https://doi.org/10.1002/fld.285

Burrichter, B., Hofmann, J., Koltermann da Silva, J., Niemann, A., & Quirmbach, M. (2023). A spatiotemporal deep learning approach for urban pluvial flood forecasting with multi-source data. *Water*, *15*(9), 1760. https://doi.org/10.3390/w15091760

Cache, T., Gomez, M. S., Beucler, T., Blagojevic, J., Leitão, J. P., & Peleg, N. (2024). Enhancing generalizability of data-driven urban flood models by incorporating contextual information. *Hydrology and Earth System Sciences Discussions*, *2024*, 1–23. https://doi.org/10.5194/hess-2024-63

Candy, A. S. (2017). A consistent approach to unstructured mesh generation for geophysical models. *arXiv preprint*. https://doi.org/10.48550/arXiv.1703.08491

Cao, X., Wang, B., Yao, Y., Zhang, L., Xing, Y., Mao, J., Zhang, R., Fu, G., Borthwick, A. G. L., & Qin, H. (2024). U-Rnn High-Resolution Spatiotemporal Nowcasting of Urban Flooding. https://doi.org/10.2139/ssrn.4935234

Caviedes-Voullième, D., Morales-Hernández, M., Norman, M. R., & Özgen-Xian, I. (2023). Serghei (serghei-swe) v1. 0: A performance-portable high-performance parallel-computing shallow-water solver for hydrology and environmental hydraulics. *Geoscientific Model Development*, *16*(3), 977–1008. https://doi.org/10.5194/gmd-16-977-2023

Chakrabortty, R., Chandra Pal, S., Rezaie, F., Arabameri, A., Lee, S., Roy, P., Saha, A., Chowdhuri, I., & Moayedi, H. (2021). Flash-flood hazard susceptibility mapping in kangsabati river basin, india. *Geocarto International*, 1–23. https://doi.org/10.1080/10106049.2021.1953618

Chakrabortty, R., Pal, S. C., Janizadeh, S., Santosh, M., Roy, P., Chowdhuri, I., & Saha, A. (2021). Impact of climate change on future flood susceptibility: An evaluation based on deep learning algorithms and gcm model. *Water Resources Management*, *35*(12), 4251–4274. https://doi.org/10.1007/s11269-021-02944-x

Chang, D.-L., Yang, S.-H., Hsieh, S.-L., Wang, H.-J., & Yeh, K.-C. (2020). Artificial intelligence methodologies applied to prompt pluvial flood estimation and prediction. *Water (Switzerland)*, *12*(12). https://doi.org/10.3390/w12123552

Chang, L.-C., Shen, H.-Y., Wang, Y.-F., Huang, J.-Y., & Lin, Y.-T. (2010). Clustering-based hybrid inundation model for forecasting flood inundation depths. *Journal of Hydrology*, *385*(1), 257–268. https://doi.org/10.1016/j.jhydrol.2010.02.028

Chen, J., Huang, G., & Chen, W. (2021). Towards better flood risk management: Assessing flood risk and investigating the potential mechanism based on machine learning models. *Journal of Environmental Management*, *293*, 112810. https://doi.org/10.1016/j.jenvman.2021.112810

Chen, L., Du, F., Hu, Y., Wang, Z., & Wang, F. (2023). Swinrdm: Integrate swinrnn with diffusion model towards high-resolution and high-quality weather forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, *37*(1), 322–330.

Chen, Y., Tang, X., Qi, X., Li, C.-G., & Xiao, R. (2022). Learning graph normalization for graph neural networks. *Neurocomputing*, *493*, 613–625. https://doi.org/10.1016/j.neucom.2022.01.003

Chicco, D., & Jurman, G. (2020). The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC genomics*, *21*(1), 1–13. https://doi.org/10.1186/s12864-019-6413-7

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint*. https://doi.org/10.48550/arXiv.1406.1078

Cho, K., van Merrienboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. https://doi.org/10.48550/arXiv.1409.1259

Chu, H., Wu, W., Wang, Q., Nathan, R., & Wei, J. (2020). An ann-based emulation modelling framework for flood inundation modelling: Application, challenges and future directions. *Environmental Modelling and Software*, *124*. https://doi.org/10.1016/j.envsoft.2019.104587

Cian, F., Marconcini, M., & Ceccato, P. (2018). Normalized difference flood index for rapid flood mapping: Taking advantage of eo big data. *Remote Sensing of Environment*, *209*, 712–730. https://doi.org/10.1016/j.rse.2018.03.006

Cortes, C., Mohri, M., & Syed, U. (2014). Deep boosting. *International conference on machine learning*, 1179–1187.

Costabile, P., Macchione, F., Natale, L., & Petaccia, G. (2015). Flood mapping using LIDAR DEM. Limitations of the 1-D modeling highlighted by the 2-D approach. *Natural Hazards*, *77*(1), 181–204. https://doi.org/10.1007/s11069-015-1606-0

Costabile, P., Costanzo, C., & Macchione, F. (2017). Performances and limitations of the diffusive approximation of the 2-d shallow water equations for flood simulation in urban and rural areas. *Applied Numerical Mathematics*, *116*, 141–156. https://doi.org/10.1016/j.apnum.2016.07.003

Costache, R., Ngo, P., & Bui, D. (2020). Novel ensembles of deep learning neural network and statistical learning for flash-flood susceptibility mapping. *Water (Switzerland)*, *12*(6).

Courant, R., Friedrichs, K., & Lewy, H. (1967). On the partial difference equations of mathematical physics. *IBM journal of Research and Development*, *11*(2), 215–234. https://doi.org/10.1147/rd.112.0215

Cunge, J., Holly, F., & Verwey, A. (1980). *Practical aspects of computational river hydraulics*. Pitman.

Darabi, H., Rahmati, O., Naghibi, S. A., Mohammadi, F., Ahmadisharaf, E., Kalantari, Z., Haghighi, A. T., Soleimanpour, S. M., Tiefenbacher, J. P., & Bui, D. T. (2022). Development of a novel hybrid multi-boosting neural network model for spatial prediction of urban flood. *Geocarto International*, *37*(19), 5716–5741. https://doi.org/10.1080/10106049.2021.1920629

De Moel, H., Asselman, N., & Aerts, J. (2012). Uncertainty and sensitivity analysis of coastal flood damage estimates in the west of the netherlands. *Natural Hazards and Earth system sciences*, *12*(4), 1045–1058.

de Brito, M. M., & Evers, M. (2016). Multi-criteria decision-making for flood risk management: A survey of the current state of the art. *Natural Hazards and Earth System Sciences*, *16*(4), 1019–1033. https://doi.org/10.5194/nhess-16-1019-2016

Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems* (pp. 1–10, Vol. 29). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2016/file/04df4d434d481c5bb723be1b6df1ee65-Paper.pdf

Delgado, R., & Tibau, X.-A. (2019). Why cohen's kappa should be avoided as performance measure in classification. *PloS one*, *14*(9), e0222916. https://doi.org/10.1371/journal.pone.0222916

Deltares. (2022). Delft3d-fm user manual [last accessed: 28/01/26]. https://content.oss.deltares.nl/delft3d/D-Flow_FM_User_Manual.pdf

Deltares. (2024). Meshkernel [last accessed: 28/01/26]. deltares.github.io/MeshKernel

de Moel, H., van Alphen, J., & Aerts, J. C. J. H. (2009). Flood maps in europe - methods, availability and use. *Natural Hazards and Earth System Sciences*, *9*(2), 289–301. https://doi.org/10.5194/nhess-9-289-2009

de Moel, H., Bouwer, L. M., & Aerts, J. C. (2014). Uncertainty and sensitivity of flood risk calculations for a dike ring in the south of the netherlands. *Science of the Total Environment*, *473*, 224–234. https://doi.org/10.1016/j.scitotenv.2013.12.015

de Moel, H., Jongman, B., Kreibich, H., Merz, B., Penning-Rowsell, E., & Ward, P. J. (2015). Flood risk assessments at different spatial scales. *Mitigation and Adaptation Strategies for Global Change*, *20*(6), 865–890. https://doi.org/10.1007/s11027-015-9654-z

den Heijer, F., & Kok, M. (2023). Assessment of ductile dike behavior as a novel flood risk reduction measure. *Risk Analysis*, *43*(9), 1779–1794. https://doi.org/10.1111/risa.14071

Destro, E., Amponsah, W., Nikolopoulos, E. I., Marchi, L., Marra, F., Zoccatelli, D., & Borga, M. (2018). Coupled prediction of flash flood response and debris flow occurrence: Application on an alpine extreme flood event. *Journal of Hydrology*, *558*, 225–237. https://doi.org/10.1016/j.jhydrol.2018.01.021

(DHPC), D. H. P. C. C. (2022). *DelftBlue Supercomputer (Phase 1)* [https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase1].

Di Baldassarre, G., Schumann, G., Bates, P. D., Freer, J. E., & Beven, K. J. (2010). Floodplain mapping: A critical discussion of deterministic and probabilistic approaches.

Hydrological Sciences Journal–Journal des Sciences Hydrologiques, *55*(3), 364–376. https://doi.org/10.1080/02626661003683389

Di Baldassarre, G., Kreibich, H., Vorogushyn, S., Aerts, J., Arnbjerg-Nielsen, K., Barendrecht, M., Bates, P., Borga, M., Botzen, W., Bubeck, P., et al. (2018). Hess opinions: An interdisciplinary research agenda to explore the unintended consequences of structural flood protection. *Hydrology and Earth System Sciences*, *22*(11), 5629–5637.

do Lago, C. A., Giacomoni, M. H., Bentivoglio, R., Taormina, R., Gomes, M. N., & Mendiondo, E. M. (2023). Generalizing rapid flood predictions to unseen urban catchments with conditional generative adversarial networks. *Journal of Hydrology*, 129276. https://doi.org/10.1016/j.jhydrol.2023.129276

do Lago, C., Brasil, J. A. T., Nóbrega Gomes, M., Mendiondo, E. M., & Giacomoni, M. H. (2024). Improving pluvial flood mapping resolution of large coarse models with deep learning. *Hydrological Sciences Journal*, (69(5)), 607–621. https://doi.org/10.1080/02626667.2024.2329268

Domeneghetti, A., Vorogushyn, S., Castellarin, A., Merz, B., & Brath, A. (2013). Probabilistic flood hazard mapping: Effects of uncertain boundary conditions. *Hydrology and Earth System Sciences*, *17*(8), 3127–3140. https://doi.org/10.5194/hess-17-3127-2013

Dong, S., Yu, T., Farahmand, H., & Mostafavi, A. (2021). A hybrid deep learning model for predictive flood warning and situation awareness using channel network sensors data. *Computer-Aided Civil and Infrastructure Engineering*, *36*(4), 402–420. https://doi.org/10.1111/mice.12629

D'Oria, M., & Maranzoni, A. (2019). Probabilistic assessment of flood hazard due to levee breaches using fragility functions. *Water Resources Research*, *55*(11), 8740–8764. https://doi.org/10.1029/2019WR025369

D'Oria, M., Mignosa, P., Tanda, M. G., & Todaro, V. (2022). Estimation of levee breach discharge hydrographs: Comparison of inverse approaches. *Hydrological Sciences Journal*, *67*(1), 54–64. https://doi.org/10.1080/02626667.2021.1996580

Dormand, J. R., & Prince, P. J. (1980). A family of embedded runge-kutta formulae. *Journal of computational and applied mathematics*, *6*(1), 19–26. https://doi.org/10.1016/0771-050X(80)90013-3

Dottori, F., & Todini, E. (2010). A 2D Flood Inundation Model Based on Cellular Automata Approach. *XVIII International Conference on Water Resources*, (2), 1–8.

Dottori, F., Alfieri, L., Bianchi, A., Skoien, J., & Salamon, P. (2021). A new dataset of river flood hazard maps for europe and the mediterranean basin region. *Earth System Science Data Discussions*, *2021*, 1–35. https://doi.org/10.5194/essd-2020-313

EuroHPC. (2025). Large unified modern infrastructure (lumi) [Online; accessed 05-August-2025]. https://www.lumi-supercomputer.eu/

European Union. (2007). Directive 2007/60/EC of the European Counil and European Parliment of 23 October 2007 on the assessment and management of flood risks. *Official Journal of the European Union*, (2455), 27–34. http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32007L0060&from=EN

Fang, Z., Wang, Y., Peng, L., & Hong, H. (2020). Predicting flood susceptibility using LSTM neural networks. *Journal of Hydrology*, (40), 125734. https://doi.org/10.1016/j.jhydrol.2020.125734

Ferrari, A., Dazzi, S., Vacondio, R., & Mignosa, P. (2020). Enhancing the resilience to flooding induced by levee breaches in lowland areas: A methodology based on numerical modelling. *Natural Hazards and Earth System Sciences*, *20*(1), 59–72. https://doi.org/10.5194/nhess-20-59-2020

Ferraro, D., Costabile, P., Costanzo, C., Petaccia, G., & Macchione, F. (2020). A spectral analysis approach for the a priori generation of computational grids in the 2-D hydrodynamic-based runoff simulations at a basin scale. *Journal of Hydrology*, *582*(December 2019), 124508. https://doi.org/10.1016/j.jhydrol.2019.124508

Fey, M., & Lenssen, J. E. (2019). Fast graph representation learning with pytorch geometric. *arXiv preprint*. https://doi.org/10.48550/arXiv.1903.02428

Finaud-Guyot, P., Delenne, C., Guinot, V., & Llovel, C. (2011). 1D-2D coupling for river flow modeling. *Comptes Rendus - Mecanique*, *339*(4), 226–234. https://doi.org/10.1016/j.crme.2011.02.001

Fortunato, M., Pfaff, T., Wirnsberger, P., Pritzel, A., & Battaglia, P. (2022). Multiscale meshgraphnets. *2nd AI4Science Workshop at the 39th International Conference on Machine Learning (ICML)*. https://doi.org/10.48550/arXiv.2210.00612

Gama, F., Isufi, E., Leus, G., & Ribeiro, A. (2020). Graphs, convolutions, and neural networks: From graph filters to graph neural networks. *IEEE Signal Processing Magazine*, *37*(6), 128–138.

Gao, H., & Ji, S. (2019). Graph u-nets. *international conference on machine learning*, 2083–2092. https://doi.org/10.48550/arXiv.1905.05178

Gao, W., Liao, Y., Chen, Y., Lai, C., He, S., & Wang, Z. (2024). Enhancing transparency in data-driven urban pluvial flood prediction using an explainable CNN model. *Journal of Hydrology*, *645*, 132228. https://doi.org/10.1016/j.jhydrol.2024.132228

Garzón, A., Kapelan, Z., Langeveld, J., & Taormina, R. (2024). Transferable and data efficient metamodeling of storm water system nodal depths using auto-regressive graph neural networks. *Water Research*, *266*, 122396. https://doi.org/10.1016/j.watres.2024.122396

Gebrehiwot, A., Hashemi-Beni, L., Thompson, G., Kordjamshidi, P., & Langan, T. E. (2019). Deep convolutional neural network for flood extent mapping using unmanned aerial vehicles data. *Sensors (Switzerland)*, *19*(7). https://doi.org/10.3390/s19071486

Gibbons, S. J., Lorito, S., Macías, J., Løvholt, F., Selva, J., Volpe, M., Sánchez-Linares, C., Babeyko, A., Brizuela, B., Cirella, A., Castro, M. J., de la Asunción, M., Lanucara, P., Glimsdal, S., Lorenzino, M. C., Nazaria, M., Pizzimenti, L., Romano, F., Scala, A., … Vöge, M. (2020). Probabilistic tsunami hazard analysis: High performance computing for massive scale inundation simulations. *Frontiers in Earth Science, Volume 8 - 2020*. https://doi.org/10.3389/feart.2020.591549

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural message passing for quantum chemistry. *International conference on machine learning*, 1263–1272.

Glenis, V., McGough, A. S., Kutija, V., Kilsby, C., & Woodman, S. (2013). Flood modelling for cities using cloud computing. *Journal of Cloud Computing: Advances, Systems and Applications*, *2*(1), 1–14. https://doi.org/10.1186/2192-113X-2-7

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning* [http://www.deeplearningbook.org]. MIT Press.

Guidolin, M., Chen, A. S., Ghimire, B., Keedwell, E. C., Djordjević, S., & Savić, D. A. (2016). A weighted cellular automata 2d inundation model for rapid flood analysis. *Environmental Modelling & Software*, *84*, 378–394. https://doi.org/10.1016/j.envsoft.2016.07.008

Guo, Z., Leitão, J. P., Simões, N. E., & Moosavi, V. (2020). Data-driven flood emulation: Speeding up urban flood predictions by deep convolutional neural networks. *Journal of Flood Risk Management*, (October 2020), 1–14. https://doi.org/10.1111/jfr3.12684

Guo, Z., Moosavi, V., & Leitão, J. P. (2022). Data-driven rapid flood prediction mapping with catchment generalizability. *Journal of Hydrology*, *609*, 127726. https://doi.org/10.1016/J.JHYDROL.2022.127726

Hall, J., & Solomatine, D. (2008). A framework for uncertainty analysis in flood risk management decisions. *International Journal of River Basin Management*, *6*(2), 85–98. https://doi.org/10.1080/15715124.2008.9635339

Hall, J. W., Manning, L. J., & Hankin, R. K. (2011). Bayesian calibration of a flood inundation model using spatial data. *Water Resources Research*, *47*(5). https://doi.org/10.1029/2009WR008541

Hashemi-Beni, L., & Gebrehiwot, A. A. (2021). Flood extent mapping: An integrated method using deep learning and region growing using uav optical data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *14*, 2127–2135. https://doi.org/10.1109/JSTARS.2021.3051873

He, J., Zhang, L., Xiao, T., Wang, H., & Luo, H. (2023). Deep learning enables super-resolution hydrodynamic flooding process modeling under spatiotemporally varying rainstorms. *Water Research*, *239*, 120057. https://doi.org/10.1016/j.watres.2023.120057

Hegnauer, M., Beersma, J., Van den Boogaard, H., Buishand, T., & Passchier, R. (2014). Generator of rainfall and discharge extremes (grade) for the rhine and meuse basins. final report of grade 2.0 [last accessed: 28/01/26]. https://publications.deltares.nl/1209424_004_0018.pdf

Hess, L., Melack, J., Filoso, S., & Wang, Y. (1995). Delineation of inundated area and vegetation along the amazon floodplain with the sir-c synthetic aperture radar. *IEEE Transactions on Geoscience and Remote Sensing*, *33*(4), 896–904. https://doi.org/10.1109/36.406675

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Horie, M., & Mitsume, N. (2022). Physics-embedded neural networks: E($\boldsymbol{n}$)-equivariant graph neural pde solvers. https://doi.org/10.48550/ARXIV.2205.11912

Hosseiny, H. (2021). A deep learning model for predicting river flood depth and extent. *Environmental Modelling & Software*, *145*, 105186. https://doi.org/10.1016/j.envsoft.2021.105186

Hou, J., Li, X., Bai, G., Wang, X., Zhang, Z., Yang, L., Du, Y., Ma, Y., Fu, D., & Zhang, X. (2021). A deep learning technique based flood propagation experiment. *Journal of Flood Risk Management*, *14*(3), e12718. https://doi.org/10.1111/jfr3.12718

Hu, R., Fang, F., Pain, C., & Navon, I. (2019). Rapid spatio-temporal flood prediction and uncertainty quantification using a deep learning method. *Journal of Hydrology*, *575*, 911–920. https://doi.org/10.1016/j.jhydrol.2019.05.087

Huang, P. C., Hsu, K. L., & Lee, K. T. (2021). Improvement of Two-Dimensional Flow-Depth Prediction Based on Neural Network Models By Preprocessing Hydrological and Geomorphological Data. *Water Resources Management*, 1079–1100. https://doi.org/10.1007/s11269-021-02776-9

Huang, T., & Merwade, V. (2024). Beyond a fixed number: Investigating uncertainty in popular evaluation metrics of ensemble flood modeling using bootstrapping analysis. *Journal of Flood Risk Management*, *17*(2), e12982. https://doi.org/10.1111/jfr3.12982

Hunter, N. M., Horritt, M. S., Bates, P. D., Wilson, M. D., & Werner, M. G. (2005). An adaptive time step solution for raster-based storage cell modelling of floodplain inundation. *Advances in Water Resources*, *28*(9), 975–991. https://doi.org/10.1016/j.advwatres.2005.03.007

Hunter, N. M., Bates, P. D., Horritt, M. S., & Wilson, M. D. (2007). Simple spatially-distributed models for predicting flood inundation: A review. *Geomorphology*, *90*(3-4), 208–225. https://doi.org/10.1016/j.geomorph.2006.10.021

Ichim, L., & Popescu, D. (2020). Segmentation of vegetation and flood from aerial images based on decision fusion of neural networks. *Remote Sensing*, *12*(15). https://doi.org/10.3390/rs12152490

Ireland, G., Volpi, M., & Petropoulos, G. P. (2015). Examining the capability of supervised machine learning classifiers in extracting flooded areas from landsat tm imagery: A case study from a mediterranean flood. *Remote Sensing*, *7*(3), 3372–3399. https://doi.org/10.3390/rs70303372

Isikdogan, F., Bovik, A. C., & Passalacqua, P. (2017). Surface water mapping by deep learning. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *10*(11), 4909–4918. https://doi.org/10.1109/JSTARS.2017.2735443

Jacquier, P., Abdedou, A., Delmas, V., & Soulaïmani, A. (2021). Non-intrusive reduced-order modeling using uncertainty-aware deep neural networks and proper orthogonal decomposition: Application to flood modeling. *Journal of Computational Physics*, *424*, 109854. https://doi.org/10.1016/j.jcp.2020.109854

Jahangir, M., Mousavi Reineh, S., & Abolghasemi, M. (2019). Spatial predication of flood zonation mapping in kan river basin, iran, using artificial neural network algorithm. *Weather and Climate Extremes*, *25*. https://doi.org/10.1016/j.wace.2019.100215

Jongejan, R., & Maaskant, B. (2015). Quantifying flood risks in the netherlands. *Risk Analysis*, *35*(2), 252–264. https://doi.org/10.1111/risa.12285

Jonkman, S. N., Kok, M., & Vrijling, J. K. (2008). Flood risk assessment in the netherlands: A case study for dike ring south holland. *Risk Analysis: An International Journal*, *28*(5), 1357–1374.

Jonkman, S., & Vrijling, J. (2008). Loss of life due to floods. *Journal of Flood Risk Management*, *1*(1), 43–56. https://doi.org/10.1111/j.1753-318x.2008.00006.x

Kabir, S., Patidar, S., Xia, X., Liang, Q., Neal, J., & Pender, G. (2020). A deep convolutional neural network model for rapid prediction of fluvial flood inundation. *Journal of Hydrology*, *590*(September), 125481. https://doi.org/10.1016/j.jhydrol.2020.125481

Kalantar, B., Ueda, N., Saeidi, V., Janizadeh, S., Shabani, F., Ahmadi, K., & Shabani, F. (2021). Deep neural network utilizing remote sensing datasets for flood hazard susceptibility mapping in brisbane, australia. *Remote Sensing*, *13*(13). https://doi.org/10.3390/rs13132638

Kamrath, P., Disse, M., Hammer, M., & Köngeter, J. (2006). Assessment of discharge through a dike breach and simulation of flood wave propagation. *Natural Hazards*, *38*(1-2), 63–78. https://doi.org/10.1007/s11069-005-8600-x

Kang, W., Xiang, Y., Wang, F., Wan, L., & You, H. (2018). Flood detection in gaofen-3 sar images via fully convolutional networks. *Sensors*, *18*(9). https://doi.org/10.3390/s18092915

Kao, I. F., Liou, J. Y., Lee, M. H., & Chang, F. J. (2021). Fusing stacked autoencoder and long short-term memory for regional multistep-ahead flood inundation forecasts. *Journal of Hydrology*, (October 2020), 126371. https://doi.org/10.1016/j.jhydrol.2021.126371

Kazadi, A., Doss-Gollin, J., & da Silva, A. L. (2024). Pluvial flood emulation with hydraulics-informed message passing. *Forty-first International Conference on Machine Learning*. https://openreview.net/forum?id=kIHIA6Lr0B

Kazakis, N., Kougias, I., & Patsialis, T. (2015). Assessment of flood hazard areas at a regional scale using an index-based approach and analytical hierarchy process: Application in rhodope–evros region, greece. *Science of The Total Environment*, *538*, 555–563. https://doi.org/10.1016/j.scitotenv.2015.08.055

Khoirunisa, N., Ku, C.-Y., & Liu, C.-Y. (2021). A gis-based artificial neural network model for flood susceptibility assessment. *International Journal of Environmental Research and Public Health*, *18*(3), 1–20. https://doi.org/10.3390/ijerph18031072

Khosravi, K., Panahi, M., Golkarian, A., Keesstra, S. D., & Saco, P. M. (2020). Convolutional neural network approach for spatial prediction of flood hazard at national scale of Iran. *Journal of Hydrology*, *591*(August), 125552. https://doi.org/10.1016/j.jhydrol.2020.125552

Kia, M. B., Pirasteh, S., Pradhan, B., Mahmud, A. R., Sulaiman, W. N. A., & Moradi, A. (2012). An artificial neural network model for flood simulation using gis: Johor river basin, malaysia. *Environmental earth sciences*, *67*(1), 251–264.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Knowles, S. (2021). Graphcore. *2021 IEEE Hot Chips 33 Symposium (HCS)*, 1–25. graphcore.ai

Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P., & Hoyer, S. (2021). Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, *118*(21), e2101784118. https://doi.org/10.1073/pnas.2101784118

Kochkov, D., Yuval, J., Langmore, I., Norgaard, P., Smith, J., Mooers, G., Klöwer, M., Lottes, J., Rasp, S., Düben, P., Hatfield, S., Battaglia, P., Sanchez-Gonzalez, A., Willson, M., Brenner, M. P., & Hoyer, S. (2024). Neural general circulation models for weather

and climate. *Nature*, *632*(8027), 1060–1066. https://doi.org/10.1038/s41586-024-07744-y

Kourgialas, N. N., & Karatzas, G. P. (2017). A national scale flood hazard mapping methodology: The case of greece – protection and adaptation policy approaches. *Science of The Total Environment*, *601-602*, 441–452. https://doi.org/10.1016/j.scitotenv.2017.05.197

Kratzert, F., Herrnegger, M., Klotz, D., Hochreiter, S., & Klambauer, G. (2019). NeuralHydrology – Interpreting LSTMs in Hydrology. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11700 LNCS*, 347–362. https://doi.org/10.1007/978-3-030-28954-6{_}19

Kratzert, F., Klotz, D., Herrnegger, M., Sampson, A. K., Hochreiter, S., & Nearing, G. S. (2019). Toward Improved Predictions in Ungauged Basins: Exploiting the Power of Machine Learning. *Water Resources Research*, *55*(12), 11344–11354. https://doi.org/10.1029/2019WR026065

Kreibich, H., Piroth, K., Seifert, I., Maiwald, H., Kunert, U., Schwarz, J., Merz, B., & Thieken, A. (2009). Is flow velocity a significant parameter in flood damage modelling? *Natural Hazards and Earth System Sciences*, *9*(5), 1679–1692. https://doi.org/10.5194/nhess-9-1679-2009

Kummu, M., De Moel, H., Ward, P. J., & Varis, O. (2011). How close do we live to water? a global analysis of population distance to freshwater bodies. *PloS one*, *6*(6), e20578. https://doi.org/10.1371/journal.pone.0020578

Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Alet, F., Ravuri, S., Ewalds, T., Eaton-Rosen, Z., Hu, W., et al. (2023). Learning skillful medium-range global weather forecasting. *Science*, *382*(6677), 1416–1421. https://doi.org/DOI:10.1126/science.adi2336

Lane, S. N., Bradbrook, K. F., Richards, K. S., Biron, P. A., & Roy, A. G. (1999). The application of computational fluid dynamics to natural river channels: Three-dimensional versus two-dimensional approaches. *Geomorphology*, *29*(1-2), 1–20. https://doi.org/10.1016/S0169-555X(99)00003-3

Lang, S., Alexe, M., Chantry, M., Dramsch, J., Pinault, F., Raoult, B., Clare, M. C., Lessig, C., Maier-Gerber, M., Magnusson, L., et al. (2024). Aifs–ecmwf's data-driven forecasting system. *arXiv preprint*. https://doi.org/10.48550/arXiv.2406.01465

Leandro, J., Chen, A. S., Djordjević, S., & Savić, D. A. (2009). Comparison of 1d/1d and 1d/2d coupled (sewer/surface) hydraulic models for urban flood simulation. *Journal of hydraulic engineering*, *135*(6), 495–504. https://doi.org/10.1061/(ASCE)HY.1943-7900.0000037

LeCun, Y., Bengio, Y., et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, *3361*(10), 1995. https://doi.org/10.5555/303568.303704

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Lei, X., Chen, W., Panahi, M., Falah, F., Rahmati, O., Uuemaa, E., Kalantari, Z., Ferreira, C. S. S., Rezaie, F., Tiefenbacher, J. P., et al. (2021). Urban flood modeling using

deep-learning approaches in seoul, south korea. *Journal of Hydrology*, *601*, 126684. https://doi.org/10.1016/j.jhydrol.2021.126684

LeVeque, R. J., et al. (2002). *Finite volume methods for hyperbolic problems* (Vol. 31). Cambridge university press.

Lhomme, J., Sayers, P., Gouldby, B., Wills, M., & Mulet-Marti, J. (2008). Recent development and application of a rapid flood spreading method. *Flood Risk Management: Research and Practice*, (October), 15–24. https://doi.org/10.1201/9780203883020.ch2

Li, L., Chen, Y., Xu, T., Huang, C., Liu, R., & Shi, K. (2016). Integration of bayesian regulation back-propagation neural network and particle swarm optimization for enhancing sub-pixel mapping of flood inundation in river basins. *Remote Sensing Letters*, *7*(7), 631–640. https://doi.org/10.1080/2150704X.2016.1177238

Li, L., Xu, T., & Chen, Y. (2016). Improved urban flooding mapping from remote sensing images using generalized regression neural network-based super-resolution algorithm. *Remote Sensing*, *8*(8). https://doi.org/10.3390/rs8080625

Li, L., Chen, Y., Xu, T., Liu, R., Shi, K., & Huang, C. (2015). Super-resolution mapping of wetland inundation from remote sensing imagery based on integration of back-propagation neural network and genetic algorithm. *Remote Sensing of Environment*, *164*, 142–154. https://doi.org/10.1016/j.rse.2015.04.009

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2020). Fourier neural operator for parametric partial differential equations. *arXiv preprint*. https://doi.org/10.48550/arXiv.2010.08895

Liang, Q., & Marche, F. (2009). Numerical resolution of well-balanced shallow water equations with complex source terms. *Advances in water resources*, *32*(6), 873–884. https://doi.org/10.1016/j.advwatres.2009.02.010

Liao, Y., Wang, Z., Chen, X., & Lai, C. (2023). Fast simulation and prediction of urban pluvial floods using a deep convolutional neural network model. *Journal of Hydrology*, *624*, 129945. https://doi.org/10.1016/j.jhydrol.2023.129945

Lin, Q., Leandro, J., Gerber, S., & Disse, M. (2020). Multistep flood inundation forecasts with resilient backpropagation neural networks: Kulmbach case study. *Water (Switzerland)*, *12*(12). https://doi.org/10.3390/w12123568

Lin, Q., Leandro, J., Wu, W., Bhola, P., & Disse, M. (2020). Prediction of maximum flood inundation extents with resilient backpropagation neural network: Case study of kulmbach. *Frontiers in Earth Science*, *8*. https://doi.org/10.3389/feart.2020.00332

Lino, M., Cantwell, C., Bharath, A. A., & Fotiadis, S. (2021). Simulating continuum mechanics with multi-scale graph neural networks. *arXiv preprint*. https://doi.org/10.48550/arXiv.2106.04900

Lino, M., Fotiadis, S., Bharath, A. A., & Cantwell, C. D. (2022). Multi-scale rotation-equivariant graph neural networks for unsteady eulerian fluid dynamics. *Physics of Fluids*, *34*(8), 087110. https://doi.org/10.1063/5.0097679

Liu, B., Li, X., & Zheng, G. (2019). Coastal inundation mapping from bitemporal and dual-polarization sar imagery based on deep convolutional neural networks. *Journal of Geophysical Research: Oceans*, *124*(12), 9101–9113. https://doi.org/10.1029/2019JC015577

Liu, J., Wang, J., Xiong, J., Cheng, W., Sun, H., Yong, Z., & Wang, N. (2021). Hybrid models incorporating bivariate statistics and machine learning methods for flash flood susceptibility assessment based on remote sensing datasets. *Remote Sensing, 13*(23), 4945. https://doi.org/10.3390/rs13234945

Liu, Q., Qin, Y., & Li, G. (2018). Fast simulation of large-scale floods based on gpu parallel computing. *Water, 10*(5), 589. https://doi.org/10.3390/w10050589

Liu, Y., Kutz, J. N., & Brunton, S. L. (2022). Hierarchical deep learning of multiscale differential equation time-steppers. *Philosophical Transactions of the Royal Society A, 380*(2229), 20210200. https://doi.org/10.1098/rsta.2021.0200

Löwe, R., Böhm, J., Jensen, D. G., Leandro, J., & Rasmussen, S. H. (2021). U-flood – topographic deep learning for predicting urban pluvial flood water depth. *Journal of Hydrology, 603*, 126898. https://doi.org/10.1016/j.jhydrol.2021.126898

Lütjens, B., Leshchinskiy, B., Requena-Mesa, C., Chishtie, F., Díaz-Rodriguez, N., Boulais, O., Piña, A., Newman, D., Lavin, A., Gal, Y., et al. (2020). Physics-informed gans for coastal flood visualization. *arXiv preprint arXiv:2010.08103*.

Lütjens, B., Leshchinskiy, B., Requena-Mesa, C., Chishtie, F., Díaz-Rodríguez, N., Boulais, O., Sankaranarayanan, A., Piña, A., Gal, Y., Raïssi, C., et al. (2021). Physically-consistent generative adversarial networks for coastal flood visualization. *arXiv preprint arXiv:2104.04785*.

Ma, X., Hong, Y., Song, Y., & Chen, Y. (2019). A super-resolution convolutional-neural-network-based approach for subpixel mapping of hyperspectral images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 12*(12), 4930–4939. https://doi.org/10.1109/JSTARS.2019.2941089

Maeland, E. (1988). On the comparison of interpolation methods. *IEEE transactions on medical imaging, 7*(3), 213–217. https://doi.org/10.1109/42.7784

Mahmoud, S. H., & Gan, T. Y. (2018). Multi-criteria approach to develop flood susceptibility maps in arid regions of middle east. *Journal of Cleaner Production, 196*, 216–229. https://doi.org/10.1016/j.jclepro.2018.06.047

Manjusree, P., Kumar, L. P., Bhatt, C. M., Rao, G. S., & Bhanumurthy, V. (2012). Optimization of threshold ranges for rapid flood inundation mapping by evaluating backscatter profiles of high incidence angle sar images. *International Journal of Disaster Risk Science, 3*(2), 113–122. https://doi.org/10.1007/s13753-012-0011-5

Martínez-Aranda, S., Fernández-Pato, J., Echeverribar, I., Navas-Montilla, A., Morales-Hernández, M., Brufau, P., Murillo, J., & García-Navarro, P. (2022). Finite volume models and efficient simulation tools (est) for shallow flows. In *Advances in fluid mechanics* (pp. 67–137). Springer. https://doi.org/10.1007/978-981-19-1438-6_3

Martinis, S., Twele, A., & Voigt, S. (2009). Towards operational near real-time flood detection using a split-based automatic thresholding procedure on high resolution terrasar-x data. *Natural Hazards and Earth System Sciences, 9*(2), 303–314. https://doi.org/10.5194/nhess-9-303-2009

Mazzoleni, M., Bacchi, B., Barontini, S., Di Baldassarre, G., Pilotti, M., & Ranzi, R. (2014). Flooding hazard mapping in floodplain areas affected by piping breaches in the po river, italy. *Journal of Hydrologic Engineering, 19*(4), 717–731. https://doi.org/10.1061/(ASCE)HE.1943-5584.0000840

Meraner, A., Ebel, P., Zhu, X. X., & Schmitt, M. (2020). Cloud removal in sentinel-2 imagery using a deep residual neural network and sar-optical data fusion. *ISPRS Journal of Photogrammetry and Remote Sensing, 166*, 333–346. https://doi.org/10.1016/j.isprsjprs.2020.05.013

Ming, X., Liang, Q., Xia, X., Li, D., & Fowler, H. J. (2020). Real-Time Flood Forecasting Based on a High-Performance 2-D Hydrodynamic Model and Numerical Weather Predictions. *Water Resources Research, 56*(7). https://doi.org/10.1029/2019WR025583

Morales-Hernández, M., Petaccia, G., Brufau, P., & García-Navarro, P. (2016). Conservative 1d–2d coupled numerical strategies applied to river flooding: The tiber (rome). *Applied Mathematical Modelling, 40*(3), 2087–2105. https://doi.org/10.1016/j.apm.2015.08.016

Muñoz, D., Muñoz, P., Moftakhari, H., & Moradkhani, H. (2021). From local to regional compound flood mapping with deep learning and data fusion techniques. *Science of the Total Environment, 782*. https://doi.org/10.1016/j.scitotenv.2021.146927

Neal, J. C., Fewtrell, T. J., Bates, P. D., & Wright, N. G. (2010). A comparison of three parallelisation methods for 2d flood inundation models. *Environmental Modelling & Software, 25*(4), 398–411. https://doi.org/10.1016/j.envsoft.2009.11.007

Nemni, E., Bullock, J., Belabbes, S., & Bromley, L. (2020). Fully convolutional neural network for rapid flood segmentation in synthetic aperture radar imagery. *Remote Sensing, 12*(16). https://doi.org/10.3390/rs12162532

Neumann, B., Vafeidis, A. T., Zimmermann, J., & Nicholls, R. J. (2015). Future coastal population growth and exposure to sea-level rise and coastal flooding-a global assessment. *PloS one, 10*(3), e0118571. https://doi.org/10.1371/journal.pone.0118571

Ngo, P. T. T., Hoang, N. D., Pradhan, B., Nguyen, Q. K., Tran, X. T., Nguyen, Q. M., Nguyen, V. N., Samui, P., & Bui, D. T. (2018). A novel hybrid swarm optimized multilayer neural network for spatial prediction of flash floods in tropical areas using sentinel-1 SAR imagery and geospatial data. *Sensors (Switzerland), 18*(11). https://doi.org/10.3390/s18113704

Nobre, A., Cuartas, L., Hodnett, M., Rennó, C., Rodrigues, G., Silveira, A., Waterloo, M., & Saleska, S. (2011). Height above the nearest drainage – a hydrologically relevant new terrain model. *Journal of Hydrology, 404*(1), 13–29. https://doi.org/10.1016/j.jhydrol.2011.03.051

Nogueira, K., Fadel, S. G., Dourado, Í. C., de O. Werneck, R., Muñoz, J. A. V., Penatti, O. A. B., Calumby, R. T., Li, L. T., dos Santos, J. A., & Torres, R. d. S. (2018). Exploiting convnet diversity for flooding identification. *IEEE Geoscience and Remote Sensing Letters, 15*(9), 1446–1450. https://doi.org/10.1109/LGRS.2018.2845549

Palmitessa, R., Grum, M., Engsig-Karup, A. P., & Löwe, R. (2022). Accelerating hydrodynamic simulations of urban drainage systems with physics-guided machine learning. *Water Research, 223*, 118972. https://doi.org/10.1016/j.watres.2022.118972

Panahi, M., Jaafari, A., Shirzadi, A., Shahabi, H., Rahmati, O., Omidvar, E., Lee, S., & Bui, D. (2021). Deep learning neural networks for spatially explicit prediction of flash flood probability. *Geoscience Frontiers, 12*(3). https://doi.org/10.1016/j.gsf.2020.09.007

Papaioannou, G., Vasiliades, L., Loukas, A., & Aronica, G. T. (2017). Probabilistic flood inundation mapping at ungauged streams due to roughness coefficient uncertainty in hydraulic modelling. *Advances in Geosciences*, *44*, 23–34. https://doi.org/10.5194/adgeo-44-23-2017

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett, Eds.). *Advances in neural information processing systems*, *32*. https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf

Peng, B., Meng, Z., Huang, Q., & Wang, C. (2019). Patch similarity convolutional neural network for urban flood extent mapping using bi-temporal satellite multispectral imagery. *Remote Sensing*, *11*(21). https://doi.org/10.3390/rs11212492

Peng, J.-Z., Wang, Y.-Z., Chen, S., Chen, Z.-H., Wu, W.-T., & Aubry, N. (2022). Grid adaptive reduced-order model of fluid flow based on graph convolutional neural network. *Physics of Fluids*, *34*(8), 087121. https://doi.org/10.1063/5.0100236

Perlin, K. (2002). Improving noise. *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, 681–682. https://doi.org/10.1145/566570.566636

Petaccia, G., Leporati, F., & Torti, E. (2016). OpenMP and CUDA simulations of Sella Zerbino Dam break on unstructured grids. *Computational Geosciences*, *20*(5), 1123–1132. https://doi.org/10.1007/s10596-016-9580-5

Petaccia, G., Natale, L., Savi, F., Velickovic, M., Zech, Y., & Soares-Frazão, S. (2013). Flood wave propagation in steep mountain rivers. *Journal of Hydroinformatics*, *15*(1), 120–137. https://doi.org/10.2166/hydro.2012.122

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., & Battaglia, P. (2021). Learning mesh-based simulation with graph networks. *International Conference on Learning Representations*. https://openreview.net/forum?id=roNqYL0_XP

Pham, B. T., Luu, C., Van Phong, T., Trinh, P. T., Shirzadi, A., Renoud, S., Asadi, S., Van Le, H., von Meding, J., & Clague, J. J. (2021). Can deep learning algorithms outperform benchmark machine learning algorithms in flood susceptibility modeling? *Journal of Hydrology*, *592*, 125615. https://doi.org/10.1016/j.jhydrol.2020.125615

Pianforini, M., Dazzi, S., Pilzer, A., & Vacondio, R. (2024). Real-time flood maps forecasting for dam-break scenarios with a transformer-based deep learning model. *Journal of Hydrology*, *635*, 131169. https://doi.org/10.1016/j.jhydrol.2024.131169

Pianforini, M., Dazzi, S., Pilzer, A., & Vacondio, R. (2025). FloodSformer: A transformer-based data-driven model for predicting the 2-D dynamics of fluvial floods. *Environmental Modelling & Software*, *193*, 106599. https://doi.org/10.1016/j.envsoft.2025.106599

Popa, M., Peptenatu, D., Draghici, C., & Diaconu, D. (2019). Flood hazard mapping using the flood and flash-flood potential index in the buzau river catchment, romania. *Water (Switzerland)*, *11*(10). https://doi.org/10.3390/w11102116

Prestininzi, P. (2008). Suitability of the diffusive model for dam break simulation: Application to a cadam experiment. *Journal of Hydrology*, *361*(1-2), 172–185. https://doi.org/10.1016/j.jhydrol.2008.07.050

Raissi, M., Perdikaris, P., & Karniadakis, G. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, *378*, 686–707. https://doi.org/10.1016/j.jcp.2018.10.045

Rijkswaterstaat. (2016). The national flood risk analysis for the netherlands : Final report [Online; accessed 03-July-2024]. https://www.helpdeskwater.nl/onderwerpen/waterveiligheid/programma-projecten/veiligheid-nederland/

Roe, P. L. (1981). Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, *135*(2), 250–258. https://doi.org/10.1006/jcph.1997.5705

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, 234–241.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

Sabbaqi, M., & Isufi, E. (2023). Graph-time convolutional neural networks: Architecture and theoretical analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *45*(12), 14625–14638. https://doi.org/10.1109/TPAMI.2023.3311912

Saeed, M., Li, H., Ullah, S., Rahman, A.-u., Ali, A., Khan, R., Hassan, W., Munir, I., & Alam, S. (2021). Flood hazard zonation using an artificial neural network model: A case study of kabul river basin, pakistan. *Sustainability*, *13*(24), 13953. https://doi.org/10.3390/su132413953

Sarker, C., Mejias, L., Maire, F., & Woodley, A. (2019). Flood mapping with convolutional neural networks using spatio-contextual pixel information. *Remote Sensing*, *11*(19). https://doi.org/10.3390/rs11192331

Savage, J. T. S., Bates, P., Freer, J., Neal, J., & Aronica, G. (2016). When does spatial resolution become spurious in probabilistic flood inundation predictions? *Hydrological Processes*, *30*(13), 2014–2032. https://doi.org/10.1002/hyp.10749

Schanze, J. (2006). Flood risk management–a basic framework. In *Flood risk management: Hazards, vulnerability and mitigation measures* (pp. 1–20). Springer. https://doi.org/10.1007/978-1-4020-4598-1_1

Schmitz, V., Erpicum, S., El kadi Abderrezzak, K., Rifai, I., Archambeau, P., Pirotton, M., & Dewals, B. (2021). Overtopping-induced failure of non–cohesive homogeneous fluvial dikes: Effect of dike geometry on breach discharge and widening. *Water Resources Research*, *57*(7), e2021WR029660. https://doi.org/10.1029/2021WR029660

Serinaldi, F., Loecker, F., Kilsby, C. G., & Bast, H. (2018). Flood propagation and duration in large river basins: a data-driven analysis for reinsurance purposes. *Natural Hazards*, *94*(1), 71–92. https://doi.org/10.1007/s11069-018-3374-0

Shao, Y., Chen, J., Zhang, T., Yu, T., & Chu, S. (2024). Advancing rapid urban flood prediction: A spatiotemporal deep learning approach with uneven rainfall and attention

mechanism. *Journal of Hydroinformatics*, *26*(6), 1409–1424. https://doi.org/10.2166/hydro.2024.024

Shirzadi, A., Asadi, S., Shahabi, H., Ronoud, S., Clague, J. J., Khosravi, K., Pham, B. T., Ahmad, B. B., & Bui, D. T. (2020). A novel ensemble learning based on bayesian belief network coupled with an extreme learning machine for flash flood susceptibility mapping. *Engineering Applications of Artificial Intelligence*, *96*, 103971. https://doi.org/10.1016/j.engappai.2020.103971

Shustikova, I., Neal, J. C., Domeneghetti, A., Bates, P. D., Vorogushyn, S., & Castellarin, A. (2020). Levee breaching: A new extension to the lisflood-fp model. *Water*, *12*(4), 942. https://doi.org/10.3390/w12040942

Sikorska, A. E., Viviroli, D., & Seibert, J. (2015). Flood-type classification in mountainous catchments using crisp and fuzzy decision trees. *Journal of the American Water Resources Association*, *5*(3), 2–2. https://doi.org/10.1111/j.1752-1688.1969.tb04897.x

Sit, M., Demiray, B. Z., Xiang, Z., Ewing, G. J., Sermet, Y., & Demir, I. (2020). A comprehensive review of deep learning applications in hydrology and water resources. *Water Science and Technology*, *82*(12), 2635–2670. https://doi.org/10.2166/wst.2020.369

Song, S., Schmalz, B., Zhang, J., Li, G., & Fohrer, N. (2016). Application of modified manning formula in the determination of vertical profile velocity in natural rivers. *Hydrology Research*, *48*(1), 133–146. https://doi.org/10.2166/nh.2016.131

Song, W., Guan, M., & Yu, D. (2025). Swinflood: A hybrid cnn-swin transformer model for rapid spatiotemporal flood simulation. *Journal of Hydrology*, *660*. https://doi.org/10.1016/j.jhydrol.2025.133280

Sridharan, B., Bates, P. D., Sen, D., & Kuiry, S. N. (2021). Local-inertial shallow water model on unstructured triangular grids. *Advances in Water Resources*, *152*(April), 103930. https://doi.org/10.1016/j.advwatres.2021.103930

Syifa, M., Park, S. J., Achmad, A. R., Lee, C.-W., & Eom, J. (2019). Flood mapping using remote sensing imagery and artificial intelligence techniques: A case study in brumadinho, brazil. *Journal of Coastal Research*, *90*(SI), 197–204. https://doi.org/10.2112/SI90-024.1

Taormina, R., & Galelli, S. (2018). Deep-learning approach to the detection and localization of cyber-physical attacks on water distribution systems. *Journal of Water Resources Planning and Management*, *144*(10), 04018065. https://doi.org/10.1061/(ASCE)WR.1943-5452.0000983

Tayefi, V., Lane, S., Hardy, R., & Yu, D. (2007). A comparison of one-and two-dimensional approaches to modelling flood inundation over complex upland floodplains. *Hydrological Processes: An International Journal*, *21*(23), 3190–3202. https://doi.org/10.1002/hyp.6523

Tehrany, M. S., Lee, M.-J., Pradhan, B., Jebur, M. N., & Lee, S. (2014). Flood susceptibility mapping using integrated bivariate and multivariate statistical models. *Environmental earth sciences*, *72*(10), 4001–4015. https://doi.org/10.1007/s12665-014-3289-3

Teng, J., Jakeman, A. J., Vaze, J., Croke, B. F., Dutta, D., & Kim, S. (2017). Flood inundation modelling: A review of methods, recent advances and uncertainty analysis. *Envi-*

ronmental Modelling and Software, 90, 201–216. https://doi.org/10.1016/j.envsoft.2017.01.006

Tien, D., Hoang, N.-d., Martínez-álvarez, F., Ngo, P.-t. T., Viet, P., Dat, T., Samui, P., & Costache, R. (2020). A novel deep learning neural network approach for predicting flash flood susceptibility : A case study at a high frequency tropical storm area. *Science of the Total Environment, 701*, 134413. https://doi.org/10.1016/j.scitotenv.2019.134413

Toro, E. F. (2013). *Riemann solvers and numerical methods for fluid dynamics: A practical introduction*. Springer Science & Business Media.

Toro, E. F. (2024). *Computational algorithms for shallow water equations*. Springer. https://doi.org/10.1007/978-3-031-61395-1

Vacondio, R., Rogers, B., Stansby, P., & Mignosa, P. (2012). Sph modeling of shallow flow with open boundaries for practical flood simulation. *Journal of Hydraulic Engineering, 138*(6), 530–541. https://doi.org/10.1061/(ASCE)HY.1943-7900.0000543

Van den Bout, B., Jetten, V., van Westen, C. J., & Lombardo, L. (2023). A breakthrough in fast flood simulation. *Environmental Modelling & Software, 168*, 105787. https://doi.org/10.1016/j.envsoft.2023.105787

Van Den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., Kavukcuoglu, K., et al. (2016). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, *12*, 1.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. *stat*, *1050*, 20.

Verheij, H., & Hydraulics, W. (2003). Aanpassen van het bresgroeimodel in his-om : Bureaustudie [Online; accessed 21-January-2025]. https://open.rijkswaterstaat.nl/open-overheid/onderzoeksrapporten/@26258/aanpassen-bresgroeimodel-his

Vorogushyn, S., Merz, B., Lindenschmidt, K. E., & Apel, H. (2010). A new methodology for flood hazard assessment considering dike breaches. *Water Resources Research*, *46*(8), 1–17. https://doi.org/10.1029/2009WR008475

Vorogushyn, S., Merz, B., & Apel, H. (2009). Development of dike fragility curves for piping and micro-instability breach mechanisms. *Natural Hazards and Earth System Sciences*, *9*(4), 1383–1401. https://doi.org/10.5194/nhess-9-1383-2009

Vreugdenhil, C. B. (1994). *Numerical methods for shallow-water flow* (Vol. 13). Springer Science & Business Media.

Wang, R., Walters, R., & Yu, R. (2020). Incorporating symmetry into deep dynamics models for improved generalization. *arXiv preprint*. https://doi.org/10.48550/arXiv.2002.03061

Wang, X., Chen, Y., & Zhu, W. (2022). A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *44*(9), 4555–4576. https://doi.org/10.1109/TPAMI.2021.3069908

Wang, Y., Fang, Z., Hong, H., & Peng, L. (2020). Flood susceptibility mapping using convolutional neural network frameworks. *Journal of Hydrology*, *582*(September 2019), 124482. https://doi.org/10.1016/j.jhydrol.2019.124482

Wang, Z., Lyu, H., Fu, G., & Zhang, C. (2024). Time-guided convolutional neural networks for spatiotemporal urban flood modelling. *Journal of Hydrology*, *645*, 132250. https://doi.org/10.1016/j.jhydrol.2024.132250

Wardhani, N. W. S., Rochayani, M. Y., Iriany, A., Sulistyono, A. D., & Lestantyo, P. (2019). Cross-validation metrics for evaluating classification performance on imbalanced data. *2019 International Conference on Computer, Control, Informatics and its Applications (IC3INA)*, 14–18. https://doi.org/10.1109/IC3INA48034.2019.8949568

Wei, G., Xia, W., He, B., & Shoemaker, C. (2024). Quick large-scale spatiotemporal flood inundation computation using integrated encoder-decoder lstm with time distributed spatial output models. *Journal of Hydrology*, 130993. https://doi.org/10.1016/j.jhydrol.2024.130993

Weinmann, P. E., & Laurenson, E. M. (1979). Approximate flood routing methods: A review. *Journal of the Hydraulics Division*, *105*(12), 1521–1536. https://doi.org/10.1061/JYCEAJ.0005329

Westerhof, S. G., Booij, M. J., Van den Berg, M. C., Huting, R. J., & Warmink, J. J. (2023). Uncertainty analysis of risk-based flood safety standards in the netherlands through a scenario-based approach. *International Journal of River Basin Management*, *21*(3), 559–574. https://doi.org/10.1080/15715124.2022.2060243

Wieland, M., & Martinis, S. (2019). A modular processing chain for automated flood monitoring from multi-spectral satellite data. *Remote Sensing*, *11*(19), 2330.

Williams, R. D., Brasington, J., & Hicks, D. M. (2016). Numerical modelling of braided river morphodynamics: Review and future challenges. *Geography Compass*, *10*(3), 102–127. https://doi.org/10.1111/gec3.12260

Wu, W., & Li, H. (2017). A simplified physically-based model for coastal dike and barrier breaching by overtopping flow and waves. *Coastal Engineering*, *130*, 1–13. https://doi.org/10.1016/j.coastaleng.2017.09.007

Xia, X., Liang, Q., Ming, X., & Hou, J. (2017). An efficient and stable hydrodynamic model with novel source term discretization schemes for overland flow and flood simulations. *Water resources research*, *53*(5), 3730–3759. https://doi.org/10.1002/2016WR020055

Xie, S., Wu, W., Mooser, S., Wang, Q., Nathan, R., & Huang, Y. (2021). Artificial neural network based hybrid modeling approach for flood inundation modeling. *Journal of Hydrology*, *592*, 125605. https://doi.org/10.1016/j.jhydrol.2020.125605

Xu, J., Pradhan, A., & Duraisamy, K. (2021). Conditionally parameterized, discretization-aware neural networks for mesh-based modeling of physical systems (M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, & J. W. Vaughan, Eds.). *Advances in Neural Information Processing Systems*, *34*, 1634–1645.

Xu, L., & Gao, L. (2024). A hybrid surrogate model for real-time coastal urban flood prediction: An application to Macao. *Journal of Hydrology*, *642*, 131863. https://doi.org/10.1016/j.jhydrol.2024.131863

Xu, Q., Shi, Y., Bamber, J. L., Ouyang, C., & Zhu, X. X. (2024). Large-scale flood modeling and forecasting with floodcast. *Water Research*, 122162. https://doi.org/10.1016/j.watres.2024.122162

Yakti, B. P., Adityawan, M. B., Farid, M., Suryadi, Y., Nugroho, J., & Hadihardaja, I. K. (2018). 2d modeling of flood propagation due to the failure of way ela natural dam. *MATEC Web of Conferences*, *147*. https://doi.org/10.1051/matecconf/201814703009

Yang, M., Isufi, E., & Leus, G. (2022). Simplicial convolutional neural networks. *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 8847–8851. https://doi.org/10.1109/ICASSP43922.2022.9746017

Yang, W., Zhang, X., Tian, Y., Wang, W., Xue, J.-H., & Liao, Q. (2019). Deep learning for single image super-resolution: A brief review. *IEEE Transactions on Multimedia*, *21*(12), 3106–3121. https://doi.org/10.1109/TMM.2019.2919431

Yin, Y., Kirchmeyer, M., Franceschi, J.-Y., Rakotomamonjy, A., & Gallinari, P. (2023). Continuous pde dynamics forecasting with implicit neural representations. *The Eleventh International Conference on Learning Representations, International Conference on Representation Learning*. https://doi.org/10.48550/arXiv.2209.14855

Yokoya, N., Yamanoi, K., He, W., Baier, G., Adriano, B., Miura, H., & Oishi, S. (2022). Breaking limits of remote sensing by deep learning from simulated data for flood and debris-flow mapping. *IEEE Transactions on Geoscience and Remote Sensing*, *60*, 1–15. https://doi.org/10.1109/TGRS.2020.3035469

You, J., Ying, R., & Leskovec, J. (2020). Design space for graph neural networks. *CoRR*, *abs/2011.08843*. https://arxiv.org/abs/2011.08843

Youssef, A. M., Pradhan, B., & Sefry, S. A. (2016). Flash flood susceptibility assessment in jeddah city (kingdom of saudi arabia) using bivariate and multivariate statistical models. *Environmental Earth Sciences*, *75*(1), 12. https://doi.org/10.1007/s12665-015-4830-8

Zhang, S., Xia, Z., Yuan, R., & Jiang, X. (2014). Parallel computation of a dam-break flow model using openmp on a multi-core computer. *Journal of hydrology*, *512*, 126–133. https://doi.org/10.1016/j.jhydrol.2014.02.035

Zhang, Z., Flora, K., Kang, S., Limaye, A. B., & Khosronejad, A. (2021). Data-driven prediction of turbulent flow statistics past bridge piers in large-scale rivers using convolutional neural networks. *Water Resources Research*, e2021WR030163.

Zhao, G., Bates, P., Neal, J., & Pang, B. (2020). Design flood estimation for global river networks based on machine learning models. *Hydrology and Earth System Sciences Discussions*, *2015*(December), 1–25. https://doi.org/10.5194/hess-2020-594

Zhao, G., Pang, B., Xu, Z., Peng, D., & Zuo, D. (2020). Urban flood susceptibility assessment based on convolutional neural networks. *Journal of Hydrology*, *590*(June), 125235. https://doi.org/10.1016/j.jhydrol.2020.125235

Zhao, G., Pang, B., Xu, Z., Cui, L., Wang, J., Zuo, D., & Peng, D. (2021). Improving urban flood susceptibility mapping using transfer learning. *Journal of Hydrology*, *602*, 126777. https://doi.org/10.1016/j.jhydrol.2021.126777

Zhao, G., Balstrøm, T., Mark, O., & Jensen, M. B. (2021). Multi-scale target-specified submodel approach for fast large-scale high-resolution 2d urban flood modelling. *Water*, *13*(3). https://doi.org/10.3390/w13030259

Zhou, Y., Wu, W., Nathan, R., & Wang, Q. J. (2021). A rapid flood inundation modelling framework using deep learning with spatial reduction and reconstruction. *Environmental Modelling and Software*, *143*, 105112. https://doi.org/10.1016/j.envsoft.2021.105112