UNIVERSITÀ DI PARMA

UNIVERSITÀ DEGLI STUDI DI PARMA

Dottorato di Ricerca in Ingegneria Civile e Architettura

Ciclo XXXVII

A deep learning framework for real-time forecasting of 2-D inundation maps: the FloodSformer model

Coordinatore:
Chiar.mo Prof. Andrea Spagnoli

Tutor:
Chiar.mo Prof. Renato Vacondio

Co-tutor:
Dott. Ing. Susanna Dazzi

Dottorando:
Matteo Pianforini

Anni Accademici 2021/2022 – 2023/2024

UNIVERSITÀ DI PARMA

UNIVERSITÀ DEGLI STUDI DI PARMA

Doctorate in Civil Engineering and Architecture

XXXVII cycle

A deep learning framework for real-time forecasting of 2-D inundation maps: the FloodSformer model

Coordinator:
Prof. Andrea Spagnoli

Supervisor:
Prof. Renato Vacondio

Co-supervisor:
Dr. Susanna Dazzi

Ph.D. student:
Matteo Pianforini

Academic years 2021/2022 – 2023/2024

Abstract

This thesis aims to contribute to the advancement of data-driven models for flood forecasting, with a specific focus on developing and implementing a deep-learning model for real-time prediction of the spatiotemporal evolution of two-dimensional (2D) inundation maps over complex large topographies. Unlike traditional flood forecasting methods, which are often computationally intensive and time-consuming, the proposed model, named FloodSformer (FS), leverages state-of-the-art deep learning techniques, integrating autoencoder and transformer architectures. This combination enables the FS model to effectively extract and process complex spatiotemporal information from a sequence of consecutive water depth maps and upstream inflow discharges, to predict inundation maps of subsequent instants through an autoregressive procedure. The FS model has been rigorously tested across a diverse range of scenarios, including both dam-break scenarios and river flood events, considering a variety of synthetic and real-world case studies. To assess the model's applicability to practical flood forecasting, its performance has been measured in terms of both predictive accuracy and computational efficiency. In addition, extensive sensitivity analyses have been conducted to explore the influence of key hyperparameters: the types of flood events used for training and the spatial resolution of the maps. Datasets were numerically generated using a 2D hydrodynamic model. The results indicate that the FS model is capable of accurately forecasting long-duration flood events over complex and large-scale bathymetries, comprising up to millions of computational cells. Remarkably, FS achieves these results with computational times on the order of minutes, which

is significantly lower than the physical time of the simulated events (with physical-to-computational time ratios up to 10,000). Furthermore, the model demonstrates predictive accuracy comparable to that of traditional physically based hydrodynamic models, achieving average root mean square errors in the range of 10-20 cm across different real-world test cases. The sensitivity analyses further indicate that larger and more diverse datasets significantly enhance the model's accuracy and generalization capabilities. Additionally, when compared to a state-of-the-art convolutional neural network architecture, the FS model consistently outperformed in terms of predictive accuracy of river flood scenarios over complex terrain topographies. In conclusion, the FS model is a versatile and high-performing tool for flood prediction, with significant potential for applications in flood risk management and emergency response planning.

Acknowledgements

I would first like to express my deepest gratitude to Prof. Renato Vacondio for his continuous, invaluable, and insightful guidance throughout my doctoral research. My sincere thanks also go to Dr. Susanna Dazzi for the constant support and the precious suggestions, without which this work would not have achieved its present form. I am also sincerely grateful to all the professors and colleagues of the Parma research group for their collaboration and constructive discussions, which have greatly enriched my academic journey. A particular thank to Dr. Andrea Pilzer for his generous technical assistance with deep-learning models and for his support in the development of the code. I also sincerely thank the reviewers of this thesis for their valuable insights and constructive feedback.

Finally, special thanks to Ilaria for everything we share every day, and to my parents and my brother for their constant closeness and affection.

Declaration of AI-assisted technologies in the writing process

During the preparation of this thesis, the author used ChatGPT, an artificial intelligence language model developed by OpenAI, to assist with language revision and proofreading. Afterward, the author reviewed and edited the content as needed and takes full responsibility for the content of the thesis.

Contents

List of Figures

List of Tables

List of Acronyms

1D	one-dimensional
2D	two-dimensional
3D	three-dimensional
Adam	Adaptive Moment Estimation
AE	autoencoder
AI	Artificial Intelligence
ANN	Artificial Neural Networks
AR	autoregressive
BC	boundary condition
BDO	Block Deactivation Optimization
BUQ	Block Uniform Quadtree
CA	cross-attention
CFL	Courant-Friedrichs-Lewy
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DB	dam-break
DEM	Digital Elevation Model
DL	deep-learning
DTM	Digital Terrain Model
EWS	Early Warning Systems
FD	Finite Difference
FE	Finite Element
FN	false negatives
FP	false positives
FS	FloodSformer
FV	Finite Volume

GAN generative adversarial network

GDL gradient difference loss

GNN Graph Neural Network

GP Gaussian Process

GPU Graphics Processing Unit

HAND Height Above Nearest Drainage

HPC High-Performance Computing

LSTM Long-Short-Term Memory

MHCA multi-head cross-attention

MHSA multi-head self-attention

ML machine-learning

MLP multilayer perceptron

MPI Message Passing Interface

MSE mean squared error

MUSCL Monotone Upwind Schemes for Scalar Conservation Laws

NLP Natural Language Processing

PDE Partial Differential Equation

PE positional encoding

ReLU Rectified Linear Unit

RMSE root-mean square error

RNN Recurrent Neural Network

SA self-attention

SGD Stochastic Gradient Descent

SM Streaming Multiprocessor

SRR spatial reduction-reconstruction

SWE Shallow Water Equations

TN true negatives

TP true positives

VidHRFormer Video High-Resolutions Transformer

VPTR Video Prediction Transformer

Introduction

Floods are among the most hazardous natural disasters globally (Wallemacq & House, 2018). Over the past fifty years, 44% of all natural disasters worldwide have been associated with inundations, leading to significant loss of life, economic damage, environmental degradation, and profound social disruptions (Douris & Kim, 2021). According to the World Meteorological Organization, between 1970 and 2019, floods caused over 330,000 deaths and led to economic losses exceeding $1 trillion globally (Douris & Kim, 2021). These losses are projected to increase further due to the impacts of climate change: Dottori et al. (2023) estimate that, in absence of further climate mitigation (i.e., 3 °C global warming by 2100) and adaptation countermeasures, the average annual damage in the European Union and the United Kingdom could rise from the current €7.6 billion to approximately €44 billion. Additionally, nearly 0.5 million Europeans could be exposed to river flooding annually, compared to the current figure of 166,000 people.

Given these alarming trends, understanding the mechanisms behind extreme flood events and improving predictive capabilities are critical challenges for the hydrological and hydraulic scientific community. Enhanced understanding is essential for improving preparedness, advancing flood mitigation strategies, and facilitating adaptation to future climate scenarios (Fernández-Nóvoa et al., 2024).

Mitigating the impact of extreme inundations can be achieved by combining structural flood defense measures, such as reservoirs and levees, with efficient emergency action plans based on Early Warning Systems (EWS). Effective EWS can reduce

flood damage by up to 35% (Rogers & Tsirkunov, 2011).

Traditionally, flood prediction has been addressed using physically based models which solve Partial Differential Equations (PDEs) to simulate the flow dynamics. These models are highly accurate for simulating flood propagation in areas with complex topography and flood dynamics. However, their application in real-time forecasting is often limited due to the substantial computational resources required (Bomers & Hulscher, 2023). To overcome this limitation, data-driven models have gained considerable attention for predicting hydrological variables, as they are often computationally efficient and therefore well-suited for real-time flood forecasting, the primary focus of this thesis.

Despite their advantages, existing data-driven models frequently struggle to accurately capture the complex spatiotemporal dynamics that characterize flood events, limiting their predictive performance and reliability. Furthermore, much of the existing research has focused on pluvial floods, often relying only on rainfall data as input, while limited attention has been given to data-driven models for real-time prediction of inundations caused by dam-break scenarios or flood propagation in rivers.

This thesis aims to contribute to the advancement of data-driven models for flood forecasting, with a specific focus on the development and implementation of a deep-learning (DL) model for the real-time prediction of the spatiotemporal evolution of inundation maps on real topographies. Unlike previous studies, the proposed model, named FloodSformer (FS), combines autoencoder (AE) and transformer architectures. This combination enables the FS model to extract spatiotemporal information from a sequence of consecutive water depth maps (past frames) and the corresponding upstream inflow discharges (in the case of river floods), to predict inundation maps of subsequent instants (future frames) using an autoregressive procedure. This novel approach aims to enhance the accuracy and efficiency of flood predictions, thereby providing a robust tool for real-time flood forecasting and management.

The proposed model has been applied to both dam-break scenarios and river flood

events, considering a variety of synthetic and real case studies. The performance of the FS model is evaluated in terms of both predictive accuracy and computational efficiency, to assess its applicability to practical situations. In addition, comprehensive sensitivity analyses have been conducted to assess the impact of different hyperparameters, the types of flood events used during training, and the spatial resolution of the maps. These evaluations are crucial for improving the model's ability to generalize to previously unseen flood events.

The water depth maps used to train and evaluate the FS model were numerically generated using a hydrodynamic model: the PARFLOOD code (Vacondio et al., 2014).

The thesis is structured as follows. Chapter 1 reviews the current state of the art in flood modelling and forecasting. Chapter 2 presents the physically based model used to generate the dataset for the surrogate model, the PARFLOOD code. In Chapter 3, an introduction to machine-learning (ML) architectures and a presentation of the most prominent DL models are provided. Chapter 4 details the development of the FS model. Subsequently, Chapters 5 and 6 present the results of applying the proposed data-driven model to dam-break scenarios and river flood events, respectively, focusing on accuracy analyses. A discussion of the results, including the analysis of computational times, is presented in Chapter 7, followed by conclusions and future perspectives.

Chapter 1

Literature review

1.1 Introduction

Rapid and accurate mathematical models that predict the spatial and temporal variation of hydrological variables are crucial for developing efficient EWS, with the aim of mitigating the impact of extreme inundations. The numerous models available for this purpose can be categorized into three main types (Zounemat-Kermani et al., 2020): physically based models, conceptual models, and data-driven models.

Physically based models solve PDEs that describe the physical processes, while conceptual models use simplified relationships to emulate these physical phenomena. In contrast, data-driven models, also known as black-box or surrogate models, entirely disregard the physical background of the processes and instead learn the nonlinear relationship between the input and output variables directly from observed data. A detailed description of each category of mathematical models is presented in Sections 1.2–1.4.

1.2 Physically based models

Among the various physically based approaches, hydrodynamic models are the most commonly adopted methodology for river flood simulations. These models are mathematical frameworks that solve equations derived from physical laws governing fluid motion (Teng et al., 2017). These equations are numerically approximated using a computational grid that discretizes the domain. Depending on the spatial representation adopted, one-dimensional (1D), two-dimensional (2D), and three-dimensional (3D) hydrodynamic models can be employed.

A 1D hydrodynamic model for unsteady open channel flows like HEC-RAS (Brunner, 2016) and MIKE 11 (DHI, 2021), solves the 1D Shallow Water Equations (SWE), also known as 1D Saint-Venant equations. These equations are derived by imposing mass and momentum conservation under the following assumptions (Cunge et al., 1980): incompressible fluid, 1D flow (i.e., uniform velocity and horizontal water level over the cross-section), hydrostatic pressure distribution, negligible viscosity effects and vertical acceleration, and small channel bed slope. Despite their computational efficiency, 1D models have several limitations, including the low accuracy in representing flow fields outside the main channel (e.g., floodplains and lowland areas) due to the assumption of a uniform velocity on the cross-section.

To address some of these issues, researchers have developed 1D-2D coupled models (e.g., Bladé et al., 2012; Morales-Hernández et al., 2013) and fully 2D hydrodynamic models (e.g., Aureli et al., 2008a; Bates & De Roo, 2000; A. Bermúdez et al., 1998; Casulli, 1990). 1D-2D coupled models use cross-sections to discretize the main channels and a 2D approach for areas outside the river region. Differently, fully 2D hydrodynamic models solve the 2D-SWE on a grid composed of computational cells covering the entire modelled area. The mesh is typically based on a Digital Elevation Model (DEM) or a Digital Terrain Model (DTM), which describe the topography of the study area. SWE schemes adopt various numerical discretisation strategies (finite element, finite difference, and finite volume methods; implicit and explicit models) and grid types (structured and unstructured meshes). The high accuracy of these

models, combined with the increasing availability of high-resolution terrain data and advancements in computational capabilities, have led to their widespread adoption for flood studies (e.g., Aureli et al., 2008a; Bates et al., 2010; Sanders et al., 2010).

However, the implementation of 2D schemes in EWS is often hindered by high computational times, especially when high spatial resolution is required (Bomers & Hulscher, 2023). To overcome this drawback, various strategies can be employed (Dazzi et al., 2021a), for example, low-resolution meshes are often used to speed up computations (e.g., Morsy et al., 2018; Vorogushyn et al., 2010). However, reducing spatial resolution can decrease the accuracy of flood dynamics simulations, particularly in domains characterized by the presence of levees, embankments, and flood walls (Vacondio et al., 2016; Wing et al., 2019).

Another approach involves using simplified versions of the SWE, which neglect one or more terms in the momentum equation, resulting in models with reduced complexity (e.g., diffusive or inertial models) (e.g., Bates et al., 2010; Costabile et al., 2017). However, these simplified schemes may encounter stability and accuracy issues in scenarios involving hydraulic jumps, rapidly varying or impulsive flows, complex topographies (e.g., urban districts floods), wet/dry fronts, low friction values, and high Froude numbers (e.g., Bates et al., 2010; Costabile et al., 2017; Dazzi et al., 2021a).

The most widely adopted strategy for reducing the computational time of 2D hydrodynamic models is the use of parallelized codes, leveraging High-Performance Computing (HPC) clusters. Over the past years, numerous studies have focused on developing parallel numerical schemes that exploit the power of Central Processing Units (CPUs) and/or Graphics Processing Units (GPUs). Examples of such models include ParBreZo (Sanders et al., 2010), PARFLOOD (Turchetto et al., 2020; Vacondio et al., 2014; Vacondio et al., 2017), Iber (García-Feal et al., 2018), HiPIMS (Xia et al., 2019), TRITON (Morales-Hernández et al., 2021), LISFLOOD-FP (Sharifian et al., 2023), and SERGHEI (Caviedes-Voullième et al., 2023). These models achieve significant speedups compared to serial codes, facilitating their integration into EWS

(Fernández-Nóvoa et al., 2024). For example, the LISFLOOD model (Van Der Knijff et al., 2010) is the main hydrological model used in the European Flood Awareness System, which, since 2012, has provided probabilistic flood forecasting information mainly for large transnational European river basins up to 10 days in advance (Smith et al., 2016). Similarly, LISFLOOD is also adopted to simulate hydrological processes in the Global Flood Awareness System of the European Commission's Copernicus Emergency Management Service (Alfieri et al., 2013). Other countries, such as the USA, Australia, and China, have also adopted EWS based on numerical models. For a detailed discussion, refer to Emerton et al. (2016) and Fernández-Nóvoa et al. (2024).

3D models enable complex representation of flow dynamics, often utilized for capturing vertical dynamics (e.g., vertical turbulence and vortices) and conducting detailed analyses of fluid-structure interaction (e.g., bridge piers and culverts). However, 3D models have never been applied in the development of EWS due to their extremely high computational requirements. Moreover, using a complex 3D representation of flow dynamics is unnecessary for practical applications. As a result, 1D and 2D hydrodynamic codes have always been preferred for efficient and rapid flood simulations (Teng et al., 2017).

1.3 Conceptual models

Conceptual models are based on simplified hydraulic concepts, not involving the detailed simulation of the physical processes of inundation, resulting in significantly shorter computational times compared to physically based models (Teng et al., 2017). These models typically provide flood inundation extents and flow depths using a DEM as input parameter. Consequently, they are commonly suitable for data-sparse areas and large-scale applications, where only the maximum flood extent and maximum water levels are required, neglecting inundation dynamics. The accuracy of conceptual models significantly decreases in cases involving complex topographies and flood

events where momentum conservation is important. Additionally, these models are not suitable for modelling dam-breaks (DBs), flash floods, tsunamis, and bank erosion studies, as they do not provide representations of flow dynamics, force fields, and flow velocities (Teng et al., 2017).

Examples of conceptual models include the Rapid Flood Spreading method (Lhomme et al., 2008), the bathtub method (Teng et al., 2015), and the Height Above Nearest Drainage (HAND) model (Nobre et al., 2011). In the HAND model, topography is normalized according to the local relative height found along the drainage network. Consequently, each cell height represents the difference in level to its respective nearest drainage cell. A cell is considered inundated if its HAND value is lower than the water level in the nearest drainage cell (Nobre et al., 2011). Despite the HAND model being adopted as a flood forecasting framework at national (Johnson et al., 2019) and continental scales (Y. Y. Liu et al., 2018), Johnson et al. (2019) has demonstrated its limited capability to accurately capture inundations. Consequently, this model can be used only for expeditious and preliminary flood analysis, with the primary purpose of guidance and emergency management.

1.4 Data-driven models

Data-driven models encompass a wide range of techniques and methodologies that leverage vast amounts of data to make predictions, recognize patterns, and optimize decisions (Abrahart et al., 2008). The high computational efficiency and versatility of these models have facilitated their application across numerous fields (Lecun et al., 2015). In the context of water management problems, data-driven modeling often employs ML and DL techniques to study a hydrological process. Unlike physically based schemes, ML and DL models completely disregard the physical behaviour of involved processes and instead learn the nonlinear relationships between input and output variables from data.

ML, a field of Artificial Intelligence (AI), utilizes appropriate algorithms to learn

relationships from input data and generalize to unseen data (Goodfellow et al., 2016). In terms of flood modeling, Mosavi et al. (2018) summarized the existing ML methods for hydrological studies, including multilayer perceptron (MLP) (e.g., Dawson & Wilby, 2001; Tayfur et al., 2018), decision tree (e.g., Choi et al., 2020), support vector machine (e.g., M. Bermúdez et al., 2019), adaptive neuro-fuzzy inference system (e.g., Talei & Chua, 2012), nonlinear autoregressive network with exogenous inputs (e.g., Bomers, 2021), wavelet neural network (e.g., Kasiviswanathan et al., 2016), and hybrid models (e.g., Nourani et al., 2011).

Despite their widespread use, conventional ML models often struggle to effectively capture complex, high-dimensional patterns in data (Lecun et al., 2015). This limitation has led to the growing adoption of DL models, which automatically learn hierarchical representations of data and have demonstrated superior performance in various domains (Lecun et al., 2015). DL techniques, such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and transformer architectures have shown remarkable success in numerous fields, including image and speech recognition, Natural Language Processing (NLP), autonomous driving, medical diagnosis, and weather forecasting (Bouallègue et al., 2024; Lecun et al., 2015). These advancements make DL a compelling choice for hydrological studies and flood prediction, where capturing complex spatiotemporal relationships is crucial.

For rainfall-runoff modeling, various types of DL models have been employed, including Artificial Neural Networks (ANN) (e.g., Dawson & Wilby, 2001), Long-Short-Term Memory (LSTM) networks (e.g., Hu et al., 2018; Kordani et al., 2024; Kratzert et al., 2018, 2019), CNNs (e.g., Van et al., 2020), transformer architectures (e.g., H. Wang et al., 2024; Xu et al., 2023; Yin et al., 2022, 2023), and hybrid algorithms (e.g., W. Li et al., 2024).

Focusing on flood prediction tasks, numerous studies have reviewed existing applications of DL models in this domain (e.g., Bentivoglio et al., 2022; Bomers & Hulscher, 2023; Karim et al., 2023). As denoted by Bomers and Hulscher (2023), early DL models were primarily focused on forecasting the temporal variation of

hydraulic variables, such as discharges and water stages, at specific river sections. These models relied on upstream hydrological variables, including precipitation, temperature, and solar radiation (e.g., Campolo et al., 1999), as well as water levels recorded at upstream river sections (e.g., Castangia et al., 2023; Dazzi et al., 2021b). Among the most commonly used DL models for these tasks are LSTM networks (e.g., Dazzi et al., 2021b; Nevo et al., 2022), CNNs (e.g., J.-H. Wang et al., 2019), and transformer architectures (e.g., Castangia et al., 2023). These models primarily aim to forecast hydrological variables at specific river sections. However, this approach only partially addresses the challenge of accurately predicting the severity and spatial extent of flood events, as it is limited to specific points along a river rather than capturing the broader flood dynamics across the floodplain.

The implementation of effective EWS requires knowledge of how hydraulic variables (e.g., water depths and/or velocities) vary in space and time during a flood event, both in river regions and in floodplains. Consequently, more advanced studies are focusing on the implementation of DL-based models for inundation map forecasting. A first category of these framework encompasses surrogate models trained to generate inundation extent maps (e.g., Fraehr et al., 2022; Sarker et al., 2019) or maps of maximum water depths and/or velocities for pluvial (e.g., Berkhahn et al., 2019; do Lago et al., 2023, 2024; Guo et al., 2021, 2022; Hofmann & Schüttrumpf, 2021; Löwe et al., 2021; Seleem et al., 2023; Zheng & Zheng, 2024) or fluvial floods (e.g., M. Bermúdez et al., 2019). However, these architectures neglect the temporal evolution of inundations, thereby providing no information about temporal dynamics of the floods.

Recent studies focused on adopting DL models for forecasting the temporal variation of inundation maps (Bomers & Hulscher, 2023). Generally, datasets used to train and evaluate these models are generated using hydrodynamic models, which provide potentially limitless data, overcoming the problem of scarcity of directly observed data (Bentivoglio et al., 2022; Bomers & Hulscher, 2023). Training a DL model for river flood forecasting requires a large dataset composed of numerous flood

scenarios with different levels of severity to correctly generalize to unseen data. This wide spectrum of inundation scenarios can only be created using a physically based model, considering different initial conditions and boundary conditions (BCs). The use of such a model is essential because observed flood maps with the necessary temporal and spatial resolution are rarely available in practice.

Applications to urban pluvial floods typically involve predicting inundation maps based on rainfall observations (e.g., Hop et al., 2024; Kao et al., 2021) and other secondary parameters, such as topographical information (e.g., DEM, slope, imperviousness), estimated runoff volume (e.g., do Lago et al., 2023), time series of sewer manholes spillings (Burrichter et al., 2023), and average inundation depth at previous time steps (Liao et al., 2023).

In the field of river flood inundation forecasting, DL models have been applied to predict the spatiotemporal propagation of floods based on time series of upstream inflow discharges (e.g., Kabir et al., 2020; Zhou et al., 2022). The primary drawback of these DL models for spatiotemporal flood prediction is that they typically use only the upstream inflow hydrographs as input data for forecasting future inundation maps, neglecting the spatiotemporal relationships between consecutive inundation maps.

For example, a first application of a 1D CNN to predict the temporal variation of inundation maps for a fluvial flood was proposed by Kabir et al. (2020). They considered real and synthetic hydrographs at three upstream BCs, with various peak values and flood duration. The proposed surrogate model predicts an inundation map at a specific instant using solely the time series of inflow discharge values at previous time steps as input data.

A different methodology for spatial inundation modelling involves combining spatial reduction-reconstruction (SRR) methods with DL models. For example, Zhou et al. (2022) applied a U-Net-based SRR method to identify representative locations in a floodplain and employed a 1D CNN model to predict the temporal variation of water depths at these selected points, using BC inflows as input data. Subsequently,

the flood surface across the entire domain is reconstructed using the SRR method, based on the forecasted water depths at the representative locations.

Nevo et al. (2022) presented the structure of the Google's operational flood forecasting system, which uses different types of surrogate models to forecast time series of water stages in river sections and produce flood inundation maps. Specifically, a LSTM architecture is employed to forecast future river stages based on previous measured stages and precipitations. Each predicted water stage is then used to create the corresponding inundation map using the ML-based manifold model (Nevo et al., 2022). The principal limitation of this ML architecture is the assumption of a quasi-steady-state condition in the river, as each inundation map depends solely on the forecasted water stage at the same instant.

More recently, Wei et al. (2024) used an autoencoder LSTM model to predict water depths and velocities in a flood detention area due to a levee breach, employing multiple surrogate models for different groups of grid cells. Generally, the use of multiple DL models can reduce computational efficiency as the number of cells discretizing the study area increases. Differently, Besseling et al. (2024) adopted a single LSTM model to predict inundation maps for dike breach scenarios, using as input data the time series of the breach outflow hydrograph.

In recent years, new flexible and efficient DL methods have been explored for flood prediction, such as Gaussian Process (GP) models combined with dimensionality reduction schemes, Graph Neural Networks (GNNs), and transformer architectures. Donnelly et al. (2022) combined a framework with principal component analysis to emulate physically based model results and quantify prediction uncertainty. Similarly, Fraehr et al. (2023) employed a low-resolution hydrodynamic model to provide a preliminary estimation of the flood dynamics, which is then enhanced to high resolution through sparse learning and empirical orthogonal function analysis. The general goal is to reproduce the outputs of a high-resolution hydrodynamic model while reducing computational times. The accuracy of the results of this model was assessed by Fraehr et al. (2024), comparing the proposed architecture with other DL

models for different case studies. However, when combining physically based and data-driven models, the prediction's efficiency may be limited by the computational time and stability of the hydrodynamic model (Fraehr et al., 2023), leading to the necessity of adopting an efficient, stable, and accurate numerical code.

GNNs represent the most developed type of geometric DL models (Bentivoglio et al., 2022). Their structure is based on graphs, defined by a set of nodes and edges. Input information, encoded in nodes, edges, and graph structures, are processed with neural networks to emulate the methodology used by hydraulic models to solve the SWE (Bomers & Hulscher, 2023). A notable application of GNNs in flood prediction is the hydraulic-based graph neural network (SWE-GNN) developed by Bentivoglio et al. (2023), designed to emulate SWE solvers for flood dynamics simulation. This model was tested on synthetic case studies to simulate dike-breach floods over unseen, randomly generated DEMs and demonstrated the ability to generalize to unseen breach locations. Despite its flexibility, the SWE-GNN struggled with accurately capturing varying propagation speeds and required numerous layers to handle large time steps, which may introduce training instability (Bentivoglio et al., 2025). Furthermore, it operated with fixed boundary conditions and required the initial time step to be provided by a numerical solver. In response to these limitations, Bentivoglio et al. (2025) introduced modifications to the SWE-GNN model, incorporating a multi-scale approach. This enhancement enables the model to handle flood simulations at varying resolutions and propagation speeds. Additionally, the improved model can accommodate time-varying boundary conditions through the introduction of a ghost cell approach. This updated version has been evaluated in simulating dike-breach floods over unseen topographies and with time-varying boundary conditions, demonstrating improvements over the original model.

Transformers, based on self-attention (SA) mechanisms, learn relationships between elements of a sequence. Unlike RNNs, which process consecutive data recursively, transformers can attend to complete sequences, allowing for modeling long-term dependencies and attending to different space-time information in the

input sequence. Moreover, transformers are faster since they can work in parallel, exploiting the power of GPUs, allowing for efficient computation of tasks with large input data. Transformer architectures (Vaswani et al., 2017) are dominating the field of NLP. For example, Bidirectional Encoder Representations from Transformers (BERT; Devlin et al., 2019) and Generative Pre-trained Transformers (GPT; OpenAI, 2023) are two of the most popular transformer architectures (Khan et al., 2022). Furthermore, transformers have become very popular in several computer vision tasks, including image classification (e.g., Dosovitskiy et al., 2020), segmentation (e.g., Z. Liu et al., 2021), captioning (e.g., G. Li et al., 2019), and generation (e.g., Parmar et al., 2018). Applications to video action recognition (e.g., Bertasius et al., 2021; Z. Liu et al., 2022), video captioning (e.g., Sun et al., 2019), and video frames prediction (e.g., Ye & Bilodeau, 2023) have also been presented. Recently, transformers have been applied in other important fields, such as medical diagnosis (e.g., He et al., 2023) and weather forecasting (e.g., Ji et al., 2024).

Applications of transformer architectures for hydrological studies are mainly focused on rainfall-runoff modelling (W. Li et al., 2024; Tang et al., 2024; H. Wang et al., 2024; W.-c. Wang et al., 2024; Xu et al., 2023; Yin et al., 2022, 2023), pluvial floods (Burrichter et al., 2024; Chaudhary et al., 2024; Jin et al., 2024), and forecasting of time series of streamflow or water level in rivers (Castangia et al., 2023; C. Liu et al., 2022; Rasiya Koya & Roy, 2024). Focusing on rainfall-runoff modelling, Yin et al. (2022) proposed a new transformer-based rainfall-runoff model for runoff prediction. Then, Yin et al. (2023) modified the previously presented model and applied it to runoff predictions in new-gauged basins. Similarly, Xu et al. (2023) evaluated the effectiveness and suitability of transferring knowledge of hydrological conditions across basins, training a transformer-based model on a data-rich basin and then transferring it to a data-sparse basin to enhance the accuracy of flood predictions. These studies have shown that transformer-based models can outperform other ML and DL models (e.g., MLP, LSTM) in forecasting runoffs.

Recent studies have applied transformer architectures for pluvial floods analyses.

For example, Burrichter et al. (2024) used a temporal fusion transformer for predicting time series of overflow from sewer manholes during heavy rainfall events. Differently, Chaudhary et al. (2024) adopted a combination of spatial convolutions and temporal attention to predict an inundation map at a specific instant. The input information to the DL model are the DEM, the current inundation map, and maps of uniform rainfall values for the forecast period. It follows that, this model does not provide the temporal evolution of the flood event but only the inundation maps in a specific instant of the future.

Focusing on river flood forecasting, only a handful of works have taken advantage of transformer-based architectures. For example, Castangia et al. (2023) predicted the daily average water level in a river station one day ahead using an encoder layer with the SA mechanism, adapting the original version of the transformer presented by Vaswani et al. (2017). Differently, C. Liu et al. (2022) adopted a cross-attention (CA) mechanism (Vaswani et al., 2017) in the decoder of a transformer-based double-encoder-enabled model to combine river flows and El Niño-Southern Oscillation observations to predict monthly river streamflows. More recently, Rasiya Koya and Roy (2024) combined a LSTM network and a transformer architecture to predict the daily streamflow in a catchment based on meteorological and hydrological data.

1.5 Gaps in flood forecasting literature

Hydrodynamic models, are invaluable for their capacity of accurately simulating flood dynamics through the solution of partial differential equations. However, these models often requires high computational times, even with the advances in parallelized computation codes utilizing HPC clusters. The need for substantial computational resources can limit their practicality for real-time flood forecasting in EWS.

On the other hand, conceptual models offer a simpler and more computationally efficient alternative. These models provide only basic information about flood extent and depth, relying on a DEM and simplified hydraulic principles. While suitable

for large-scale and data-sparse areas, their simplicity renders them inadequate for accurately capturing the complexities of flood dynamics, particularly in areas with intricate topographies.

Data-driven models offer a promising middle ground by providing computational efficiency and the ability to learn complex relationships from data. These models can rapidly predict inundation dynamics, making them suitable for real-time forecasting applications. However, existing data-driven models often fail to fully exploit the spatiotemporal dynamics inherent in flood events, which can limit their predictive accuracy and reliability. Additionally, most studies in this domain have focused on the analysis of pluvial floods, primarily using rainfall observations as input data.

Despite the significant potential demonstrated by transformer models in various fields, their application within flood prediction studies remains considerably underdeveloped. Current research has largely overlooked the utilization of attention mechanisms for the specific task of predicting the spatiotemporal propagation of floods. This gap in the literature highlights the need of improving the accuracy and efficiency of flood prediction systems. Moreover, existing frameworks typically handle homogeneous data in terms of dimensionality (either all vectors or all matrices), whereas simulating river floods involves the challenge of correlating information from heterogeneous data sources (i.e., time series of BC values and sequences of inundation maps).

Furthermore, the literature lacks AI-based models for real-time prediction of inundations resulting from DB scenarios. DBs are catastrophes that have caused significant loss of life and economic damage in recent centuries (Aureli et al., 2021). Unlike river floods, DBs are characterized by a rapid and unexpected flood propagation downstream of the dam following its collapse. The release of a considerable volume of water typically results in an outflow discharge significantly higher than that expected during the most severe river floods. Consequently, DB events often lead to overflows and subsequent inundation of floodplains. Previous studies on the application of AI-based models to DB scenarios have mainly focused on predicting outflow discharges

from reservoirs (e.g., Ma & Fu, 2012). Preliminary applications of ML models have been restricted to 1D analysis (C. Li et al., 2023) or the development of surrogate models for computational fluid dynamics, focusing on multiphase flows (Boosari, 2019) and fluid-structure interactions (S. Li et al., 2023) for 2D synthetic DBs. Notably, these surrogate models are not applicable for forecasting DB scenarios on real 2D topographies. The use of data-driven models for DB flood prediction remains an unexplored topic.

This literature gaps underscores the need for the development of an innovative DL model for real-time forecasting of the spatiotemporal evolution of inundation maps, with the aim of improving the accuracy, efficiency, and speed of flood predictions.

Chapter 2

Hydrodynamic model

2.1 Introduction

In the present thesis, a hydrodynamic model is employed to generate the datasets used to train and evaluate the proposed surrogate model. It is fundamental that the numerical model adopted is both highly accurate and computationally efficient, thereby ensuring the reliability of the ground-truth data for training the surrogate model and minimizing the computational time required for dataset generation. Furthermore, given that the ultimate objective of this study is the implementation of a surrogate model for predicting inundation maps, the dataset samples must consist of 2D water depth maps. The use of a physically based 1D approach might lead to significant inaccuracies in reproducing flood dynamics outside the main river channel (see Section 1.2). Similarly, 3D models are excluded due to their prohibitively high computational times, which render them impractical for simulating floods in river regions. In contrast, 2D hydrodynamic models offer a balanced approach, providing accurate results within reasonable computational times, particularly when utilizing parallelized code capable of harnessing the computing power of GPUs. For these reasons, this work adopts the parallelized 2D-SWE solver developed by Vacondio

et al. (2014) and Vacondio et al. (2017), named PARFLOOD.

In this Chapter, an overview of the PARFLOOD code is provided, detailing its underlying numerical methods and innovative features designed to enhance simulation accuracy and efficiency. Then, the Compute Unified Device Architecture (CUDA) implementation of the PARFLOOD code is recalled, highlighting the strategies employed to optimize computational performance and effectively handle large-scale flood simulations. Finally, a review of previous works is provided, demonstrating the code's accuracy and efficiency through various test cases and real-world flood event simulations.

2.2 PARFLOOD code

The PARFLOOD code solves the complete 2D-SWE with an explicit Finite Volume (FV) scheme. The SWE are derived from the physical principles of mass and momentum conservation, by depth-averaging the Navier-Stokes equations, obtaining a system of non-linear partially differential equations of hyperbolic type (Toro, 2001). The assumptions underlying the SWE are the following (Cunge et al., 1980):

- incompressible fluid;
- horizontal length scale much larger than the vertical one;
- hydrostatic pressure distribution (i.e., small streamline curvatures and negligible vertical accelerations);
- effects of friction can be accounted for through steady-state resistance laws;
- small channel bed slope;
- the effect of viscosity and surface tension is negligible.

Generally, the SWE are used for modelling both slow and rapidly varying free surface flows, such as flood waves in rivers and DB waves.

The conservative form of the 2D-SWE can be written as (Toro, 2001):

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} = \mathbf{S}_0 + \mathbf{S}_f \tag{2.1}$$

in which $\mathbf{U}$ is the vector of conserved variables, $\mathbf{F}$ and $\mathbf{G}$ are fluxes in the x and y directions, respectively. $\mathbf{S}_0$ and $\mathbf{S}_f$ are the bed and friction slope source terms, respectively. t is the time.

In the PARFLOOD code, the terms in Eq. (2.1) are defined using the well-balanced formulation of Liang and Borthwick (2009):

$$\mathbf{U} = \begin{bmatrix} \eta \\ uh \\ vh \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} uh \\ u^2h + \frac{1}{2}g\left(\eta^2 - 2\eta z\right) \\ uvh \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} vh \\ uvh \\ v^2h + \frac{1}{2}g\left(\eta^2 - 2\eta z\right) \end{bmatrix},$$

$$\mathbf{S}_0 = \begin{bmatrix} 0 \\ -g\eta\frac{\partial z}{\partial x} \\ -g\eta\frac{\partial z}{\partial y} \end{bmatrix}, \quad \mathbf{S}_f = \begin{bmatrix} 0 \\ -gh\frac{n_f^2 u\sqrt{u^2+v^2}}{h^{4/3}} \\ -gh\frac{n_f^2 v\sqrt{u^2+v^2}}{h^{4/3}} \end{bmatrix}. \tag{2.2}$$

where h is the water depth [m], u and v are the velocity components [m/s] along x and y, respectively. z is the bed elevation [m a.s.l.] (fixed) and $\eta = h + z$ is the water surface elevation relative to the mean sea level [m a.s.l.]. g is the gravitational acceleration [m/s^2] and n_f is the Manning's roughness coefficient [sm$^{-1/3}$].

The update of the conserved physical quantities $\mathbf{U}_{i,j}$ in time can be obtained using both a first order or a second order accuracy. For the first order approximation the update in time is obtained using the following formulation (Toro, 2001):

$$\mathbf{U}_{i,j}^{t+\Delta t} = \mathbf{U}_{i,j}^{t} - \frac{\Delta t}{\Delta x}\left(\mathbf{F}_{i+\frac{1}{2},j} - \mathbf{F}_{i-\frac{1}{2},j}\right) - \frac{\Delta t}{\Delta y}\left(\mathbf{G}_{i,j+\frac{1}{2}} - \mathbf{G}_{i,j-\frac{1}{2}}\right) + \Delta t\left(\mathbf{S}_0 + \mathbf{S}_f\right). \tag{2.3}$$

Differently, the second order in time is achieved adopting a second-order Runge-Kutta

method (Toro, 2001):

$$\mathbf{U}_{i,j}^{t+\Delta t} = \mathbf{U}_{i,j}^{t} + \tfrac{1}{2}\Delta t \left[\mathbf{D}_i\left(\mathbf{U}_{i,j}^{t}\right) + \mathbf{D}_i\left(\mathbf{U}_{i,j}^{t+\frac{\Delta t}{2}}\right)\right] \tag{2.4}$$

where:

$$\mathbf{U}_{i,j}^{t+\frac{\Delta t}{2}} = \mathbf{U}_{i,j}^{t} + \Delta t \mathbf{D}_i\left(\mathbf{U}_{i,j}^{t}\right) \tag{2.5}$$

and

$$\mathbf{D}_i\left(\mathbf{U}_{i,j}^{t}\right) = -\frac{\mathbf{F}_{i+\frac{1}{2},j} - \mathbf{F}_{i-\frac{1}{2},j}}{\Delta x} - \frac{\mathbf{G}_{i,j+\frac{1}{2}} - \mathbf{G}_{i,j-\frac{1}{2}}}{\Delta y} + \mathbf{S}_0 + \mathbf{S}_f. \tag{2.6}$$

In Eqs. (2.3)–(2.6), the subscripts i, j and Δx, Δy are the cell position and the grid size [m] in x and y directions, respectively. Δt is the time step [s] computed accordingly to the Courant-Friedrichs-Lewy (CFL) stability condition (Toro, 2001):

$$\Delta t = \frac{1}{2} Cr \min\left(\frac{\Delta x}{|u| + \sqrt{gh}}, \frac{\Delta y}{|v| + \sqrt{gh}}\right) \tag{2.7}$$

where Cr is the Courant number [-], which must be lower than one to guarantee the stability of explicit schemes.

The system of non-linear partially differential equations in Eq. (2.1) is solved on a structured grid using a FV scheme. Fluxes are computed adopting a HLLC approximate Riemann solver (Toro, 1999), while a Monotone Upwind Schemes for Scalar Conservation Laws (MUSCL) interpolation with minmod limiter (Toro, 2001) is adopted for ensuring the second order of accuracy in space. Furthermore, to obtain a balance between fluxes and source terms and guarantee the *C-property*, defined as the capability of preserving still water at rest, the formulation proposed by Liang and Marche (2009) is adopted.

The MUSCL reconstruction is calculated as follows. Considering a generic cell with coordinates (i, j), the generic conserved variable ψ is reconstructed in the x direction at the left side (superscript L) and right side (superscript R) of the cell as

follows:

$$\begin{aligned}\psi^L_{i+\frac{1}{2},j} &= \psi_{i,j} + \tfrac{1}{2}\phi_{i,j}\left(\psi_{i,j} - \psi_{i-1,j}\right),\\ \psi^R_{i-\frac{1}{2},j} &= \psi_{i,j} - \tfrac{1}{2}\phi_{i,j}\left(\psi_{i+1,j} - \psi_{i,j}\right)\end{aligned} \tag{2.8}$$

where $\phi_{i,j}$ is the minmod limiter function (Toro, 2001), which is evaluated separately for each considered variable. If the cell is wet, the velocities are calculated as:

$$\begin{aligned}u^L_{i+\frac{1}{2},j} &= \frac{(uh)^L_{i+\frac{1}{2},j}}{h^L_{i+\frac{1}{2},j}}, \quad u^R_{i-\frac{1}{2},j} = \frac{(uh)^R_{i-\frac{1}{2},j}}{h^R_{i-\frac{1}{2},j}},\\ v^L_{i+\frac{1}{2},j} &= \frac{(vh)^L_{i+\frac{1}{2},j}}{h^L_{i+\frac{1}{2},j}}, \quad v^R_{i-\frac{1}{2},j} = \frac{(vh)^R_{i-\frac{1}{2},j}}{h^R_{i-\frac{1}{2},j}}\end{aligned} \tag{2.9}$$

while the bed elevation at the same cell interface is computed as:

$$z^L_{i+\frac{1}{2},j} = \eta^L_{i+\frac{1}{2},j} - h^L_{i+\frac{1}{2},j}, \quad z^R_{i-\frac{1}{2},j} = \eta^R_{i-\frac{1}{2},j} - h^R_{i-\frac{1}{2},j}. \tag{2.10}$$

Then, a single value of the bottom elevation is computed considering the left and right values:

$$z_{i+\frac{1}{2},j} = \max\left[z^L_{i+\frac{1}{2},j}, z^R_{i+\frac{1}{2},j}\right], \quad z_{i-\frac{1}{2},j} = \max\left[z^L_{i-\frac{1}{2},j}, z^R_{i-\frac{1}{2},j}\right] \tag{2.11}$$

and the water depths are corrected to ensure positivity:

$$h^{L*}_{i+\frac{1}{2},j} = \max\left[0, \eta^L_{i+\frac{1}{2},j} - z_{i+\frac{1}{2},j}\right], \quad h^{R*}_{i-\frac{1}{2},j} = \max\left[0, \eta^R_{i-\frac{1}{2},j} - z_{i-\frac{1}{2},j}\right]. \tag{2.12}$$

Finally, the Riemann states are calculated as follows:

$$\eta^{L*}_{i+\frac{1}{2},j} = h^{L*}_{i+\frac{1}{2},j} + z_{i+\frac{1}{2},j}, \quad \eta^{R*}_{i-\frac{1}{2},j} = h^{R*}_{i-\frac{1}{2},j} + z_{i-\frac{1}{2},j}, \tag{2.13}$$

$$\begin{aligned}uh^{L*}_{i+\frac{1}{2},j} &= h^{L*}_{i+\frac{1}{2},j} u^L_{i+\frac{1}{2},j}, \quad uh^{R*}_{i-\frac{1}{2},j} = h^{R*}_{i-\frac{1}{2},j} u^R_{i-\frac{1}{2},j},\\ vh^{L*}_{i+\frac{1}{2},j} &= h^{L*}_{i+\frac{1}{2},j} v^L_{i+\frac{1}{2},j}, \quad vh^{R*}_{i-\frac{1}{2},j} = h^{R*}_{i-\frac{1}{2},j} v^R_{i-\frac{1}{2},j}.\end{aligned} \tag{2.14}$$

The MUSCL reconstruction is analogous in y direction.

To guarantee that the second order numerical scheme is well-balanced, the bed slope source term $\mathbf{S}_0$ has to be calculated in compatibility with the MUSCL reconstruction. Considering the x direction, the discretization is obtained as (Liang & Marche, 2009):

$$S_x = -g\frac{\eta^{R*}_{i-\frac{1}{2},j} + \eta^{L*}_{i+\frac{1}{2},j}}{2}\left[\frac{z_{i+\frac{1}{2},j} - z_{i-\frac{1}{2},j}}{\Delta x}\right] + S^- + S^+ \tag{2.15}$$

in which:

$$\begin{aligned} S^- &= g\Delta z_{i-\frac{1}{2},j}\frac{z_{i+\frac{1}{2},j} - z_{i-\frac{1}{2},j} + \Delta z_{i-\frac{1}{2},j}}{2\Delta x}, \\ S^+ &= g\Delta z_{i+\frac{1}{2},j}\frac{z_{i+\frac{1}{2},j} - z_{i-\frac{1}{2},j} + \Delta z_{i+\frac{1}{2},j}}{2\Delta x} \end{aligned} \tag{2.16}$$

where the two corrective terms $\Delta z_{i\pm\frac{1}{2},j}$ guarantee the respect of the *C-property* also in presence of wet/dry fronts. They are formulated as follows:

$$\begin{aligned} \Delta z_{i-\frac{1}{2},j} &= \max\left[0, z_{i-\frac{1}{2},j} - \eta^{R}_{i-\frac{1}{2},j}\right], \\ \Delta z_{i+\frac{1}{2},j} &= \max\left[0, z_{i+\frac{1}{2},j} - \eta^{L}_{i+\frac{1}{2},j}\right]. \end{aligned} \tag{2.17}$$

Analogous discretization is applied in the y direction.

If a first order numerical scheme is adopted, the MUSCL reconstruction is not applied. Consequently, the free surface elevations at the interface correspond to the value in the corresponding cell: $\eta^{R*}_{i-\frac{1}{2},j} = \eta^{L*}_{i+\frac{1}{2},j} = \eta_{i,j}$. Moreover, the bed slope source term is discretized with the same formulation of the second order of accuracy, while the bed elevation at the interface is obtained as follows:

$$z^{L}_{i+\frac{1}{2},j} = \max\left[z_{i,j}, z_{i+1,j}\right], \quad z^{R}_{i-\frac{1}{2},j} = \max\left[z_{i-1,j}, z_{i,j}\right]. \tag{2.18}$$

To avoid instabilities at small water depths, the friction source term $\mathbf{S}_f$ is discretized using the implicit formulation proposed by Caleffi et al. (2003). Furthermore, to prevent the formation of spurious non-physical flow velocities near the wet-dry

front, the specific discharge uh is corrected as follows (Kurganov & Petrova, 2007):

$$uh_c = uh \frac{\sqrt{2}}{\sqrt{1 + \max\left[1, \left(\frac{\epsilon}{h}\right)^4\right]}} \tag{2.19}$$

where uh_c is the corrected specific discharge in x direction and ϵ is a water depth threshold. Analogous correction is applied to the specific discharge in y direction vh.

A Cartesian grid or a non-uniform structured grid, named Block Uniform Quadtree (BUQ) grid (Vacondio et al., 2017), can be adopted to discretize the domain. The BUQ grid uses cell blocks with uniform resolution, while allowing different resolutions for different blocks of cells. This implementation enables to reduce the computational burden, with a substantial decrease of the number of stored cells and of the runtime compared to uniform grids (Vacondio et al., 2017).

The model also features an extension to include levee/dam breaches using a physically based erosion model, able to predict the opening and evolution of breaches forming in levees built with either cohesive or non-cohesive material (Dazzi et al., 2019). Furthermore, hydraulic structures can be modelled thanks to the adoption of internal BCs (Dazzi et al., 2020).

2.2.1 CUDA implementation

The resolution of the fully 2D-SWE over large domains is a computationally intensive process, often resulting in unfeasible computational times for practical applications, especially when a fine spatial resolution (e.g., 1-5 m) is adopted. Consequently, in order to reduce the high computational cost, the PARFLOOD code was efficiently implemented in C++ and CUDA languages (NVIDIA, 2024). CUDA is a high-level language introduced by NVIDIA that allows for the efficient utilization of both CPU (host) and GPU (device). Generally, the main part of the code is executed on the CPU, which controls the execution flow, the time advancement of the simulation, and calls specific functions, named kernels, that are executed in

parallel on a GPU, allowing the offloading of intensive computations. In CUDA, the basic work unit is the thread (Fig. 2.1a) which represents, in the PARFLOOD code, a computational cell used to discretize the physical domain. Many threads (i.e., adjacent cells) are grouped to form a regular block of size 8×8 or 16×16 and are processed in parallel by a Streaming Multiprocessor (SM) of the GPU (Fig. 2.1b), taking advantage of fast memory communication (shared memory cache area), which is fundamental for cell neighborhood information exchange (Vacondio et al., 2017). Each block of threads is executed independently and is scheduled on any available SM within a GPU, in any order, so that a compiled CUDA program can execute on any number of multiprocessors, as illustrated in Fig. 2.1b. Consequently, a GPU with more SMs will automatically execute a program in less time than a GPU with fewer SMs (NVIDIA, 2024).

In the PARFLOOD code, when the domain is discretized with a Cartesian grid, data are stored as a simple 2D array. Consequently, the coordinate system used in the program corresponds to the matrix indexes used to store the array. Differently, when the BUQ grid is adopted, blocks with various resolutions are tiled according to their code index and therefore the original neighborhood relationships among blocks is not maintained in the final tiling (Vacondio et al., 2017). Blocks with different resolution have different size in the physical space, whereas they have the same size in the memory space, since each block contains information about the same number of cells (8×8 or 16×16). Fig. 2.2 shows an example of how a BUQ grid with three levels of resolutions and 10 blocks (Fig. 2.2a) are rearranged into a 4×4 matrix of blocks (Fig. 2.2b).

In the PARFLOOD code, the progress of a time step in the second order scheme (i.e., the update of the conservative variables $\mathbf{U}$ from time t to $t + \Delta t$ using Eq. (2.4)) consists in a sequence of consecutive steps (Fig. 2.3), each one corresponding to a distinct CUDA kernel invoked by the CPU (Ferrari, 2018):

1. BCs definition;

2. Current time step computation and sorting among wet and dry blocks (Block

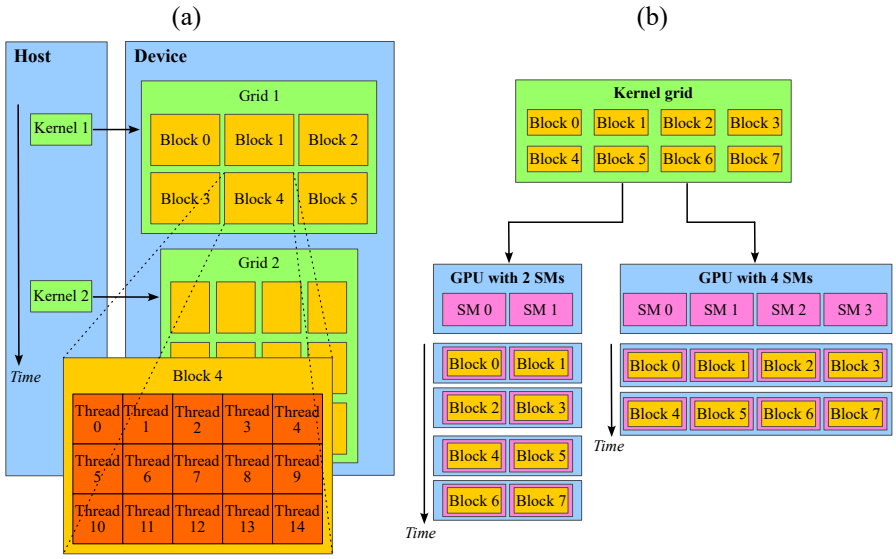

FIGURE 2.1 – (a) CUDA work units and (b) automatic scalability of a kernel on GPUs with different number of SMs. Image modified from NVIDIA (2024).

Deactivation Optimization (BDO) procedure);

3. MUSCL reconstruction (1st half step);

4. Flux computation (1st half step);

5. MUSCL reconstruction (2nd half step);

6. Flux computation (2nd half step);

7. Sum of the 1st and the 2nd half steps.

At the end of each time step, a communication phase between host and device is performed. In order to reduce the computational time, it is fundamental to limit the quantity of information exchanged to avoid bottlenecks due to data movements (NVIDIA, 2024). During the first task, the BC values are imposed using a implicit local ghost approach (Vacondio et al., 2014), which allows to consider complex boundary geometries (e.g., different BCs on diverse edges of the same ghost cell). For

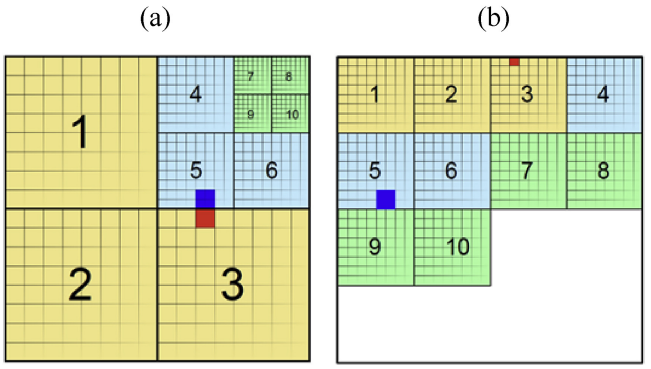

FIGURE 2.2 – (a) Example of a BUQ grid represented in the physical space with 8×8 cell blocks and three resolution levels and (b) its memory allocation. Different colors correspond to different resolution levels. Image from Vacondio et al. (2017).

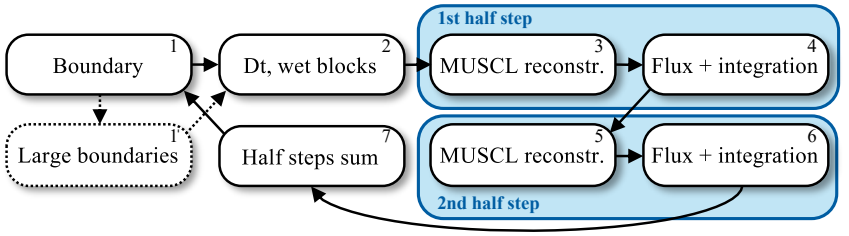

FIGURE 2.3 – Iteration flux diagram in case of a second order scheme. Each phase is handled by a GPU kernel. Image modified from Vacondio et al. (2014).

the second task, a kernel computes the current time step using Eq. (2.7) and tackles the BDO procedure which is used to reduce the number of blocks to be processed during a time step. Specifically, for each time step, calculations are performed only on wet blocks (with at least one wet cell) or the potentially wet blocks (with a cell that might become wet at the end of the time step, based on compliance with the CFL condition), while the cells with water depth lower than a predefined threshold (e.g., 10^{-10} m) are considered dry and consequently excluded from the calculations. This procedure reduces the number of cells to be considered during computations (Vacondio et al., 2014).

For further details on the model description, the reader is referred to Vacondio et al. (2014) and Vacondio et al. (2017).

A multi-GPUs version of the PARFLOOD code, that uses a Message Passing Interface (MPI) API to exchange information between different GPUs, has also been implemented (Turchetto et al., 2020). The reader is referred to Turchetto et al. (2020) for a comprehensive description of the multi-GPUs implementation.

2.2.2 Validation and configuration of the PARFLOOD model

The PARFLOOD code's accuracy and efficiency have been extensively tested for challenging case studies. For example, Vacondio et al. (2014) and Vacondio et al. (2017) applied the numerical scheme for simulating synthetic (e.g., a cylindrical DB, a steady vortex circulation) and real test cases. Moreover, Dazzi et al. (2021a, 2022), Mignosa et al. (2018), and Vacondio et al. (2016) used the PARFLOOD code to reproduce real river flood events and levee-breach floods that occurred in recent years. The results of numerical simulations were compared with observed data (e.g., recorded water levels, observed flood extension, arrival time of the inundation) to validate the PARFLOOD code's accuracy in reproducing real flood events.

The numerical model showed significant speedups compared to single-core CPU applications, with computational time reductions of two order of magnitude (Vacondio et al., 2014). Furthermore, the PARFLOOD code can simulate flood events with a high ratio of physical to computational time (Dazzi et al., 2024; Ferrari et al., 2020). These results confirm the suitability of the PARFLOOD code to rapidly simulate real flood events.

All the numerical simulations performed in the present thesis employ a first-order scheme and Cartesian grids. Unless otherwise specified, a Courant number (Cr) of 0.8 is adopted. Additionally, the water depth threshold (ϵ in Eq. (2.19)) was selected based on the scale of the case study considered. For real-world scenarios, a threshold value in the range of 0.05 m to 0.2 m was chosen, whereas for laboratory cases, the threshold was set at 10^{-4} m. This approach effectively prevented the formation of spurious, non-physical flow velocities near the wet-dry front.

Chapter 3

Deep learning models

3.1 Introduction

DL has revolutionized the field of AI by enabling the development of models capable of learning complex patterns from vast amounts of data. These models, often composed of multiple layers of ANN, have demonstrated remarkable performance across a wide range of applications, including NLP and computer vision tasks. The core strength of DL models lies in their ability to automatically extract hierarchical features from raw input data, thus facilitating the identification of intricate relationships and underlying structures that traditional ML techniques may fail to capture.

The FS model integrates various DL architectures, including CNNs, AEs, and transformers, to address complex prediction tasks. Therefore, in this Chapter, following an introduction about the fundamental principles of ML (Section 3.2), a comprehensive background on the DL models employed is provided, with the aim of clarifying their roles and functionalities within the FS framework. Additionally, Section 3.6 presents the structure of the Video Prediction Transformer (VPTR) model, which serves as a foundational component of the FS model.

3.2 Machine learning basics

A ML framework is an algorithm that is able to learn from data, improving its performance on a specific task through experience (Goodfellow et al., 2016). Mitchell (1997) defined the concept of *learning* as: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E". In general, to define a learning problem, it is essential to correctly identify the class of tasks, the performance measure, and the source of experience.

A wide range of tasks can be addressed using ML techniques. Common tasks include (Goodfellow et al., 2016):

- Classification: in this task, the ML algorithm assigns each input to one of k categories. The task can be described by a function $y = f(\boldsymbol{x})$, where $f : \mathbb{R}^n \to \{1, \ldots, k\}$, $\boldsymbol{x} \in \mathbb{R}^n$ is the input vector, and y is a number identifying a category or a probability distribution over classes. An example of this task is the object recognition in images.

- Regression: this task involves predicting a continuous outcome based on one or more inputs using a function $f : \mathbb{R}^n \to \mathbb{R}$. Typical applications include healthcare predictions, stock price forecasting, and trend analysis.

- Machine translation: this task focuses on translating a sentence of symbols from one language to another and is commonly applied in NLP tasks.

- Denoising: denoising involves reducing or eliminating noise from input data.

- Anomaly detection: in this task, the ML algorithm identifies unusual or atypical input data. An example of an anomaly detection task is the credit card fraud detection.

Once a specific task is defined, an appropriate performance measure must be selected to quantify the algorithm's accuracy in performing that task. The choice of

metric depends on the nature of the task. For classification tasks, accuracy is typically measured by the proportion of correctly classified examples. Common metrics include accuracy, recall, and F1 score (see Section 4.7). For regression tasks, the performance is measured using a continuous-valued score, which reflects the difference, or residuals, between the predicted and actual target values. Common regression metrics include mean absolute error (MAE), the coefficient of determination (R^2), and the root-mean square error (RMSE). During training, this performance measure is defined as the loss function.

ML algorithms can be categorized into two broad classes based on the nature of the experience: unsupervised learning and supervised learning. In unsupervised learning, the algorithm analyses multiple instances of an input x to implicitly or explicitly learn the probability distribution $p(x)$, or to identify interesting properties of that distribution. For example, clustering algorithms such as k-means can group a set of unlabeled data points based on their similarity, without any prior knowledge of the data structure. Another common unsupervised learning task is dimensionality reduction with methods like principal component analysis, where the algorithm identifies the most important features in the data. Conversely, supervised learning involves observing multiple instances of input x along with corresponding target values y (referred to as labels), and the algorithm learns to predict y from x, typically by estimating $p(y|x)$. A classic example of supervised learning is image classification, where an algorithm is trained on labeled images (e.g., cats or dogs), learning to predict the correct label for unseen images. Another example is linear regression, where the goal is to predict a continuous value y, such as the temperature based on features like time of day and weather conditions. In supervised learning, the target y provides the algorithm with guidance on how to perform the task, whereas, in unsupervised learning, the algorithm must learn from the data without such guidance.

The foundation of any ML model lies in the quality and structure of the dataset used for training. The ability of the model to learn patterns and make accurate predictions depends on the richness and relevance of the provided data. A well-curated

dataset should comprehensively represent the problem domain while minimizing biases and noise that could interfere with the learning process.

A key step in preparing a dataset is dividing it into three subsets: training, validation, and testing, each serving a distinct purpose in the model development pipeline (Chollet, 2021):

- Training Set: the training data is used to fit the model's parameters. During this phase, the model learns to map input features to the desired output by adjusting internal parameters (e.g., weights in an ANN). It is essential that the training set is large and diverse enough to capture underlying patterns, enabling the model to generalize effectively.

- Validation Set: the validation set is used to tune hyperparameters and assess the model's performance during training. Unlike the training set, the model does not learn from the validation data. Instead, this set provides an unbiased evaluation of model performance, allowing for iterative adjustments without overfitting to the training data.

- Test Set: after all training and validation steps are completed, the model is evaluated on the test set to determine how well it generalizes to unseen data. The test set should only be used for final evaluation, as making decisions based on its results would compromise its role as an unbiased performance estimator.

This separation ensures a robust model evaluation pipeline, enabling the optimization of the model and a realistic assessment of its performance.

In ML, parameters and hyperparameters play critical but distinct roles. Parameters are internal variables that the model learns from the data during training, such as the weights and biases in a neural network (Section 3.2.1). These parameters are automatically adjusted through optimization algorithms to minimize the loss function (Section 3.2.2). Hyperparameters, on the other hand, are external configurations that must be set before the training process begins. They control how the model learns, and include settings such as learning rates, the number of layers, and the depth of

an ANN. Hyperparameters are not learned from the data and are typically selected through methods such as grid search, random search, or Bayesian optimization, often evaluated on the validation set. Proper tuning of hyperparameters is crucial for optimizing model performance and for mitigating issues like overfitting or underfitting (Aggarwal, 2018), which are common challenges in ML models (see Section 3.2.2 for further details).

3.2.1 Basic architecture of neural networks

A basic example of a ML algorithm is a single-layer neural network, commonly used to solve linear regression problems. This model, also known as a perceptron (Fig. 3.1a), takes a vector $\boldsymbol{x} \in \mathbb{R}^n$ as input and predicts a scalar value $y \in \mathbb{R}$ as follows:

$$y = \boldsymbol{w}\ \boldsymbol{x} + b = \sum_{j=1}^{n} w_j x_j + b, \tag{3.1}$$

where b is the bias and $\boldsymbol{w} \in \mathbb{R}^n$ is a vector of parameters, which represent the weights that determine the influence of each feature on the prediction. These parameters are optimized during the training process, as detailed in Section 3.2.2.

Multilayer neural networks consist of more than one computational layer (Fig. 3.1b), unlike the perceptron, which only has input and output layers. In a perceptron, the input layer simply passes the data to the output layer, where all computations are directly observable by the user. In contrast, multilayer neural networks include additional layers, known as hidden layers, situated between the input and output layers. These hidden layers allow the model to capture more complex patterns within the data, which are not directly visible to the user.

The architecture of these networks is typically referred to as a feed-forward network, where data flows in a single direction, from the input layer through the hidden layers to the output layer. The standard structure of feed-forward networks assumes full connectivity, meaning every node in one layer is connected to every node in the subsequent layer. This configuration facilitates the flow of information and

enables the network to learn hierarchical features from the input data (Goodfellow et al., 2016).

Activation functions

Activation functions are critical in neural networks, as they introduce non-linearity, allowing the model to learn complex patterns. Fig. 3.2 shows the commonly used activation functions including the sigmoid, tanh, and Rectified Linear Unit (ReLU).

The sigmoid function, defined as:

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}}, \tag{3.2}$$

maps input values x to a range between 0 and 1, making it particularly suitable for binary classification tasks.

Similarly, the tanh function outputs values in the range of -1 to 1 and offers stronger gradients than the sigmoid, particularly for negative inputs. The tanh function is expressed as:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}. \tag{3.3}$$

Both the sigmoid and tanh functions suffer from the vanishing gradient problem, where gradients become extremely small during backpropagation (see Section 3.2.2), particularly in deep networks. This issue arises when inputs fall into the saturated

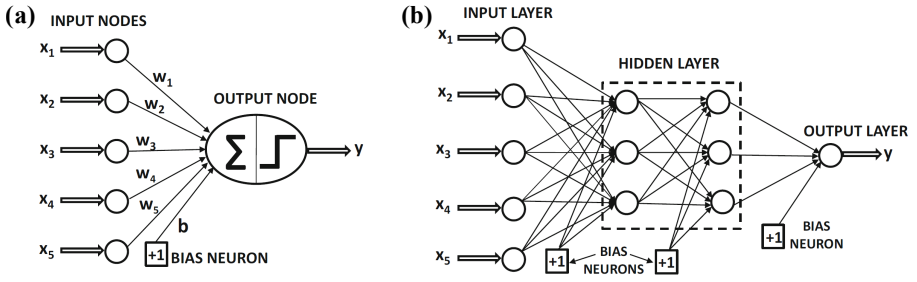

FIGURE 3.1 – The basic architecture of a feed-forward network. (a) A single-layer perceptron. (b) Feed-forward network with two hidden layers and a single output layer. Images from Aggarwal (2018).

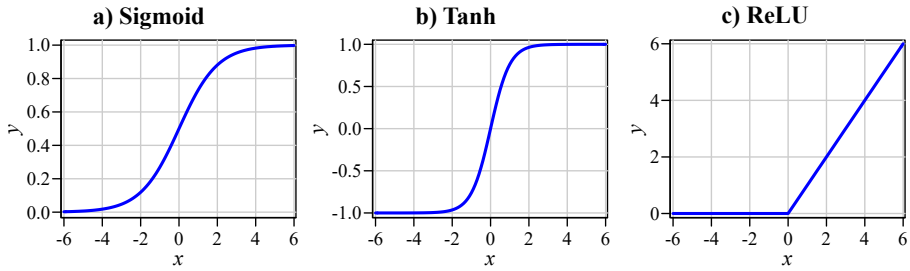

FIGURE 3.2 – Common activation functions in ML algorithms: a) sigmoid, b) tanh, and c) ReLU.

regions of the functions, resulting in slow or stalled learning because weight updates become negligible. Conversely, the exploding gradient problem occurs when gradients increase exponentially during backpropagation, causing unstable weight updates (see Section 3.2.2). This problem is more common in deep networks, where gradients can grow significantly as they propagate through multiple layers.

To address these challenges, the ReLU activation function is widely used in modern ANNs. It is defined as:

$$\text{ReLU}(x) = \max(0, x) = \begin{cases} x & \text{if } x > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{3.4}$$

ReLU alleviates the vanishing gradient problem by maintaining non-zero gradients for positive inputs, leading to faster and more efficient training (Aggarwal, 2018). However, ReLU can still suffer from the "dying ReLU" problem, where neurons output a constant zero if their weights are updated such that they only produce negative inputs. Despite this limitation, ReLU remains one of the most widely used activation functions in DL due to its simplicity and effectiveness (Goodfellow et al., 2016).

3.2.2 Training a neural network

When training a ML model, two fundamental issues must be addressed: optimization and generalization (Chollet, 2021). The optimization problem involves adjusting

the model's parameters to improve performance on the training dataset, while the generalization problem refers to the model's ability to perform well on unseen data beyond the training dataset. Both aspects must be balanced to avoid underfitting and overfitting. Underfitting occurs when a model is too simple and fails to capture the underlying patterns in the data, resulting in poor performance on both the training and test datasets. Overfitting, on the other hand, happens when the model is too complex and learns not only the patterns but also the noise within the training data, leading to excellent performance on the training set but poor generalization to new, unseen data (Chollet, 2021).

Fig. 3.3 illustrates the typical learning curves of the loss function computed on the training and validation dataset during the training process. Initially, the model is in the underfitting zone, as it is not already trained. In this phase, optimization and generalizations are correlated, with the loss decreasing on both the training and validation datasets. After several training iterations, or epochs, the validation metric stops improving and begins to degrade, indicating the onset of overfitting, where the model starts learning patterns specific to the training data.

To prevent overfitting, the training process is often terminated early using an early stopping mechanism. This technique halts training if the validation metric fails to improve after a predetermined number of epochs, and the model parameters

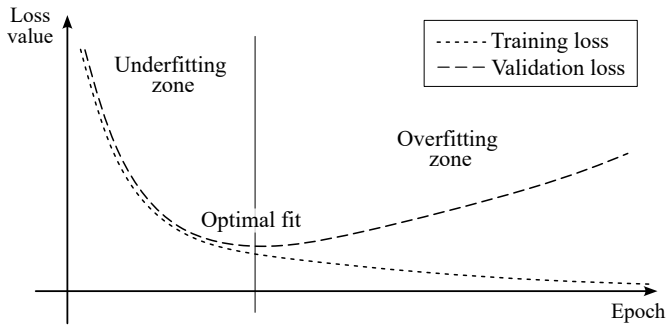

Figure 3.3 – Example of canonical learning curves obtained during the training of a ML model.

corresponding to the best validation metric are selected as the final weights.

Backpropagation

In a single-layer neural network, the training process is relatively straightforward, as the error (or loss function) can be directly computed from the weights, making gradient calculation simple. However, in multi-layer networks, the loss is a complex function composed of weights from earlier layers. To compute the gradient in such cases, the backpropagation algorithm is employed. This algorithm uses the chain rule from differential calculus to compute error gradients as a sum of local-gradient products across the various paths from a node to the output. Despite the exponential growth in the number of paths, backpropagation efficiently computes these gradients using dynamic programming (Aggarwal, 2018).

Backpropagation consists of two main phases: the forward phase and the backward phase.

1. Forward phase: during this step, the input data is passed through the network, and computations propagate forward across the layers based on the current weights. The predicted output is then compared with the target output, and the derivative of the loss function with respect to the output is calculated.

2. Backward phase: this step computes the gradient of the loss function with respect to the weights using the chain rule. These gradients are then used to update the weights.

Once the gradients have been computed, optimization algorithms are used to update the network parameters to minimize the loss, thus improving the performance of the ML architecture (Goodfellow et al., 2016).

Optimization algorithms

After calculating gradients through backpropagation, the next step is to adjust the network's parameters to minimize the loss function. This process is managed by

optimization algorithms, which include:

- Stochastic Gradient Descent (SGD): it is one of the most fundamental and widely used optimization algorithms for training neural networks. It updates the network's parameters iteratively by computing the gradient of the loss function with respect to each parameter. The update rule for SGD is given as (Robbins & Monro, 1951):

$$\theta_t = \theta_{t-1} - \eta \nabla_\theta J(\theta_{t-1}) \tag{3.5}$$

  where θ represents the parameters of the network, η is the learning rate, and $\nabla_\theta J(\theta)$ is the gradient of the loss function $J(\theta)$ with respect to the parameters. The term *stochastic* refers to the fact that parameter updates are based on individual mini-batches (i.e., small subsets of the training data), which introduces noise into the gradient estimation. While this noise can help the model escape local minima or saddle points and enhance generalization, it may also result in less stable convergence compared to batch gradient descent. In batch gradient descent, the true gradient of the total cost function decreases and eventually vanishes as the model approaches a minimum. However, the noise in SGD prevents the gradient from reaching zero, which can be addressed by adopting a decreasing learning rate (Goodfellow et al., 2016).

- Adaptive Moment Estimation (Adam): it is one of the most effective and widely used optimizers for DL tasks. Adam maintains individual adaptive learning rates for each parameter, relying on estimates of both the first-order moment (mean) and the second-order moment (uncentered variance) of the gradients. This makes Adam more robust to noisy gradients and better suited for sparse gradients (Goodfellow et al., 2016). The Adam update rule involves two moving

averages of the gradients (Kingma & Ba, 2015):

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla_\theta J(\theta_{t-1}) \tag{3.6}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)(\nabla_\theta J(\theta_{t-1}))^2 \tag{3.7}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{3.8}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{3.9}$$

$$\theta_t = \theta_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{3.10}$$

where m_t and v_t represent the first and second moment estimates, β_1 and β_2 are hyperparameters controlling the decay rates of these moment estimates, and ϵ is a small constant added for numerical stability (Kingma & Ba, 2015). Adam's adaptive learning rate and bias correction make it robust and typically faster to converge than SGD, particularly in high-dimensional and noisy datasets.

- AdamW: a variant of Adam, AdamW improves the application of weight decay (L2 regularization). Traditional Adam applies weight decay to the gradient update, but this can be problematic due to Adam's adaptive learning rates, which can interfere with regularization. AdamW decouples weight decay from gradient updates, applying regularization only to the parameter values, thus improving generalization (Loshchilov & Hutter, 2019). The weight update rule for AdamW is as follows:

$$\theta_t = \theta_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} - \eta \lambda \theta_{t-1} \tag{3.11}$$

where λ is the weight decay factor, and the rest of the terms follow the Adam update. This decoupling leads to better regularization and reduces overfitting in DL models (Loshchilov & Hutter, 2019).

The choice of optimization algorithm can significantly affect the training efficiency and performance of DL models. The appropriate method should be selected based

on the specific characteristics of the problem at hand.

3.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of DL models that utilize convolutions instead of general matrix multiplications in at least one of their layers (Goodfellow et al., 2016). The convolution of two functions x and w of real-valued arguments is computed as follows:

$$z(t) = (x * w)(t) := \int_{-\infty}^{\infty} x(\tau)w(t-\tau)d\tau \tag{3.12}$$

In CNN architectures, the functions x and w are commonly referred to as the input and the kernel, respectively. The output z is referred to as the feature map. Assuming that x and w are defined only on integer t, the 1D discrete convolution can be written as:

$$z(t) = (x * w)(t) = \sum_{\tau=-\infty}^{\infty} x(\tau)w(t-\tau). \tag{3.13}$$

In ML applications, the input is usually a multidimensional array of data, known as a tensor, while the kernel is a multidimensional array of parameters that are adapted during the training procedure. Consequently, the convolution operator can be extended also to multidimensional domains. For example, considering a 2D matrix $\boldsymbol{X}$ as input data, the convolution is formulated as follows (Goodfellow et al., 2016):

$$\boldsymbol{Z}(i,j) = (\boldsymbol{X} * \boldsymbol{K})(i,j) = \sum_m \sum_n \boldsymbol{X}(m,n)\boldsymbol{K}(i-m, j-n) \tag{3.14}$$

where $\boldsymbol{K}$ is a 2D kernel. Similarly, it can be also computed as follows:

$$\boldsymbol{Z}(i,j) = (\boldsymbol{K} * \boldsymbol{X})(i,j) = \sum_m \sum_n \boldsymbol{X}(i-m, j-n)\boldsymbol{K}(m,n) \tag{3.15}$$

since convolution is commutative. This formulation involves flipping the kernel relative to the input (i.e., when m increases, the index into the input increases, while the index into the kernel decreases). Generally, Eq. (3.15) is more straightforward to implement in a DL library, as the values of m and n have a lower range of variation (Goodfellow et al., 2016). Nevertheless, many neural network libraries implement a different function called the cross-correlation (Goodfellow et al., 2016):

$$\boldsymbol{Z}(i,j) = (\boldsymbol{K} * \boldsymbol{X})(i,j) = \sum_m \sum_n \boldsymbol{X}(i+m, j+n)\boldsymbol{K}(m,n). \tag{3.16}$$

This function is analogous to the convolution except that the kernel is not flipped (i.e., in the cross-correlation function, an increase in m results in an increase in both the indices in the input and kernel matrices).

Similarly to other ANN, CNNs are feed-forward networks in which the input data provide the initial information that propagates up to the hidden units at each layer and finally produces an output.

The output layer of a CNN is an activation function (see Section 3.2.1).

In a CNN, the most important hyperparameters are:

- Kernel size (L): represents the size of the convolving kernel, typically a square matrix (e.g., 3×3) for 2D data.
- Stride (S): indicates the number of cells by which the kernel translates during the convolution process. This parameter can be used to reduce the dimensionality of the latent feature.
- Padding (P): the process of adding zero-value cells on the borders of the input matrix to control the kernel width and the size of the output independently.
- Depth (C_{out}): represents the number of kernels in a single layer of the CNN, corresponding to the number of channels in the output tensor.

Once these hyperparameters are defined, assuming $\boldsymbol{X} \in \mathbb{R}^{H_{in} \times W_{in} \times C_{in}}$ as the input tensor, the dimension of the output tensor $\boldsymbol{Z} \in \mathbb{R}^{H_{out} \times W_{out} \times C_{out}}$ can be

computed as follows:

$$\begin{aligned} H_{out} &= \frac{H_{in} + 2P_h - L_h}{S_h} + 1 \\ W_{out} &= \frac{W_{in} + 2P_w - L_w}{S_w} + 1 \end{aligned} \tag{3.17}$$

where H_{in}, W_{in}, and C_{in} are the height, width, and number of channels of the input tensor, respectively. Similarly, H_{out}, W_{out}, and C_{out} are the height, width, and number of channels of the output tensor, respectively. P_h, L_h, and S_h are the dimensions of the padding, the kernel, and the stride in the x direction, respectively. Analogously, P_w, L_w, and S_w are the corresponding dimensions in the y direction.

Another layer commonly used in CNN is the pooling layer, which is a form of non-linear down-sampling. Unlike the convolution operation, which applies filters on all the channels of the input tensor to produce a single feature value, pooling independently operates on small grid regions of size $D \times D$ in each channel of the input feature map to produce another feature map. Consequently, the depth created using pooling is the same as that of the tensor on which the pooling operation was performed (i.e., $C_{in} = C_{out}$). For each square region of size $D \times D$ in the input feature map, the corresponding output value is the maximum of the values in that region. This approach is referred to as max pooling. Similarly, the average pooling operation calculates the average value of the values in the square region. Considering a stride of the pooling $S_p \geq 1$, the output feature map will have a size of $\frac{H_{in}-D}{S_p} + 1 \times \frac{W_{in}-D}{S_p} + 1 \times C_{in}$. Therefore, pooling drastically reduces the spatial dimensions of the feature map input to this layer (Aggarwal, 2018).

Numerous CNN architectures have been developed in recent decades. Early networks were quite shallow. For example, the *LeNet-5* network (Lecun et al., 1998), one of the earliest CNN, uses only two convolution layers, two pooling layers, and three fully connected layers (Fig. 3.4a). Modern architectures, however, are deeper and use a variety of computational, architectural, and hardware tricks to efficiently train these networks with large amounts of data. Furthermore, the advent of modern GPU architectures allowed for the exploration of more complex and larger CNN

models. Examples of these CNN architectures include AlexNet, GoogLeNet, and ResNet (Aggarwal, 2018).

For instance, the ResNet model (He et al., 2016) has a depth of up to 152 layers. The main challenge in training such deep networks is that the gradient flow between layers is impeded by the large number of operations in deep layers, which can increase or decrease the size of the gradients, leading to the problems of vanishing and exploding gradients (see Section 3.2.2). To overcome this issue, the ResNet architecture uses skip connections, also known as residual connections, between layers to bypass some levels and link layer activations to subsequent layers (Fig. 3.4c). While most feed-forward networks contain connections only between layers i and $i+1$, ResNet contains connections between layers i and $i+r$, for $r > 1$. This skip connection simply copies the input of layer i and adds it to the output of layer $i+r$. An example of residual connection with $r = 2$ is shown in Fig. 3.4b.

A residual block can be defined as (He et al., 2016):

$$\boldsymbol{y} = \mathcal{F}(\boldsymbol{x}, \{W_i\}) + \boldsymbol{x} \tag{3.18}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are the input and output vectors of the layers considered, respectively. W_i denotes the weight of i^{th} layer in this block. The function $\mathcal{F}(\boldsymbol{x}, \{W_i\})$ represents the residual mapping to be learned. For the example in Fig. 3.4b that has two layers (i.e., $r = 2$), $\mathcal{F} = W_2\sigma(W_1\boldsymbol{x})$ where σ denotes the ReLU function and the biases are omitted for simplicity. The shortcut connections introduce neither extra parameter nor computation complexity. In Eq. (3.18) the dimensions of $\boldsymbol{x}$ and $\mathcal{F}$ must be equal. However, when changing the input/output channels, a linear projection W_s is performed by the shortcut connections to match the dimensions (He et al., 2016):

$$\boldsymbol{y} = \mathcal{F}(\boldsymbol{x}, \{W_i\}) + W_s\boldsymbol{x}. \tag{3.19}$$

Generally, including skip connections allows any layer that degrades architecture performance to be omitted, reducing the likelihood of encountering problems of

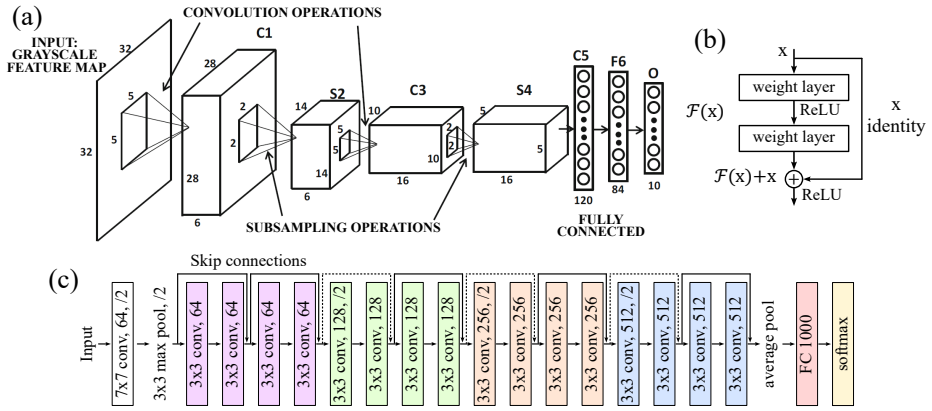

FIGURE 3.4 – (a) Architecture of *LeNet-5* network. (b) Skip-connections in a residual module. (c) *ResNet-18* architecture with 18 parameter layers. The continuous and dashed skip connections are computed with Eq. (3.18) and Eq. (3.19), respectively. Image (a) modified from Aggarwal (2018) while images (b) and (c) modified from He et al. (2016).

vanishing or exploding gradients.

These kinds of CNNs have been widely applied in NLP tasks, computer vision applications (e.g., detection, segmentation and recognition of objects and regions in images, video analysis), and time series forecasting (Lecun et al., 2015).

3.3.1 1D CNN model for flood prediction (Kabir et al., 2020)

In this study, the 1D CNN model developed by Kabir et al. (2020) is utilized as a benchmark to evaluate the accuracy of the proposed surrogate model. The 1D CNN model consists of two 1D convolutional layers followed by three fully connected layers (see Fig. 3.5). The surrogate model forecasts water depths at time t by utilizing inflow discharge data from time t to $t - r$ as input. Thus, the prediction is based exclusively on temporal data from BCs and does not incorporate prior inundation maps or spatiotemporal relationships between consecutive maps.

This specific model architecture has been employed as a benchmark in various studies aimed at simulating flood events across study areas of different sizes, up to approximately 1,500 km^2, with the number of cells ranging from 100k to 3.7M (e.g.,

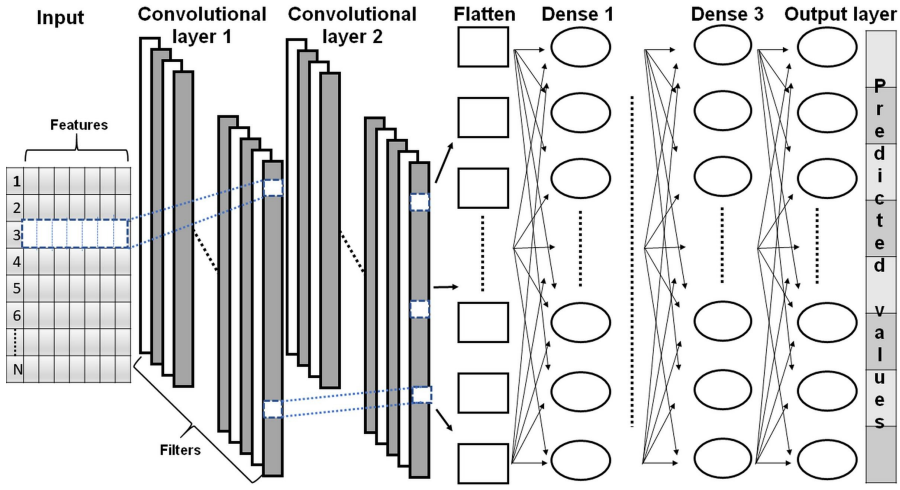

FIGURE 3.5 – Structure of the 1D CNN model used as a benchmark. The model inputs, referred to as features, are arranged in a table where each row represents a specific instant t of the flood event, composed of N time steps. The model's output is an array of water depths, which is then converted into a spatially distributed inundation map. Image from Kabir et al. (2020).

Donnelly et al., 2022; Fraehr et al., 2024).

3.4 Autoencoder model

An autoencoder (AE) is a type of ANN characterized by an output layer with the same dimensionality as the input layer. The primary objective is to reconstruct the input data by passing it through the network, effectively learning an efficient representation of the data (Aggarwal, 2018). Consequently, an AE is trained to take an input $\boldsymbol{x}$ and to reconstruct the corresponding output $\hat{\boldsymbol{x}}$, which aims to be as close as possible to $\boldsymbol{x}$.

The AE architecture consists of two main components (Fig. 3.6a): an encoder function $\boldsymbol{z} = F(\boldsymbol{x})$, which produces a compressed representation of the input data, and a decoder function that reconstructs the input as $\hat{\boldsymbol{x}} = G(\boldsymbol{z})$. Here, $\boldsymbol{z}$ represents the code, also known as latent feature or embedding. Typically, the number of units in the middle layer is fewer than in the input and output layers, leading to a

compressed representation. This compression forces the model to prioritize which aspects of the input are most important, often resulting in the learning of useful data properties (Goodfellow et al., 2016).

The loss function L, used for training an AE framework, minimizes the differences between the input and the reconstructed output, thereby forcing the output to be as similar as possible to the input:

$$L(\boldsymbol{x}, G(F(\boldsymbol{x}))). \tag{3.20}$$

The simplest version of an AE uses a MLP architecture with a single hidden layer of $k \ll d$ units between the input and output layers, each having d units. This basic AE model is commonly employed for matrix factorization (Aggarwal, 2018). More complex models can be derived by adding additional hidden layers. Deep encoders and decoders offer numerous advantages, including increased data compression, reduced computational cost, and a decrease in the amount of training data required to learn certain functions (Goodfellow et al., 2016).

A specific type of AE used for dimensionality reduction and feature learning is the convolutional AE, which employs CNNs in the encoder and decoder layers. Unlike traditional AEs, convolutional AEs leverage spatial relationships within the data to extract features with a visual interpretation through spatial convolution

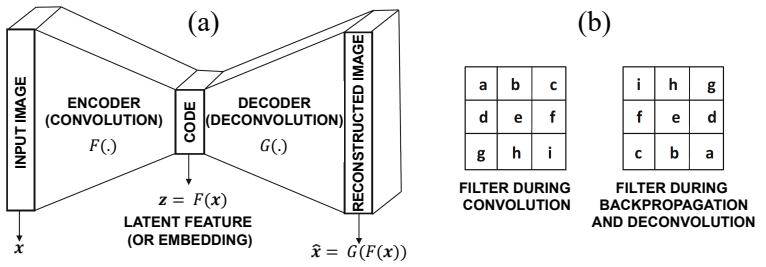

FIGURE 3.6 – (a) Sketch of a convolutional AE. (b) Comparison of a convolution kernel (left) and the transposed and inverted kernel used for deconvolution (right). The deconvolution operation mirrors the process used in backpropagation. Images modified from Aggarwal (2018).

operations in the intermediate layers (Aggarwal, 2018). The encoder compresses the input using convolutions, while the decoder reconstructs the input using deconvolutions, also known as transposed convolutions (see Fig. 3.6a). The deconvolution operation involves a convolution with a filter derived by transposing and inverting the original convolution filter tensor (see Fig. 3.6b), similar to the operation used for backpropagation in CNNs (Aggarwal, 2018). The architecture of convolutional AEs need not be symmetric; for instance, deeper encoders than decoders can be utilized, resulting in an asymmetric structure.

A widely adopted convolutional AE is the U-Net architecture (Ronneberger et al., 2015), originally developed for biomedical image segmentation (see Fig. 3.7). The U-Net comprises a contracting path (the encoder on the left side of Fig. 3.7) and an expansive path (the decoder on the right side). The encoder consists of a sequence of two convolutions, each followed by a ReLU activation and a max pooling operation for downsampling. Each step of the decoder involves upsampling the feature map followed by a sequence of convolutions and ReLU activations. During the contraction, the spatial dimensions are reduced while the number of feature channels is doubled. Conversely, in the reconstruction process, the number of feature channels is progressively reduced while the spatial dimension is restored. Additionally, the U-Net architecture employs skip connections (represented by gray arrows in Fig. 3.7) between corresponding layers in the encoder and decoder, similar to those used in ResNet blocks (see Section 3.3).

Convolutional AEs are crucial for dimensionality reduction tasks as they effectively capture and compress spatial hierarchies and intricate patterns in high-dimensional data, leading to more efficient and meaningful feature representations. Moreover, using compressed latent features reduces the runtime and memory requirements of the training process (Goodfellow et al., 2016).

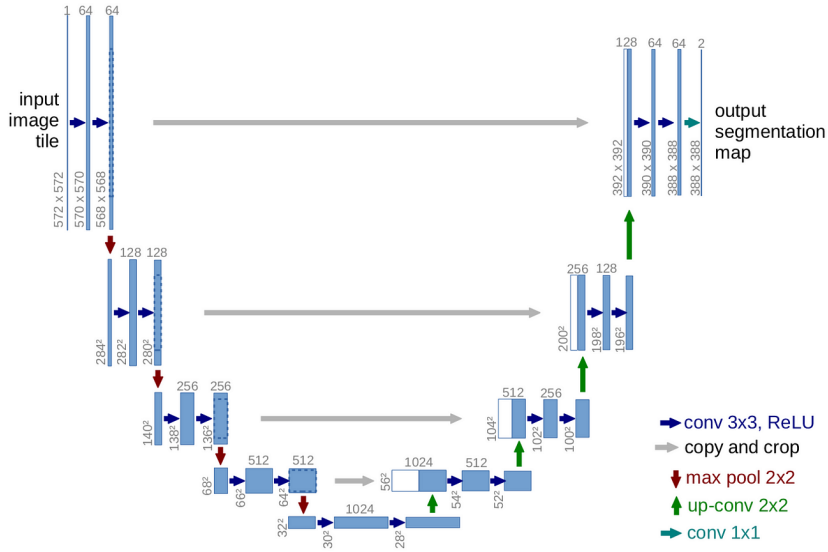

FIGURE 3.7 – U-Net architecture with 32×32 pixels in the lower resolution. Each multi-channel feature map is represented by a blue rectangle, while the copied feature maps are denoted by white rectangles. The number of channels and the dimensions of the images are indicated on top and lower left edges of the rectangles, respectively. The arrows denote the different operations. Image from Ronneberger et al. (2015).

3.5 Transformer architecture

The transformer architecture, introduced by Vaswani et al. (2017), employs the attention mechanism to analyse long-range dependencies in sequential data. Although this architecture was initially proposed to address machine translation problems, it rapidly evolved into a state-of-the-art model for both NLP and computer vision tasks. Unlike other DL architectures, such as CNN and RNN, transformers rely solely on the attention mechanism.

Consider a sequence of n entities $(x_1, \ldots, x_n)$ denoted as $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, where d is the embedding dimension representing each entity. Self-attention (SA) aims to encode each entity using global contextual information, thereby capturing the interactions among all n entities. This is achieved by projecting the input sequence $\boldsymbol{X}$ onto three

matrices: query $\boldsymbol{Q}$, key $\boldsymbol{K}$, and value $\boldsymbol{V}$, computed as follows (Vaswani et al., 2017):

$$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}_Q, \qquad \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}_K, \qquad \boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}_V \tag{3.21}$$

where $\boldsymbol{W}_Q \in \mathbb{R}^{d \times d_q}$, $\boldsymbol{W}_K \in \mathbb{R}^{d \times d_k}$, and $\boldsymbol{W}_V \in \mathbb{R}^{d \times d_v}$ are learnable weight matrices used to project the input sequence $\boldsymbol{X}$.

The SA computation, represented in Fig. 3.8a, involves computing the scaled dot-product of the query with the key, which is then normalized using the Softmax operator (Eq. (3.23)) to obtain the attention scores. This attention matrix is subsequently multiplied by the value matrix to yield the output of the SA operation $\boldsymbol{Z} \in \mathbb{R}^{n \times d_v}$:

$$\boldsymbol{Z} = \text{Softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right)\boldsymbol{V} \tag{3.22}$$

The Softmax function for an input vector $\boldsymbol{x} = (x_1, \dots x_n) \in \mathbb{R}^n$ is defined as (Goodfellow et al., 2016):

$$\text{Softmax}(\boldsymbol{x})_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)}. \tag{3.23}$$

In certain scenarios, SA must be computed only on specific data within the input sequence. This is typically applied when training a model to predict the next element in a sequence. In such case, a masked SA is employed in the decoder to ensure that predictions for a given position depend solely on known outputs at preceding positions, thereby preventing the model from using unknown future data for predictions. This is achieved by setting future entities to zero using an upper-triangular matrix $\boldsymbol{M} \in \mathbb{R}^{n \times n}$, resulting in the masked SA operation:

$$\text{Softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}} \circ \boldsymbol{M}\right) \tag{3.24}$$

where $\circ$ denotes the Hadamard product.

To capture complex relationships among different elements in the sequence, a multi-head self-attention (MHSA) mechanism is typically employed (see Fig. 3.8b). In the

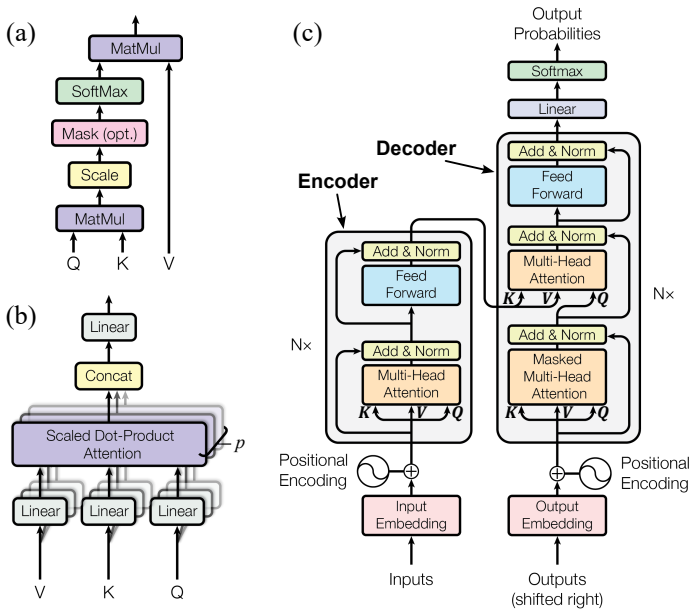

FIGURE 3.8 – (a) Scaled dot-product attention. (b) Multi-head attention. (c) Encoder-decoder structure of the transformer architecture proposed by Vaswani et al. (2017). The encoder uses a MHSA. The decoder comprises a masked MHSA and a MHCA, wherein the value ($\boldsymbol{V}$) and key ($\boldsymbol{K}$) matrices are obtained from the encoder's output, while the query ($\boldsymbol{Q}$) matrix is derived from the decoder. Images modified from Vaswani et al. (2017).

MHSA with p heads, the input matrix $\boldsymbol{X}$ is linearly projected using distinct learnable weight matrices $\{\boldsymbol{W}_{Q_i}, \boldsymbol{W}_{K_i}, \boldsymbol{W}_{V_i}\}$, where $i \in (0, p-1)$, to obtain the queries, keys, and values matrices. The attention function is then performed in parallel by the p heads, producing p output matrices. These outputs are concatenated into a single matrix $[\boldsymbol{Z}_0, \ldots, \boldsymbol{Z}_{p-1}] \in \mathbb{R}^{n \times p \cdot d_v}$ and projected using a weight matrix $\boldsymbol{W}_P \in \mathbb{R}^{p \cdot d_v \times d}$. The multi-head mechanism can be formulated as follows (Vaswani et al., 2017):

$$\text{MultiHead} = \text{Concat}(\boldsymbol{Z}_0, \ldots, \boldsymbol{Z}_{p-1})\boldsymbol{W}_P. \tag{3.25}$$

Another type of attention computation is the cross-attention (CA) mechanism, which allows for attending to information from different input sequences. Unlike the SA operation, where the query, key, and value matrices are computed from the input

matrix $\boldsymbol{X}$ using Eq. (3.21), CA computes these matrices using two different input matrices, $\boldsymbol{X}_Q$ and $\boldsymbol{X}_{KV}$:

$$\boldsymbol{Q} = \boldsymbol{X}_Q \boldsymbol{W}_Q, \qquad \boldsymbol{K} = \boldsymbol{X}_{KV} \boldsymbol{W}_K, \qquad \boldsymbol{V} = \boldsymbol{X}_{KV} \boldsymbol{W}_V. \tag{3.26}$$

Similar to the SA mechanism, CA and multi-head cross-attention (MHCA) are computed using Eqs. (3.22) and (3.25), respectively.

The CA architecture is commonly employed in the decoder component of DL models for machine translation (e.g., Vaswani et al., 2017) and multi-modal tasks, such as visual-question answering and caption-based image retrieval (e.g., Lu et al., 2019).

In the original implementation of the transformer model by Vaswani et al. (2017), an encoder-decoder architecture (see Fig. 3.8c) is utilized to address machine translation tasks. The encoder processes the source sequence using 6 stacked blocks ($N = 6$ in Fig. 3.8c), each consisting of two sub-layers: a MHSA operator and a fully connected feed-forward network, comprising two linear transformations with a ReLU activation in between. Additionally, layer normalization and residual connections (see Section 3.3) are adopted to facilitate gradient flow during backpropagation and to prevent the destruction of important information between layers, respectively (Chollet, 2021). The decoder, which also comprises 6 identical blocks, differs from the encoder due to the inclusion of a masked MHSA layer to process the decoder's input data, thus preventing the extraction of information from future data. Moreover, an additional MHCA operator is introduced in the decoder to integrate information from different input sequences. Specifically, the MHCA layer generates the value ($\boldsymbol{V}$) and key ($\boldsymbol{K}$) matrices by projecting the embedding information from the encoder's output, while the query ($\boldsymbol{Q}$) matrix is derived from the output of the masked MHSA layer of the decoder. Finally, a linear layer and a Softmax function are applied after the decoder's output to generate the final output (see Fig. 3.8c).

Since the transformer model is based solely on the attention mechanism and consequently lacks recurrence or convolution layers, the order of data in the input

sequence is not inherently preserved. To address this, Vaswani et al. (2017) introduced positional encoding (PE) to inject information about the relative or absolute positions of the data in the sequence. Specifically, PE is added to the input embedding at the base of the encoder and decoder blocks (see Fig. 3.8c). The use of an increasing sequence of integers may not be ideal due to the rapid increases in PE values, which can disrupt the ranges of values in the embedding vector (Chollet, 2021). Consequently, Vaswani et al. (2017) adopted sine and cosine functions to define positional embedding, constraining the values within the range $[-1, 1]$. This strategy enables the unique characterization of the position of data in the input sequence using a vector (or matrix for 2D data) of small values. Additionally, learnable positional embeddings can also be utilized.

The success of transformers in NLP has facilitated their rapid development in computer vision tasks, including popular recognition tasks (e.g., image classification, object detection, segmentation, and action recognition), multi-modal tasks (e.g., visual-question answering and visual reasoning), generative modeling, low-level vision (e.g., image super-resolution and image enhancement), video processing (e.g., action recognition, video forecasting), and three-dimensional analysis (e.g., point cloud classification and segmentation) (Khan et al., 2022).

3.6 Video Prediction Transformer (VPTR) model

The Video Prediction Transformer (VPTR) model, proposed by Ye and Bilodeau (2023), is a transformer-based architecture designed for video frames prediction. This model features an efficient transformer block, referred to as Video High-Resolutions Transformer (VidHRFormer), which employs separate spatial and temporal attention mechanisms to reduce the complexity typically associated with standard transformers. A video can be conceptualized as a sequence of consecutive images (i.e., frames) that exhibit spatiotemporal relationships. The primary objective of the VPTR model is to forecast N future frames given L past frames.

The VPTR model comprises three consecutive components (see Fig. 3.9): a CNN encoder, a VPTR module, and a CNN decoder. The encoder extracts visual features from the past frames, the VPTR module predicts the visual features of each future frame based on the past frame features, and the decoder reconstructs the future frames using these predicted visual features.

The encoder and decoder blocks can be considered as components of a classical AE, (see Section 3.4). Specifically, Ye and Bilodeau (2023) adapted the ResNet-based AE from the Pix2Pix model (Isola et al., 2017). Unlike traditional AE models (e.g., the U-Net architecture; see Section 3.4), this particular architecture does not utilize skip connections between the encoder and decoder blocks (i.e., the grey arrows in Fig. 3.7). This configuration is fundamental to prevent any information from bypassing the bottleneck of the AE, where the VPTR block is introduced. Additionally, the use of residual connections could allow the model to produce an output identical to the input data, thereby neglecting temporal variations, which are modelled by the VPTR block.

The VPTR module consists of B consecutive VidHRFormer blocks (depicted in blue in Fig. 3.10). The main layers of the VidHRFormer block include: a local spatial MHSA, which computes spatial attention on local patches of size 4×4; a convolutional feed-forward neural network (Conv FNN), which exchanges spatial information between local patches using convolutional layers; and a temporal MHSA, which models temporal dependencies between frames using a SA mechanism. A MLP

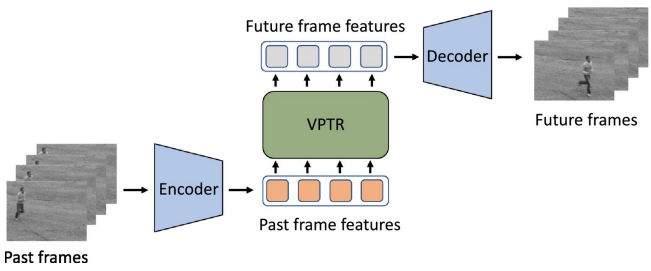

FIGURE 3.9 – Overall workflow of the VPTR model. Image from Ye and Bilodeau (2023).

and normalization layers complete the VidHRFormer block. Furthermore, the spatial attention layer employs a 2D relative PE, as Ye and Bilodeau (2023) demonstrated that it yields better results compared to the absolute PE. Conversely, a fixed absolute 1D PE is used in the temporal MHSA for simplicity.

Ye and Bilodeau (2023) proposed three variants of the VPTR model (see Fig. 3.10), which are distinguished by the structure of the VPTR modules and the method used for predicting the future frames:

1. Fully autoregressive (FAR): in this variant, the prediction of the next frame is conditioned on all previous frames. The VPTR-FAR model consists of B consecutive VidHRFormer blocks (Fig. 3.10a), with an attention mask applied to the temporal MHSA layer. As this model can predict only one frame into the future, an autoregressive (AR) procedure may be employed to forecast a sequence of future frames.

2. Partially autoregressive (PAR): this implementation features an encoder-decoder structure (Fig. 3.10b). The encoder is a standard VidHRFormer block, while the decoder incorporates additional masked temporal MHSA and Conv FNN layers. This configuration is less practical for applications, as predicting future frames in the range $[L+1, L+N]$ requires the frames at instants $[L+1, L+N-1]$ as input data, despite these being unknown.

3. Non-autoregressive (NAR): this variant has a structure similar to the VPTR-PAR implementation, but no masking is applied to the temporal attention layers (Fig. 3.10c). Furthermore, unlike the VPTR-PAR model, the VPTR-NAR uses a trainable query sequence ($q_{L+1}, \ldots, q_{L+N}$ in Fig. 3.10c) for predicting future frames. This model takes the past frames ($\boldsymbol{z}_1, \ldots, \boldsymbol{z}_L$) as input data and predicts all N future frames ($\hat{\boldsymbol{z}}_{L+1}, \ldots, \hat{\boldsymbol{z}}_{L+N}$) in a single step.

The authors showed that the FAR and PAR variants achieve better accuracy in predicting the initial few frames, but performance progressively degrades due to the accumulation of errors resulting from using previously predicted frames to forecast

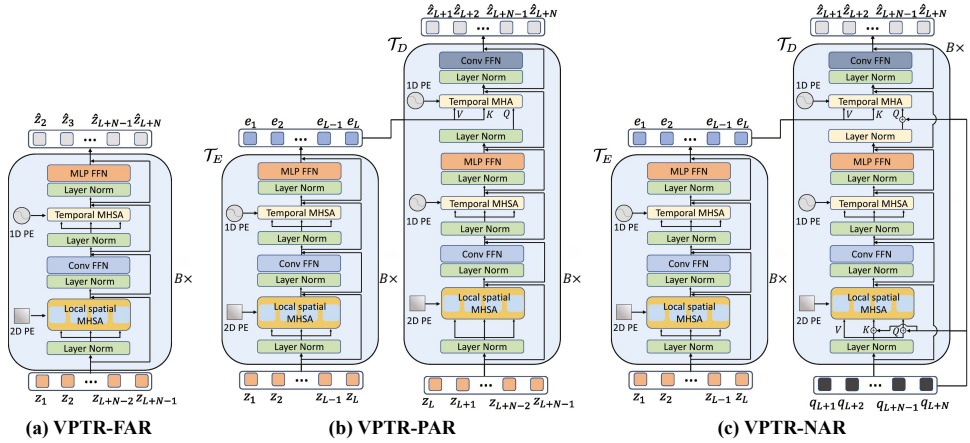

FIGURE 3.10 – Overview of the (a) fully autoregressive (FAR), (b) partially autoregressive (PAR), and (c) non-autoregressive (NAR) implementations of the VPTR module. Image modified from Ye and Bilodeau (2023).

subsequent instants. Conversely, the VPTR-NAR model is not affected by error accumulation, as future frames are predicted in a single step. Additionally, it offers faster inference speed compared to the other variants. However, training the NAR scheme is more complex due to its large number of parameters, necessitating a more sophisticated loss function to effectively capture the spatiotemporal relationships in the input frames (Ye & Bilodeau, 2023). Furthermore, GPU memory and time consumption required for training increase with the raise in the number of forecasted frames. This imposes a limitation on the total number of frames that can be predicted in a single step. Consequently, if the desired number of future frames exceeds this limit, the VPTR-NAR model must be used in an AR mode, using the N predicted frames from the first iteration to forecast the subsequent N frames.

For a comprehensive description of the VPTR model's details, the reader is referred to Ye and Bilodeau (2023).

Chapter 4

FloodSformer model

4.1 Introduction

The FloodSformer (FS) model represents an innovative DL framework designed for real-time forecasting of flood inundation maps with negligible computational time. The model leverages an AE framework to efficiently analyze the spatial information in the input maps and reduce their dimensionality, while the transformer-based VPTR architecture captures the complex spatiotemporal relationships within the data and predicts future maps. The FS model can forecast long-lasting events using an AR procedure, offering significant improvements in accuracy compared to other DL models, and computational efficiency over traditional hydrodynamic models.

This Chapter provides a comprehensive overview of the FS model, detailing its architecture and implementation, dataset generation, training and prediction methodology, hyperparameter definitions, and performance indicators used to assess the effectiveness of the surrogate model.

4.2 Model description

The FS model is a transformer-based DL architecture designed to predict the temporal evolution of inundation maps, emulating the results of classical physically based numerical schemes. Exploiting the analogies between inundation maps and images, a DL model, originally proposed for computer vision tasks, has been adapted to predict inundation maps. Specifically, each water depth map can be conceptualized as a greyscale image with dimensions $H \times W$, where each pixel corresponds to a computational cell of a Cartesian grid. Furthermore, subsequent inundation maps have spatial and temporal correlations with each other, similar to the relationships between consecutive frames in videos. Consequently, the FS model draws inspiration from the VPTR model (see Section 3.6), originally applied for video frame prediction tasks (Ye & Bilodeau, 2023). The FS model consists of three consecutive blocks (see Fig. 4.1a): a 2D CNN encoder, a VPTR module, and a 2D CNN decoder.

The general structure of the encoder and decoder blocks, constituting an AE framework, has been previously presented in Section 3.6. The VPTR module is based on the fully AR version of the VPTR model (see Section 3.6), which comprises B consecutive VidHRFormer blocks (see Fig. 3.10a). For simplicity, unless otherwise specified, the fully AR version of the model will be referred as VPTR.

In this thesis, two different variants of the FS architecture are proposed to address different flood scenarios, specifically, dam-breaks (DBs) (Chapter 5) and river floods (Chapter 6). This distinction is fundamental due to the different types of input information required for their simulation. For both scenarios, the spatiotemporal relationships between consecutive inundation maps are of primary importance for accurately reproducing flood dynamics. Furthermore, the initial condition plays a critical role, particularly for DB studies, where the initial water level in the upstream reservoir defines the amount of water volume stored and, consequently, the severity of the flood event. In contrast, information about time series of upstream BCs is relevant for the analysis of river flood events only, whereas upstream inflow can usually be neglected in DB scenarios. Consequently, the need for different types

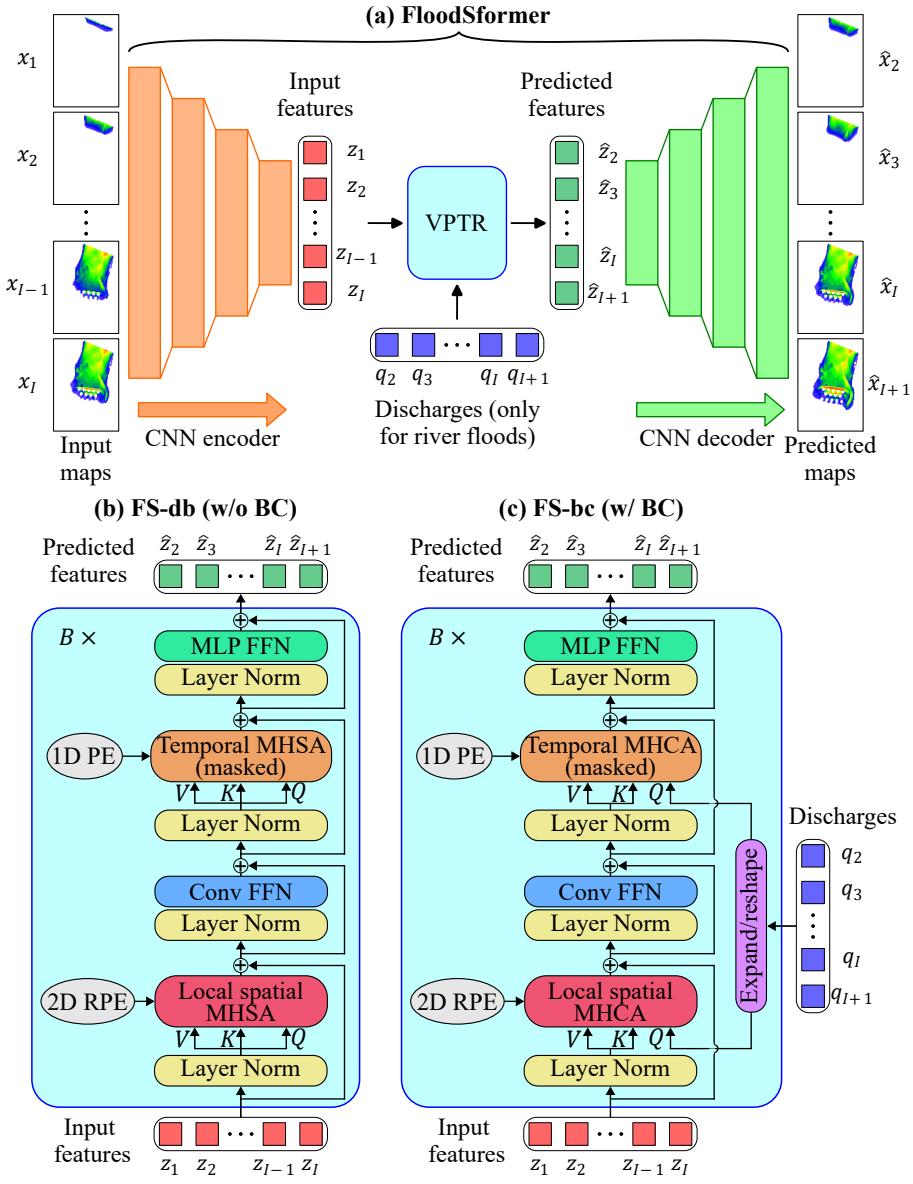

FIGURE 4.1 – Overview of the FloodSformer model architecture. (a) Schematic description of the general workflow of the proposed model. The time series of discharge inflow is used only when addressing river floods (i.e., FS-bc model). (b-c) Description of one of the B VidHRFormer block composing the VPTR module of the (b) FS-db model and (c) FS-bc model. The latter model considers the BC as additional input data.

of input data and the inherent differences in these phenomena (e.g., dam-break floods are characterized by very rapid dynamics, whereas river floods typically exhibit slower progression), necessitate tailored architectural designs for accurate and reliable simulations. However, both models are integrated into a unified code framework, providing a seamless user experience. The end user simply selects the type of phenomenon (dam break or river flood) when running the model. This modular approach balances the need for specialized architectures with practical usability, addressing the specificities of each flood type while maintaining operational simplicity.

The general workflow of the model is represented in Fig. 4.1a. Considering a sequence of $I+1$ water depth maps, the CNN encoder uses the input maps $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_I)$ to extract the spatial information and reduce their dimensionality, creating the latent features $(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_I)$. These features are then passed to the VPTR block. Furthermore, when simulating river floods, the inflow discharges $(q_2, \ldots, q_{I+1})$ are used by the VPTR block as additional input information. The VPTR block predicts the latent features at the next instant $(\hat{\boldsymbol{z}}_2, \ldots, \hat{\boldsymbol{z}}_{I+1})$. Finally, the CNN decoder reconstructs the predicted water depth maps $(\hat{\boldsymbol{x}}_2, \ldots, \hat{\boldsymbol{x}}_{I+1})$ based on the predicted latent features. It is important to emphasize that the VPTR block analyzes different types of information depending on the flood event considered: only water depth maps for DB scenarios, and both water depth maps and inflow discharges for river flood studies. As a result, the two versions of the FS model employ different architectures to analyze the appropriate input information (see Fig. 4.1b,c).

Focusing on the prediction of DB scenarios, the first version of the FS model, referred to as FS-db, uses as input data only a sequence of water depth maps. Consequently, the VidHRFormer block (Fig. 4.1b) comprises a local spatial MHSA and a temporal MHSA, both based on the SA mechanism (Eq. (3.22)). These layers analyze the spatial and temporal correlations between input inundation maps to predict future frames. The key, value, and query matrices are computed using Eq. (3.21), where the input tensor ($\boldsymbol{X}$ in Eq. (3.21)) represents the sequence of latent

features $\boldsymbol{Z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_I] \in \mathbb{R}^{I \times h \times w \times d_{\text{model}}}$ provided by the CNN encoder.

The structure of the FS-db model can be summarized as follows:

$$\begin{aligned} \boldsymbol{z}_t &= Enc\left(\boldsymbol{x}_t\right), && t \in [1, \ldots, I] \\ \hat{\boldsymbol{z}}_t &= \mathcal{T}(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{t-1}), && t \in [2, \ldots, I+1] \\ \hat{\boldsymbol{x}}_t &= Dec\left(\hat{\boldsymbol{z}}_t\right), && t \in [2, \ldots, I+1] \end{aligned} \tag{4.1}$$

where $\boldsymbol{x}_t \in \mathbb{R}^{H \times W \times C}$ and $\hat{\boldsymbol{x}}_t \in \mathbb{R}^{H \times W \times C}$ are the ground-truth and predicted maps at time t, respectively. $\boldsymbol{z}_t \in \mathbb{R}^{h \times w \times d_{\text{model}}}$ and $\hat{\boldsymbol{z}}_t \in \mathbb{R}^{h \times w \times d_{\text{model}}}$ are the latent features at time t in input and output from the VPTR block, respectively. $Enc(\ldots)$, $Dec(\ldots)$ and $\mathcal{T}(\ldots)$ are the encoder, decoder, and VPTR blocks, respectively. H and W are the height and width (in cells) of the water depth maps along the south-north and west-east directions, while $h = H/2^k$ and $w = W/2^k$ are the height and width of the latent features, respectively. k is the number of convolutional layers of the AE, d_{model} is the number of channels of the latent features, and C is the number of channels in the input and output maps, which is equal to 1 in this thesis. The hyperparameter I represents the number of input frames, i.e., the maximum length of the sequence of input maps for the FS model. Correctly defining the value of this hyperparameter is crucial for accurately reproducing flood dynamics (see Section 5.2.3).

As previously mentioned, predicting river floods requires the introduction of the upstream BC as input data to the surrogate model. Consequently, the second version of the FS model, referred to as FS-bc, replaces the SA mechanism with a CA mechanism to facilitate attention across different input sequences, namely latent features of inundation maps and time series of inflow discharge. Fig. 4.1c shows the layers composing the adapted VidHRFormer block. Unlike the FS-db version (Fig. 4.1b), a sequence of inflow discharges $(q_2, \ldots, q_{I+1})$ is passed to the local spatial MHCA and temporal MHCA as the query ($\boldsymbol{Q}$) matrix, while latent features $(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_I)$ are passed as the value ($\boldsymbol{V}$) and key ($\boldsymbol{K}$) matrices. Consequently, for the CA mechanism, in Eq. (3.26) the input tensor $\boldsymbol{X}_{KV}$, used to compute key and value matrices, represents the sequence of latent features $\boldsymbol{Z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_I] \in \mathbb{R}^{I \times h \times w \times d_{\text{model}}}$

in output from the CNN encoder, while the query matrix is computed projecting the $\boldsymbol{X}_Q$ tensor, which is obtained by expanding and reshaping the sequence of discharge values $[q_2, \ldots, q_{I+1}] \in \mathbb{R}^I$ to match the dimensions of $\boldsymbol{X}_{KV}$. Consequently, the VPTR block takes as input the embedded water depth maps $[\boldsymbol{z}_1, \ldots, \boldsymbol{z}_I]$, and the corresponding inflow discharge values $[q_2, \ldots, q_{I+1}]$, and predicts the embedded maps $[\hat{\boldsymbol{z}}_2, \ldots, \hat{\boldsymbol{z}}_{I+1}]$ (see Fig. 4.1c). The temporal shift of one frame between maps and discharge values used as input data (e.g., the MHCA correlates the feature at time $t = 1$ with the discharge at instant $t = 2$) is essential to predict the map at the subsequent instant (e.g., $t = 2$).

The structure of the FS-bc model can be summarized as follows:

$$\begin{aligned} \boldsymbol{z}_t &= Enc\left(\boldsymbol{x}_t\right), && t \in [1, \ldots, I] \\ \hat{\boldsymbol{z}}_t &= \mathcal{T}(\boldsymbol{K}, \boldsymbol{V} : [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{t-1}]; \boldsymbol{Q} : [q_2, \ldots, q_t]), && t \in [2, \ldots, I+1] \\ \hat{\boldsymbol{x}}_t &= Dec\left(\hat{\boldsymbol{z}}_t\right), && t \in [2, \ldots, I+1] \end{aligned} \tag{4.2}$$

where $q_t \in \mathbb{R}$ is the inflow discharge at time t.

A schematic description of the general workflow for dataset generation, training, testing, and real-time prediction of the FS-bc model is represented in Fig. 4.2. For the FS-db version of the model, the workflow differentiates as inflow hydrographs are not used in the dataset generation, which is composed only of sequences of water depth maps.

The dataset generation requires the simulation of a large number of flood events using the physically based hydrodynamic model (Fig. 4.2a). The simulated water depth maps, which represent the ground-truth maps, are used to create the training, validation, and testing datasets. A detailed description of the dataset generation is available in Section 4.3.

The training process of the FS model (Fig. 4.2c) aims to minimize the differences between the inundation maps predicted by the surrogate model and the target maps obtained with the physically based model, using an appropriate loss function. A comprehensive description of the training strategy is available in Section 4.4. Once

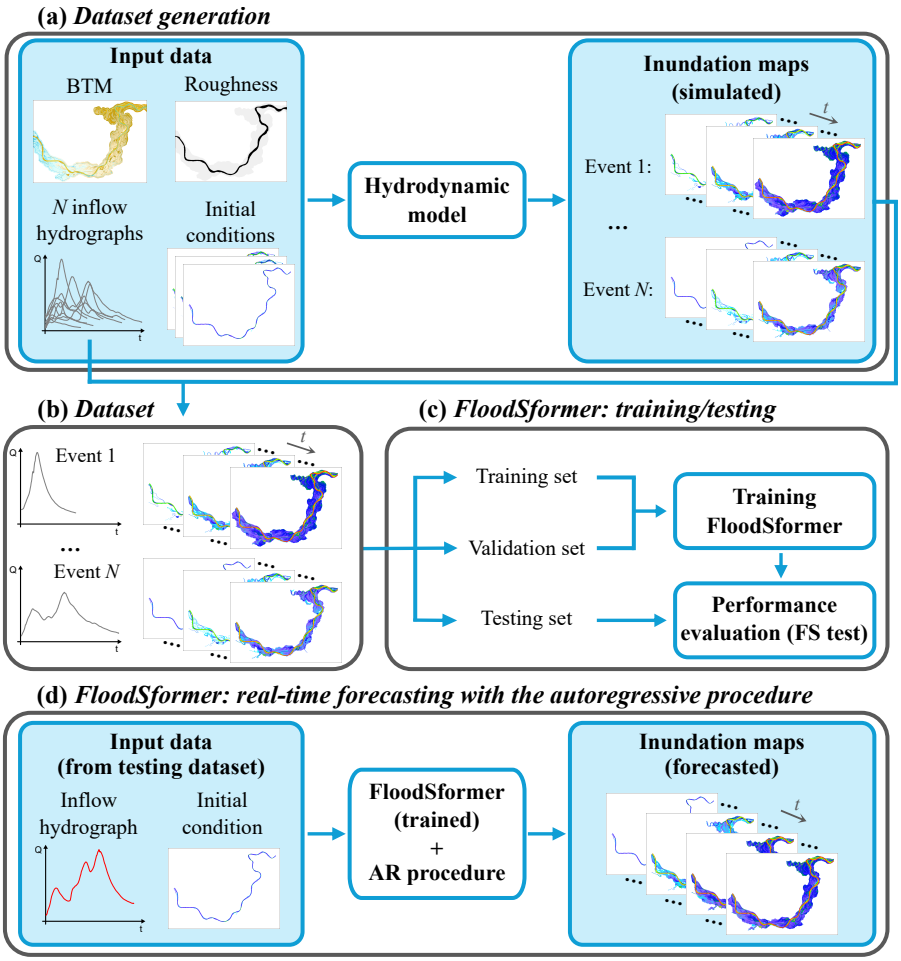

FIGURE 4.2 – Schematic description of the FS-bc model workflow: (a-b) dataset generation, (c) training, testing, and (d) autoregressive procedure for real-time forecasting.

the FS model is trained, the accuracy in forecasting the map one frame ahead is assessed using the unseen samples in the testing dataset. This procedure is referred to as FS test (Fig. 4.2c).

Once the FS model is trained and tested, real-time forecasts of entire flood events is obtained using an AR method (Fig. 4.2d), detailed in Section 4.5. This method employs a recursive technique, iteratively replacing input maps with predicted ones.

Consequently, starting with P past frames, the AR process sequentially forecasts F future frames. The prediction lead time is limited by potential accuracy degradation due to error accumulation during the recursive procedure. As shown in Fig. 4.2d, the AR procedure of the FS-bc model requires the entire time series of upstream discharges as input data. In EWS the inflow time series are commonly derived from physically based rainfall-runoff models. Consequently, the length of the time series of inflow discharges available can be another limiting factor for the total lead time when predicting river floods.

4.3 Dataset generation

The creation of a robust and comprehensive dataset is essential for training any DL model, including the FS, as it directly influences the model's ability to learn, generalize, and perform accurately during inference. Datasets commonly used for training large DL models in NLP and computer vision tasks typically contain from a few thousand to tens of millions of samples. For instance, the ImageNet-21k dataset (Deng et al., 2009), widely adopted for image classification, consists of approximately 14M images subdivided in 21k classes. Creating such extensive datasets requires significant effort, resources, and expertise, which are typically available only to large organizations. However, for specialized models like the FS, targeted efforts in curating and annotating datasets tailored to specific applications can yield highly effective results even with comparatively smaller datasets, especially when employing transfer learning (Goodfellow et al., 2016). Transfer learning is a technique that leverages pre-trained models on large datasets to transfer learned features to a new model, which can then be fine-tuned on a smaller, domain-specific dataset (see Section 4.6). This approach significantly reduces the amount of data and computational resources required to achieve high performance in specific tasks (Aggarwal, 2018).

The dataset used to train and evaluate the FS model was generated by simulating various flood events using the PARFLOOD code (Fig. 4.2a). Numerically

generated inundation maps overcome the challenge of unavailable directly observed maps representing inundation dynamics at adequate temporal resolution. However, the quality of the training data depends on the accuracy of the physically-based model in reproducing flood dynamics. Therefore, adopting an accurate and stable hydrodynamic model is fundamental to ensure the reliability of ground-truth maps.

The dataset differentiates based on the version of the FS model adopted:

- For the forecasting of DB scenarios using the FS-db model, different flood scenarios were simulated by considering various initial water levels in the upstream reservoir. The output maps of one specific simulation were selected to generate the testing dataset, while the maps from the remaining simulations were randomly split to create the training and validation datasets, with a ratio of 95% and 5%, respectively. The dataset samples were obtained by extracting sequences of $I + 1$ consecutive water depth maps using a sliding window with a stride of 1 (see Fig. 4.3a). This procedure was repeated for all numerically simulated events.

  The flood event in the testing dataset was also used to evaluate the accuracy of the FS-db model in predicting DB scenarios using the AR procedure (see Section 4.5). The input data for the AR forecasting consists of a sequence of $P \leq I$ past water depth maps from the initial instants of the flood event, representing the initial condition for the prediction, while the output is the sequence of F future water depth maps, spanning the entire duration of the flood scenario (see Fig. 4.3b).

- For river flood predictions, using the FS-bc model, the water depth maps comprising the dataset were numerically generated by running the PARFLOOD code multiple times, varying the upstream BC to produce different flood events (Fig. 4.2a). The dataset samples consist of sequences of $I + 1$ consecutive inundation maps and the corresponding inflow discharges (see Fig. 4.3a). A sliding window of length $I+1$ was applied over time to extract multiple samples

from each simulation, and this process was repeated for all scenarios. Each flood event with T instants ultimately provided $T - I$ samples. The training and validation datasets were obtained by randomly splitting a large number of numerically simulated events, with a proportion of 95% for training and 5% for validation. In contrast, the testing samples consist of inundation maps for flood events different from those used for training and validation.

The AR forecasting of the FS-bc model (Fig. 4.2d), which uses the flood events in the testing dataset, requires $P \leq I$ water depth maps representing the initial condition and the time series of inflow discharges for the entire event duration (i.e., $P + F$ instants), as represented in Fig. 4.3b.

All samples in the datasets are non-dimensionalized by dividing them by a value slightly higher than the maximum simulated water depth for the specific case study. For river flood studies, the inflow discharges are also normalized using a value exceeding the largest peak discharge expected for the specific case study. Consequently, all the samples in the datasets have values in range of $[0, 1]$. This dataset normalization procedure is a common practice in the development of DL models (Goodfellow et al., 2016).

4.4 Training methodology

Training a DL model involves optimizing its parameters to accurately map inputs to desired outputs, thereby enabling the model to generalize well to new, unseen data. This optimization is typically achieved by minimizing a loss function, which quantifies the discrepancy between the model's predictions and the actual target values. For the FS model, the training process aims to minimize the differences between predicted inundation maps and those generated by the hydrodynamic model, ensuring the surrogate model can effectively simulate flood dynamics.

Given the complex structure of the FS model and the large number of parameters that must be optimized, training the AE and transformer blocks together can be

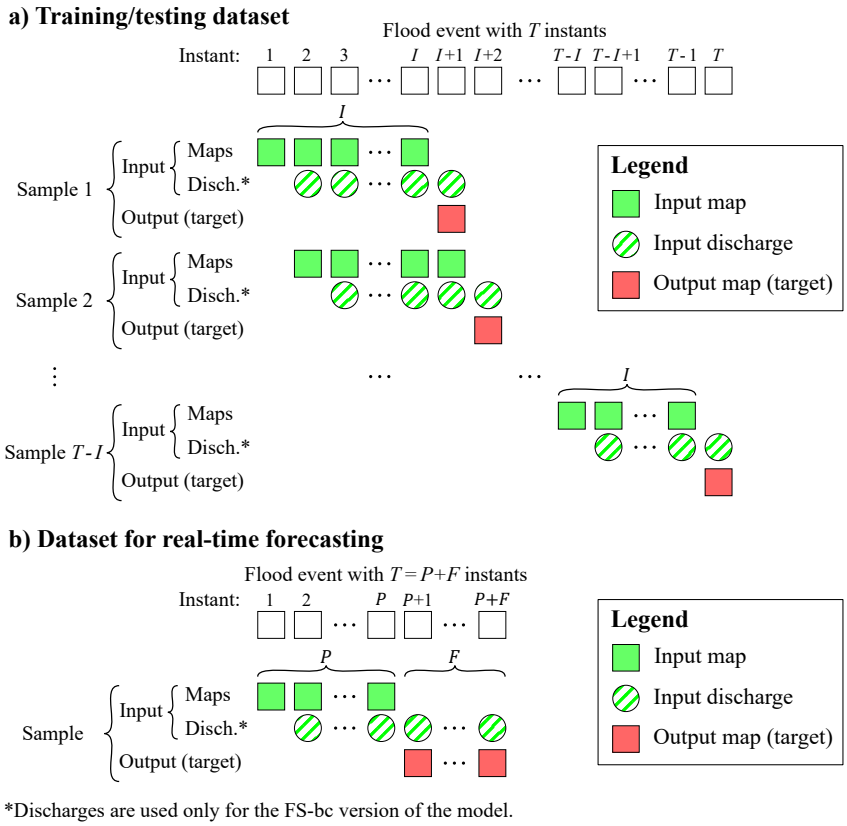

FIGURE 4.3 – Schematic overview of the dataset generation methodology. For clarity, only one flood event is shown in the sketch. The process is repeated across all flood events, and the final dataset consists of the union of all samples generated from the full set of flood events. The inflow discharges are employed only for the generation of the dataset used by the FS-bc model. In the Figure, T represents the total number of maps of an event, while I, P, and F denote the numbers of input maps, past maps, and future maps, respectively. a) Dataset for training, validation, and FS test. b) Dataset for the real-time forecasting of inundation events obtained with the AR procedure.

infeasible. Since the CNN encoder and CNN decoder can be considered as a standalone AE architecture, the most effective method to train the FS model is to divide the procedure into two distinct stages. This approach also reduces the GPU memory demand and computation time of the training process.

The first stage, referred to as "AE training", focuses on training the convolutional

AE, which is obtained by concatenating the CNN encoder and CNN decoder blocks. The primary goal of this training phase is features extraction, achieved by analyzing the spatial information in the input maps. Specifically, the encoder compresses each input map $\boldsymbol{x}_t$ using convolution layers to obtain the latent feature $\boldsymbol{z}_t$. Then, the decoder reconstructs the original map $\hat{\boldsymbol{x}}_t$ from this latent feature. Consequently, this phase of training disregards temporal information and BC information. The batch consists of N randomly selected maps $\boldsymbol{X} \in \mathbb{R}^{N\times H\times W\times 1}$. The corresponding latent features and reconstructed maps are $\boldsymbol{Z} \in \mathbb{R}^{N\times h\times w\times d_{\text{model}}}$ and $\hat{\boldsymbol{X}} \in \mathbb{R}^{N\times H\times W\times 1}$, respectively.

The loss function of the AE training procedure is as follows:

$$\mathcal{L}_{AE} = \mathcal{L}_{MSE} + \lambda_{GDL}\mathcal{L}_{GDL} + \lambda_{GAN} \arg\min_{G}\max_{D} \mathcal{L}_{GAN}(D,G) \tag{4.3}$$

where:

$$\mathcal{L}_{MSE} = \frac{1}{N}\sum_{n=1}^{N} \left(\boldsymbol{x}^n - \hat{\boldsymbol{x}}^n\right)^2 \tag{4.4}$$

$$\begin{aligned}\mathcal{L}_{GDL} = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{W}\sum_{j=1}^{H} \Big(&\big||x^n_{i,j} - x^n_{i-1,j}| - |\hat{x}^n_{i,j} - \hat{x}^n_{i-1,j}|\big|^{\alpha} \\ &+ \big||x^n_{i,j-1} - x^n_{i,j}| - |\hat{x}^n_{i,j-1} - \hat{x}^n_{i,j}|\big|^{\alpha}\Big)\end{aligned} \tag{4.5}$$

$$\mathcal{L}_{GAN}(D,G) = \mathbb{E}_{\boldsymbol{X}}\left[\log D(\boldsymbol{X})\right] + \mathbb{E}_{\hat{\boldsymbol{X}}}\left[\log\left(1 - D(G(\boldsymbol{X}))\right)\right] \tag{4.6}$$

where $\mathcal{L}_{MSE}$, $\mathcal{L}_{GDL}$, and $\mathcal{L}_{GAN}$ represent the mean squared error (MSE), the gradient difference loss (GDL), and the generative adversarial network (GAN) loss, respectively. The GDL is designed to minimize differences between the gradients of water depths in the original ($\boldsymbol{X} = [\boldsymbol{x}^1, \ldots, \boldsymbol{x}^N]$) and reconstructed ($\hat{\boldsymbol{X}} = [\hat{\boldsymbol{x}}^1, \ldots, \hat{\boldsymbol{x}}^N]$) maps. The GAN loss encompasses both a generator G, which represents the AE, and the PatchGAN discriminator D. More information about the PatchGAN discriminator is provided by Isola et al. (2017). The values of the hyperparameters λ_{GDL}, λ_{GAN}, and α are defined in Section 4.6.

The second training phase, named "VPTR training", focuses on training the VPTR

block of the FS model. During this stage, the CNN encoder and CNN decoder blocks are not trained; instead, the weights optimized during the AE training process are utilized. Unlike the first training phase, which analyzes only spatial information, the VPTR training stage considers both temporal and spatial information. Additionally, the training varies based on the version of the FS model used (see Fig. 4.4):

- For the FS-db model (Fig. 4.4a), which considers only water depth maps as input data, each batch comprises a sequence of $I + 1$ inundation maps $\boldsymbol{X} \in \mathbb{R}^{N\times(I+1)\times H\times W\times 1}$.

- During the FS-bc model training (Fig. 4.4b), time series of inflow discharge are also used to predict future frames. Consequently, a batch consists of the sequence of $I + 1$ inundation maps $\boldsymbol{X}$ and the corresponding inflow discharges $\boldsymbol{q} \in \mathbb{R}^{N\times(I+1)}$ shifted by one time step.

During the VPTR training phase, the trained encoder extracts latent features $[\boldsymbol{z}_1, \ldots, \boldsymbol{z}_I]$ from the input maps $[\boldsymbol{x}_1, \ldots, \boldsymbol{x}_I]$. These features are utilized by the

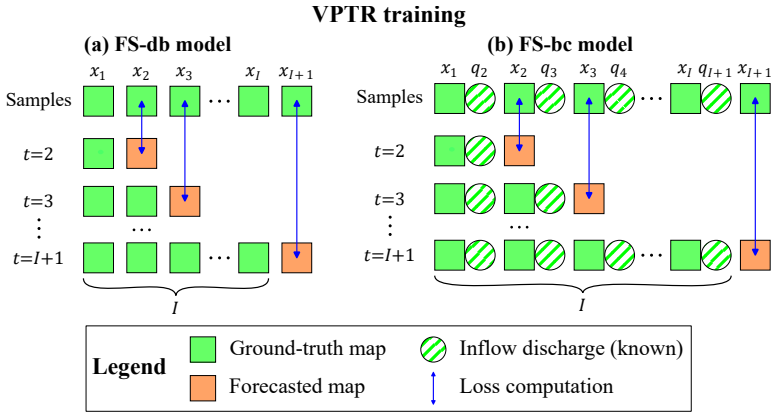

FIGURE 4.4 – Sketch of the VPTR training process for the (a) FS-db model and (b) FS-bc model. The squares represent the water depth maps, while the circles represent the discharge values. For simplicity, only the input and output maps are illustrated, while latent feature computations are neglected. For the FS-bc model, the prediction of the frame at instant $t = I + 1$ (red square) is achieved by considering the ground-truth maps at time $t \in [1, I]$ (green squares) and the discharge values at instants $t \in [2, I + 1]$ (green circles) as inputs.

VPTR block, in conjunction with the inflow discharge values $[q_2, \ldots, q_{I+1}]$ (only in the FS-bc model), to predict the latent features $[\hat{\boldsymbol{z}}_2, \ldots, \hat{\boldsymbol{z}}_{I+1}]$ (see Fig. 4.4). These features are then used to reconstruct the predicted maps $[\hat{\boldsymbol{x}}_2, \ldots, \hat{\boldsymbol{x}}_{I+1}]$ through the previously trained decoder.

The VPTR training procedure aims to minimize the following loss function:

$$L_{VPTR} = \frac{1}{I}\sum_{t=2}^{I+1}(\mathcal{L}_{MSE}(\boldsymbol{x}_t, \hat{\boldsymbol{x}}_t)) + \lambda_{GDL}\frac{1}{I}\sum_{t=2}^{I+1}(\mathcal{L}_{GDL}(\boldsymbol{x}_t, \hat{\boldsymbol{x}}_t)) \tag{4.7}$$

where the $\mathcal{L}_{MSE}$ and $\mathcal{L}_{GDL}$ are computed using Eqs. (4.4) and (4.5), respectively.

Unlike the training loss used for the AE (Eq. (4.3)), the VPTR training loss (Eq. (4.7)) incorporates a summation over time, accounting for the I predicted maps within the range $2 \leq t \leq I + 1$.

For practical applications, the predicted frames within the range $2 \leq t \leq I$ are typically less significant during the forecasting process since the corresponding target maps are already known and included in the input sequence. However, incorporating these forecasted frames into the loss computation enhances prediction performance. This approach also enables the AR procedure to use fewer maps as initial conditions (i.e., the hyperparameter P) compared to the I frames used during training (see Section 4.5).

4.5 Autoregressive prediction

Once trained and tested, the FS model can be utilized for real-time forecasting of inundation maps. As illustrated in Fig. 4.1, the trained model can predict only one frame ahead (i.e., at instant $t = I + 1$). However, for practical applications, it is crucial that the model can forecast multiple future maps to provide sufficient lead time for the adoption of emergency measures in the event of flood alerts. To address this, an autoregressive (AR) inference procedure has been implemented for the prediction of future maps. This methodology uses the predicted maps as input

data for forecasting subsequent maps, thereby enabling the prediction of many future frames (see Fig. 4.2d).

The input data for the AR procedure depend on the type of flood event considered (see Figs. 4.5 and 4.6). Specifically, when simulating DB scenarios (i.e., using the FS-db model), the AR procedure uses only a sequence of P past frames as input data, representing the initial condition of the prediction (green squares in Fig. 4.5). The hyperparameter P defines the number of maps used as the initial condition, and its value must be between 1 and I. A reduced value of P decreases the prior information required for the AR prediction. The ability to use a value of $P < I$ stems from the structure of the AR procedure and the range of predictions provided by the FS model. Specifically, as detailed in Eqs. (4.1) and (4.2), the model forecasts not only the map at instant $t = I + 1$, but also the maps within the range $2 \leq t \leq I$. As a result, the maps predicted at time steps $t < I + 1$, while redundant during training, are used during the initial iterations of the AR procedure when $P < I$ (Fig. 4.5). The influence of the hyperparameter P on prediction accuracy is analysed in Section 5.4.3.

In the AR procedure of the FS-db model, the prediction of F future frames is obtained as depicted in Fig. 4.5. Initially, the ground-truth maps of the P past frames $[\boldsymbol{x}_1, \ldots, \boldsymbol{x}_P]$ (green squares in Fig. 4.5) are used to predict the first unknown future map $\hat{\boldsymbol{x}}_{P+1}$ (red square at $t = P + 1$). In the second step, the forecasted map $\hat{\boldsymbol{x}}_{P+1}$ (orange square at $t = P + 2$) is concatenated at the end of the past frames. This new sequence is then fed to the FS model to predict the next future map $\hat{\boldsymbol{x}}_{P+2}$ (red square at $t = P + 2$). This procedure is repeated until all the F future frames are predicted. It is important to underline that the maximum number of frames that can be used as input data by the FS model is constrained to I, the hyperparameter defined during the training procedure (see Section 4.4). Consequently, starting from the prediction of the map at time $t = I + 2$ (i.e., when the sum of past frames P and concatenated ones exceeds I), the oldest maps are discarded to constrain the length of the input sequence to I. Furthermore, after $I + P$ iterations, all the

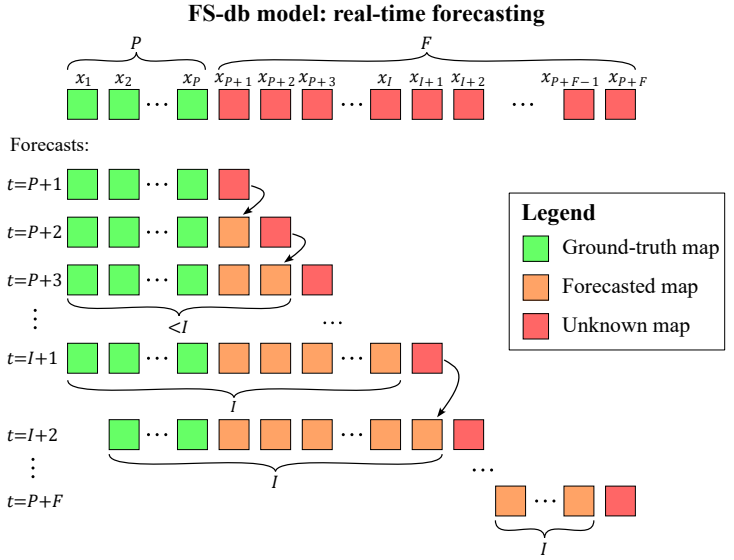

FIGURE 4.5 – Sketch of the AR procedure of the FS-db model. The squares represent the water depth maps. Each forecasted future map (orange squares) is used to make the prediction of the next future map (red squares). Starting from the prediction of the map at instant $t = I+2$, a sliding window is introduced to constrain the length of the input sequence to I. In this illustration $P < I < F$ is assumed and, for simplicity, only the input and output maps of the prediction are illustrated, neglecting the latent feature computations.

ground-truth maps have been discarded, and all subsequent frames are predicted using only previously forecasted maps.

Generally, the AR procedure of the FS-db model can be summarized as follows:

$$\hat{\boldsymbol{x}}_j = \begin{cases} Dec\left(\mathcal{T}(Enc([\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{j-1}]))\right) & \text{if } P+1 < j \leq I+1 \\ Dec\left(\mathcal{T}(Enc([\boldsymbol{x}_{j-P}, \ldots, \boldsymbol{x}_P, \hat{\boldsymbol{x}}_{P+1}, \ldots, \hat{\boldsymbol{x}}_{j-1}]))\right) & \text{if } I+1 < j \leq I+P \\ Dec\left(\mathcal{T}(Enc([\hat{\boldsymbol{x}}_{j-I}, \ldots, \hat{\boldsymbol{x}}_{j-1}]))\right) & \text{if } j > I+P \end{cases} \quad (4.8)$$

The AR procedure of the FS-bc model differs from the aforementioned approach due to the inclusion of the BC as additional input information. Consequently, in addition to the sequence of P past frames, the sequence of upstream discharge values for the entire duration of the flood event (i.e., $P + F$ values) is used for the AR

procedure. As shown in Fig. 4.6, each water depth map $\boldsymbol{x}_t$ is associated with the inflow discharge at the subsequent instant q_{t+1}. It is important to note that the entire sequence of upstream inflow must be provided as input data to the surrogate model (represented by green circles in Fig. 4.6). This requirement stems from the physical nature of the phenomenon being modeled, rather than a limitation of the proposed surrogate model.

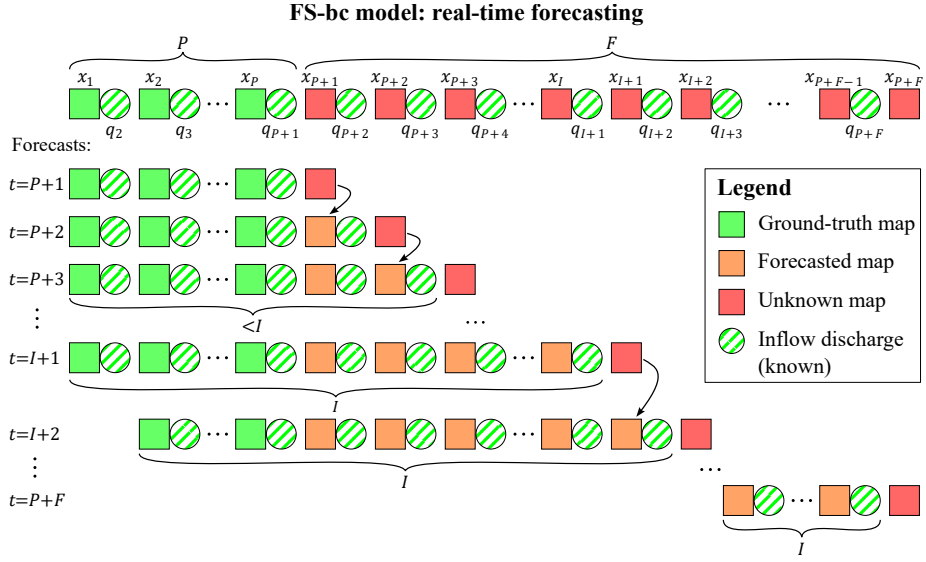

FIGURE 4.6 – Sketch of the AR procedure of the FS-bc model. The squares represent the water depth maps, while the circles represent the discharge values. The inflow hydrograph spanning the entire duration of the prediction must be known during the AR procedure. Each forecasted future map (orange squares) is concatenated with the inflow discharge (green circles) of the subsequent instant and used to predict the next future map (red squares). Starting from the prediction of the map at instant $t = I + 2$, a sliding window is introduced to constrain the length of the input sequence to I. In this illustration $P < I < F$ is assumed and, for simplicity, only the input and output maps of the prediction are illustrated, neglecting the latent feature computations.

For the FS-bc model, the AR procedure is based on the following formulation:

$$\hat{\boldsymbol{x}}_j = \begin{cases} Dec(\mathcal{T}(\boldsymbol{K}, \boldsymbol{V} : Enc([\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{j-1}]); \\ \qquad \boldsymbol{Q} : [q_2, \ldots, q_j])) & \text{if } P+1 < j \leq I+1 \\ Dec(\mathcal{T}(\boldsymbol{K}, \boldsymbol{V} : Enc([\boldsymbol{x}_{j-P}, \ldots, \boldsymbol{x}_P, \hat{\boldsymbol{x}}_{P+1}, \ldots, \hat{\boldsymbol{x}}_{j-1}]); \\ \qquad \boldsymbol{Q} : [q_{j-P+1}, \ldots, q_j])) & \text{if } I+1 < j \leq I+P \\ Dec(\mathcal{T}(\boldsymbol{K}, \boldsymbol{V} : Enc([\hat{\boldsymbol{x}}_{j-I}, \ldots, \hat{\boldsymbol{x}}_{j-1}]); \\ \qquad \boldsymbol{Q} : [q_{j-I+1}, \ldots, q_j])) & \text{if } j > I+P \end{cases} \tag{4.9}$$

For DB studies, the number of future frames that the FS-db model can predict with an acceptable accuracy is limited by the error accumulation due to the AR procedure. Conversely, for river flood predictions (i.e., when using the FS-bc model), the availability of a sufficiently extended forecast period for the inflow hydrograph may also impose an additional constraint on the number of future frames that can be predicted.

4.6 Implementation details and hyperparameters definition

The FS model was developed as a tool for real-time forecasting of inundation maps over bathymetry of real domains. Typically, these maps can contain millions of cells, especially when large domains are discretized with a high spatial resolution. In contrast, the original implementation of the VPTR model by Ye and Bilodeau (2023) predicted video frames with dimensions of 64×64 pixels. Consequently, the FS model must handle maps that are 2-3 orders of magnitude larger than video frames. It is therefore essential to modify the structures of the CNN encoder and CNN decoder blocks to increase the compression rate of the input maps. This aims to limit the dimensionality of the latent features input to the VPTR block, ensuring the training remains feasible in terms of memory and computational time requirements.

To address this, the numbers of convolutional layers (k) in the AE block and the number of channels in the latent features (d_{model}) were increased compared to Ye and Bilodeau (2023), depending on the dimensions of the maps for the specific case study. Each convolutional layer uses a kernel size (L) of 3×3, a stride (S) of 2×2, and zero padding (P) of 1×1 (refer to Section 3.3 for definitions of these hyperparameters). Consequently, each layer of the encoder halves the input map size (see Eq. (3.17)) while doubles the number of channels. Thus, a higher value of k increases d_{model} while decreasing the height and width of the latent features to $h = H/2^k$ and $w = W/2^k$, respectively. These adaptations, combined with the separation of spatial and temporal MHSAs and the use of local patches in the spatial MHSA, are crucial to overcoming the quadratic complexity of the SA mechanism (Ye & Bilodeau, 2023).

Following the original VPTR implementation, the corresponding block structure in the FS model includes 12 consecutive VidHRFormer blocks. In the MHSA and MHCA computations, the number of parallel attention heads (p) is set to 8, and the local patch size is 4×4. The output layer of the FS model uses a Sigmoid activation function (see Fig. 3.2a and Eq. (3.2)), returning values within the range $[0, 1]$. As the dataset samples are normalized in the same range (see Section 4.3), the formation of non-physical negative water depths is automatically prevented.

As discussed in Section 4.4, the training process consists of two stages. For the AE training, the first phase, the Adam optimizer (see Section 3.2.2) with beta values set to (0.5, 0.999) and learning rates of 2×10^{-4} and 10^{-5} , depending on the case study, were adopted. Differently, for the VPTR training process, the second phase, an AdamW optimizer (see Section 3.2.2) with beta values of (0.9, 0.999), a weight decay factor of 0.01, and a learning rate of 10^{-3} were employed. The batch size, defined through a trial-and-error approach during the training process, ranged from 4 to 12, depending on the case study. The number of input frames (I), fundamental hyperparameters with physical significance (representing the number of precedent time steps used by the model for predictions), was optimized through a sensitivity

analysis performed in Section 5.2.3.

For the training loss computation (Eqs. (4.3) and (4.7)), the values of λ_{GDL} and λ_{GAN} were determined through a trial-and-error procedure. The optimized values depend on the training phase and the case study. Generally, λ_{GDL} is calibrated to ensure $\mathcal{L}_{MSE}$ (Eq. (4.4)) and $\mathcal{L}_{GDL}$ (Eq. (4.5)) losses are of the same order of magnitude. Differently, the value of λ_{GAN} varies during the AE training. Specifically, until the $\mathcal{L}_{GAN}$ loss (Eq. (4.6)) converges, λ_{GAN} is set to 0.1 or 0.01. After convergence, λ_{GAN} is set to 0, excluding the $\mathcal{L}_{GAN}$ loss from the total loss computation. The number of epochs required for $\mathcal{L}_{GAN}$ convergence varies depending on the AE model size, typically ranging between 5 and 50 epochs. α is set to 1 in the $\mathcal{L}_{MSE}$ loss computation.

Training a large DL model from scratch presents considerable challenges and often results in low predictive performance, especially when data resources are constrained, as is common with domain-specific datasets (see Section 4.3). Consequently, the FS model was trained using the transfer learning, which involves initializing the surrogate model weight with those pre-trained on a larger dataset. Specifically, pre-trained weights from Ye and Bilodeau (2023), trained on the MovingMNIST dataset composed of thousands of video frame sequences, were employed. This configuration minimizes training costs and addresses the overfitting problem (see Section 3.2.2).

To further mitigate overfitting and improve the performance of the AE prediction of the surrogate model, an early stopping technique with automatic training restarts was implemented. This approach halts the training process if the metric computed on the validation dataset does not improve after a specified number of epochs, subsequently restarting it for a predefined number of iterations.

4.7 Performance indicators

The accuracy of the FS model is assessed by comparing the water depth maps generated by the surrogate model with those produced by the hydrodynamic model,

which serve as the ground truth. Two performance metrics are used: the RMSE, a regression metric that measures the discrepancies between the target and predicted maps, and the F1 score, a classification metric that evaluates the surrogate model's ability to accurately predict the extent of flooded areas. These metrics are defined as follows:

$$\mathrm{RMSE}_t = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(y_t^n - \hat{y}_t^n)^2} \quad \text{for} \quad t \in [1, T] \tag{4.10}$$

$$\mathrm{F1} = 2 \times \frac{\mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} = \frac{2 \times \mathrm{TP}}{2 \times \mathrm{TP} + \mathrm{FN} + \mathrm{FP}} \tag{4.11}$$

where:

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{4.12}$$

$$\mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}. \tag{4.13}$$

In Eq. (4.10), y_t^n and $\hat{y}_t^n$ represent the ground-truth and forecasted water depths in the n-th wet cell at time t, respectively. N denotes the number of wet cells in the map, defined as cells with water depths exceeding a specified threshold ϵ_{wet}. T represents the total number of temporal frames analyzed. In Eqs. (4.11)–(4.13), TP (true positives) indicates the number of cells correctly predicted as flooded, FN (false negatives) denotes the number of cells incorrectly predicted as non-flooded, and FP (false positives) represents the cells incorrectly predicted as flooded.

As previously mentioned, a water depth threshold ϵ_{wet} was used to distinguish between wet and dry cells in both the predicted and ground-truth maps during the calculation of metrics. This method was chosen to avoid artificially lowering the RMSE in cases where the maps predominantly consist of true negatives (TN; i.e., dry cells in both the target and predicted maps), which would result in zero errors. This threshold should be a small percentage of the maximum expected water depth in the dataset, denoted by H_{max} (e.g., $\frac{\epsilon_{wet}}{H_{max}} \approx 0.5\%$). This approach ensures that almost-dry cells are excluded from the computations, and that only cells classified as TP, FP, and FN are included in the RMSE calculation (Eq. (4.10)).

When analysing river floods, the testing dataset is composed of 3-4 flood events with different intensities. Consequently, to facilitate comparison across varying flood magnitudes, a non-dimensional RMSE was calculated by dividing the RMSE at a given time t by the average water depth of the ground-truth map at the same time:

$$\text{RMSE\_ND}_t = \frac{\text{RMSE}_t}{\frac{1}{N}\sum_{n=1}^{N} y_t^n}. \tag{4.14}$$

Chapter 5

Dam-break scenarios

5.1 Introduction

In this Chapter, the effectiveness of the FS-db model in predicting dam-break (DB) scenarios is evaluated through three distinct case studies (Table 5.1). The first involves simulating a DB event in a channel with a parabolic cross section (Section 5.2). The second case examines the forecasting of flood dynamics for a laboratory-scale DB within a rectangular tank (Section 5.3). The final case involves the hypothetical failure of the flood-control reservoir dam of the Parma River (Section 5.4). While the first two test cases feature simple geometry and bathymetry, the last case involves a real bathymetry and urban environments, resulting in a more complex flood evolution.

The results of the autoregressive prediction of dam-break scenarios are firstly presented with the number of past frames (P) equal to the number of input frames (I). Subsequently, for the Parma River case study, the performance of the FS-db model in predicting a dam-break scenario on a real bathymetry using only the water depth map in the upstream reservoir prior to the dam break as past frame (i.e., $P = 1$) is analyzed (see Section 5.4.3). The ability to utilize solely the initial water level for forecasting entire dam-break scenarios significantly enhances the potential

applicability of the proposed model in EWS, as it reduces the amount of prior information required for predicting such flood events.

As outlined in Section 4.3, the dataset for each case study was generated using the PARFLOOD code, varying the initial water levels in the upstream reservoir across different simulations (see Table 5.2). For all simulations, both inflow into the reservoir and initial streamflow conditions in the downstream watercourse were neglected, assuming a dry floodable area at the start of the event. This simplification is widely accepted in the study of dam-break scenarios (Rizzo et al., 2023).

A specific simulation's output maps were selected as samples for the test dataset, while the maps from the remaining simulations were randomly divided into training and validation datasets (see Table 5.2). The flood scenario in the test dataset was also employed for the real-time forecasting of the entire event using the AR procedure.

As detailed in Section 4.2, once the surrogate model is trained, the FS test procedure is applied to evaluate its performance in forecasting the inundation map one time step ahead. For forecasting entire flood events, the AR procedure is employed. Since this method uses the predicted map at one time step to forecast the subsequent ones, the performance metrics calculated on the recursively predicted maps are worse than those from the FS test. It is crucial to emphasize that the AR procedure is the one used for real-time flood forecasting, making its performance metrics more significant than those of the FS test.

Each case study utilized a distinct temporal interval between the inundation maps composing the dataset (Table 5.1). Selecting an appropriate temporal resolution between consecutive maps is crucial to ensure that dynamic changes are captured accurately, avoiding abrupt or overly gradual transitions. This aspect is related to the physics of the type of event studied. Specifically, given the rapid propagation velocity of DB floods, a small temporal resolution is necessary. Conversely, river flood events, which typically exhibit slower dynamics, allow for a longer temporal resolution (see Chapter 6).

Table 5.1 also details the water depth threshold ϵ_{wet} employed to differentiate

TABLE 5.1 – Dam-break scenarios: case studies summary and surrogate model hyperparameters.

	Parabolic channel	Rectangular tank	Parma River flood	
			Res. 20 m	Res. 10 m
Case study number	1	2	3a	3b
Spatial resolution [m]	10	0.01	20	10
Number of cells	122,880	32,768	114,688	458,752
Temporal resolution [s]	20	0.02	120	
Maximum water depth in test dataset [m]	10.0	0.15	14.2	
Depth normalization value [m]	16.0	0.22	16.0	
ϵ_{wet} (for metric computation) [m]	0.05	5×10^{-4}	0.05	
CNN layers (k)	4	4	4	5
Latent feature size ($h \times w \times d_{\text{model}}$)	$60 \times 8 \times 512$	$16 \times 8 \times 512$	$28 \times 16 \times 512$	$28 \times 16 \times 768$
AE parameters [million]	45.5	45.5	45.5	105
VPTR parameters [million]	129	90	125	244

between wet and dry cells during the computation of performance metrics (as described in Section 4.7) for each case study. It is important to note that while cases 1 and 3 are field-scale simulations, the case 2 is conducted at the laboratory scale, resulting in significantly lower water depths. Accordingly, certain parameters, including spatial and temporal resolutions, normalization factors, and the wet-dry threshold (ϵ_{wet}), have been appropriately scaled. Furthermore, also the results and errors associated to each case study are characterized by different orders of magnitude.

This Chapter focuses on presenting the results in terms of model accuracy, while details on the computational times are provided in Section 7.1. Furthermore, the case studies and results presented here are derived from Pianforini et al. (2024e).

TABLE 5.2 – Training configurations for DB studies. The training and validation datasets were obtained by dividing the total number of samples with a ratio of 95% for the training and 5% for the validation. The flood event included in the testing datasets differ from those used to generate the training datasets. The column "Initial water levels" indicates the level range (e.g., 87-95) and the uniform interval (e.g., every 2 m) at which levels are sampled within the range.

DB case study	Case study number	Training and validation		Testing		AR prediction
		Initial water levels [m a.s.l.]	# of samples	Initial water levels [m a.s.l.]	# of samples	
Parabolic channel	1	87-95 every 2 m	530	90	106	Testing event
Tank	2	0.06-0.20 every 0.02 m	1138	0.15	121	
Parma River	3a-3b	98-100-102 104-105.6	560	105	91	

5.2 Dam-break in a parabolic channel

5.2.1 Case study description

The first test case involves a DB scenario in a non-horizontal channel with a parabolic cross-section. This case was chosen due to its simple geometry, which, combined with the symmetry and predominant flow direction, provides a controlled environment to assess the performance of the surrogate model under simplified conditions. Additionally, the straightforward nature of this case study facilitates sensitivity analyses on the model's hyperparameters, enabling a focused evaluation of their impact before applying the model to more complex, real-world scenarios.

The domain measures 9600 m in length and 1280 m in width, with a channel slope of 1% (Fig. 5.1). The terrain profile is defined by the following equation:

$$z = 96 + 0.0001x^2 - 0.01y, \quad x \in [-640, 640], \quad y \in [0, 9600]. \tag{5.1}$$

The parabolic channel is divided into two regions by a vertical gate located at $y = 1600$ m (Fig. 5.1). The upstream area ($0 < y < 1600$ m) represents the reservoir,

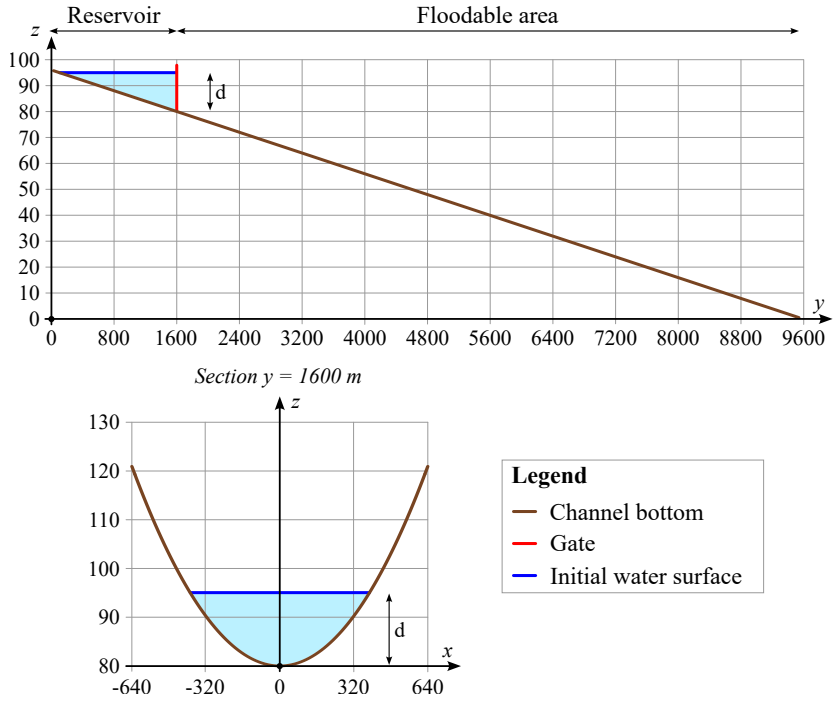

FIGURE 5.1 – Case study 1: sketch of the non-horizontal channel with a parabolic cross section. All the measurements are in meters.

while the downstream region ($1600 < y < 9600$ m) represents the floodable channel. The 12.3 km^2 domain is uniformly discretized with a 10-m resolution grid, resulting in 122,880 cells.

For the numerical simulation of this synthetic case study, a uniform Manning roughness coefficient of 0.05 $sm^{-1/3}$ was adopted. This value is consistent with the friction characteristics of natural channels (Chow, 1959).

The initial condition was set by imposing a constant water surface elevation in the upstream reservoir, with the downstream channel initially dry. Additionally, a uniform-flow rating curve was applied at the downstream boundary. An instantaneous gate removal was assumed to simulate DB scenarios.

For generating the training and validation datasets, five numerical simulations were conducted with initial water surface elevations in the upstream reservoir ranging

from 87 m to 95 m a.s.l. in increments of 2 m (Table 5.2). This corresponds to maximum water depths upstream of the gate ranging from 7 m to 15 m (labeled as d in Fig. 5.1). In contrast, the testing dataset samples were obtained from a hydrodynamic simulation performed with an initial water surface elevation of 90 m a.s.l. (i.e., $d = 10$ m) in the upstream reservoir.

All simulations were terminated shortly after the upstream basin was emptied and the flood reached the downstream boundary. The total simulation time ranged from 25 and 50 minutes, depending on the initial water level in the upstream reservoir. To ensure accurate flood dynamics representation, output maps were generated with a temporal resolution of 20 s.

5.2.2 Results

The results for the first case study are summarized in Table 5.3, which presents the average performance metrics (RMSE and F1 score) obtained by setting $P = I = 8$ and $F = 98$, which correspond to a lead time of about 33 minutes. Although this lead time is insufficient for evacuation purposes from a forecasting perspective, the case study is synthetic and serves solely to validate the FS-db model.

Fig. 5.2 illustrates the comparison of RMSEs values between the FS test and AR prediction of all 98 future maps. In the FS test (represented by the black line), the RMSE varies between 0.4 cm and 2.5 cm. The rapid flood dynamics and the

Table 5.3 – Performance metrics for the FS test and AR forecasting using the FS-db model across each DB-related case study. The number of frames is set to $I = P = 8$, with F values of 98, 113, and 83 for the three respective test cases.

Case study number	FS test		AR procedure	
	RMSE [cm]	F1 [-]	RMSE [cm]	F1 [-]
1	0.7	0.998	8.4	0.986
2	0.03	0.998	0.29	0.994
3a	4.2	0.967	10.4	0.937
3b	5.9	0.928	15.4	0.886

high velocity of the wet/dry front propagation in the initial moments after the gate removal result in higher errors in the first 10 predicted maps (i.e., frames in the range of 8-17). Subsequently, the surrogate model improves its accuracy, reducing errors near the wet/dry front as the flood evolution velocity gradually decreases. The average RMSE of 0.7 cm constitutes a small fraction of the maximum water depth in the testing dataset (i.e., 10 m). Thus, it can be concluded that the SA mechanism within the VPTR block effectively extracts spatiotemporal information from latent features to predict maps one instant ahead with good accuracy. The FS-db model's ability to accurately identify flooded areas in this case study is further confirmed by the extremely high average F1 score (Table 5.3).

Focusing on the results of the AR procedure, the red line in Fig. 5.2 shows the RMSE computed for each recursively predicted future map, while Fig. 5.3 compares the ground-truth and predicted maps at representative instants.

As shown in Fig. 5.2, during the recursive prediction of the first 15 future maps (i.e., frames in range 9-23), the RMSE increases from 2.5 to 12.5 cm, due to the higher errors encountered when forecasting the initial future frames, similar to those observed in the FS test (black line), and the accumulation of errors, which progressively increases the discrepancies between the target and predicted maps. Starting approximately from frame 24 until frame 72, the RMSE gradually decreases due to the reduced flood propagation velocity and the corresponding lower error near the wet/dry front. Subsequently, the RMSE rises again due to error accumulation until frame t=100, when the flood inundation reaches the downstream boundary, resulting in the disappearance of the wet/dry front and a sudden RMSE drop of about 5 cm. The average RMSE for all 98 predicted maps is reported in Table 5.3.

Fig. 5.3 indicates that the differences between the ground-truth and predicted maps generally remain below 25 cm, except near the flood front, where local differences can reach up to 80 cm at certain instants. Nevertheless, these errors are acceptable for real-time forecasting purposes. Additionally, the FS-db model demonstrates a strong ability to accurately reproduce the symmetry of the inundation event. It is

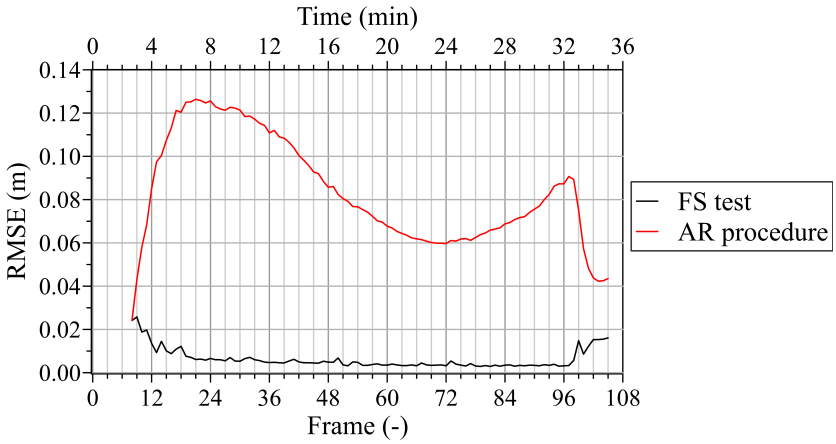

FIGURE 5.2 – Case study 1: RMSE for the FS test and AR prediction using the FS-db model. The number of frames is set to $I = P = 8$ and $F = 98$.

noteworthy that no explicit information regarding symmetry was provided to the surrogate model during training.

As anticipated, the AR procedure of the FS model results in higher RMSEs compared to the single-frame prediction of the FS test, due to error accumulation. Nevertheless, the average F1 score only slightly decreases for the recursive prediction (see Table 5.3), maintaining a value close to one.

5.2.3 Sensitivity analysis to the number of input frames

To determine the optimal value of the hyperparameter I, representing the number of input frames, a sensitivity analysis was conducted by training the FS-db model six times using values of I equal to 1, 2, 4, 8, 12, and 16. This analysis was performed using the case study 1. Since the number of input frames only affects the operation of the VPTR module, the same optimized AE parameters were maintained across all configurations. To evaluate the optimal configuration, the RMSE and the computational time of the training process were used as performance metrics.

Generally, reducing the number of input frames decreases the GPU memory consumption, allowing for an increase in the batch size. Preliminary experiments

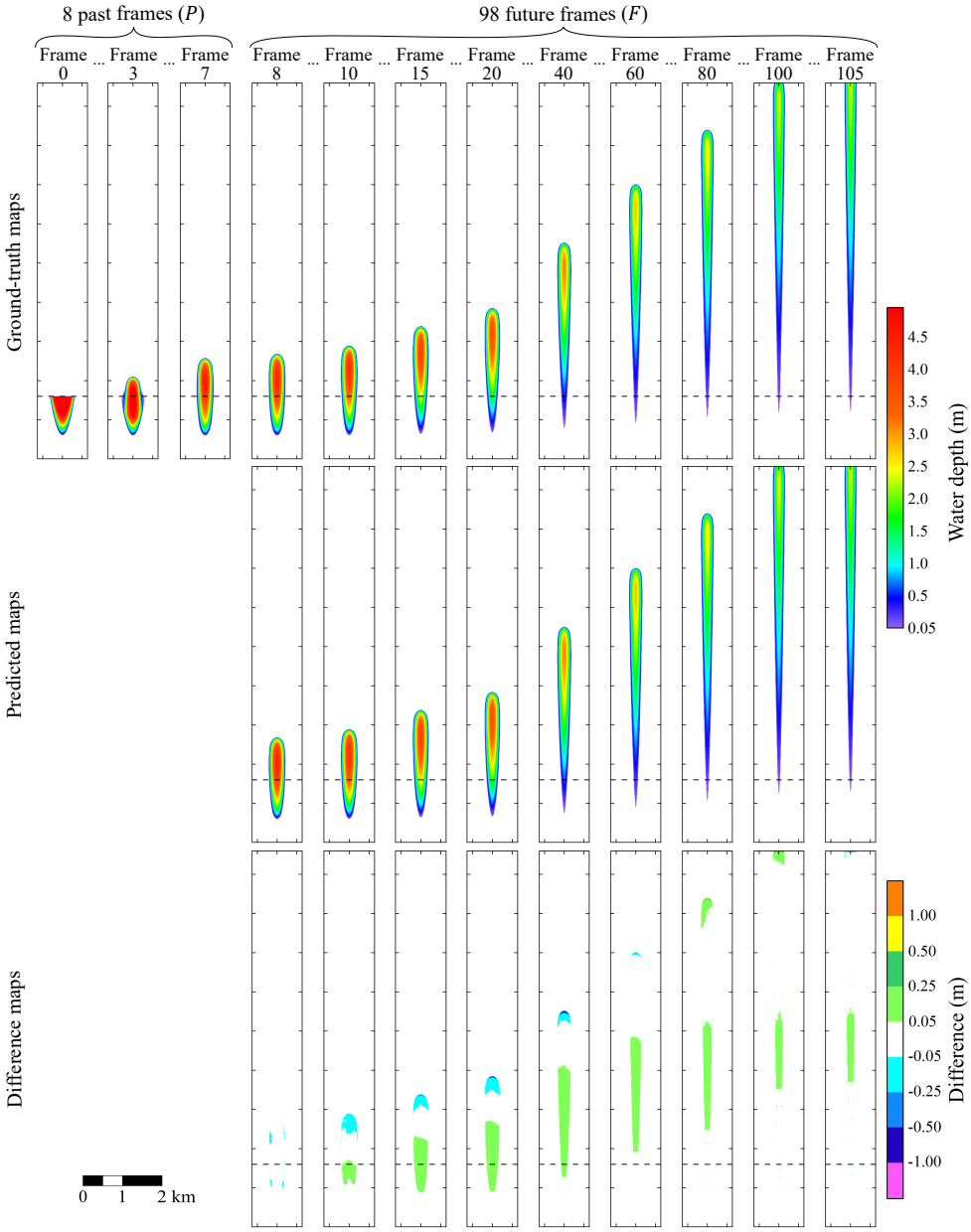

FIGURE 5.3 – Case study 1: comparison between ground-truth and predicted maps for selected instants of the testing scenario. The forecasted maps are generated through the AR procedure of the FS-db model. The last row shows the difference between predicted and ground-truth maps. Dashed black lines indicate the position of the gate at $y = 1600$ m. The number of frames is set to $I = P = 8$ and $F = 98$.

suggested that increasing the batch size reduces the training time per epoch, but may slow convergence, requiring more epochs to reach an optimal solution. To minimize the influence of the batch size on the results of this analysis, a constant value equal to 4 was used across all training configurations. Additionally, all other hyperparameters were held constant, equal to the values defined in Section 4.6.

Table 5.4 summarizes the performance metrics considered in this sensitivity analysis. To ensure a fair comparison, the average RMSE of the FS test is calculated using the forecasted maps at the same time steps for all simulations (i.e., frames in range of 16 to 75, as the maximum I value considered is 16). Overall, the average RMSE remains nearly constant as I varies, except for $I = 1$, which results in a slightly higher RMSE. This consideration is further supported by Fig. 5.4a, which illustrates the temporal variation of the RMSE. To enhance clarity, only the first 10 minutes of the flood event (i.e., frames in the range of 1 to 30) are shown, as the RMSE stabilizes and remains nearly identical across all configurations in the later stages. Training the surrogate model with $I = 4$ results in lower errors in predicting the first three future maps (i.e., frames 4 to 6) compared to predictions made with $I = 2$ (see Fig. 5.4a). Similar improvements are observed with $I = 8$ over $I = 4$ for frames 8 to 10, while further increasing the number of input frames (e.g., $I = 12$ or 16) leads to only modest RMSE improvements.

Regarding the training time, increasing the number of input frames from 1 to 2 results in a slight increase in the average epoch training time (Table 5.4). For higher I values (from 4 to 16) an almost linear increase in computational times is obtained.

Given that the FS model is ultimately intended for real-time flood scenarios forecasting, this sensitivity analysis also considers the outcomes of the AR prediction. In this study, the number of past frames is set equal to the number of input frames (i.e., $P = I$), while the number of future frames is set to $F = 60$. To facilitate comparison across different values of the hyperparameter I, all configurations focus on predicting the same 60 future maps (i.e., frames 16 to 75). Fig. 5.4b compares the temporal variation of the RMSEs for each training configuration. Notably, the model

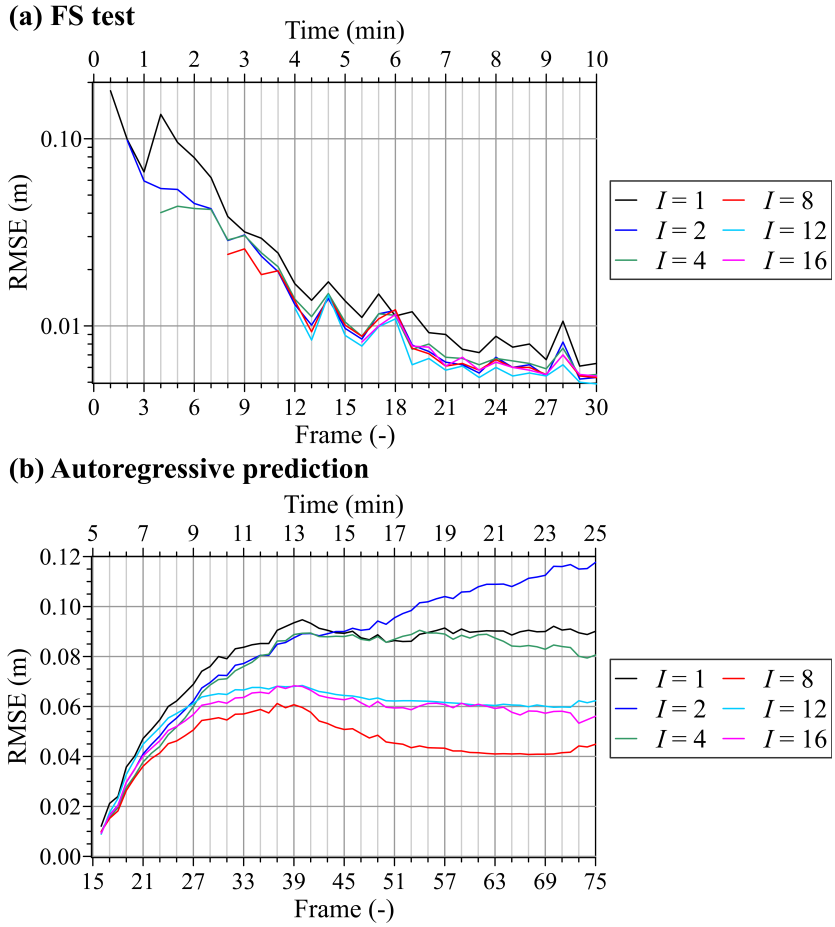

FIGURE 5.4 – Sensitivity analysis to the number of input frames (I). (a) Comparison of the RMSE computed for the FS test varying the number of input frames. For clarity of representation, only the first 30 predicted frames are shown, as RMSEs are nearly identical across all configurations in the later instants. The number of input frames is set to $I = [1, 2, 4, 8, 12, 16]$. (b) Comparison of the RMSE computed for the AR procedure of the FS model varying the number of input frames (I). All the simulations predict the same temporal frames in range $[16, 75]$. The number of frames is set to $I = [1, 2, 4, 8, 12, 16]$ and $F = 60$.

TABLE 5.4 – Comparison of prediction performance with varying numbers of input frames (I), where $P = I$ and $F = 60$. The number of training epochs is determined automatically using the early stopping technique. The average RMSE of the FS test and AR prediction are computed over frames in range 16-75. The final configuration ($I = 8$) is highlighted in bold.

Input frames (I)	Training epochs	Training time [min]	Time/epoch [min]	RMSE [cm]	
				FS test	AR prediction
1	104	65	0.63	0.60	8.0
2	110	75	0.68	0.56	8.5
4	100	90	0.90	0.56	7.5
8	**116**	**125**	**1.08**	**0.54**	**4.5**
12	120	155	1.29	0.53	5.9
16	136	200	1.47	0.53	5.6

trained with $I = 8$ achieves the lowest RMSE during the AR procedure. Higher values of I (e.g., 12 or 16) reduce performance due to information redundancy in the input sequence, while lower values (e.g., 1 to 4) decrease accuracy as fewer maps are available to capture the essential spatiotemporal information. The average RMSEs summarized in Table 5.4 confirms that the surrogate model trained with $I = 8$ outperforms the other configurations in recursively forecasting the 60 future maps.

In conclusion, the optimal value of I obtained for the first case study is assumed as ideal also for the other cases considered in this thesis.

5.3 Dam-break in a rectangular tank

5.3.1 Case study description

The second case study investigates a DB flow interacting with an obstacle at the laboratory scale, inspired by the experimental setup described by Aureli et al. (2008b). The test is conducted within a rectangular tank, measuring 2.56 m by 1.28 m, with a flat bottom (Fig. 5.5). The tank is divided into two compartments by a wall, which features a 0.30-m-wide gate at its center. The upstream compartment, measuring

0.79 m by 1.28 m, is the reservoir, while the remaining area of the tank represents the downstream floodplain. A non-submersible prismatic block, with a rectangular base of 0.30 m by 0.15 m and vertical walls, is positioned 0.60 m downstream of the gate. The interaction of the flow with this obstacle generates a hydraulic jump and multiple wave reflections, significantly complicating the flow dynamics.

Although this study is not experimental, it is designed analogously to the laboratory cases examined by Aureli et al. (2008b), who demonstrated that flood propagation within the tank can be effectively simulated using a 2D-SWE model. Consequently, the PARFLOOD code was employed to generate water depth maps, which served as ground truth data for the training and testing of the surrogate model.

A spatial resolution of 0.01 m was adopted for discretizing the study area, resulting in a computational grid of 32,768 cells. Given the similarity to the experiments conducted by Aureli et al. (2008b), the same Manning roughness coefficient of $0.007\,\mathrm{sm}^{-1/3}$ was used. This value was derived from a calibration process based on the propagation time of the wetting front.

The initial conditions for the hydrodynamic model included 8 different constant water levels in the upstream reservoir, ranging from 6 cm to 20 cm in 2 cm increments (see Table 5.2). The output maps from these numerical simulations were used to generate the training dataset. Additionally, a simulation was conducted with a 15 cm initial water level to create the testing dataset. Similar to the methodology employed in case study 1 (Section 5.2), the testing event was derived using a different initial condition than those used in the training set, thereby ensuring the surrogate model's performance could be evaluated on unseen flood scenarios.

In all simulations, the downstream floodable area was initially dry. To simulate the DB scenarios, an instantaneous gate removal was assumed, and the simulation was terminated when the water reflected off the downstream wall, which occurred between 2 s and 4 s after the gate removal. The surrogate model dataset consisted of consecutive water depth maps with a temporal resolution of 0.02 s, a value considered appropriate for accurately capturing flood dynamics in this laboratory-scale case.

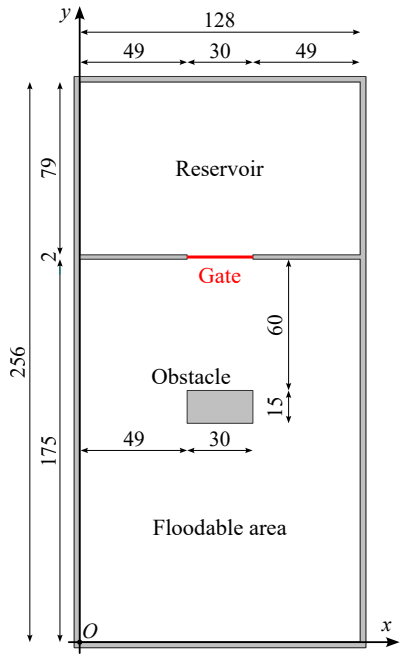

FIGURE 5.5 – Case study 2: sketch of the rectangular tank domain. All the measurements are in centimeters.

5.3.2 Results

For the second case study, the RMSE values for both the FS test and the AR procedure of the FS-db model are presented in Fig. 5.6, with the corresponding average values summarized in Table 5.3.

For the FS test, the RMSE remains below 0.5 mm across all predicted maps, which represents less than 0.5% of the initial water level in the upstream reservoir (i.e., 15 cm). As observed in the previous case study, the FS model demonstrates a high degree of accuracy in forecasting one map ahead.

For the AR prediction, the number of past and future frames was set to 8 and 113, respectively, encompassing all 121 maps from the testing event. A comparative analysis of ground-truth and predicted maps, shown in Fig. 5.7, indicates that the temporal evolution of inundation within the tank's floodable area is accurately

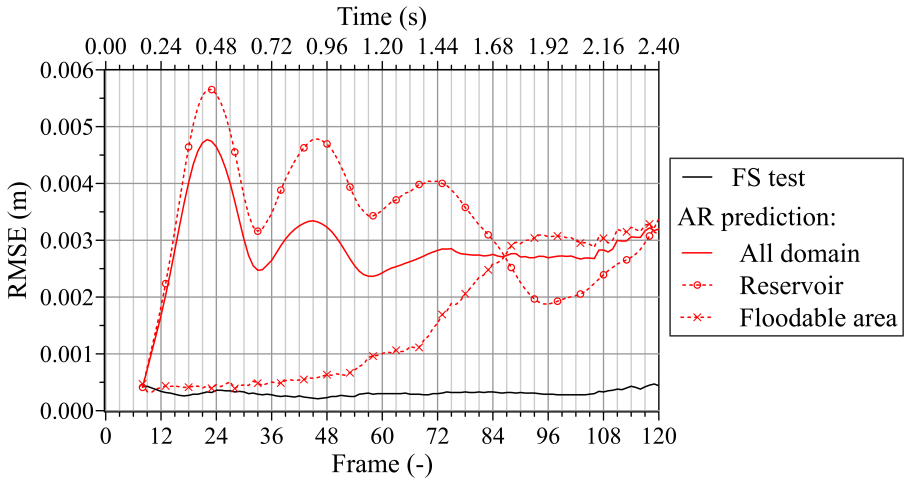

FIGURE 5.6 – Case study 2: RMSE for the FS test and AR prediction using the FS-db model. In this case, the metric computation is divided between the upstream reservoir and the downstream floodable area. The number of frames is set to $I = P = 8$ and $F = 113$.

captured. For the first 70-80 predicted maps, discrepancies between the target and forecasted maps generally remain below 2.5 mm. For subsequent instants, higher errors arise near the lateral and downstream boundaries of the floodable area due to wave reflections from the walls, which introduce complex dynamics and higher water depths.

Upstream the prismatic block, the differences remain under 5 mm, despite water depths reaching up to 10 cm in this region. The surrogate model accurately captures the presence of the block, a feature that was not explicitly provided during the FS model's training, indicating the model's ability to infer this information from the training data.

The prediction accuracy diminishes in the upstream reservoir, where larger errors are observed. These discrepancies are likely attributable to the complex flow characteristics within the reservoir, involving intricate interactions with the boundaries. Initially, a rarefaction wave moves upstream, followed by reflections from the walls, which complicate the flow dynamics and amplify error accumulation in the recursively forecasted maps. Consequently, differences between the target and

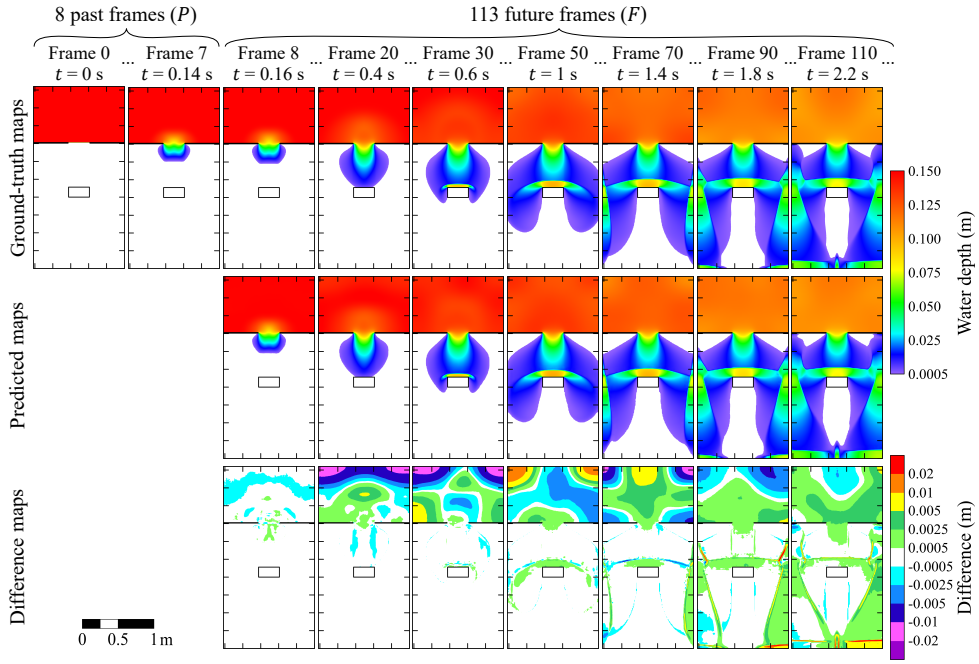

FIGURE 5.7 – Case study 2: comparison between ground-truth and predicted maps for selected instants of the testing scenario. The forecasted maps are generated through the AR procedure of the FS-db model. The last row shows the difference between predicted and ground-truth maps. The black rectangle in the downstream floodable area indicate the position of the prismatic block. The number of frames is set to $I = P = 8$ and $F = 113$.

predicted maps in the upstream reservoir can reach up to 13-14 mm (see Fig. 5.7). Nonetheless, these errors do not significantly impair the accuracy of flood prediction in the downstream area.

To further analyze the error distribution, Fig. 5.6 presents the RMSE values calculated separately for the upstream reservoir and the downstream floodable area. It is evident that the RMSE for the entire domain is predominantly influenced by errors in the upstream reservoir during the first 80 predicted maps. Specifically, the RMSE for the upstream basin increases sharply from 0.5 mm to approximately 5.6 mm within the first 16 forecasted maps (i.e., frames in the range 8-23), primarily due to high prediction errors near the northern edge of the tank (as shown in Fig. 5.7), and then follows an oscillatory pattern. In contrast, the RMSE for the downstream

area remains below 1 mm until 1.2 s after the gate removal (i.e., frame 60), after which it gradually rises to around 3.0-3.3 mm. This increase is attributable to error accumulation in the southern part of the tank, where the inundation reaches the downstream boundary approximately at instant $t = 1.4$ s (i.e., frame 70), resulting in localized increases in water depth up to 75 mm due to wall reflections. The FS model tends to overestimate the speed of the reflected wave's front propagation, leading to water depth differences of up to 35 mm.

The average RMSE across the 113 recursively forecasted maps is 2.9 mm (see Table 5.3), which is less than 2% of the initial water level in the upstream reservoir (i.e., 15 cm). Moreover, the average RMSEs for the upstream reservoir and downstream floodable area are approximately 3.4 mm and 1.6 mm, respectively.

The F1 score remains exceptionally high for both the FS test and the AR prediction (see Table 5.3), indicating that the FS model accurately reproduces the extent of the flooded area in this case study. Additionally, as observed in the first case study, the model effectively preserves the symmetry of the flood in the downstream floodable area (Fig. 5.7), with the most pronounced asymmetries occurring near the lateral walls, where wave reflections influence the flood dynamics, and in the upstream reservoir, where error accumulation is more significant.

5.4 Dam-break of the Parma River flood-control dam

5.4.1 Case study description

The third test case involves the hypothetical failure of the Parma River flood-control reservoir dam. Unlike the previous case studies, this scenario is characterized by the use of a real bathymetry, providing a more complex and realistic environment for evaluating the surrogate model.

The in-stream detention reservoir of the Parma River, located a few kilometers

upstream from the city of Parma in Italy, has a storage capacity of approximately 12×10^6 m^3. The maximum retaining water level is 105.6 m a.s.l., corresponding to a maximum water depth of about 14.5 m immediately upstream of the dam. The study area covers approximately 45 km^2, encompassing the floodplain between the control reservoir and the city center of Parma (Fig. 5.8). Unlike the previous case studies, this scenario employs a real DTM to represent the topography.

Two DTMs, with spatial resolutions of 10 m (458,752 cells) and 20 m (114,688 cells), derived from a LiDAR survey, were considered. A uniformly distributed Manning's roughness coefficient value of 0.05 sm$^{-1/3}$ was adopted, based on a prior calibration process using historical flood events of the Parma River. For all the numerical simulations, a far-field BC was applied at the downstream section, while the river region was considered initially dry.

Different flood scenarios have been created adopting different water levels in the upstream reservoir as initial conditions (see Table 5.2). Specifically, numerical simulations were conducted for initial water levels ranging from 98 m to 104 m, in 2-m increments, to create samples for the training and validation datasets. Additionally, the maximum retaining water level of 105.6 m a.s.l. was included to further enrich these datasets. Incorporating the maximum expected water level in the upstream reservoir helps to prevent the need for extrapolation beyond the training data range during predictive applications (for further details see Section 7.2). Differently, the testing dataset was created using the output maps from a numerical simulation initialized with a water level of 105 m a.s.l. in the upstream reservoir.

Each simulation was run for a specific duration (between 2 and 3 hours) to ensure the completion of flood propagation within the study area, including the emptying of the reservoir and the reaching of the maximum flood extent. Water depth maps were sampled at 2-minute intervals, a time-frame sufficient to accurately capture the rapid flood propagation associated with a DB scenario on a real bathymetry.

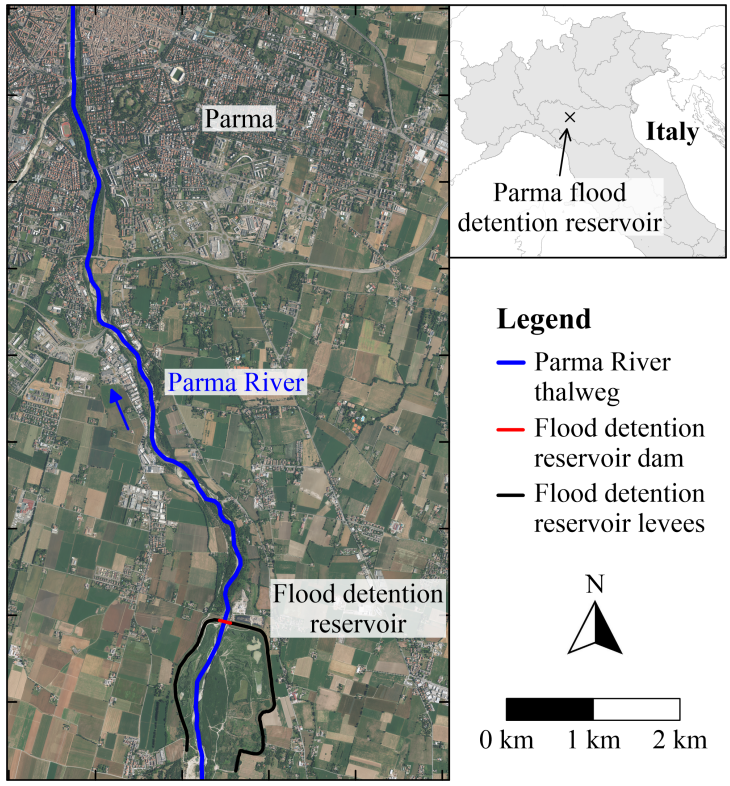

FIGURE 5.8 – Case studies 3a and 3b: study area for the hypothetical collapse of the Parma River flood-control reservoir dam. The computational domain is the entire area visible in the left figure.

5.4.2 Results

In this Section, the findings from the FS model training considering maps with a spatial resolution of 20 m (case study 3a) are firstly presented. Subsequently, these results are compared with those obtained from the surrogate model trained using the maps with a 10 m resolution (case study 3b).

Spatial resolution of 20 m

For case study 3a, the solid lines in Fig. 5.9 illustrate the RMSEs for both the FS test and the AR procedure of the FS-db model. A summary of the average values for

these performance metrics is provided in Table 5.3. As anticipated, the AR prediction exhibits a higher RMSE compared to the FS test due to error accumulation, resulting in recursively forecasted maps that are less accurate than those generated by the FS test procedure, which only predicts the map one step ahead.

The RMSE for the FS test ranges from 3 cm to 6 cm, with an average value of 4.2 cm. This value represents less than 0.5% of the maximum water depth observed in this case study (i.e., 14.2 m). Additionally, the F1 score for the test procedure is nearly 1 (see Table 5.3), confirming that the FS model is highly accurate in predicting a single frame ahead, even in the context of a DB scenario on real bathymetry.

For the AR prediction, the parameters were set to $I = P = 8$ and $F = 83$, encompassing all 91 maps of the testing event (see Table 5.2). The 83 recursively predicted frames correspond to a lead time of about 2.8 hours, allowing the inundation to fully propagate over the study area. Fig. 5.10 presents a comparison between the target and predicted inundation maps at representative instants of the testing flood event.

The first forecasted frame (i.e., frame 8 as $P = 8$) has a RMSE of approximately 5 cm (Fig. 5.9). For the second to eighth predicted maps (i.e., frames in range 9-15), the error gradually increases to 9 cm, primarily due to the underestimation of water depths near the wet/dry front in the river region (Fig. 5.10). From frame 16 onwards (i.e., starting from 32 minutes after the DB), as the inundation reaches the downstream boundary, the error associated with the wet/dry front dissipates, leading to a reduction in RMSE by about 2 cm (Fig. 5.9). However, the RMSE increases again due to error accumulation, reaching approximately 16 cm by the final forecasted map (i.e., frame 90, about 3 hours after the DB). Fig. 5.10 reveals that, in general, the differences between the forecasted and target maps outside the river region are less than 25 cm, except in certain locations where errors increase to as much as 1 m. These significant discrepancies begin to appear around the 60th forecasted map (i.e., 2 hours after the DB), primarily due to error accumulation at the northeastern edge of the flood. Additionally, for these later maps, the progressive

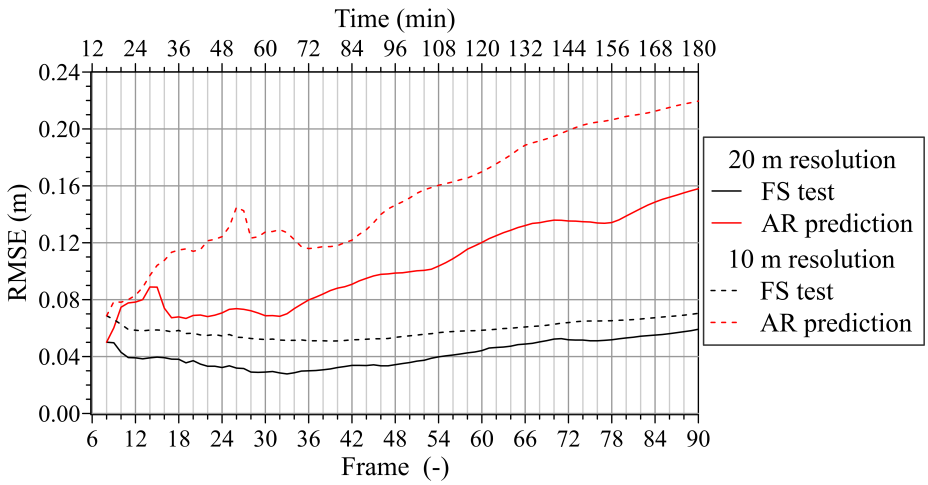

FIGURE 5.9 – Case study 3a and 3b: RMSE for the FS test and AR prediction using the FS-db model. The number of frames is set to $I = P = 8$ and $F = 83$.

rise in RMSE is primarily attributed to the overestimation of water depths in the river region during the recession limb of the flood. For example, when considering only the domain outside the river region, the RMSE for frame 90 (the final frame) is 9.8 cm, which is 6.1 cm lower than the RMSE calculated for the entire study area (Fig. 5.9). Similarly, for frame 60, the RMSE decreases from 12 cm to 8.5 cm when considering only the cells outside the river region. Moreover, during the recession limb, the flood propagation dynamics in the lowland area are not affected by the overestimated water depths in the river region, as the water surface elevations in this area remain below the riverbank elevations, preventing overtopping.

Fig. 5.11a provides a direct comparison between the target and recursively predicted water depths for each cell in the domain across the 83 forecasted frames during the AR procedure. In this scatter plot, the water depths are categorized into classes with 0.1 m intervals, and the color of each point, representing a pair of predicted-target values, indicates the frequency of occurrence computed as the ratio of the number of grid cells with a given pair to the total number of wet cells (i.e., approximately 2.1M). For clarity, true negative values (i.e., dry cells in both target

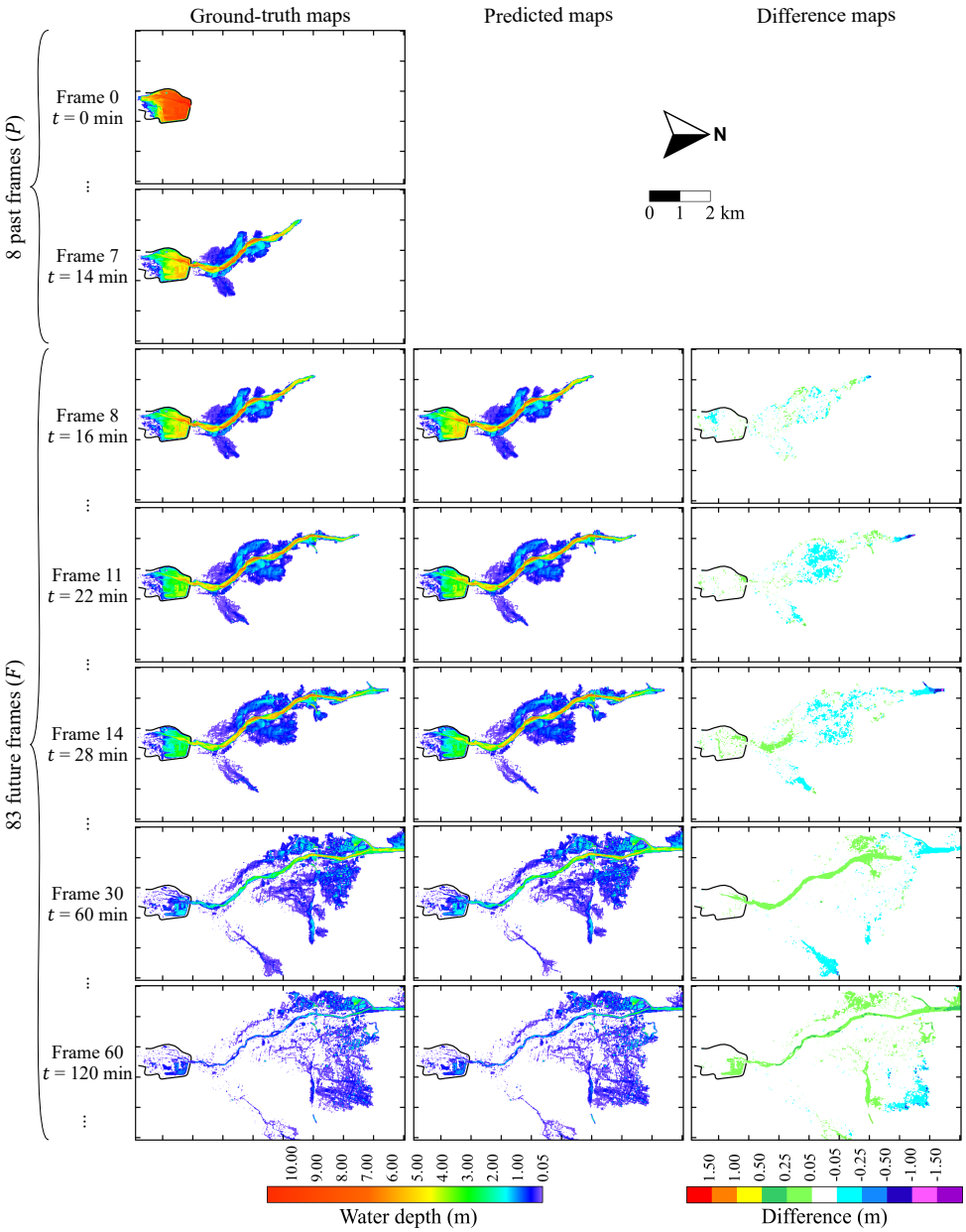

FIGURE 5.10 – Case study 3a: comparison between ground-truth and predicted maps for selected instants of the testing event. The forecasted maps are generated through the AR procedure of the FS-db model. The last row shows the difference between predicted and ground-truth maps. The number of frames is set to $I = P = 8$ and $F = 83$.

and predicted maps), which account for approximately 7.4M pairs, are excluded from the color scale representation. The majority of points align closely with the dashed black line, representing accurate predictions. The surrogate model shows a slight tendency in overestimating the water depths, especially for values lower than 5 m.

Fig. 5.11b indicates that approximately 87.4% of the wet cells have an error of less than 10 cm. This percentage increases to about 95.8% when considering an error threshold of 20 cm. Moreover, only 0.05% of the wet cells exhibit errors exceeding 1 m.

The average RMSE and F1 score for the 83 recursively forecasted maps are 10.4 cm and 0.937, respectively. Consequently, considering the application to DB scenarios on real bathymetry, the model provides sufficiently accurate predictions for practical purposes.

Spatial resolution of 10 m

In this subsection, the results from case study 3a, discussed in the previous subsection, are compared with those from case study 3b, where the surrogate model is trained using inundation maps with double spatial resolution. The objective is to assess the impact of map resolution on the performance of the surrogate model, evaluating its robustness and scalability across different resolutions. The same frame configurations used in case study 3a are adopted here (i.e., $I = P = 8$ and $F = 83$).

Fig. 5.9 presents the RMSE comparison between the two configurations. Overall, the errors associated with the 10 m resolution are slightly higher than those observed with the coarser grid. Specifically, as summarized in Table 5.3, the average RMSEs for both the FS test and the AR prediction are 1.7 cm and 5.0 cm greater, respectively, for the finer resolution.

As shown by the comparison of Figs. 5.10 and 5.12, the forecasting of the first 9 future maps (i.e., frames in range 8-16) indicate that the FS-db model trained with the 10 m resolution grid reduces errors near the wet/dry front but introduces more spatially distributed discrepancies, ranging from 5 cm to 25 cm, across the

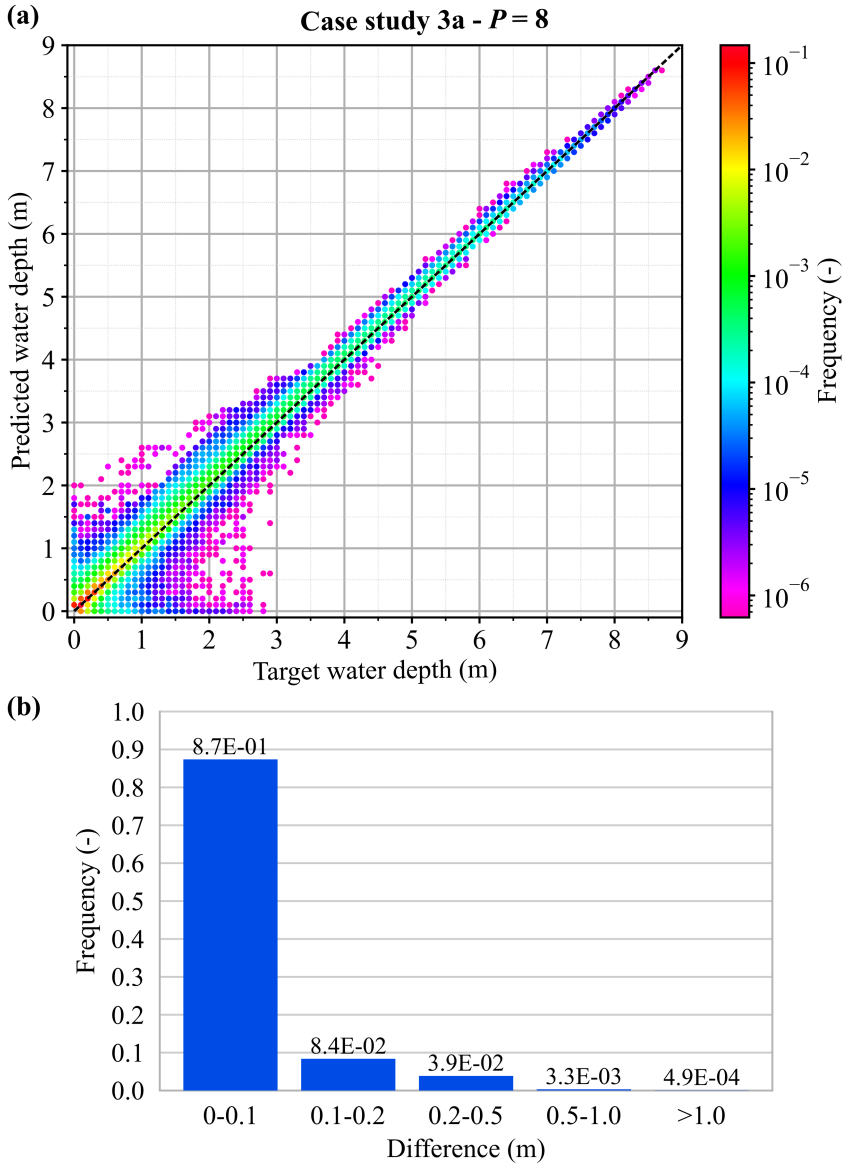

FIGURE 5.11 – Case study 3a: comparison of predicted and target water depths obtained setting $P = 8$ and $F = 83$. (a) Scatter plot comparing predicted and target water depths across all grid cells and time steps of the testing flood event. The color scale indicates the frequency, computed using as normalization factor the total number of wet cells (about 2.1M). The dashed black line represents the correct forecast. (b) Histogram showing the distribution of the differences between target and predicted water depths.

flooded area, compared to case study 3a. For frames in range 17-35 (i.e., the 10th to 28th forecasted maps), these discrepancies increase to as much as 0.8-1.2 m in certain locations on the hydraulic left of the river, where the local flow field is notably complex. In subsequent frames, similar to the results obtained with the coarser grid, the model tends to overestimate water depths in the river region by up to 0.5-0.8 m, and the largest errors (up to 1.0-1.2 m) occur at the northeastern edge of the flooded area. In contrast to case study 3a, these discrepancies are primarily attributed to the underestimation of the flooded area. Indeed, as shown in Table 5.3, the average F1 score for case study 3b is slightly lower than that of case study 3a.

For the 10 m resolution maps, error accumulation in the river region leads to a progressive increase in RMSE, reaching up to 22 cm for the final forecasted maps (Fig. 5.9). When considering only the cells outside the riverbanks, the RMSE for frame 90 is 14.8 cm, which is 7.3 cm lower than the RMSE computed for the entire study area, while for frame 60 it is 12.5 cm. However, these values are 4-6 cm higher than the results observed in case study 3a. Thus, refining the map resolution amplifies the effect of error accumulation in long-term predictions, while also allowing for a more detailed representation of the inundation's temporal evolution, thanks to the finer discretization of the domain. This reduction in accuracy is primarily attributed to the increased complexity of the FS model when the spatial resolution is enhanced, as it is characterized by a greater number of layers and parameters requiring optimization. Consequently, this complexity complicates the training process and may necessitate a larger dataset for more accurate results. Moreover, as the FS model analyses spatial and temporal correlations between adjacent cells for prediction, increasing the spatial resolution leads to less smooth flood maps in terms of water depth values. This generates sharper differences in water depths between adjacent cells, thereby increasing the difficulty of the forecasting process.

The reduction in the prediction performance of the FS-db model when increasing the spatial resolution is also evident when comparing Fig. 5.13a and Fig. 5.11a, which show the comparison of target and predicted water depths for all grid cells and time

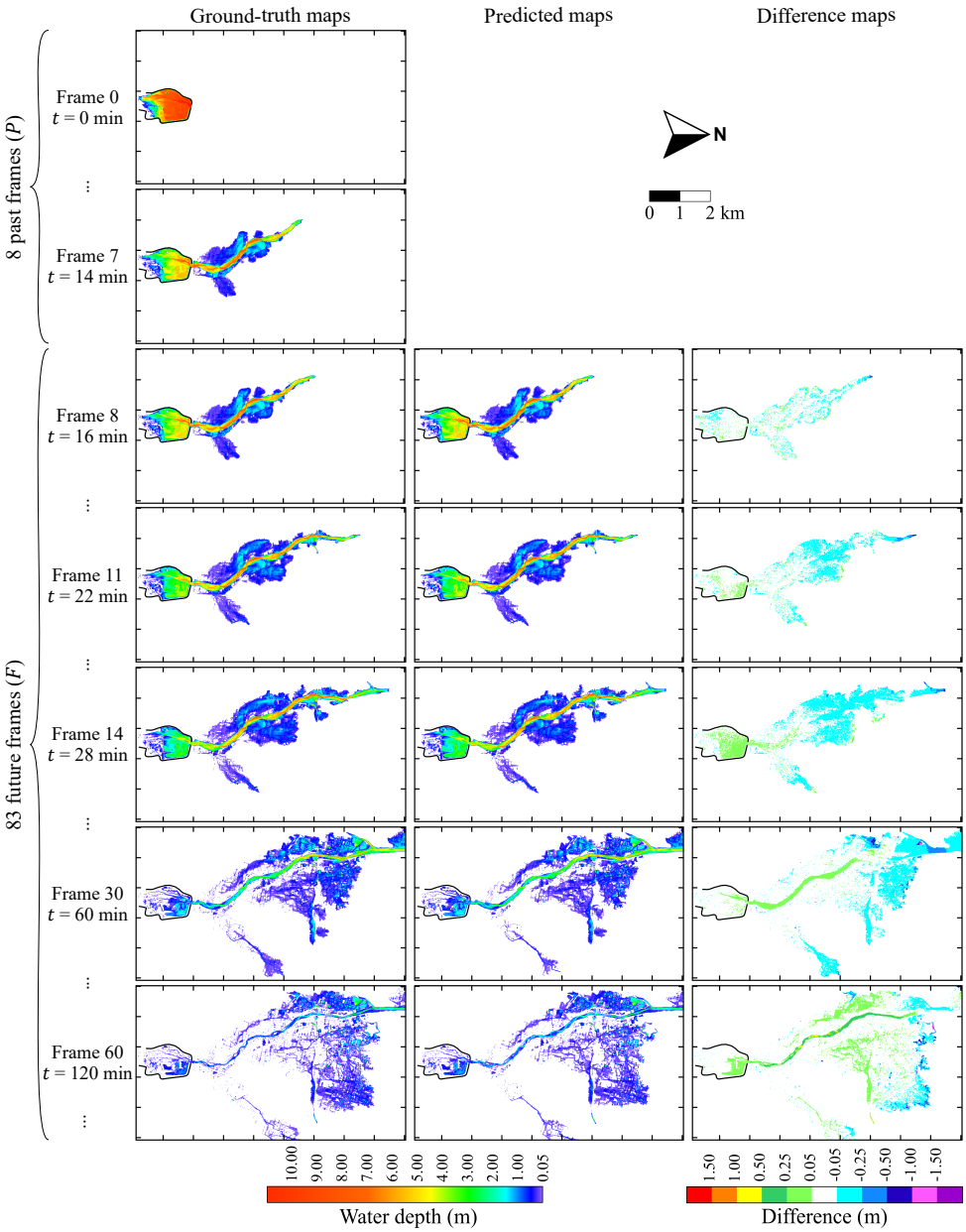

FIGURE 5.12 – Case study 3b: comparison between ground-truth and predicted maps for selected instants of the testing scenario. The forecasted maps are generated through the AR procedure of the FS-db model. The last row shows the difference between predicted and ground-truth maps. The number of frames is set to $I = P = 8$ and $F = 83$.

steps of the AR prediction. Fig. 5.13a indicates a greater frequency of points that deviate from the dashed black line, representing perfect prediction. The surrogate model shows a more pronounced tendency to overestimate predicted water depths, particularly for values in range of 0-3 m. Additionally, the surrogate models generates a higher number of cells with FN prediction compared to FP, mainly due to the underestimation of the velocity of propagation of the wet/dry front. This leads to cells that should be predicted as flooded being incorrectly classified as dry.

Nevertheless, approximately 80.8% of the wet cells in all the recursively predicted maps (i.e., about 8.9M) exhibit differences lower than 10 cm, and 93.3% of the wet cells have errors lower than 20 cm (see Fig. 5.13b). These results confirm that the model's accuracy is acceptable for practical applications.

5.4.3 Sensitivity analysis to the number of past frames

In Section 5.4.2, the outcomes of AR forecasting for DB scenarios using a number of past frames equal to the number of input frames defined during the training process (i.e., $P = I = 8$) were discussed. In this configuration, implementing the FS-db model to predict the temporal evolution of inundation maps for a real-world emergency would necessitate access to the first eight water depth maps of the flood event (i.e., the past frames). These maps can be generated by a physically based model. However, the integration of numerical and surrogate models may compromise overall computational efficiency and presents challenges for real-time forecasting applications. To address these issues, the FS-db model's capability to forecast water depth maps by reducing the number of past frames (P) below the number of input frames (I) used during training is explored. The optimal configuration would use $P = 1$, as it would only require knowledge of the initial water level in the upstream reservoir (i.e., before the dam-break occurs) to recursively forecast the entire flood event. The reservoir water level is typically recorded by a gauging station near the dam, and it can be easily converted into a water depth map under lake-at-rest conditions.

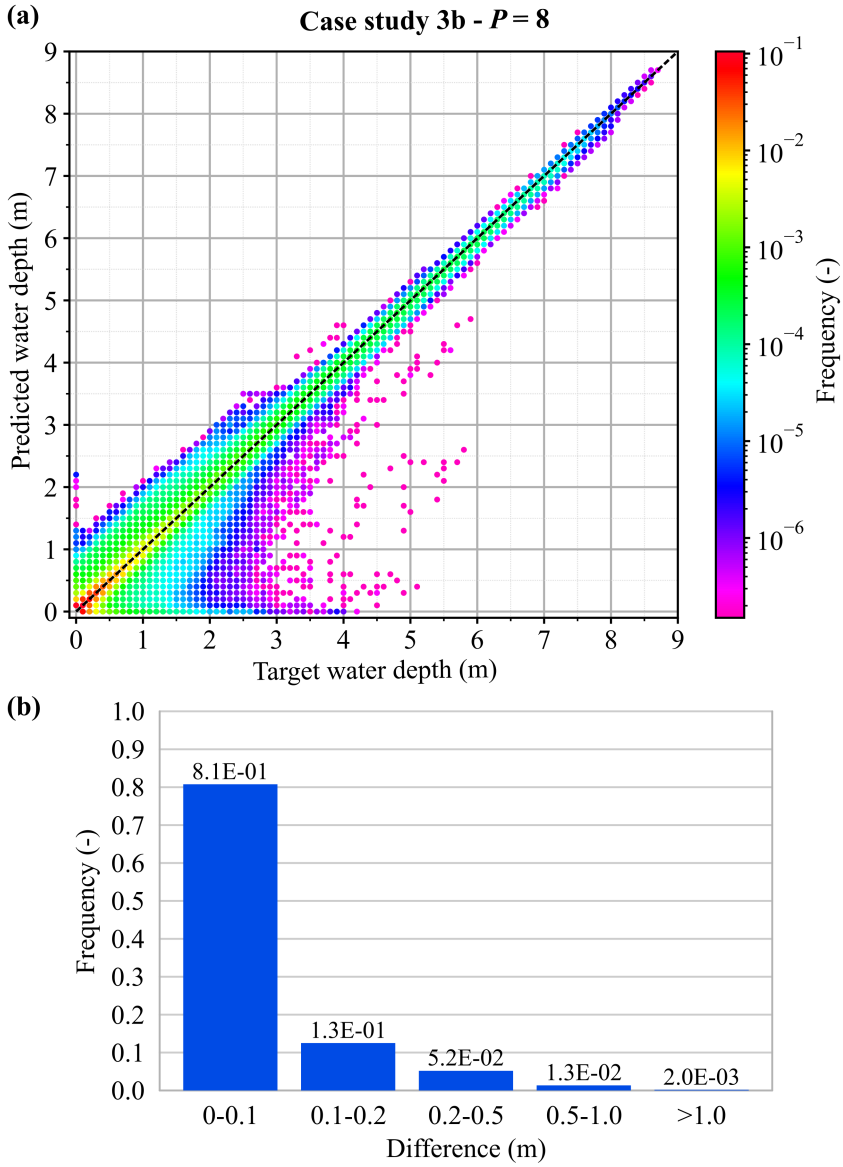

FIGURE 5.13 – Case study 3b: comparison of predicted and target water depths obtained setting $P = 8$ and $F = 83$. (a) Scatter plot comparing predicted and target water depths across all grid cells and time steps of the testing flood event. The color scale indicates the frequency, computed using as normalization factor the total number of wet cells (about 8.9M). The dashed black line represents the correct forecast. (b) Histogram showing the distribution of the differences between target and predicted water depths.

For this investigation, the DB scenario of the Parma River dam with a spatial resolution of 20 m (i.e., case 3a) was selected, maintaining $I = 8$ to utilize the previously trained FS parameters. In addition to the AR prediction obtained setting $P = 8$, as discussed in Section 5.4.2, also configurations with P set to 4, 2, and 1 were tested. The corresponding number of future frames (F) was 87, 89, and 90, respectively (i.e., $P + F = 91$, matching the number of time steps of the testing event).

Fig. 5.14 presents the RMSEs for the recursively forecasted maps across different P values, while Fig. 5.15 illustrates the discrepancies between the predicted and ground-truth maps for each forecasting configuration. Reducing the number of past frames results in an initial increase in RMSE for the early frames of the sequence (Fig. 5.14). For $P = 1$, the RMSE of the first forecasted map (i.e., frame 1) is relatively high, at approximately 31 cm. As shown in Fig. 5.15, water depths at this instant are particularly significant, reaching values up to 11-12 m, with the largest discrepancies in the order of 1 m, occurring near the dam and in the southern reservoir area. The RMSE for subsequent predicted maps decreases progressively, reaching around 7 cm for the 17th predicted frame. Comparable results were observed with $P = 2$, though the RMSE was notably lower, particularly for frames in the range of 9-16. For $P = 4$, the RMSE for frames in range of 4-9 is 1.5-3.5 cm lower than that for $P = 2$. Similar findings were obtained when comparing $P = 8$ and $P = 4$, with differences of 1.5-2.5 cm for frames 8-12. Starting from frame 17 (i.e., 34 minutes after the DB, approximately when the flood reaches the downstream boundary), RMSE values are comparable across all simulations.

The comparison of the scatter plots in Fig. 5.11a and Fig. 5.16a, which represent the results obtained with P set to 8 and 1, respectively, shows that reducing the required input information to the minimum (i.e., setting $P = 1$) results in only a minor decrease in prediction performance. Specifically, comparing the percentage of wet cells with errors lower than 10 cm decreases from 87.4% to 86.7%, representing a reduction of less than 1% (see Fig. 5.11b and Fig. 5.16b).

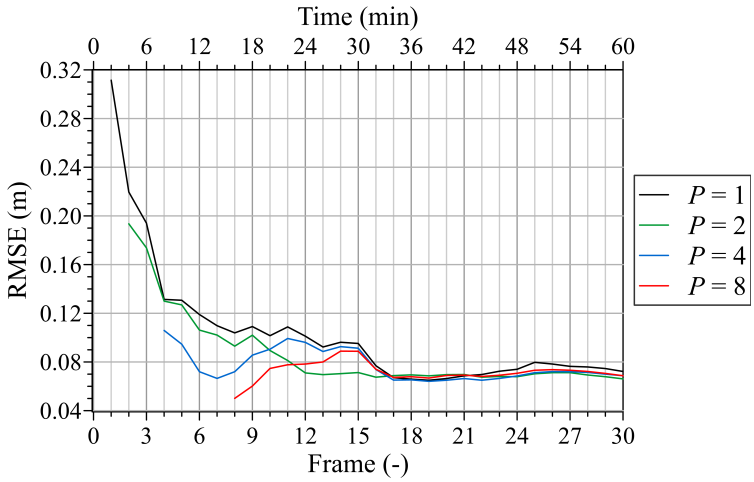

FIGURE 5.14 – RMSE comparison for different number of past frames (P) used to forecast the testing event of the case study 3a, adopting the AR procedure of the FS-db model. The number of frames is set to $I = 8$, $P = [1, 2, 4, 8]$, and $F = [90, 89, 87, 83]$. Only the RMSE of the first 30 predicted frames is represented.

The average RMSEs and F1 scores for the four simulations are summarized in Table 5.5. The average F1 score remains nearly consistent across all simulations, indicating that the FS-db model's ability to predict the extent of the flooded area is only marginally influenced by the number of past frames. Similarly, the average RMSE remains nearly constant, except for $P = 1$, which results in an increase of approximately 1 cm. Despite these minor discrepancies, the results are acceptable for real-time forecasting of DB scenarios on realistic bathymetry.

TABLE 5.5 – Comparison of prediction performance with different number of past frames (P). The number of frames is set to $I = 8$, $P = [1, 2, 4, 8]$, and $F = [90, 89, 87, 83]$.

Past frames (P)	RMSE [m]	F1 [-]
1	0.114	0.936
2	0.106	0.939
4	0.105	0.937
8	0.104	0.937

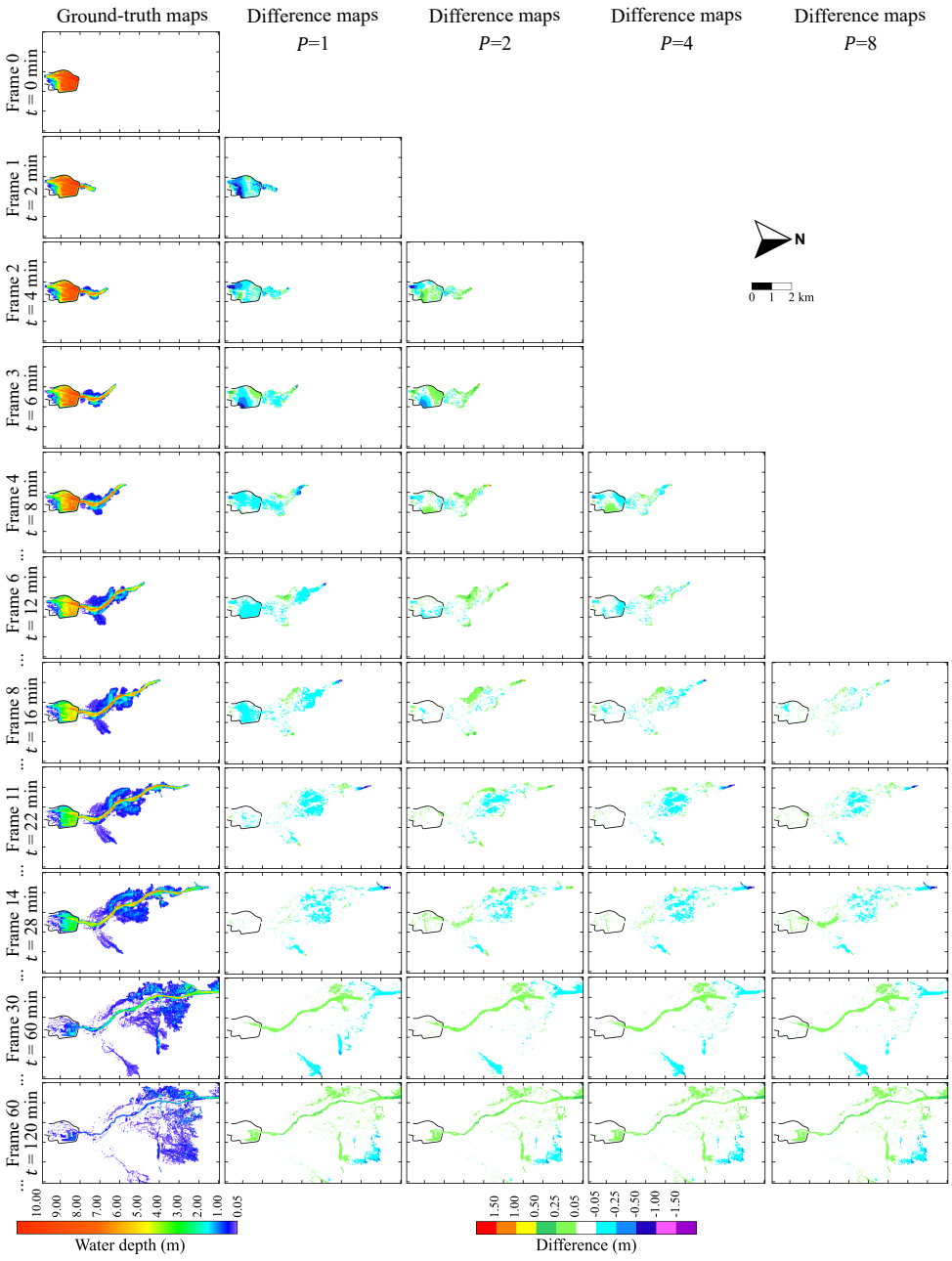

FIGURE 5.15 – Comparison of prediction performance with different number of past frames (P). The maps show the differences between predicted (not shown) and ground-truth maps. The number of frames is set to $I = 8$, $P = [1, 2, 4, 8]$, and $F = [90, 89, 87, 83]$.

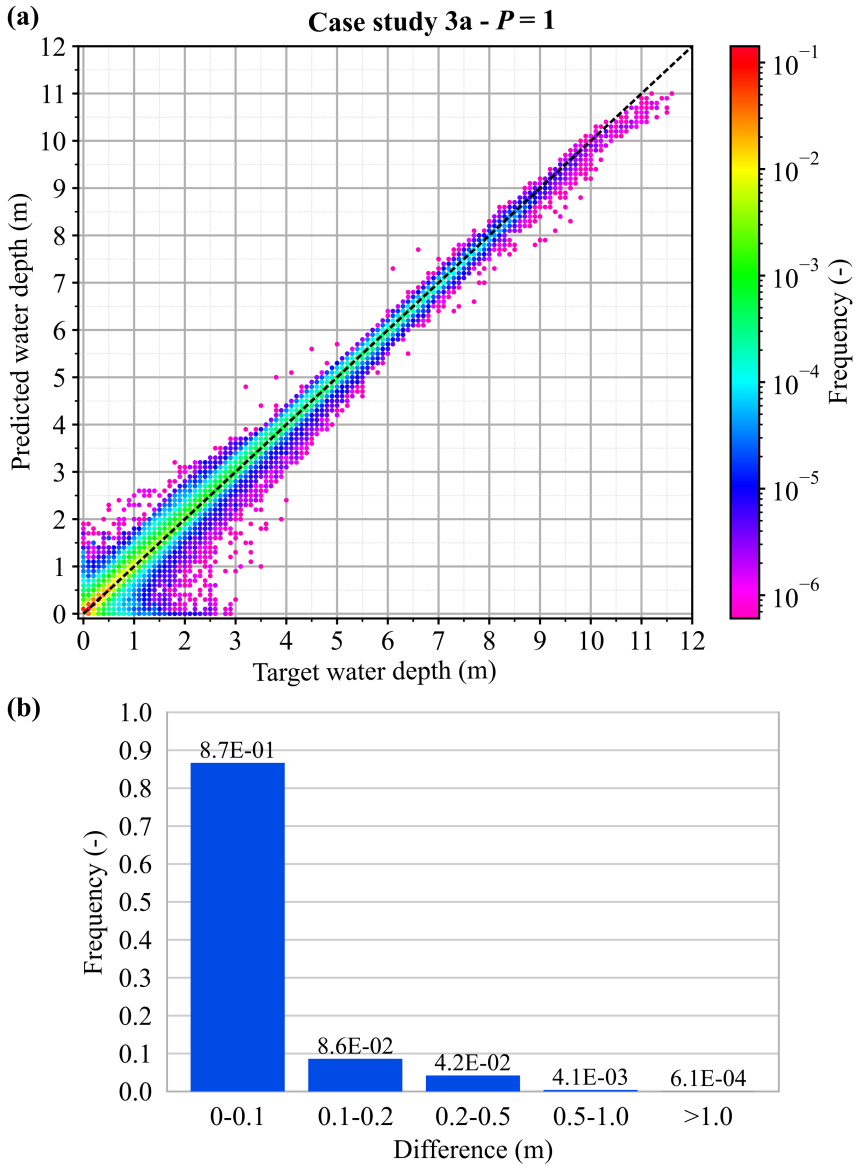

FIGURE 5.16 – Case study 3a: comparison of predicted and target water depths obtained setting $P = 1$, and $F = 90$. (a) Scatter plot comparing predicted and target water depths across all grid cells and time steps of the testing flood event. The color scale indicates the frequency, computed using as normalization factor the total number of wet cells (about 2.2M). The dashed black line represents the correct forecast. (b) Histogram showing the distribution of the differences between target and predicted water depths.

5.5 Concluding remarks

In this Chapter, the FS-db model was applied for the first time to real-time prediction of DB scenarios. Three case studies were conducted to assess the performance of this surrogate model, each considered to test the model's predictive capabilities under varying conditions of geometry, scale, and flow dynamic complexity.

The first two case studies, involving a channel with a parabolic cross-section and a rectangular tank, respectively, served as synthetic benchmarks to validate the FS-db model's performance. The findings underscored the model's ability to replicate the expected flood evolution during a DB event.

The third case study focused on the Parma River flood-control dam, which introduced additional challenges related to real-world bathymetry. The model's application in this scenario shown its capacity to accurately forecast inundation dynamics over time, achieving an average RMSE of 10.4 cm. The analysis also highlighted the importance of spatial resolution in the model's predictive performance, with higher resolution maps offering more detailed, though slightly less accurate, forecasts due to slightly larger error accumulation. Specifically, doubling the spatial resolution resulted in a 5 cm increase in the average RMSE, resulting in a final value of 15.4 cm.

Additionally, the sensitivity analysis of the number of past frames provided crucial insights into the model's flexibility. Specifically, by reducing the number of past frames required for accurate forecasting, it was shown that the FS-db model could maintain reliable performance even with minimal input data (e.g., the water level in the upstream reservoir before the dam-break), thereby enhancing its potential for real-time applications. For instance, reducing the number of past frames from $P = 8$ to $P = 1$ caused only a marginal increase of 1 cm in the average RMSE.

In summary, the FS-db model has proven to be a versatile and effective tool for predicting DB scenarios. The findings from the case studies analysed suggest that the model is applicable to both theoretical and real-world DB scenarios, offering a valuable resource for flood risk management and emergency response planning.

Chapter 6

River flood scenarios

6.1 Introduction

In this Chapter, the FS-bc model is applied for real-time forecasting of river flood events. The primary objective is to assess the model's performance in forecasting flood events characterized by varying flow dynamics and different topographies.

Three distinct case studies, each with increasing complexity of topography and flood dynamics, are analysed (Table 6.1). The first case study involves simulating river flood events in the Baganza River, in Italy (Section 6.2). Although real terrain topography is considered, this case is quite simple, as the flooded area is restricted to the main channel throughout the event here considered. In contrast, the second case study addresses an impulsive flood event in the urbanized valley of the Toce River, reproduced at the laboratory scale (Section 6.3). The presence of an urban district, combined with the particularly high intensity of the floods, results in complex flood dynamics. Furthermore, the Toce River case has been widely employed in the literature for validating physically based numerical models (e.g., Costabile et al., 2017; Ferrari et al., 2019; Xia et al., 2017), making it a suitable case for evaluating the performance of the proposed FS-bc model. The last case study focuses on forecasting

river flood events along a 48 km-long stretch of the Po River in Italy (Section 6.4). Unlike the previous case studies, this river region is characterized by both open and defended floodplains, which drastically increases the difficulty of predicting flood events in this study area. Moreover, Po River floods are distinguished by slow flow dynamics and long propagation times, with events lasting from days to weeks.

In contrast to the DB scenarios analysed in Chapter 5, simulating river floods requires knowledge of the upstream BC, typically provided as time series of inflow discharges in a specific river section. When simulating historical flood events, the

TABLE 6.1 – River floods: case studies summary and surrogate model hyperparameters.

	Baganza River flood	Toce River flood	Po River flood	
			Res. 20 m	Res. 10 m
Case study number	4	5	6	
Spatial resolution [m]	5	0.05	20	10
Number of cells	1,802,240	16,384	1,306,624	5,226,496
Temporal resolution	0.5 h	0.5 s	3 h	
Maximum water depth in test dataset [m]	3.5	0.14	25.6	
Depth normalization value [m]	4.4	0.18	28.0	
Discharge normalization value [m^3/s]	230	0.25	13,000	
ϵ_{wet} (for metric computation) [m]	0.05	0.001	0.2	
CNN layers (k)	5	3	5	6
Latent feature size ($h \times w \times d_{model}$)	$44 \times 40 \times 768$	$16 \times 16 \times 256$	$44 \times 29 \times 1024$	$44 \times 29 \times 1280$
AE parameters [million]	105	11	182	300
VPTR parameters [million]	462	33	585	826

inflow discharges are commonly obtained by converting the water levels recorded by a gauging station using a rating curve. Differently, for the real-time simulation of river floods, this information usually derives partially or completely from rainfall-runoff models.

For each case study, different flood events have been simulated. The dataset samples were generated as detailed in Section 4.3. The resulting samples were randomly divided into training and validation datasets, with 95% allocated to training and 5% to validation.

To ensure comprehensive training, the datasets for the Toce River and Po River case studies included both real and synthetic flood events (see Table 6.2).

To evaluate the generalization capability of the FS-bc model beyond the training data, at the end of the training procedure, its performance was tested on a separate dataset composed of additional flood events not used during training (see Table 6.2). To further assess the model's predictive ability across floods of varying severity, testing events with diverse intensities and flow dynamics were included.

As previously mentioned, the FS test procedure is used at the end of the training phase to evaluate the surrogate model's accuracy in forecasting one time step ahead. For real-time flood forecasting, which involves predicting an entire flood event, the AR procedure is used.

The water depth maps in the datasets were normalized by dividing them by a value slightly higher than the maximum simulated water depth for each specific case study (refer to Table 6.1). Similarly, for the inflow discharges, a value slightly higher than the highest peak discharge was used (Table 6.1). This normalization step ensured that all samples were scaled within the range of $[0, 1]$, thereby enhancing the effectiveness of the training process.

Insignificant water depths in the simulated maps were removed by setting values below a certain threshold to zero. Specifically, thresholds of 0.001 m, 10^{-5} m, and 0.05 m were applied to the Baganza, Toce, and Po case studies, respectively.

Table 6.1 outlines also the various configurations and hyperparameters used. It

TABLE 6.2 – Training configurations for river flood studies. The training and validation datasets were obtained by dividing the total number of samples with a ratio of 95% for the training and 5% for the validation. The flood events included in the testing datasets differ from those used to generate the training datasets.

Training case	Spatial resolution [m]	Training and validation		Testing		AR prediction
		Flood events	Number of samples	Flood events	Number of samples	
Baganza River	5	11 real	752	2 real + 1 synthetic	360	Testing events (i.e., 3)
Toce River						
Toce1	0.05	15 *Rapid*	1056	3 *Rapid* (*Low*, *Medium*, *High*) + 1 *Gradual*	396	Testing events (i.e., 4)
Toce2	0.05	15 *Rapid* + 7 *Gradual*	1782			
Po River						
Po1	20	19 real	2320	3 real (Nov 2011, Nov 2014, and Jun 2020)	423	Testing events (i.e., 3)
Po2	20	19 real + 6 synthetic	3144			
Po3	10	19 real + 6 synthetic	3144			

is important to note that the Toce River case study, being at the laboratory scale, involves different spatial and temporal scales compared to the other cases and requires adjustments to parameters such as spatial and temporal resolutions, normalization factors, and the wet-dry threshold (ϵ_{wet}) to fit its specific characteristics. Similarly, the results and errors associated with each case study also vary in terms of magnitude.

As shown in Section 5.4.3, the recursive forecasting of DB scenarios is only marginally influenced by the number of past frames (P) used as the initial condition. Preliminary analysis confirmed that this observation holds true also for river flood predictions as well. Consequently, to reduce the amount of prior information required for real-time forecasting with the FS-bc model, the results presented in this Chapter

were obtained using only a single map as the initial condition (i.e., setting $P = 1$). This means that the FS-bc model can forecast the sequence of inundation maps for an entire flood event using just one initial condition map and the inflow discharge for the entire event duration, making it highly suitable for real-time forecasting applications.

In addition to presenting the results concerning model accuracy, this Chapter includes a benchmark comparison of the FS-bc model against a state-of-the-art DL model (see Section 6.5). Details regarding the computational times of the proposed model will be discussed in the next Chapter (see Section 7.1).

The case studies and results presented here are derived from Pianforini et al. (2024a).

6.2 Baganza River flood

6.2.1 Case study description

In this case study, a 10 km-long stretch of the Baganza River in Italy, located just upstream of the city of Parma, is considered (see Fig. 6.1). The upstream section is positioned downstream the Sala Baganza bridge, while the downstream boundary is set at the confluence of the Baganza and Parma rivers, near the city center of Parma. The primary objective of this case study is to evaluate the performance of the FS-bc model in predicting flood propagation within a river channel.

The main channel of the Baganza River varies in width from 40 m to 200 m. To accurately reproduce the flood events, a DTM with a spatial resolution of 5 m, derived from a LIDAR survey, was utilized (Table 6.1). Furthermore, a uniform Manning's roughness coefficient of $0.04\,\mathrm{sm}^{-1/3}$ was applied, based on a prior calibration carried out by simulating historical flood events. For the initial condition, a steady-flow numerical simulation was conducted with an inflow discharge of $10\,\mathrm{m}^3/\mathrm{s}$. A uniform flow condition was applied at the downstream boundary.

To generate the training and validation datasets, 11 real flood events (illustrated in Fig. 6.2a) were simulated using the PARFLOOD code. For the testing dataset

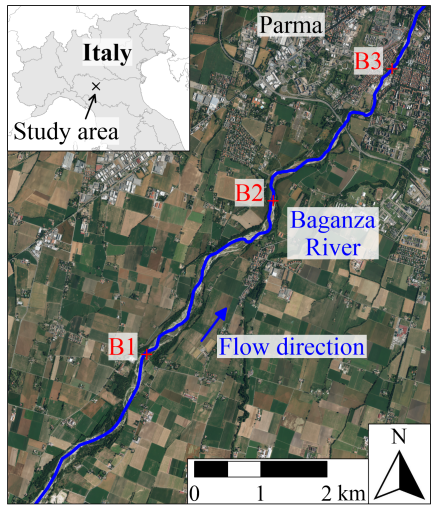

FIGURE 6.1 – Case study 4: Baganza River study area.

(Fig. 6.2b), two additional real flood events, which occurred in December 2019 and January 2021, along with a synthetic flood event featuring a double peak, were simulated. The output maps from the hydrodynamic model were sampled at 30-min intervals, which is sufficient to accurately capture the flow variations in the Baganza River.

Three control points (B1, B2, and B3), located along the Baganza River, as shown in Fig. 6.1, were selected to compare the predicted and target water depths at these locations.

6.2.2 Results

Focusing on the FS test, which evaluates the performance of the surrogate model in predicting one map ahead, Table 6.3 summarizes the average performance metrics for all the testing events. The average RMSE is less than 4.5% of the average water depth across the testing events, reported in Table 6.3. The temporal variation of this metric for each flood event is illustrated in Fig. 6.3. In general, the RMSEs are below 5-6 cm, except during the initial stages of the December 2019 event, where the

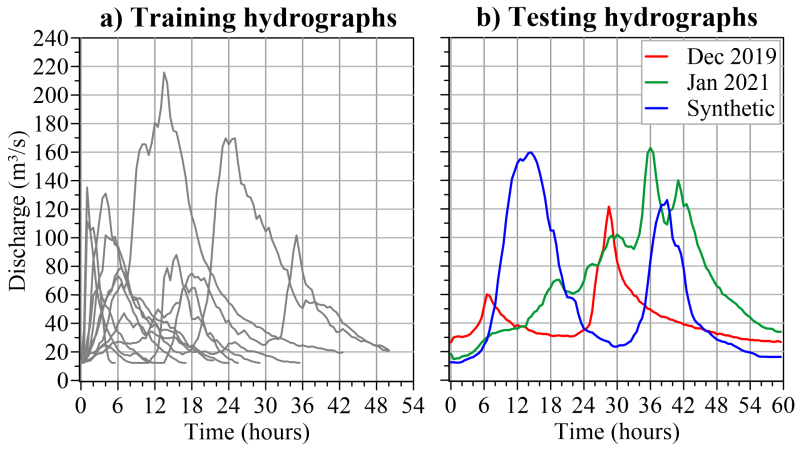

FIGURE 6.2 – Inflow hydrographs of the Baganza River case study. (a) 11 real events used to create the training dataset. (b) 2 real events (December 2019 and January 2021) and 1 synthetic event used to generate the testing dataset.

influence of the adopted initial condition is more pronounced, as explained later.

The consistently low RMSE values and the F1 scores close to 1 confirm the excellent performance of the FS-bc model in predicting one future map, and allows for the application of the model also for forecasting entire flood events.

For the AR prediction, the numbers of past frames (P) and future frames (F) were set to 1 and 118, respectively. Table 6.4 presents the average performance metrics for the three recursively forecasted events in the testing dataset. The temporal variation of the RMSE and RMSE_ND is shown in Fig. 6.3. Generally, the average RMSEs are only slightly higher than those obtained in the FS test (Table 6.3), further

TABLE 6.3 – Baganza River case study: average RMSE, RMSE_ND, and F1 score for the FS test procedure. The average water depth ($\bar{h}$) computed considering all wet cells for each event is also provided.

Testing event	$\bar{h}$ [m]	RMSE [m]	RMSE_ND [-]	F1 [-]
Dec 2019	0.54	0.020	0.037	0.987
Jan 2021	0.63	0.026	0.040	0.987
Synthetic	0.58	0.026	0.043	0.987

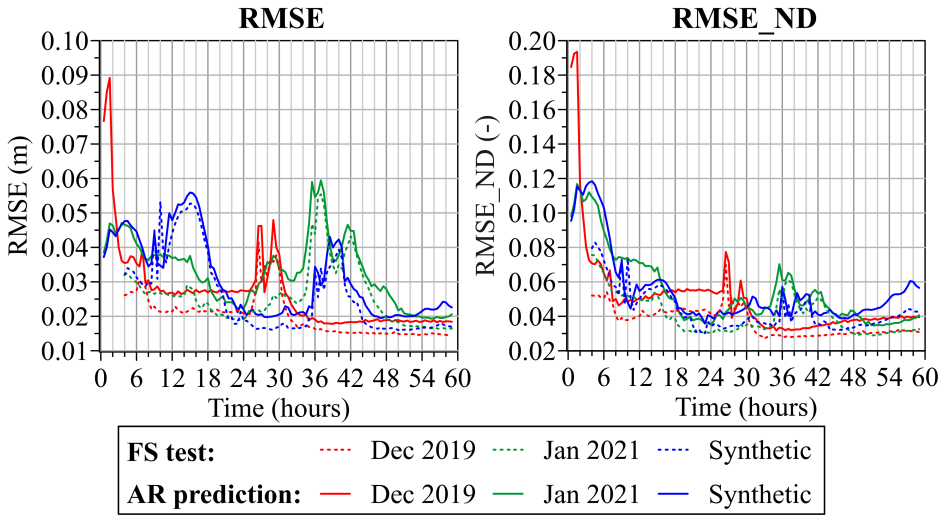

FIGURE 6.3 – Baganza River case study: RMSE and RMSE_ND for the three testing events computed for the FS test and AR prediction.

demonstrating the high accuracy of the surrogate model in predicting entire flood events in the Baganza River region.

Fig. 6.4 compares the time series of water depths extracted from the target and predicted maps at the three control points (see Fig. 6.1 for locations). The surrogate model shows high accuracy in predicting water depth variations at all control points, with average RMSE values ranging from 0.7 cm to 3.1 cm. Some discrepancies occur for the first predicted maps due to the initial condition, especially for the December 2019 event, but the model soon "forgets" the initial condition data, following the expected trend after a few frames. As a result, the overall accuracy of the inference

TABLE 6.4 – Baganza River case study: average RMSE, RMSE_ND, and F1 score for the AR prediction of the testing flood events.

Testing event	RMSE [m]	RMSE_ND [-]	F1 [-]
Dec 2019	0.027	0.051	0.984
Jan 2021	0.033	0.055	0.984
Synthetic	0.031	0.055	0.984

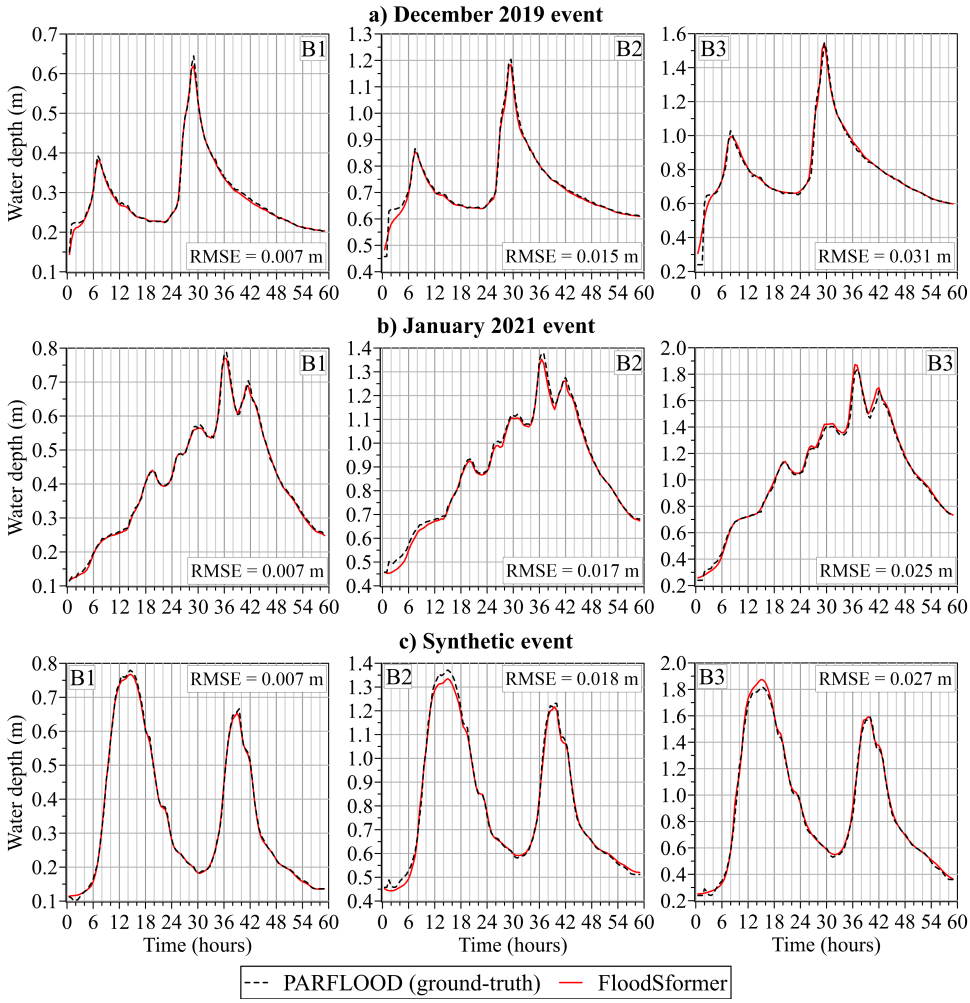

FIGURE 6.4 – Comparison between target and predicted water depths at control points for the Baganza River case study. For each control point, the average RMSE is reported. (a) December 2019 flood event. (b) January 2021 flood event. (c) Synthetic flood event.

process is only marginally affected by the map used as the past frame (see also Section 6.4.3). Despite the multiple peak behavior of the flood events considered for testing, the surrogate model accurately reproduces the expected flood dynamics.

Figs. 6.5 and 6.6 show the forecasted inundation maps for selected moments of the January 2021 flood event, generated using the AR procedure, compared with the

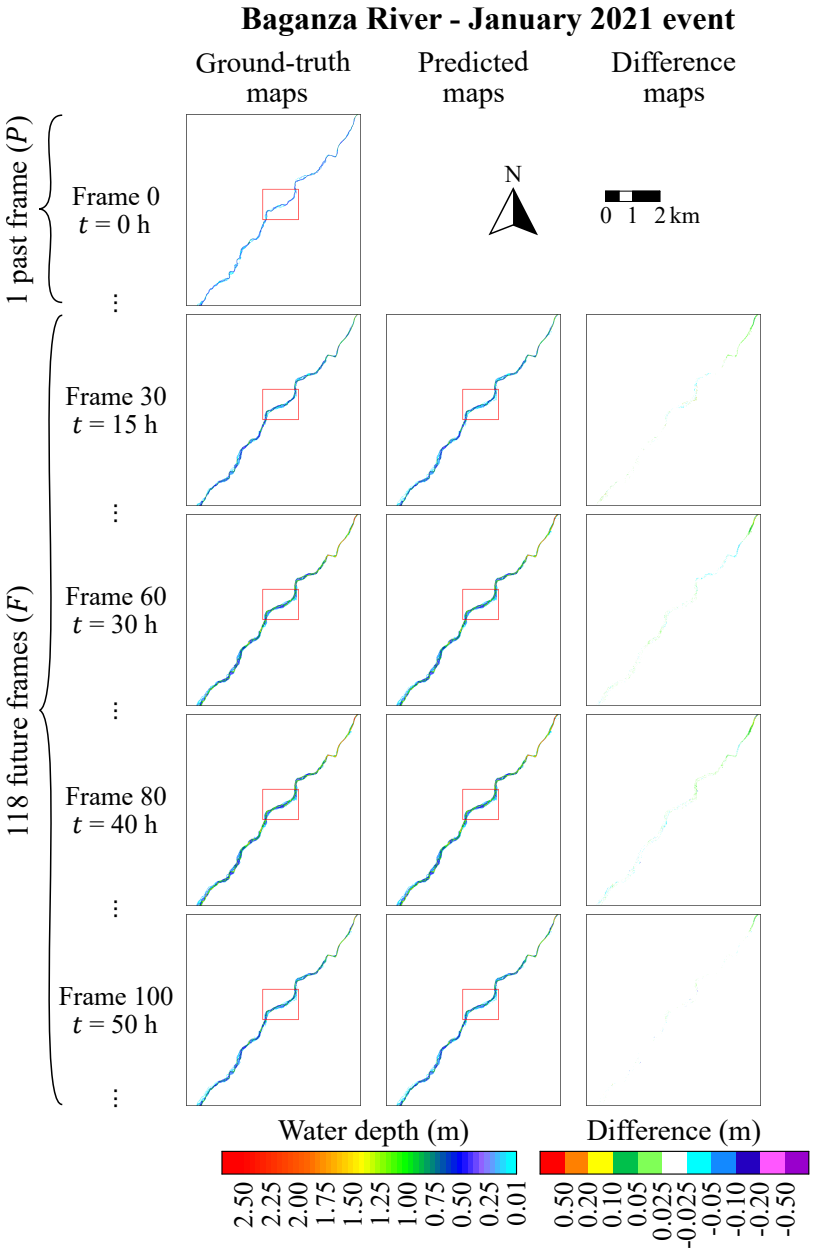

FIGURE 6.5 – Baganza River case study: real-time forecasting of the January 2021 flood event using the AR procedure of the FS-bc model. The columns represent the target maps from the hydrodynamic model, the maps predicted from the surrogate model, and the maps of differences between the predicted and target maps. Only selected time steps are shown. The red square indicates the magnified area shown in Fig. 6.6.

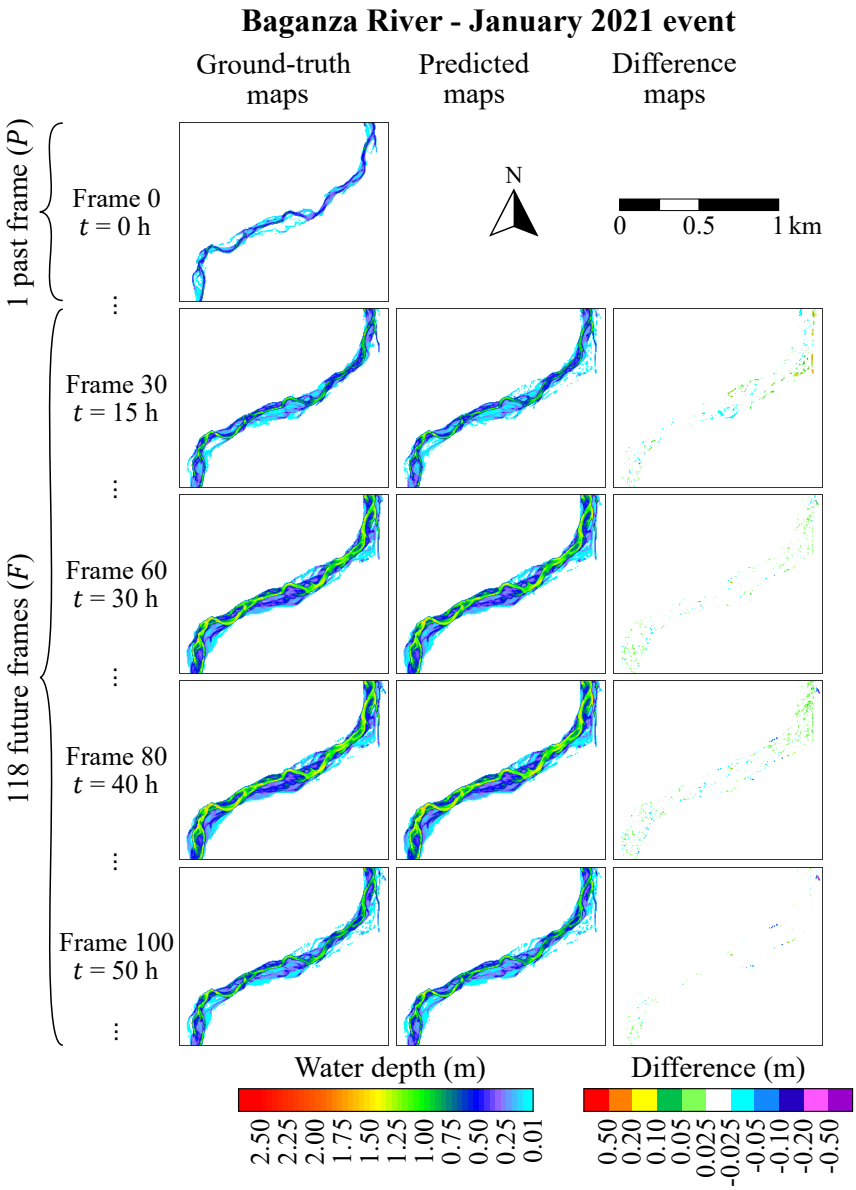

FIGURE 6.6 – Baganza River case study: zoom into the region marked in Fig. 6.5. The same time steps of Fig. 6.5 are shown.

target maps. The difference maps confirm the high accuracy of the surrogate model. Similar results were obtained for the other two flood events in the testing dataset, which are not displayed.

6.3 Toce River flood

6.3.1 Case study description

This case study investigates an urban flash flood at a laboratory scale. The study's geometry is based on a physical model developed by Testa et al. (2007), which represents a 1:100 scale model of a segment of the Toce River valley in Italy. The model incorporates 18 concrete cubic blocks arranged in a staggered pattern in the center of the domain to simulate an urban district (Fig. 6.7). In their experiments, Testa et al. (2007) analyzed three inflow hydrographs with varying peak discharge levels, referred to as *Low*, *Medium*, and *High* (Fig. 6.8c). Water depths were continuously monitored at nine gauge points (P2 to P10 in Fig. 6.7) throughout the duration of the experiments.

To generate the ground-truth water depth maps for training and testing the surrogate model, the PARFLOOD model was employed, utilizing a DTM with a spatial resolution of 0.05 m. The numerical simulations assumed a far-field BC downstream and an initially dry domain. Water depth maps were sampled every 0.5 s to capture the rapid flood propagation throughout the study area.

To create a comprehensive dataset of water depth maps, the three hydrographs recorded by Testa et al. (2007), along with 23 synthetic events (see Fig. 6.8), were used as upstream BC in the numerical simulations. The synthetic events were derived from discharge records at a gauging station on the Toce River. These recorded hydrographs were scaled in both time and magnitude to match the physical model's scale.

This case study serves two main purposes: (a) to validate the performance of the FS-bc model in predicting impulsive flood events with complex flow dynamics, and

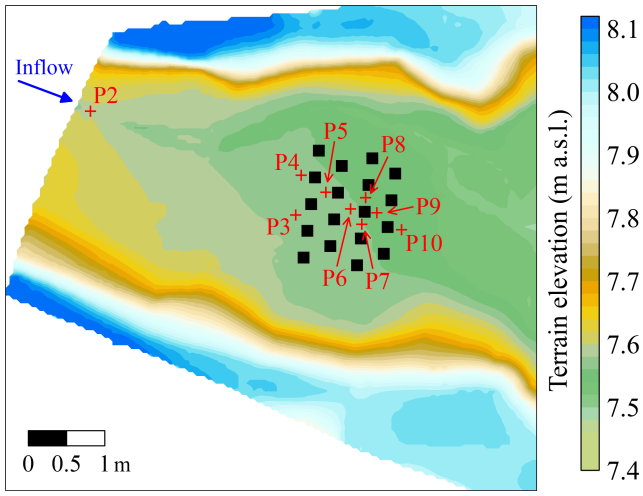

FIGURE 6.7 – Case study 5: Toce River study area. The black squares represent the urban district.

(b) to assess the impact of the types of flood event used in the training dataset on the FS-bc model's accuracy in forecasting unseen floods with different characteristics. Accordingly, two distinct datasets were used for training the surrogate mode (see Table 6.2).

The first training scenario, termed *Toce1*, comprised a dataset of 15 synthetic events (Fig. 6.8a), referred to as *Rapid* hydrographs. These hydrographs are characterized by a steep rising limb and a peak occurring within the first 6 s, similar to the ones of the experimental tests.

The second training scenario, referred to as *Toce2*, included the same 15 *Rapid* flood events along with 7 additional synthetic hydrographs, labeled *Gradual* (Fig. 6.8b), which feature a slower rising limb compared to the *Rapid* hydrographs.

For both training scenarios, the testing dataset remained consistent (Table 6.2), containing samples from 3 experimental test simulations and 1 synthetic hydrograph with a *Gradual* trend (Fig. 6.8c). These testing hydrographs were excluded from the training processes in order to evaluate the model's ability to predict unseen events.

Calibration of the PARFLOOD code

For this case study, the PARFLOOD model was calibrated by comparing simulated water depths at 9 gauge points (depicted in Fig. 6.7) with observed data from three experimental hydrographs (*Low*, *Medium*, and *High*) provided by Testa et al. (2007). This comparison is crucial to validate the accuracy of the ground-truth maps used for training the surrogate model. Figs. 6.9–6.11 present the time series of water

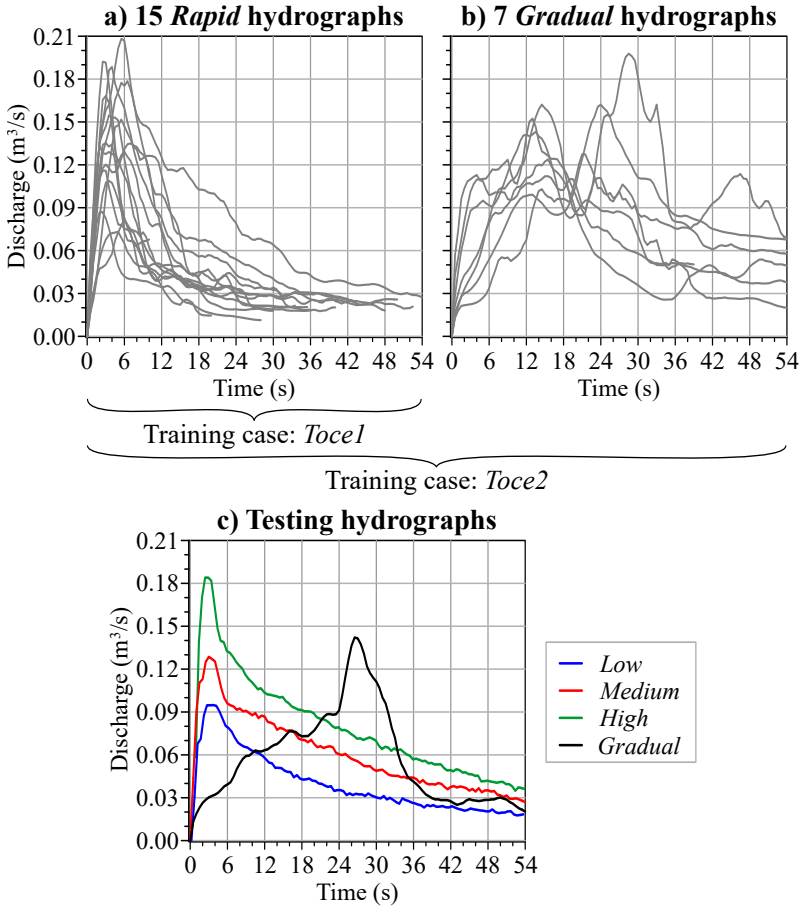

FIGURE 6.8 – Inflow hydrographs for the Toce River case study. (a) 15 *Rapid* hydrographs used to generate the training dataset for the *Toce1* case. (b) 7 additional *Gradual* hydrographs used to create the *Toce2* training dataset. (c) 3 recorded hydrographs (*Low*, *Medium*, and *High*) and one synthetic hydrograph (*Gradual*) used to generate the testing dataset.

depths extracted at the control points for the three experimental events, using a uniform Manning's roughness coefficient of $0.0162\,\mathrm{sm}^{-1/3}$, as recommended by Testa et al. (2007). Although some discrepancies are evident, particularly within the urban area (e.g., gauges P6-P8), which is a common challenge for various numerical models (e.g., Xia et al., 2017), the PARFLOOD model successfully captures the overall flood dynamics. This validation underscores the reliability of using numerically generated datasets for training and testing the surrogate model in this case study.

6.3.2 Results and comparison between different training configurations

As detailed in Section 6.3.1, the surrogate model was trained using the two configurations, *Toce1* and *Toce2*, which have been detailed in Table 6.2. The two training configurations have been compared by forecasting the four flood events from the testing dataset (Fig. 6.8c).

For the FS test procedure, the average performance metrics for both the training configurations are summarized in Table 6.5, while the temporal variation of the RMSE and RMSE_ND is presented in Figs. 6.12–6.15. The average RMSEs range between 0.6 mm and 1.9 mm, which corresponds to less than 4% of the average water depth for the specific testing scenario (see Table 6.5). For the three experimental hydrographs, the RMSE of the FS test procedure is generally below 2 mm across all events, with minimal differences between the two training configurations (Figs. 6.12–6.14). However, for the *Gradual* hydrograph, higher RMSEs are observed near the peak of the flood event, around $t = 28$ s (Fig. 6.15). The *Toce2* configuration helps mitigate these errors.

An F1 score close to 1 (see Table 6.5) further confirms the high accuracy of the FS-bc model in predicting the temporal variation of the flood extent one map ahead.

For the AR prediction of river flood events in the Toce River valley, only one map was used as the initial condition (i.e., $P = 1$), while the number of future frames (F) recursively predicted by the surrogate model was set to 106, corresponding to

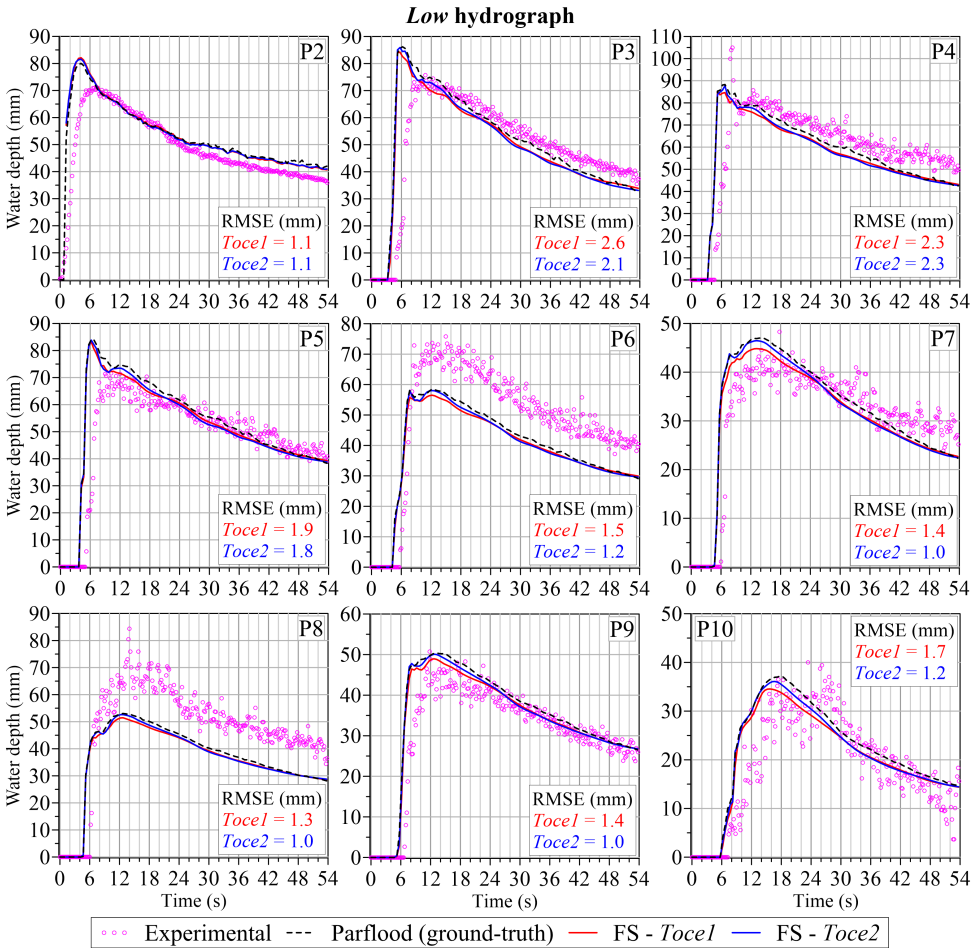

FIGURE 6.9 – Toce River case study: recorded and simulated water depths at control points for the *Low* event. The magenta circles represent the recorded water depths from the experimental analysis (Testa et al., 2007). Average RMSEs comparing ground-truth and predicted water depths for both the *Toce1* and *Toce2* training configurations are shown in each graph.

a lead time of 53 s. Table 6.6 summarizes the average performance metrics for the recursively predicted maps for different training configurations and testing events. The high F1 scores across the scenarios highlight the FS-bc model's accuracy in predicting the temporal variation of flood extent.

For the *Low*, *Medium*, and *High* hydrographs (see Figs. 6.12–6.14), the RMSE is

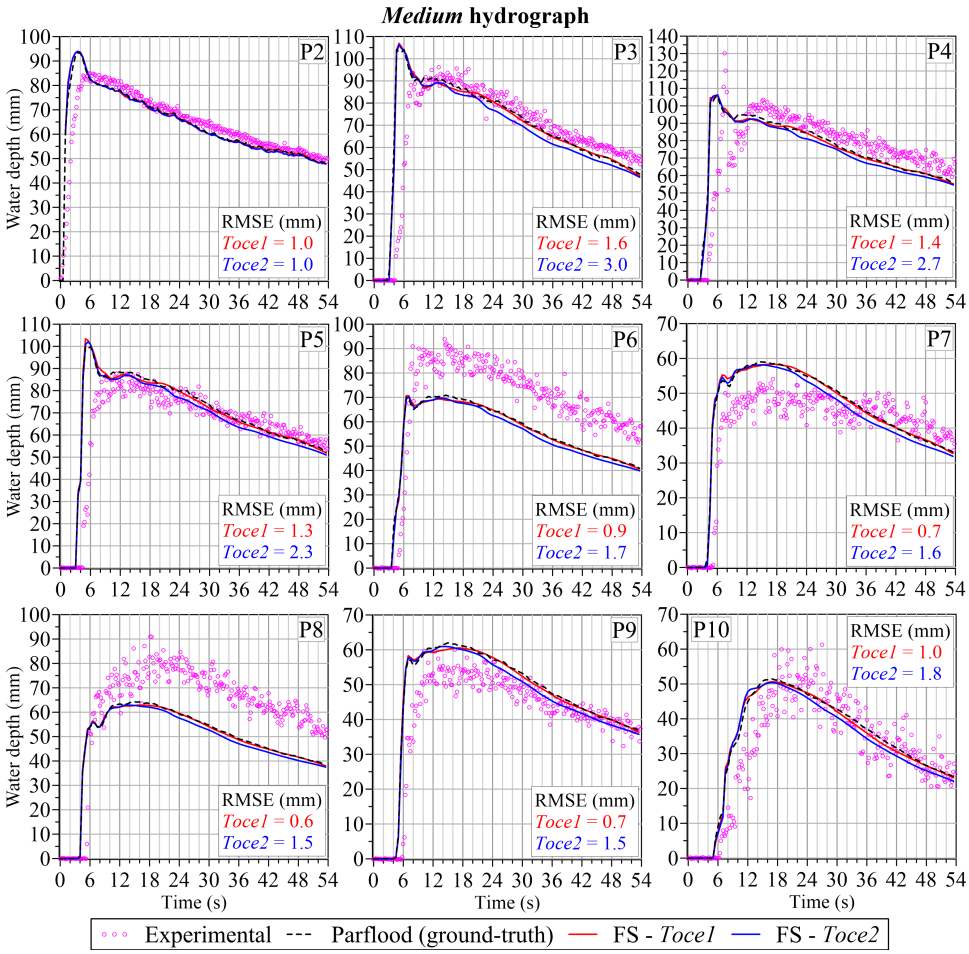

FIGURE 6.10 – Toce River case study: recorded and simulated water depths at control points for the *Medium* event. The magenta circles represent the recorded water depths from the experimental analysis (Testa et al., 2007). Average RMSEs comparing ground-truth and predicted water depths for both the *Toce1* and *Toce2* training configurations are shown in each graph.

relatively high at the initial predicted maps due to underestimation of the flood's wet/dry front, but the errors decrease as the flood progresses through the urban district. For the *Gradual* hydrograph (see Fig. 6.15), accuracy is more sensitive to the training dataset, particularly in the initial predicted maps. The model trained with the *Toce1* configuration exhibits higher RMSEs in the first 30-36 s, whereas the *Toce2*

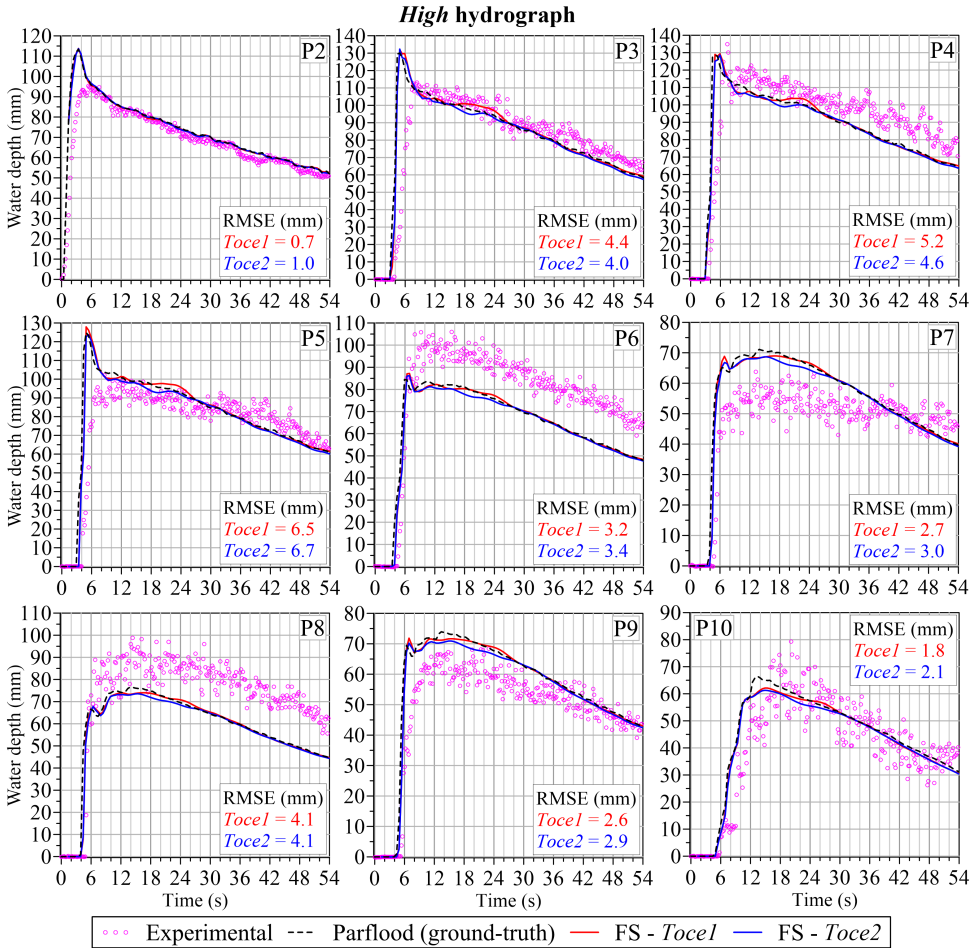

FIGURE 6.11 – Toce River case study: recorded and simulated water depths at control points for the *High* event. The magenta circles represent the recorded water depths from the experimental analysis (Testa et al., 2007). Average RMSEs comparing ground-truth and predicted water depths for both the *Toce1* and *Toce2* training configurations are shown in each graph.

configuration shows significantly lower errors in the initial 18-24 s, owing to a broader representation of similar events in its training data. During the period immediately following the flood's peak (around 28 s) the RMSE remains only marginally lower using the *Toce2* configuration, stabilizing around 1 cm.

A more detailed analysis of the predictions using the AR procedure from the

TABLE 6.5 – Toce River case study: average RMSE, RMSE_ND, and F1 score computed for the FS test. The average water depth ($\bar{h}$) computed considering all wet cells for each event is also provided.

Testing event	$\bar{h}$ [mm]	RMSE [mm]		RMSE_ND [-]		F1 [-]	
		Toce1	*Toce2*	*Toce1*	*Toce2*	*Toce1*	*Toce2*
Low	33.5	0.9	0.8	0.027	0.026	0.996	0.996
Medium	43.6	0.7	0.6	0.016	0.013	0.996	0.997
High	52.0	1.0	0.8	0.018	0.015	0.996	0.996
Gradual	40.9	1.9	1.3	0.044	0.030	0.993	0.995

TABLE 6.6 – Toce River case study: average RMSE, RMSE_ND, and F1 score for the AR prediction of the testing flood events across different training configurations.

Testing event	RMSE [mm]		RMSE_ND [-]		F1 [-]	
	Toce1	*Toce2*	*Toce1*	*Toce2*	*Toce1*	*Toce2*
Low	2.1	1.8	0.061	0.056	0.994	0.995
Medium	2.0	2.6	0.044	0.060	0.995	0.995
High	3.5	3.2	0.067	0.061	0.991	0.991
Gradual	4.8	3.6	0.123	0.083	0.988	0.991

FS-bc model trained with both configurations is discussed next.

***Toce1* training configuration**

Figs. 6.9–6.11 and 6.16 present a comparison between the ground-truth water depths (dashed black lines) and the recursively predicted depths (red lines) at control points for all testing events. The surrogate model exhibits high accuracy in predicting the flood arrival time at all control points and accurately replicates water depth peaks, particularly for the experimental hydrographs (*Low*, *Medium*, and *High*).

For the *Medium* hydrograph (Fig. 6.10), the model shows its highest accuracy, with average RMSEs below 1.6 mm. Dividing these RMSEs by the maximum expected water depth at each control point (e.g., 95 mm for point P2), the relative errors are less than 2%. Although the model slightly underestimates water depths at some

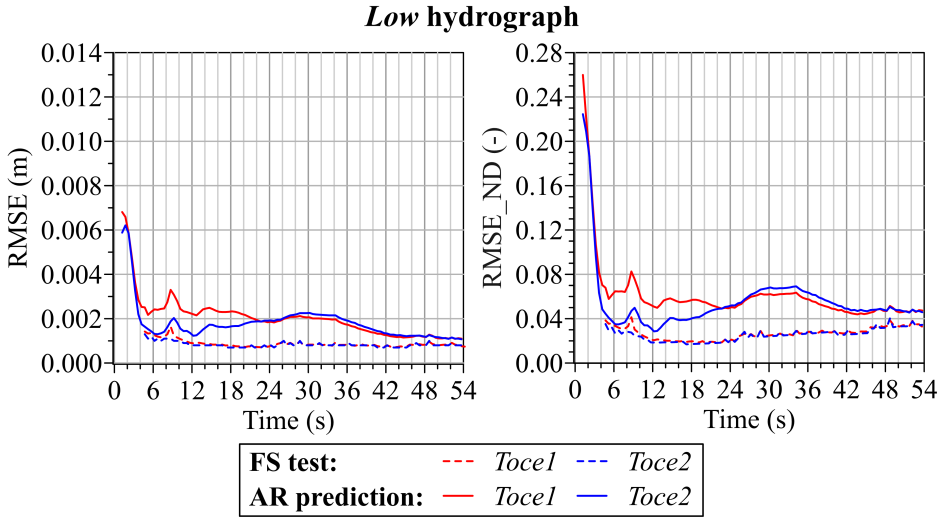

FIGURE 6.12 – Toce River case study: RMSE and RMSE_ND for the *Low* event computed for the FS test (dashed lines) and AR prediction (continuous lines) using the two training configurations.

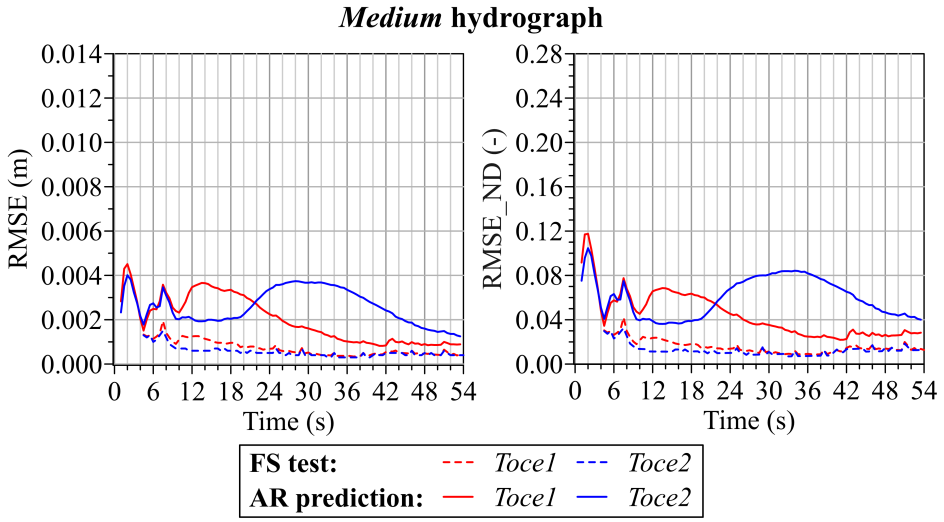

FIGURE 6.13 – Toce River case study: RMSE and RMSE_ND for the *Medium* event computed for the FS test (dashed lines) and AR prediction (continuous lines) using the two training configurations.

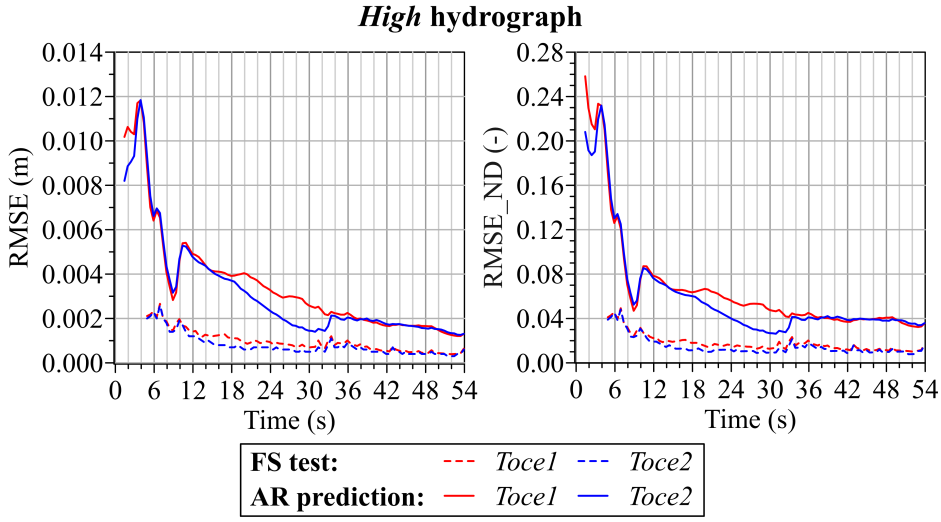

FIGURE 6.14 – Toce River case study: RMSE and RMSE_ND for the *High* event computed for the FS test (dashed lines) and AR prediction (continuous lines) using the two training configurations.

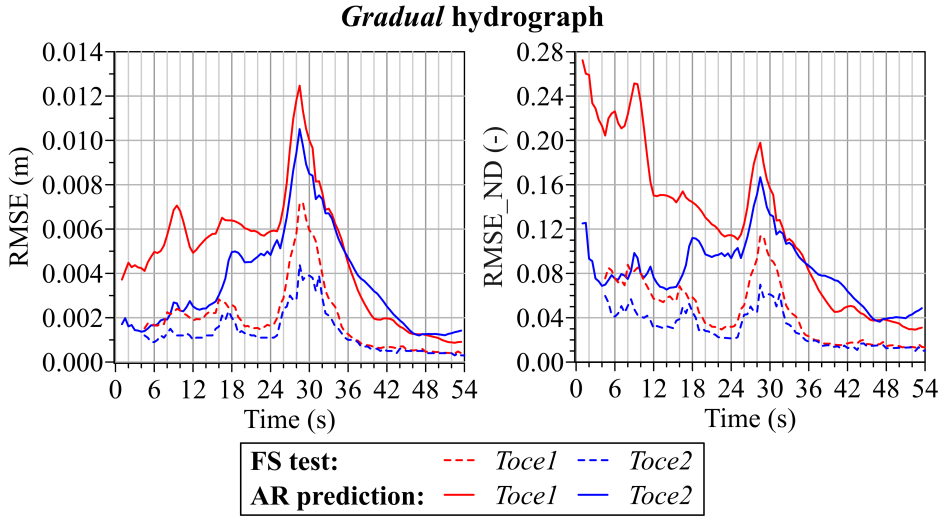

FIGURE 6.15 – Toce River case study: RMSE and RMSE_ND for the *Gradual* event computed for the FS test (dashed lines) and AR prediction (continuous lines) using the two training configurations.

control points for the *Low* and *High* hydrographs (Figs. 6.9 and 6.11, respectively), the relative errors remain below 5%, confirming the robust performance of the FS-bc model.

As anticipated, the FS model trained with the *Toce1* dataset performs well in predicting flood propagation for the three experimental hydrographs, as these events are similar to the *Rapid* hydrographs in the training dataset. However, the model's

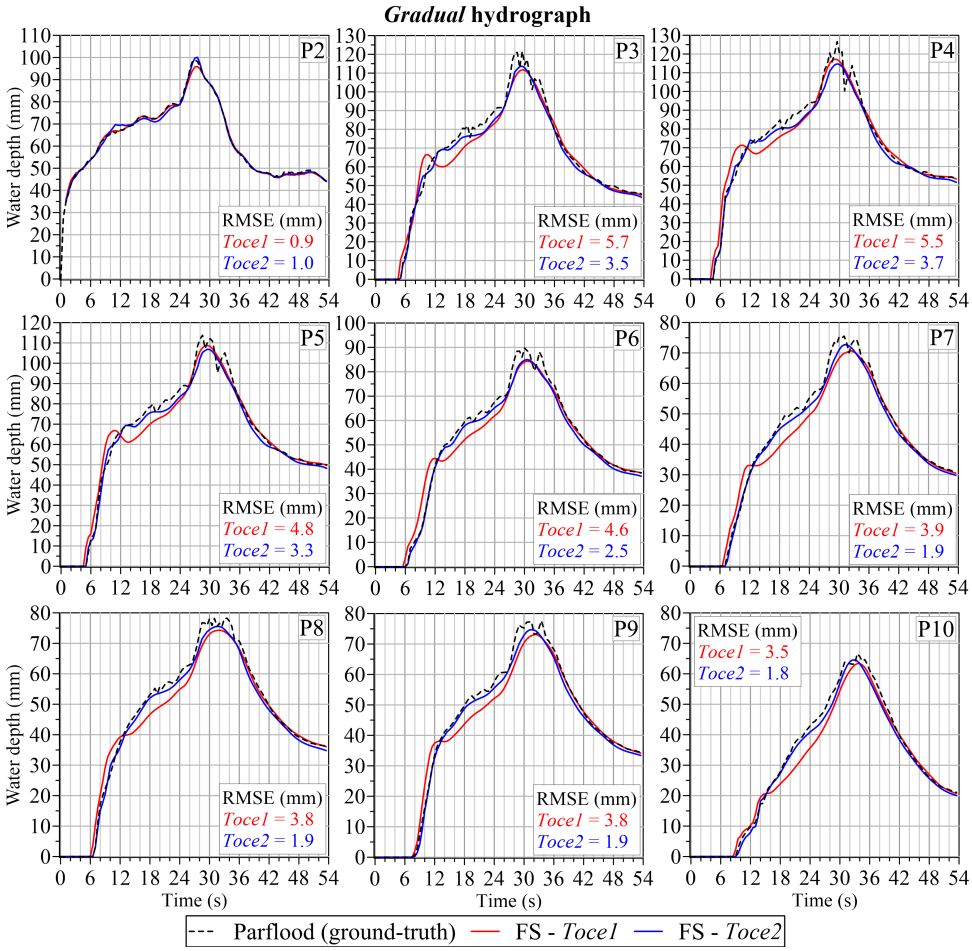

FIGURE 6.16 – Toce River case study: recorded and simulated water depths at control points for the *Gradual* event. Average RMSEs comparing ground-truth and predicted water depths for both the *Toce1* and *Toce2* training configurations are shown in each graph.

accuracy diminishes when predicting the flood generated by the *Gradual* hydrograph, which presents significantly different characteristics compared to the *Toce1* training dataset. For instance, at P3 in Fig. 6.16, the predicted water depth begins to decrease around 10.5 s, contrary to the expected trend of a monotonic increase. Similar issues, though less pronounced, are observed at other control points.

Figs. 6.17–6.20 illustrate the comparison between predicted and ground-truth inundation maps at representative instants of the three flood events for the *Toce1* training case. For the experimental hydrographs, discrepancies are primarily observed near the wet/dry front during the initial predicted maps. In contrast, for the *Gradual* hydrograph (Fig. 6.20), the surrogate model generates more spatially distributed errors, particularly in the vicinity of the urban district.

Toce2 training configuration

The second training configuration (*Toce2*) is designed to evaluate how the accuracy of the FS-bc model evolves when both *Rapid* and *Gradual* flood events are included in the training dataset (Fig. 6.8a,b).

A comparison of water depths extracted at control points for the two training configurations (with *Toce2* results shown in blue) is presented in Figs. 6.9–6.11 and 6.16. The FS model trained with the *Toce2* dataset delivers more accurate predictions for the *Gradual* hydrograph, with average RMSEs roughly halved. The predicted water depths also exhibit a more consistent trend during the rising limb of the flood, indicating that incorporating *Gradual* hydrographs into the training data partially addresses the limitations identified in the *Toce1* configuration. This enhancement is further demonstrated by comparing the forecasted water depth maps for both trained models in Fig. 6.20.

The inclusion of *Gradual* hydrographs in the *Toce2* training dataset does not compromise the model's performance when predicting *Rapid* floods. The accuracy remains comparable for the *Low* and *High* events (Figs. 6.9 and 6.11), with only a minor increase in RMSEs observed at certain control points during the *Medium*

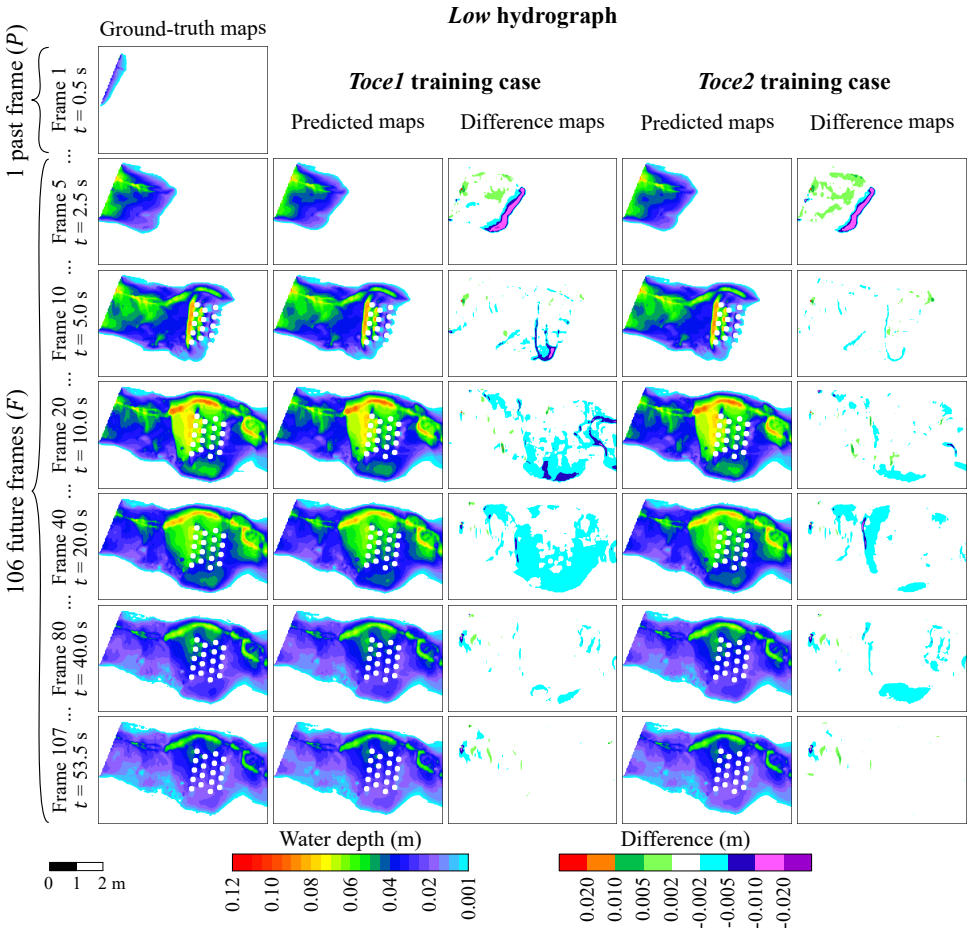

FIGURE 6.17 – Toce River case study: comparison of predicted maps for the *Low* event using the surrogate model trained with the *Toce1* and *Toce2* configurations. The first column presents the ground-truth maps generated by the hydrodynamic model. The second and fourth columns show the predicted maps for the *Toce1* and *Toce2* configurations, respectively. The third and fifth columns depict the differences between the predicted and target maps for the *Toce1* and *Toce2* configurations, respectively. Only selected representative time steps are presented.

event (Fig. 6.10). Overall, the ratio of average RMSE to the maximum water depth at a control point remains within 2.5% to 3.5%.

A detailed comparison of predicted water depth maps for the two training configurations is provided in Figs. 6.17–6.20, showing that the surrogate model maintains

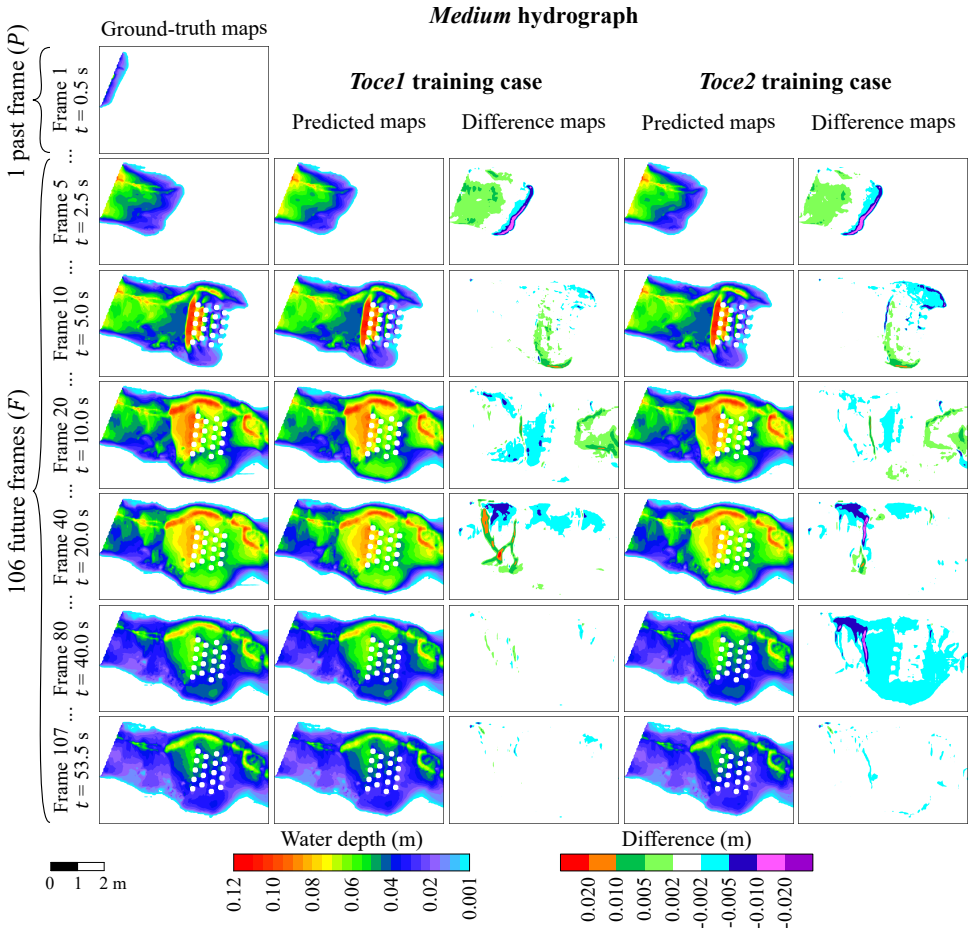

FIGURE 6.18 – Toce River case study: comparison of predicted maps for the *Medium* event using the surrogate model trained with the *Toce1* and *Toce2* configurations. The first column presents the ground-truth maps generated by the hydrodynamic model. The second and fourth columns show the predicted maps for the *Toce1* and *Toce2* configurations, respectively. The third and fifth columns depict the differences between the predicted and target maps for the *Toce1* and *Toce2* configurations, respectively. Only selected representative time steps are presented.

high accuracy throughout the flood events. For all events, the largest discrepancies occur near the hydraulic jump located upstream of the urban district and at the wet/dry front during the early stages of the flood. It is important to emphasize that accurately predicting the hydraulic jump is particularly challenging due to the

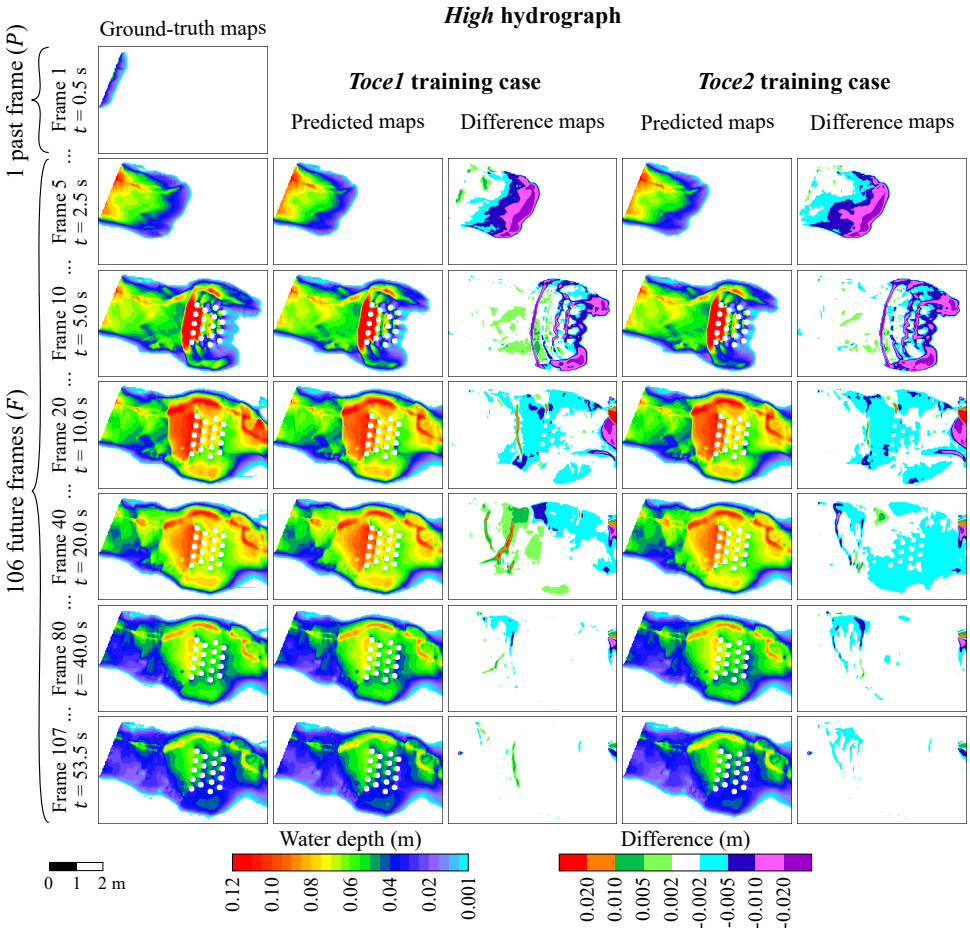

FIGURE 6.19 – Toce River case study: comparison of predicted maps for the *High* event using the surrogate model trained with the *Toce1* and *Toce2* configurations. The first column presents the ground-truth maps generated by the hydrodynamic model. The second and fourth columns show the predicted maps for the *Toce1* and *Toce2* configurations, respectively. The third and fifth columns depict the differences between the predicted and target maps for the *Toce1* and *Toce2* configurations, respectively. Only selected representative time steps are presented.

abrupt change in water depth. As a result, even a slight error in forecasting the location of this discontinuity can lead to substantial differences between the target and predicted water depths in the vicinity of the jump.

In summary, the inclusion of diverse inflow hydrographs in the training dataset

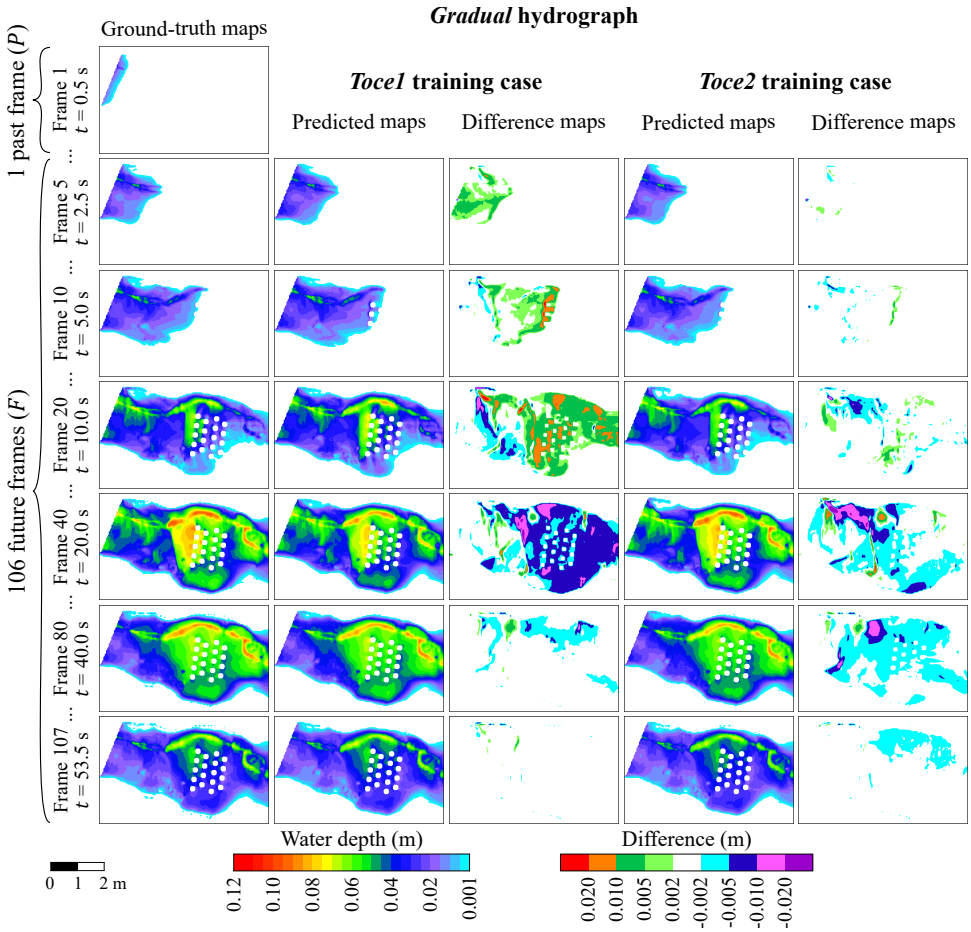

FIGURE 6.20 – Toce River case study: comparison of predicted maps for the *Gradual* event from the testing dataset using the surrogate model trained with the *Toce1* and *Toce2* configurations. The first column presents the ground-truth maps generated by the hydrodynamic model. The second and fourth columns show the predicted maps for the *Toce1* and *Toce2* configurations, respectively. The third and fifth columns depict the differences between the predicted and target maps for the *Toce1* and *Toce2* configurations, respectively. Only selected representative time steps are presented.

significantly enhances the overall performance of the FS-bc model.

6.4 Po River flood

6.4.1 Case study description

The last case study focuses on predicting river floods along a 48 km-long stretch of the Po River in Italy, located between the Casalmaggiore and Borgoforte gauging stations (see Fig. 6.21). In this segment, the river width ranges from 300 m to 3 km, while the main channel consistently maintains a width of approximately 300 m. The region is characterized by defended floodplains, protected by minor levees designed to withstand floods with a return period up to 50 years. Additionally, the surrounding lowlands are defended by main embankments, which exceed 8 m in height and are designed to offer protection against 1-in-200-year flood events. These structural defenses significantly complicate flood propagation dynamics, requiring the use of fully 2D hydraulic models for accurate simulations (Dazzi et al., 2021a). Consequently, the 2D SWE solver, PARFLOOD, was employed to generate the ground-truth maps used to train the surrogate model.

A DTM with a 2 m resolution was developed for the study area by merging LIDAR and bathymetric surveys. To reduce computational demands, the DTM was downsampled to resolutions of 10 m and 20 m, resulting in grids with approximately 5.2M and 1.3M cells, respectively (see Table 6.1). These resolutions were deemed sufficient for simulating flood propagation in the Po River region, given that the main channel exceeds 200 meters in width. To ensure accurate modeling of embankment overtopping, the crest elevations of both major and minor levees were preserved in the downsampled grids using terrestrial survey data. All simulations assumed that the embankments were non-erodible, meaning the defended floodplains would only be inundated if the minor levees were overtopped.

The upstream boundary of the model was imposed at the Casalmaggiore section, while a rating curve was applied at the downstream Borgoforte section. To construct the dataset, 22 historical flood events from 2000 to 2021 were considered. The corresponding water levels, recorded at the Casalmaggiore gauging station, were

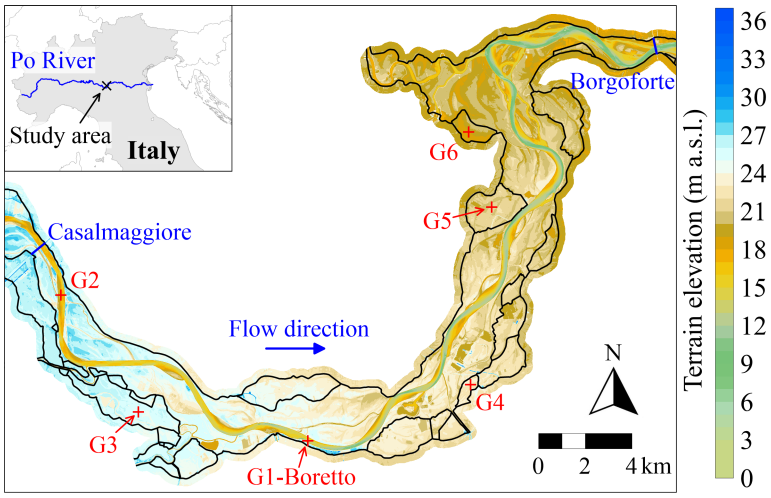

FIGURE 6.21 – Case study 6: Po River study area. The black lines represent the Po River levees.

converted to discharge values (Fig. 6.22a,c) using a rating curve. In addition to these historical events, 6 synthetic flood scenarios with peak discharges exceeding 8,000 m^3/s were incorporated into the dataset (Fig. 6.22b). The temporal resolution for both the inflow hydrographs and the resulting inundation maps was set at 3 hours, which is appropriate for capturing the slow flood propagation characteristic of the Po River.

Six different initial water depth maps were generated through steady-flow numerical simulations, using upstream discharge values ranging from 500 m^3/s to 2,500 m^3/s in increments of 500 m^3/s. For each flood event, the initial condition corresponding to the discharge value most closely matching the actual initial discharge of the event was employed for the numerical simulation.

The FS model was trained using three distinct configurations, detailed in Table 6.2. The two main objectives were: (a) to evaluate how the type and severity of flood events included in the training dataset affect the model's ability to generalize to unseen floods, and (b) to assess how spatial resolution influences both accuracy and computational efficiency. For the first objective, the training configurations

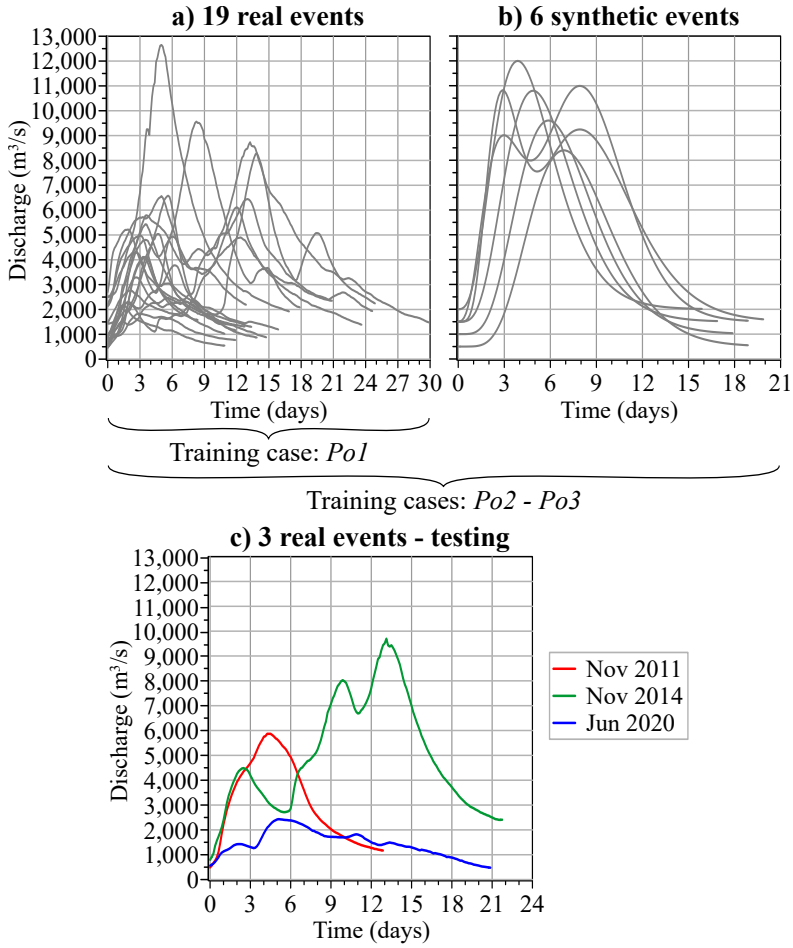

FIGURE 6.22 – Inflow hydrographs of the Po River case study. (a) 19 real flood events between 2000 and 2021 used in the training dataset for the *Po1* case. (b) 6 additional synthetic scenarios in the training dataset for the *Po2* and *Po3* cases. (c) Recorded events in November 2011, November 2014, and June 2020 used to generate the testing dataset.

Po1 and *Po2* were compared, both using a spatial resolution of 20 m. In *Po1*, the training dataset consisted of 19 historical flood events, with inflow hydrographs featuring peak discharges ranging from approximately 2,000 m^3/s to 13,000 m^3/s and flood durations between 11 and 30 days (see Fig. 6.22a). Many of these events exhibited multiple peaks and fluctuating discharge trends. To address the scarcity of

low-frequency, high-intensity floods in the historical record, the second dataset, *Po2*, included 6 additional synthetic hydrographs with high peak discharges (Fig. 6.22b). This dataset aimed to assess whether a surrogate model trained on a more diverse and balanced set of flood events would achieve higher accuracy than one trained solely on historical data.

For the second objective, a third training configuration, *Po3*, was introduced. In this configuration, the surrogate model was trained and tested using inundation maps with a spatial resolution of 10 m. The *Po3* configuration considered the same flood events in the training dataset as the *Po2* configuration, enabling a direct comparison of the impact of spatial resolution on model performance.

To assess the model's accuracy in predicting unseen flood events, three real floods that occurred in November 2011, November 2014, and June 2020 (shown in Fig. 6.22c) were simulated. These hydrographs differed in peak discharges, ranging from 2,500 m^3/s to 9,700 m^3/s, and in flood duration, which varied between 13 and 22 days. The extent of inundation varied significantly across the events: the June 2020 flood, characterized by the lowest peak discharge, remained confined to the main channel, whereas the November 2011 flood affected the open floodplains of the Po River due to the higher peak discharge. The most severe event of the testing dataset, the November 2014 flood, inundated the majority of the defended floodplains. These diverse testing events provided a comprehensive evaluation of the surrogate model's performance under varying flood intensities, from minor to severe. As in previous case studies, the testing dataset consisted only of unseen flood events and remained consistent across all three training configurations (see Table 6.2).

To facilitate a detailed comparison between the numerical and surrogate models, water depths were extracted at six predefined control points located along the main channel and within the defended floodplains (labeled G1-G6 in Fig. 6.21).

Calibration of the PARFLOOD code

In the selected stretch of the Po River, a gauging station, named Boretto, is located approximately 15 km downstream of the Casalmaggiore river section (refer to point G1 in Fig. 6.21). The water depths recorded at this station were utilized for calibrating the numerical model. Specifically, the three real flood events in the testing dataset (i.e., November 2011, November 2014, and June 2020) were employed to calibrate the roughness coefficient. Distinct Manning's coefficients were used for the main channel and the floodplains, set at $0.03\,\mathrm{sm}^{-1/3}$ and $0.045\,\mathrm{sm}^{-1/3}$, respectively. These values were chosen to minimize discrepancies between the simulated and observed water levels at the Boretto gauging station, represented respectively by the dashed black line and the magenta circles at the control point G1 in Fig. 6.23. The PARFLOOD code accurately reproduces the temporal variation of water depths at the target station of the Po River, thereby validating the reliability of the numerically generated ground-truth maps used to train the surrogate model.

6.4.2 Results and comparison between different training configurations

To assess the influence of different spatial resolutions and the variety of flood scenarios within the training dataset, the FS-bc model was trained using the three configurations detailed in Table 6.2. For each training iteration, the surrogate model's predictive accuracy was evaluated by testing its performance on unseen flood events.

Considering the FS test procedure, Table 6.7 summarizes the average performance metrics for each training configuration. Additionally, the corresponding temporal variations of RMSE and RMSE_ND are depicted in Figs. 6.24–6.26. Overall, the *Po1* and *Po2* configurations demonstrated comparable performance in predicting one frame ahead (i.e., the FS test). Differently, the *Po3* configuration resulted in higher RMSE values, with increases ranging between 3 cm to 8 cm when compared to the other setups. This decline in accuracy can be attributed primarily to the increased

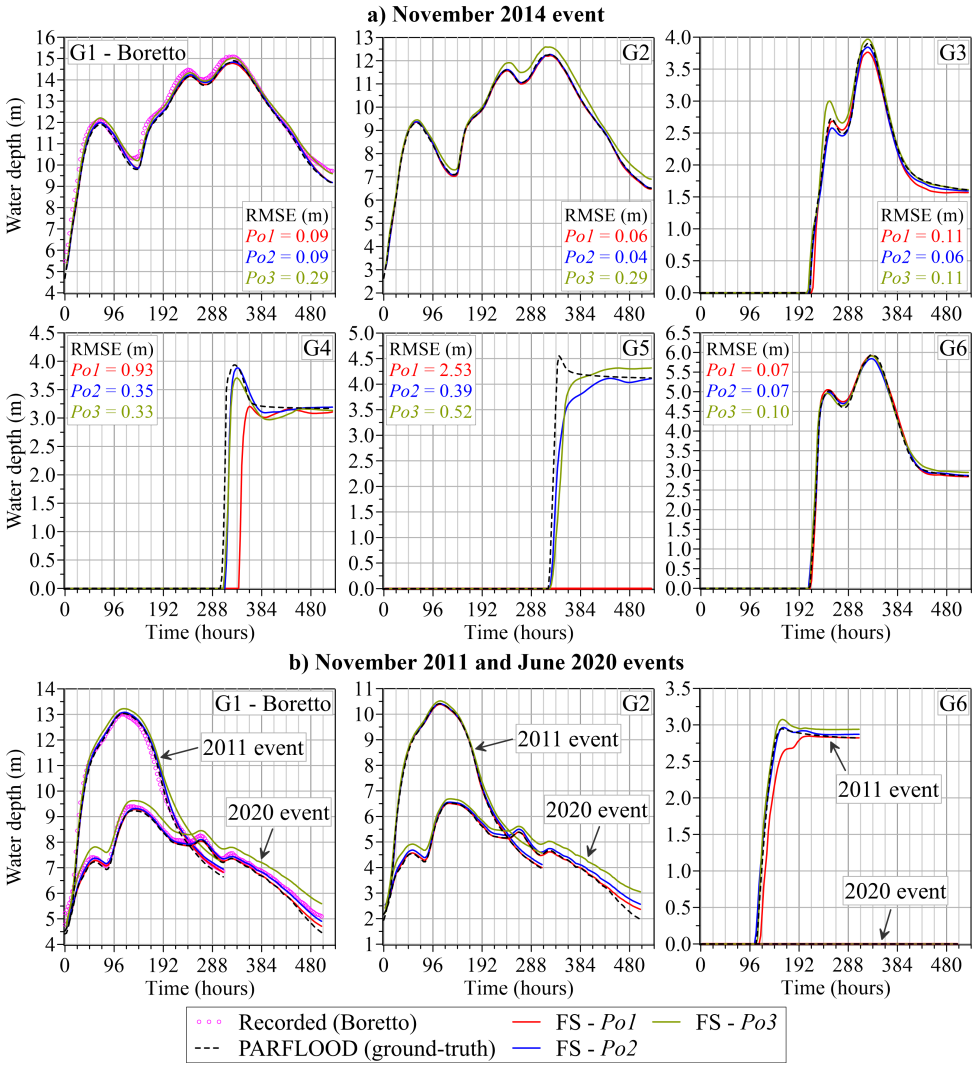

FIGURE 6.23 – Comparison of target and forecasted water depths at control points for the Po River case study. The magenta circles indicate recorded water depths at the Boretto gauging station (point G1). (a) November 2014 flood event. For each control point, the average RMSE of the time series of ground-truth and predicted water depths using the surrogate model trained with the three configurations is reported. (b) November 2011 and June 2020 flood events.

number of parameters in the surrogate model under the *Po3* configuration, as the maps contain four times more cells, while the size of the training dataset remains unchanged. A more detailed discussion of this issue is provided later.

The promising results obtained from the FS test provide a solid foundation for applying the AR procedure to forecast entire flood events. In this process, the number of past frames (P) was set to 1, while the number of future frames (F) varied according to the duration of the flood event being simulated. Specifically, the

TABLE 6.7 – Po River case study: average RMSE, RMSE_ND, and F1 score for the FS test. The average water depth ($\bar{h}$) computed considering all wet cells for each event is also provided.

Testing event	$\bar{h}$ [m]	RMSE [m]			RMSE_ND [-]			F1 [-]		
		Po1	*Po2*	*Po3*	*Po1*	*Po2*	*Po3*	*Po1*	*Po2*	*Po3*
Nov 2011	3.45	0.07	0.07	0.11	0.022	0.021	0.032	0.983	0.983	0.981
Nov 2014	4.06	0.08	0.07	0.11	0.019	0.018	0.027	0.990	0.991	0.989
Jun 2020	4.16	0.06	0.06	0.13	0.015	0.016	0.034	0.990	0.990	0.988

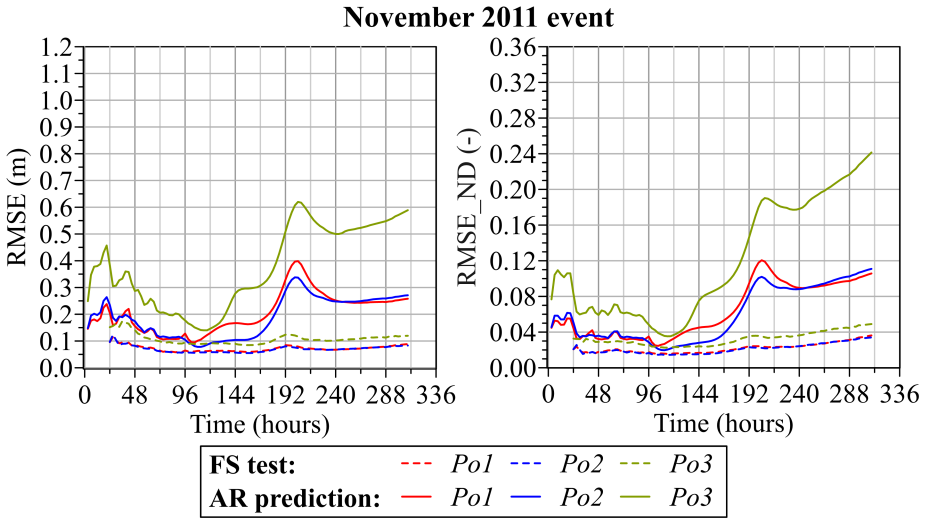

FIGURE 6.24 – Po River case study: RMSE and RMSE_ND for the November 2011 flood event, computed for the FS test (dashed lines) and AR prediction (continuous lines) using the different training configurations.

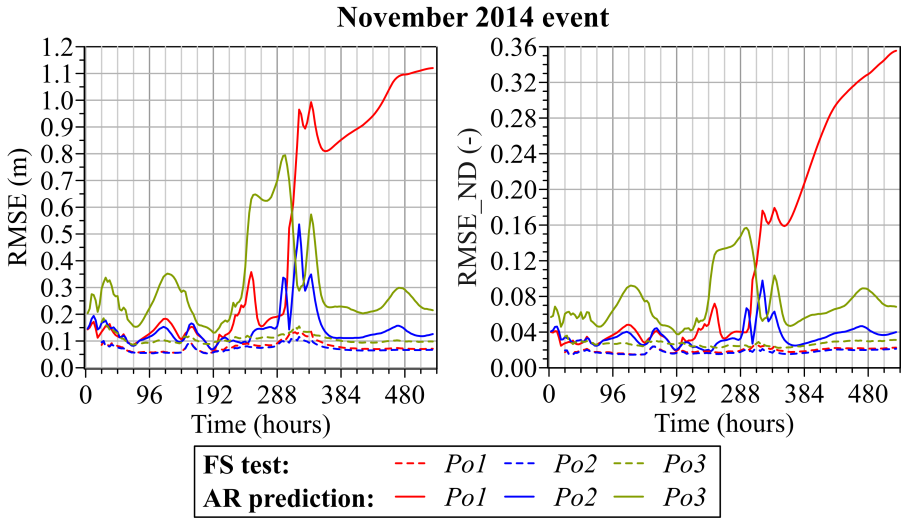

FIGURE 6.25 – Po River case study: RMSE and RMSE_ND for the November 2014 flood event, computed for the FS test (dashed lines) and AR prediction (continuous lines) using the different training configurations.

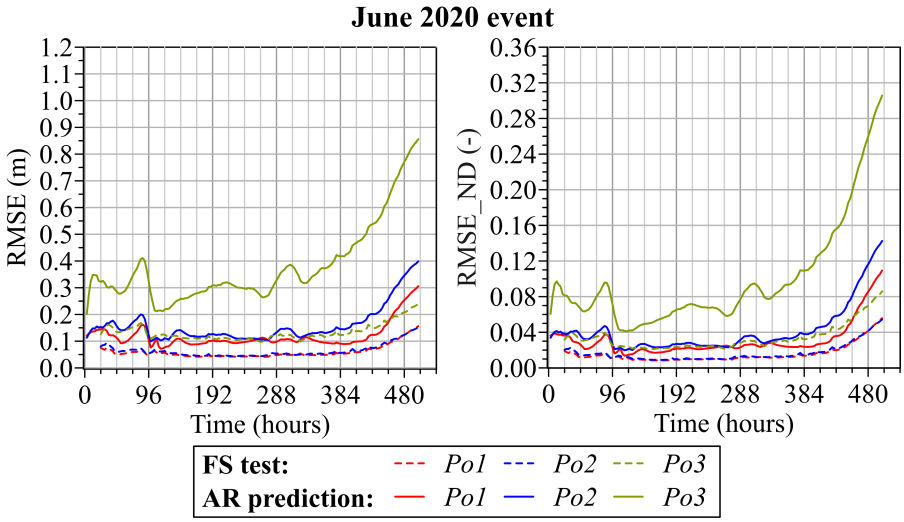

FIGURE 6.26 – Po River case study: RMSE and RMSE_ND for the June 2020 flood event, computed for the FS test (dashed lines) and AR prediction (continuous lines) using the different training configurations.

values of F were set to 103, 174, and 167 for the 2011, 2014, and 2020 flood events, respectively, corresponding to lead times between 13 and 22 days.

Table 6.8 summarized the average performance metrics for the recursively forecasted maps across different training configurations and testing events, while the trend of the RMSE and RMSE_ND are presented in Figs. 6.24–6.26. The results clearly show that the accuracy of the AR prediction is highly dependent on the choice of training configuration.

A comprehensive discussion on the influence of various training configurations on the performance of the AR procedure is provided in the following subsections.

Po1 training configuration

In the first training scenario, named *Po1*, a spatial resolution of 20 m was employed, with a training dataset consisting of 19 recorded flood events (Fig. 6.22a). Focusing on the AR prediction of the test events, the average RMSE is particularly low for the 2011 and 2020 floods, ranging from approximately 0.15 m to 0.2 m, as reported in Table 6.8. Differently, for the 2014 flood event, the average RMSE increases to approximately 0.5 m. Fig. 6.25 shows a steep rising of the RMSE to 1 m after the flood event's peak.

To investigate the reasons behind these discrepancies, the time series of water depths at control points, derived from both ground-truth (dashed black lines) and predicted (red lines) maps for all testing events, are compared in Fig. 6.23. Addi-

Table 6.8 – Po River case study: average RMSE, RMSE_ND, and F1 score for the AR prediction of the testing flood events across different training configurations.

Testing event	RMSE [m]			RMSE_ND [-]			F1 [-]		
	Po1	*Po2*	*Po3*	*Po1*	*Po2*	*Po3*	*Po1*	*Po2*	*Po3*
Nov 2011	0.21	0.19	0.38	0.064	0.060	0.112	0.972	0.973	0.963
Nov 2014	0.48	0.15	0.31	0.127	0.036	0.073	0.974	0.988	0.979
Jun 2020	0.12	0.16	0.38	0.031	0.041	0.099	0.985	0.982	0.955

tionally, Fig. 6.27 presents a comparison of water depth maps for selected instants during the November 2014 flood event.

Overall, the surrogate model exhibits high accuracy in predicting water depths within the main channel. For the most extreme flood scenario in the testing dataset, the November 2014 event, the average RMSE at control points G1 and G2, both located within the main channel, is less than 0.1 m (Fig. 6.23a). Additionally, the model effectively captures the temporal variations in water depths and the flood arrival time at control points G3 and G6, located in two protected floodplains, with average RMSEs below 0.11 m.

However, the surrogate model encounters limitations in predicting water depths in other protected floodplains, as can also be seen in Fig. 6.27. For instance, the model predicts a delay of approximately 30 hours in the flood's arrival at control point G4 (Fig. 6.23a), and the maximum water depth is underestimated by about 0.7 m. More critically, the model entirely fails to predict the inundation of the protected floodplain where control point G5 is situated. During the November 2014 event, the surrogate model incorrectly predicts that this area remains dry, while the numerical model indicates water depths exceeding 4 m.

The failure of the surrogate model to accurately simulate flood dynamics in certain protected floodplains can be attributed to the nature of the flood events included in the *Po1* training dataset. Approximately 80% of the recorded hydrographs have peak discharges below 7,000 m^3/s, with only a few events characterized by low-frequency and high-intensity floods. Moreover, many protected floodplains are only inundated during floods with peak discharges exceeding 8,000-10,000 m^3/s. As a result, the model struggles to learn flood dynamics in these regions due to the limited number of relevant examples in the training dataset. Specifically, in the case of the 2014 flood event (Fig. 6.23a and Fig. 6.27), the surrogate model accurately predicts water depths at control points G3 and G6, which are located in protected floodplains that are flooded during relatively low-intensity flood events (e.g., control point G6 is also flooded during the November 2011 event, as shown in Fig. 6.23b). In contrast, control

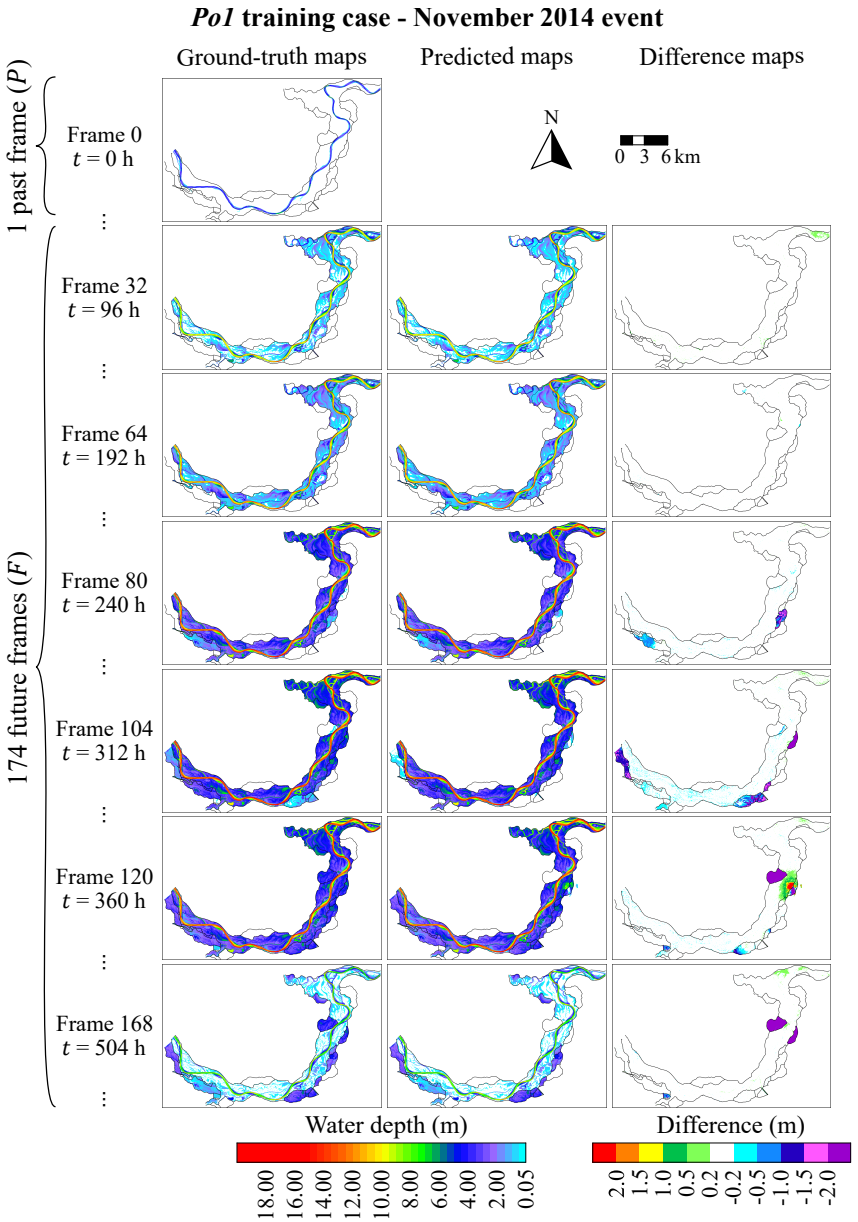

FIGURE 6.27 – Po River case study: real-time forecasting of the November 2014 flood event using the FS-bc model with the *Po1* training configuration. The columns represent, respectively, the ground-truth maps obtained from the hydrodynamic model, the maps predicted by the surrogate model, and the difference maps between the predicted and ground-truth maps. Only selected representative instants are shown.

points G4 and G5, which exhibit larger prediction errors, are situated in regions protected by higher levees and therefore experience flooding less frequently.

Po2 training configuration

To address the limitation caused by the underrepresentation of high-intensity flood events in the *Po1* training dataset, the *Po2* configuration was developed by incorporating synthetically generated flood scenarios (Fig. 6.22b). These synthetic events include peak discharges exceeding 8,000 m^3/s, resulting in the inundation of most defended floodplains. The water depths predicted by the newly trained model are depicted as blue lines in Fig. 6.23. Focusing on the 2014 flood event (Fig. 6.23a), the inclusion of a more balanced training dataset substantially improves the model's accuracy in predicting flood dynamics within defended floodplains.

The *Po2* training configuration significantly enhances the accuracy of predictions concerning the temporal variation of water depths at control points G4 and G5 located in defended floodplains. Previous issues, such as delays in predicting the flood arrival time and failure to forecast flooding at control point G5, are effectively resolved. However, the model still underestimates water depths during the initial hours following the flood's arrival at this control point. In contrast, the model's accuracy remains consistently high at the other control points, with results comparable to the ones obtained with the *Po1* configuration.

In the less severe flood scenarios within the testing dataset (i.e., the 2011 and 2020 events shown in Fig. 6.23b), the surrogate model trained with the *Po2* configuration demonstrates high accuracy in forecasting water depths, both within the main channel and the floodplains. However, it slightly overestimates water depths in the main channel during the recession limb of the floods. This is further corroborated by the increase in RMSE shown in Figs. 6.24 and 6.26. The discrepancies during the final phase of the AR predictions stem from the higher errors generated by the FS test for these concluding maps (as seen, for instance, in the dashed blue line of Fig. 6.26). Consequently, greater errors are expected when these maps are predicted by the

AR procedure. Additionally, using a more balanced dataset enhances the model's prediction accuracy for high-severity floods but may slightly reduce its performance in forecasting low-intensity events. Nevertheless, these slight increase of the error in the recession limb of low-severity floods are considered negligible for practical purposes.

Table 6.8 summarizes the average performance metrics calculated for the recursively forecasted maps across the entire duration of each flood event. The surrogate model's accuracy in predicting the temporal evolution of the flood extent is confirmed by the extremely high F1 values.

For the 2014 flood event, the *Po2* configuration reduces the average RMSE by approximately 70% compared to the *Po1* case, from 0.48 m to 0.15 m. In contrast, for the 2020 event, the average RMSE increases from 0.12 m to 0.16 m, representing a 30% increase compared to the *Po1* configuration. This rise in error is attributable to the overestimation of water depths during the flood's recession limb, as previously discussed. However, errors of this magnitude remain acceptable for practical applications. These findings are supported by Figs. 6.24–6.26, which compare the temporal variation of the RMSE and RMSE_ND for the different training configurations.

As shown in Fig. 6.25, the use of the *Po2* configuration significantly reduces errors during and after the peak of the November 2014 flood event (around $t = 312$ h). However, an oscillatory trend in the RMSE, with values reaching up to 0.5 m, is notable near the flood peak, gradually decreasing to values below 0.15 m in subsequent instants. To better understand these discrepancies, an analysis of the spatial distribution of the errors is essential. Fig. 6.28 compares the RMSEs for the November 2014 flood event across different areas of the river region: (a) the entire region (blue lines), (b) the main channel and open floodplains (magenta lines), and (c) the defended floodplains (green lines). As previously noted, the RMSE over the entire flooded area remains below 0.2 m during the early and late stages of the event, with higher errors occurring as defended floodplains begin to flood, coinciding with the flood peak. In contrast, the RMSE for the main channel remains consistently

below 0.2 m throughout the entire event, representing an extremely high level of accuracy. The RMSE for the defended floodplains, on the other hand, exhibits peaks at the onset of flooding in various protected floodplains, followed by a reduction to values below 0.2 m. These peaks are attributed to the temporal delay in the filling process (as seen, for example, at control point G5 in Fig. 6.23a) and the relatively small number of wet cells in the floodplains used to compute the RMSE. Thus, the higher RMSE values across the entire flooded area can be primarily attributed to the discrepancies observed during the initial flooding of the defended floodplains.

Figs. 6.29–6.31 provide a comparison between target maps and predicted maps generated by the FS-bc model trained with the *Po2* configuration for the three flood events in the testing dataset. These figures confirm the surrogate model's high accuracy in predicting flood dynamics in the Po River region. Specifically, the differences between the predicted and target maps remain below 0.2 m across most time steps of all three flood events. This error, approximately 5% of the average water depth across the entire flood scenario (Table 6.7), is acceptable for practical applications.

A comparison between Figs. 6.27 and 6.30 highlights the superior performance of the surrogate model trained with the *Po2* configuration compared to the *Po1* configuration. In particular, errors in the defended floodplains are significantly reduced with the *Po2* training approach.

Figs. 6.32a, 6.33a, and 6.34a provide scatter plots comparing the predicted and target water depths for each grid cell across all forecasted time frames for the three test events. These scatter plots categorize water depths into 0.1 m intervals. The color of each point represents the frequency of occurrence of each pair, computed as the ratio of the number of grid cells with a given pair of predicted and target values to the total number of wet cells for each event, which ranges from approximately 7.6M to 32.9M. TN values (i.e., dry cells in both the target and predicted maps), which range from 120M to 210M, are excluded from the color scale for clarity. Generally, most points closely align with the dashed black line, indicating accurate predictions.

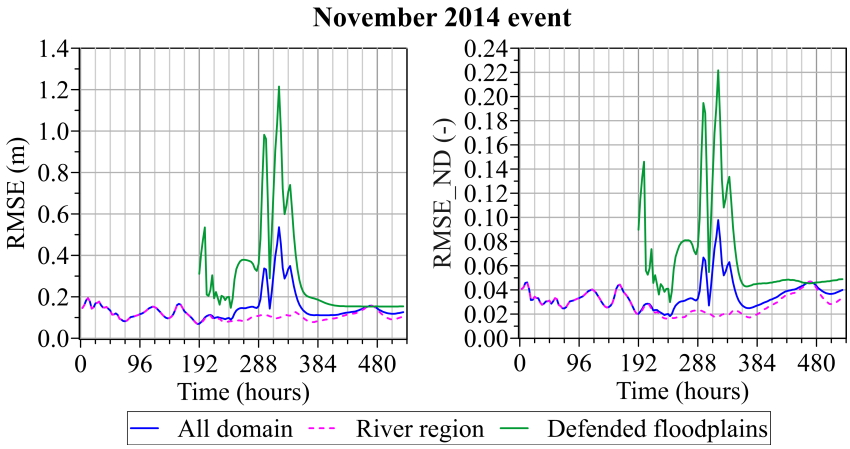

FIGURE 6.28 – November 2014 flood event: RMSE and RMSE_ND computed separately for the main river region (including the main channel and open floodplains) and the defended floodplains. These results were obtained with the *Po2* training configuration.

However, in the November 2014 event (Fig. 6.33a), the model tends to slightly underestimate water depths, particularly in cells with depth below 4-5 m or exceeding 24 m. Conversely, in the less severe 2011 and 2020 floods (Figs. 6.32a and 6.34a), the model exhibits a slight tendency to overestimate water depths by a few centimeters.

The corresponding histograms in Figs. 6.32b, 6.33b, and 6.34b illustrate the distribution of differences between target and predicted water depths. For the November 2014 event (Fig. 6.33b), where about 33M cells for all the predicted maps exhibit water depths greater than 0.05 m (the threshold used to exclude negligible depths, as explained in Section 6.1), more than 70% of these flooded cells exhibit errors below 0.1 m. This percentage rises to approximately 94% for errors between 0 m and 0.2 m. Additionally, less than 0.5% of wet cells display errors exceeding 1 m. These discrepancies, primarily located in the defended floodplains (as shown in Fig. 6.30), are mainly associated with a slight temporal shift in flood arrival time, as previously discussed. For the other two events, the percentage of cells with errors lower than 0.2 m decreases to about 85%, still represents a very high level of accuracy.

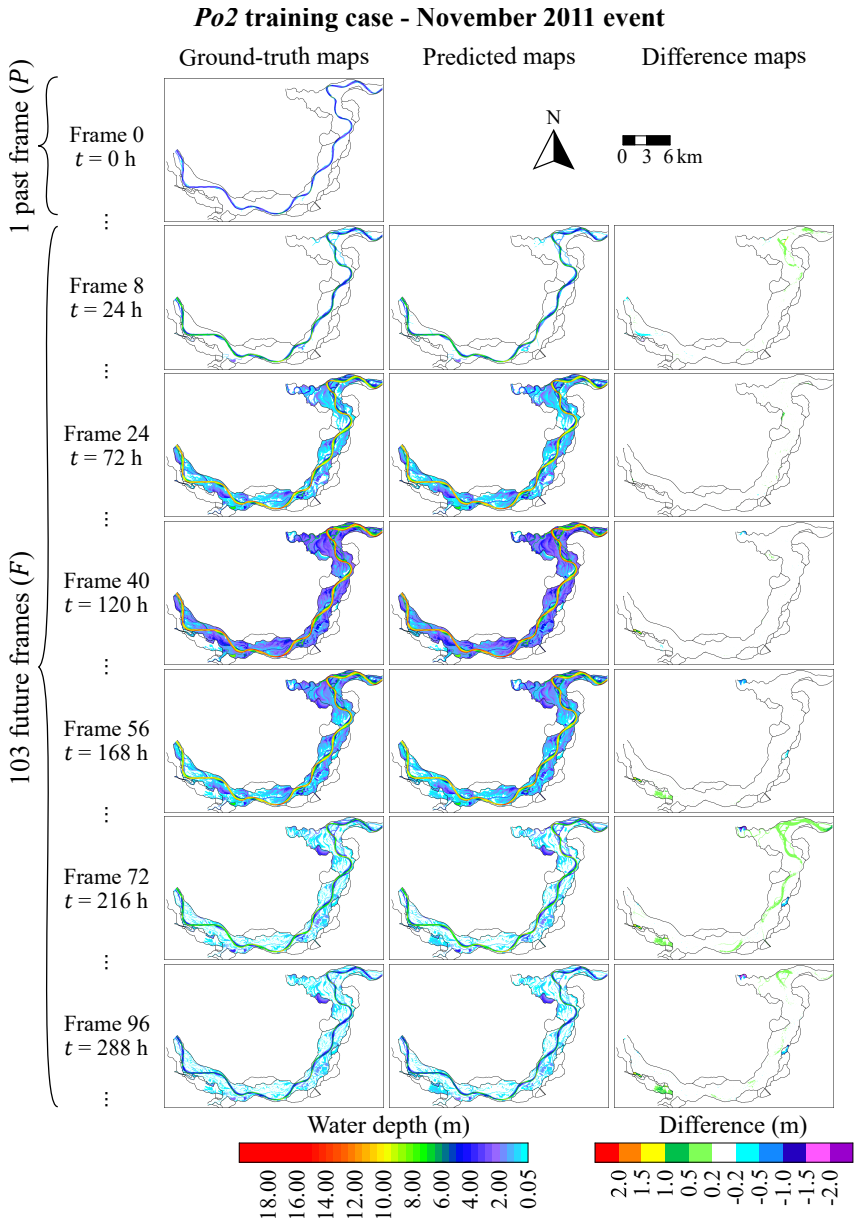

FIGURE 6.29 – Po River case study: real-time forecasting of the November 2011 flood event using the FS-bc model with the *Po2* training configuration. The columns represent, respectively, the ground-truth maps obtained from the hydrodynamic model, the maps predicted by the surrogate model, and the difference maps between the predicted and ground-truth maps. Only selected representative instants are shown.

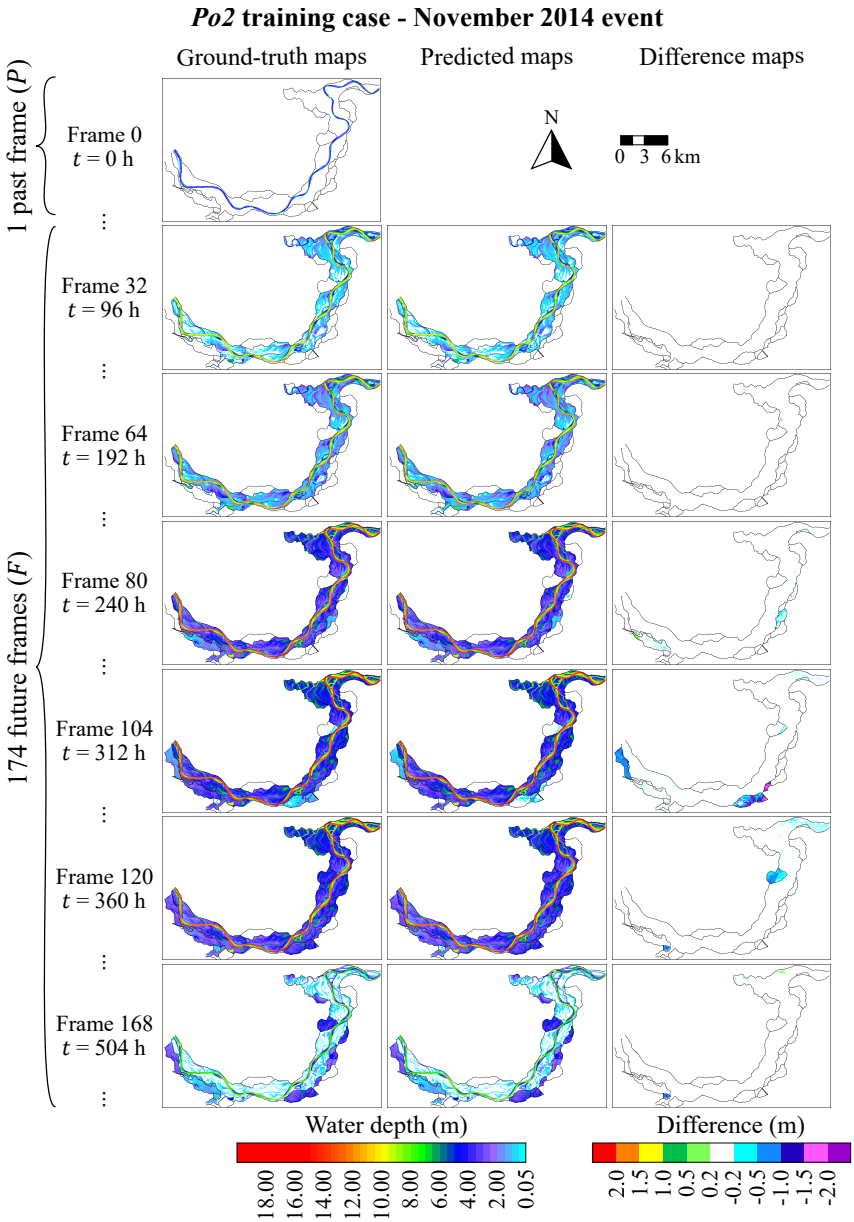

FIGURE 6.30 – Po River case study: real-time forecasting of the November 2014 flood event using the FS-bc model with the *Po2* training configuration. The columns represent, respectively, the ground-truth maps obtained from the hydrodynamic model, the maps predicted by the surrogate model, and the difference maps between the predicted and ground-truth maps. Only selected representative instants are shown.

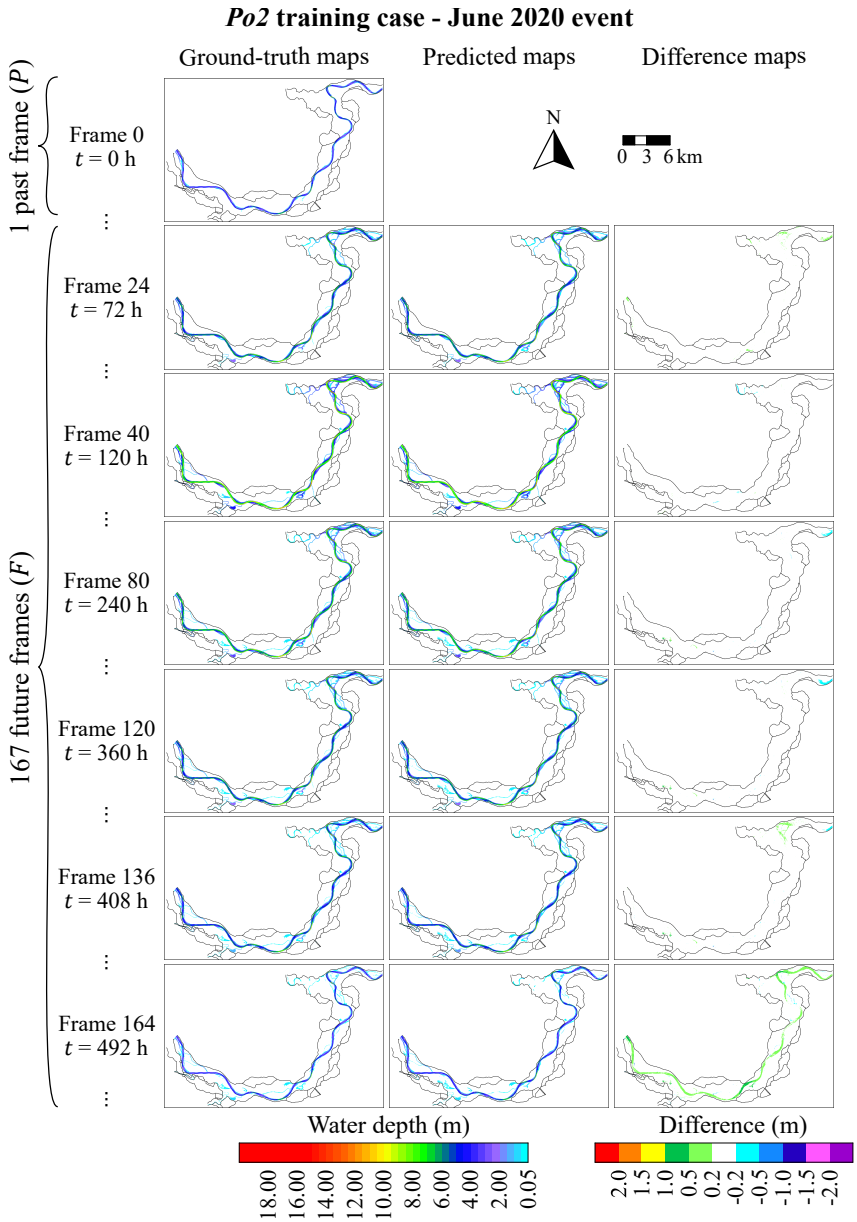

FIGURE 6.31 – Po River case study: real-time forecasting of the June 2020 flood event using the FS-bc model with the *Po2* training configuration. The columns represent, respectively, the ground-truth maps obtained from the hydrodynamic model, the maps predicted by the surrogate model, and the difference maps between the predicted and ground-truth maps. Only selected representative instants are shown.

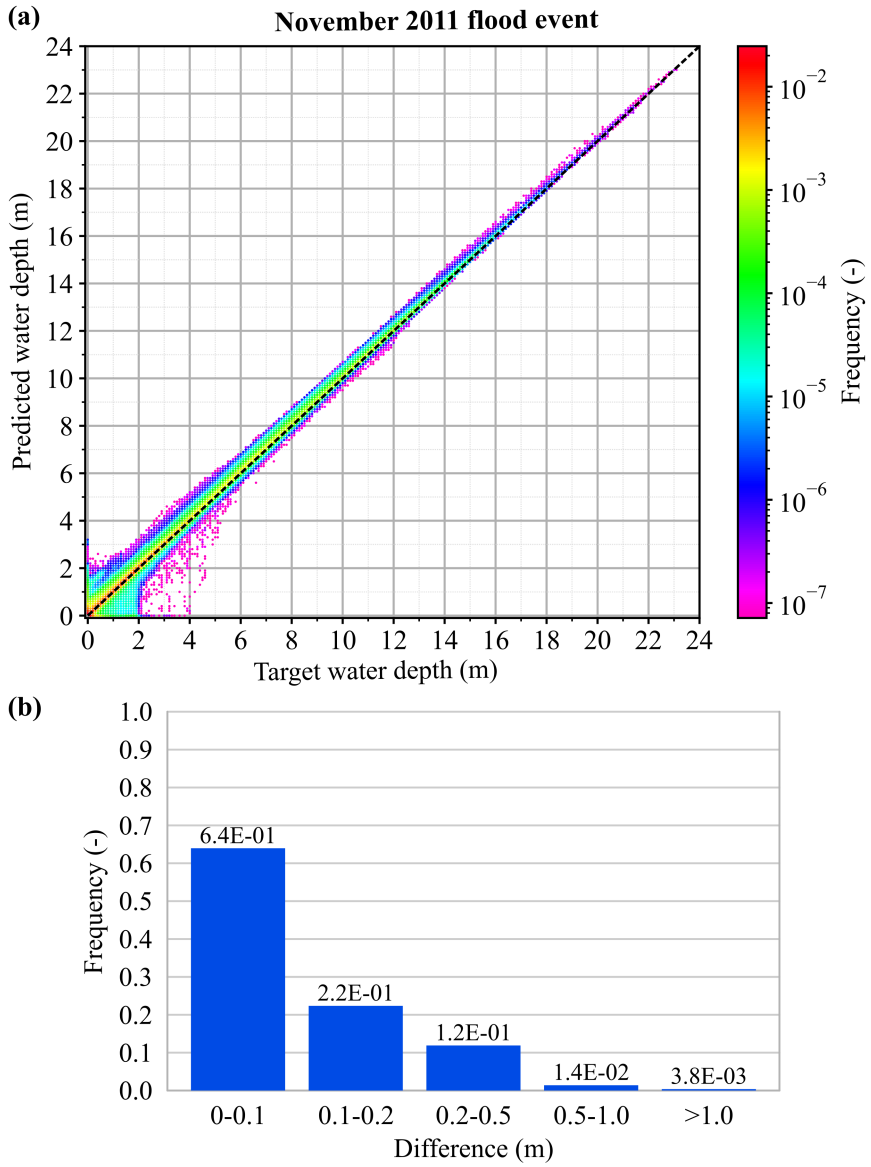

FIGURE 6.32 – Po River case study: comparison of predicted and target water depths for the November 2011 flood event, using the FS-bc model trained with the *Po2* configuration. (a) Scatter plot comparing predicted and target water depths across all grid cells and time steps of the November 2011 flood event. The color scale indicates the frequency, computed using the total number of wet cells (approximately 14M) as the normalization factor. The dashed black line represents the correct forecast. (b) Histogram showing the distribution of the differences between target and predicted water depths.

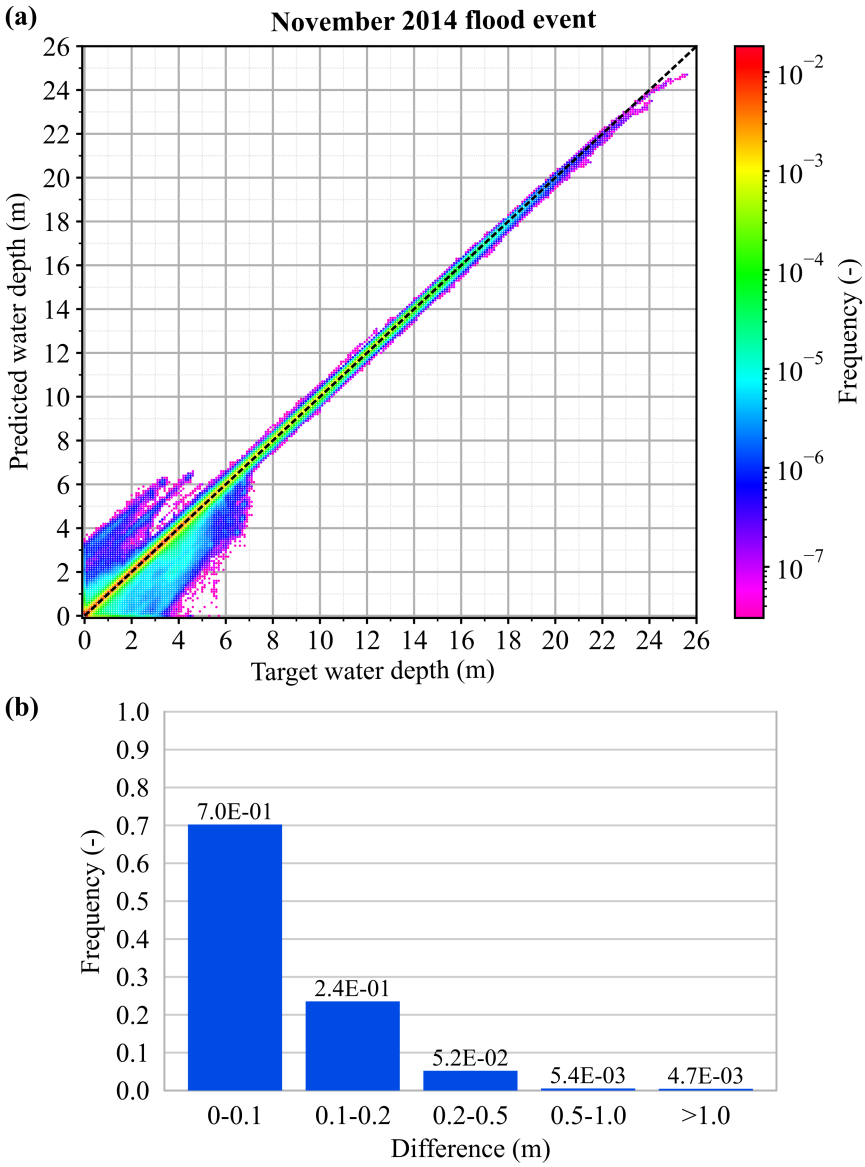

FIGURE 6.33 – Po River case study: comparison of predicted and target water depths for the November 2014 flood event, using the FS-bc model trained with the *Po2* configuration. (a) Scatter plot comparing predicted and target water depths across all grid cells and time steps of the November 2014 flood event. The color scale indicates the frequency, computed using the total number of wet cells (approximately 32.9M) as the normalization factor. The dashed black line represents the correct forecast. (b) Histogram showing the distribution of the differences between target and predicted water depths.

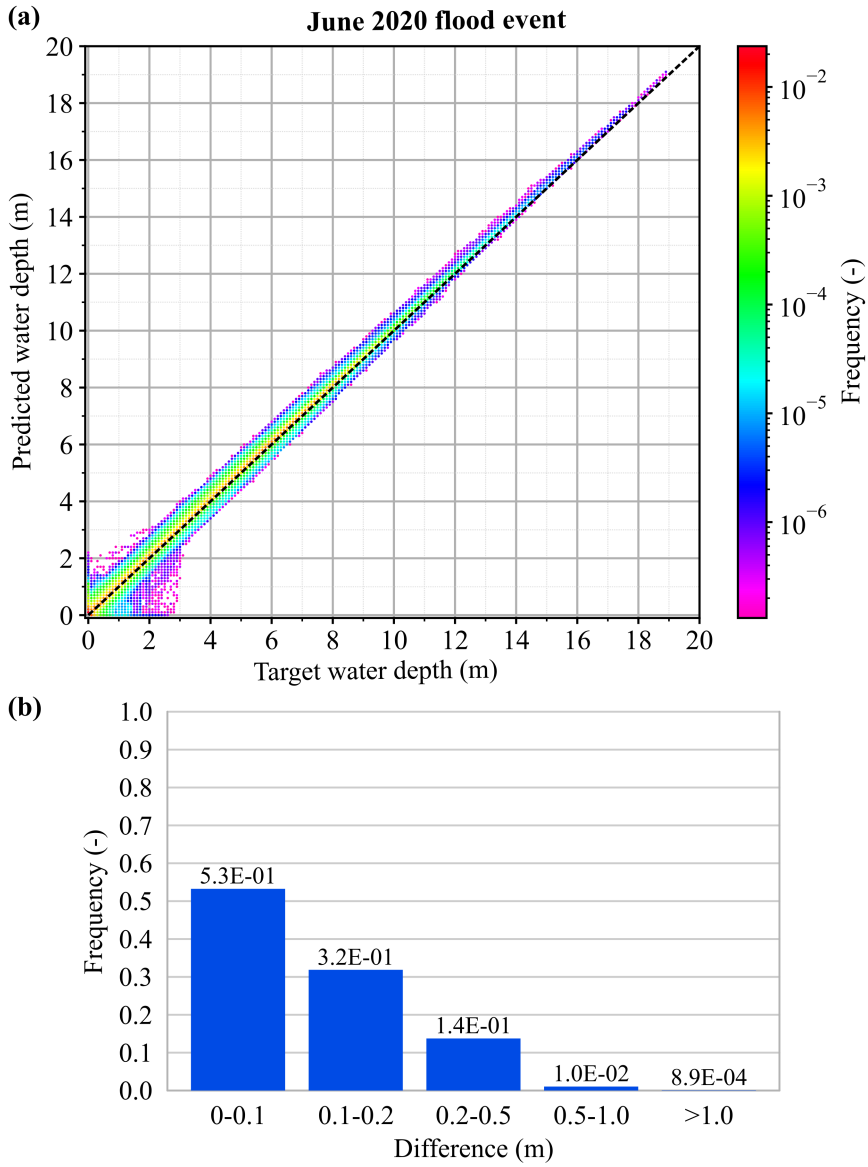

FIGURE 6.34 – Po River case study: comparison of predicted and target water depths for the June 2020 flood event, using the FS-bc model trained with the *Po2* configuration. (a) scatter plot comparing predicted and target water depths across all grid cells and time steps of the June 2020 flood event. The color scale indicates the frequency, computed using the total number of wet cells (approximately 7.6M) as the normalization factor. The dashed black line represents the correct forecast. (b) Histogram showing the distribution of the differences between target and predicted water depths.

Po3 training configuration

In the final training configuration, named *Po3*, maps with a spatial resolution of 10 m were employed. The training dataset remained identical to that used in the *Po2* configuration, incorporating both real and synthetic flood events (see Table 6.2). The objective of utilizing a higher resolution, resulting in maps with more than 5.2M of cells, was to assess how the FS-bc model scales with increased model size, in terms of both predictive accuracy and computational time required for training and forecasting. Here, only the model accuracy is discussed, while details regarding runtimes are provided in Section 7.1.

When comparing the results of the *Po3* configuration with those of *Po2*, Table 6.8 shows an increase in the average RMSE by 0.15-0.2 m, accompanied by a decrease in the F1 score due to the larger cell count. These higher RMSE values are also reflected in the graphs presented in Figs. 6.24–6.26.

The green lines in Fig. 6.23, which represent the forecasted water depths at control points for the *Po3* configuration, indicate that the increase in spatial resolution introduces slightly greater discrepancies at certain locations. For example, during the November 2014 flood event, water depths at control point G2 are overestimated by approximately 0.3 m at the flood peak. Additionally, the accuracy of predictions at some control points located in defended floodplains, such as G4 and G5, is lower than in the coarser resolution. These trends are also evident in other flood scenarios within the testing dataset (Fig. 6.23b). For example, in the June 2020 event, a more pronounced overestimation of water depths in the main channel is evident. This is further confirmed by Fig. 6.26, which displays a gradual increase in the RMSE, highlighting the accumulation of errors due to the recursive nature of the AR procedure.

Fig. 6.35 compares the ground-truth and forecasted maps for the November 2014 flood event using the *Po3*-trained surrogate model. Similar to the results obtained with the *Po2* configuration (Fig. 6.30), good accuracy is maintained in the main channel and open floodplains. However, notable discrepancies are observed in the

defended floodplains. Specifically, the surrogate model tends to anticipate flood arrival in some defended floodplains, leading to notable differences at certain instants due to sudden increases in water depths.

The reduced accuracy of the AR process may be attributed to the increase in model size, which involves a significantly larger number of parameters to optimize during training compared to the *Po2* configuration (Table 6.2). However, since the size of the training dataset remained the same, this may have hindered the optimization of the model trained with the finer resolution maps. This issue could potentially be mitigated by employing a larger training dataset. Nevertheless, the primary goal of this analysis was to compare results across different spatial resolutions using the same dataset, with further investigations to be reserved for future work.

Despite the observed reduction in accuracy, the predictions generated by the FS-bc model trained with the *Po3* configuration maintain sufficient fidelity for real-time flood prediction applications.

6.4.3 Sensitivity analysis to the initial condition

As discussed in Section 5.4.3, the FS-bc model can predict flood scenarios by using a single initial condition map, representing a past frame, to initiate the AR process. This initial map is generated from a steady state simulation conducted using a physically based model. However, in the context of real-time flood forecasting, performing a steady flow numerical simulation at the onset of the forecast can introduce significant computational delays. To address this issue, this Section examines the impact of the initial condition on the flood predictions generated by the FS-bc model.

For this sensitivity analysis, the November 2014 flood event was analyzed using the FS-bc model trained with the *Po2* configuration (refer to Table 6.2). The "true" initial condition was derived from a steady flow simulation with a discharge rate of 500 m^3/s, which closely matches the initial values of the November 2014 hydrograph. In addition, two alternative steady flow conditions were simulated with discharge

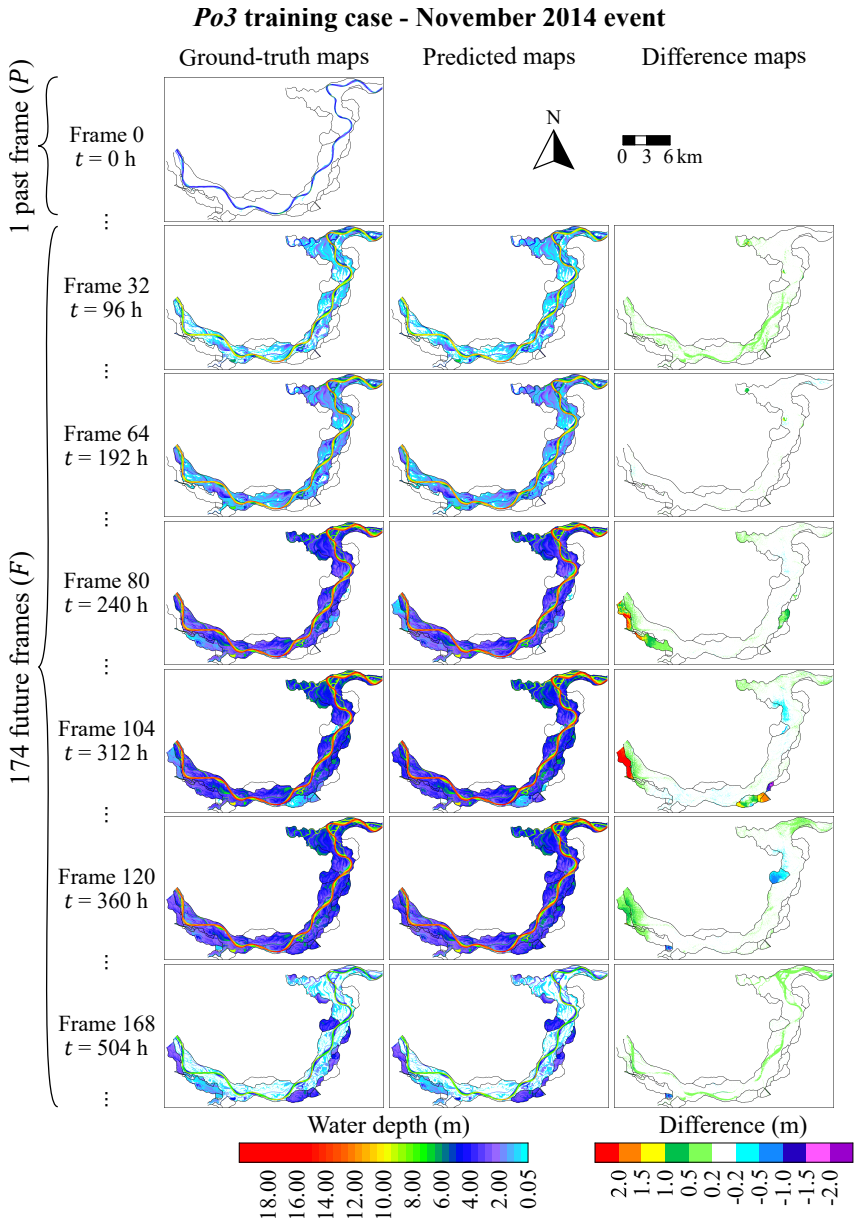

FIGURE 6.35 – Po River case study: real-time forecasting of the November 2014 flood event using the FS-bc model with the *Po3* training configuration. The columns represent, respectively, the ground-truth maps obtained from the hydrodynamic model, the maps predicted by the surrogate model, and the difference maps between the predicted and ground-truth maps. Only selected representative instants are shown.

values of 1,500 m^3/s and 2,500 m^3/s. Water depth maps from these three steady state conditions were used as the past frame for initiating the AR forecast.

Fig. 6.36a compares the water depths extracted at the Boretto gauging station (point G1) for the different configurations. Notably, the FS-bc model quickly becomes insensitive to the initial condition after the first few time steps. Similar behavior is observed in other areas of Po River region (not shown). Fig. 6.36b illustrates the temporal variation of the RMSE across the different configurations. Initially, RMSE values vary based on the initial condition, but they rapidly converge to similar values after just a few time steps. The closer the assumed initial condition is to the "true" initial condition, the faster this convergence occurs.

These findings suggest that the FS-bc model's predictions are largely independent of the initial condition after a brief warm-up period. Therefore, for operational purposes, a small database of water depth maps corresponding to various steady flow conditions could be precomputed. These maps could then serve as past frames

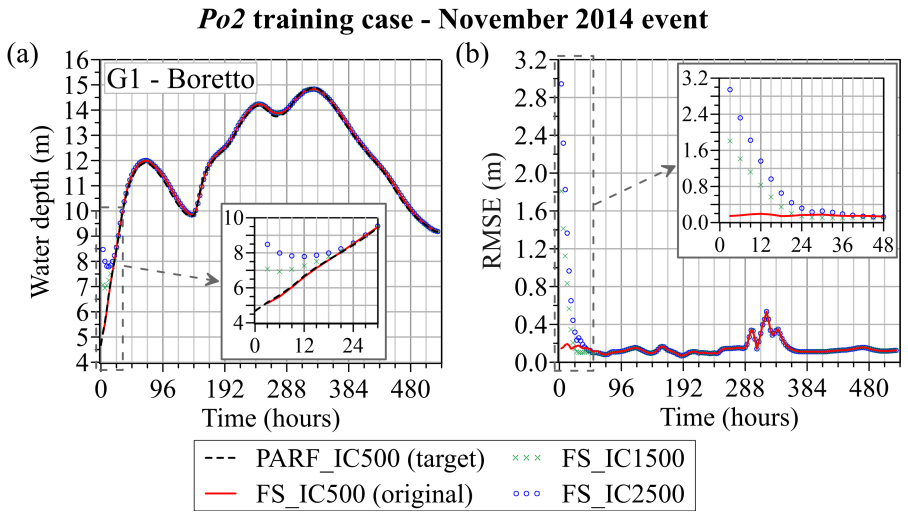

FIGURE 6.36 – Comparison of results for different initial conditions. The surrogate model forecasts consider different past frame maps. (a) Comparison of water depths extracted at the Boretto gauging station. (b) RMSE computed on the forecasted maps with the three initial conditions analyzed.

for the AR procedure of the FS-bc model. The most appropriate map could be selected based on the discharge value closest to the real-time event initial condition. This approach avoids the need for a time-consuming numerical simulation with the physically based model to generate the initial condition. The number and range of discharge values to be included in this database depend on the specific characteristics of the river under analysis. For the Po case study, for instance, discharge values between 500 m^3/s and 3,000 m^3/s could be considered for creating the database.

6.5 Benchmark comparison

This Section presents a benchmark comparison between the FS-bc model and the 1D CNN model proposed by Kabir et al. (2020). Details regarding the architecture of the 1D CNN model can be found in Section 3.3.1. To evaluate the robustness of the proposed architecture, the prediction of river floods in both the Toce River (case 5) and the Po River (case 6) is analyzed. Specifically, the surrogate models were trained using the *Toce2* and *Po2* configurations (refer to Table 6.2) for the Toce River and Po River case studies, respectively.

Following the original implementation of the 1D CNN model (Kabir et al., 2020), a temporal window size of 8 time steps ($r = 8$, as defined in Section 3.3.1) was employed during the training process. Consequently, the convolutional model receives a sequence of 9 inflow discharges from $t - 8$ to t and generates an inundation map for time step t. This setup is similar to the one of the FS-bc model, which uses a temporal window of 8 time steps (i.e., $I = 8$). The 1D CNN model was trained with a batch size of 10, employing the MSE loss function and the Adam optimizer, with learning rates set to 5×10^{-4} and 10^{-3} for the Toce River and Po River case studies, respectively. The learning rates and batch size values were determined through an iterative trial-and-error process.

Regarding the number of parameters, the 1D CNN model contains approximately 8.5M and 670M parameters for the Toce River and Po River case studies, respectively.

These values are of the same order of magnitude as those of the FS-bc model, as detailed in Table 6.1.

Table 6.9 summarizes the performance metrics of both surrogate models for the two case studies. In all test events, the FS-bc model outperformed the 1D CNN model in terms of RMSEs and F1 scores.

Toce River case study

Focusing on the Toce River case study, Fig. 6.37 compares the inundation maps of the *Medium* hydrograph as predicted by the two DL models. The FS-bc model generally produces lower errors across the forecasted maps compared to the convolutional model, which exhibits localized high errors in the initial instants and more spatially distributed errors after the flood peak.

The water depths extracted at the 9 control points (see Fig. 6.7 for locations) are compared in Fig. 6.38. The models show comparable performance at points upstream of the urban district (i.e., P2-P4). However, at the points where the flow is strongly influenced by the presence of the blocks (i.e., P6-P10), the 1D CNN model presents lower accuracy, with notable overestimates in water depth predictions. Furthermore, the temporal trend of the water depths predicted by the convolutional model is less smooth, especially at point P10, where oscillatory spikes are observed. These errors are likely due to the oscillatory behavior of the recession limb in the *Medium* hydrograph (see Fig. 6.8c), as the 1D CNN model relies solely on upstream BCs, and consequently it is strongly influenced by its trend. In contrast, the FS-bc model also incorporates information from previous inundation maps, which significantly reduces the occurrence of nonphysical oscillations caused by the upstream BC.

Similar results were observed for the other flood events in the testing dataset (not shown).

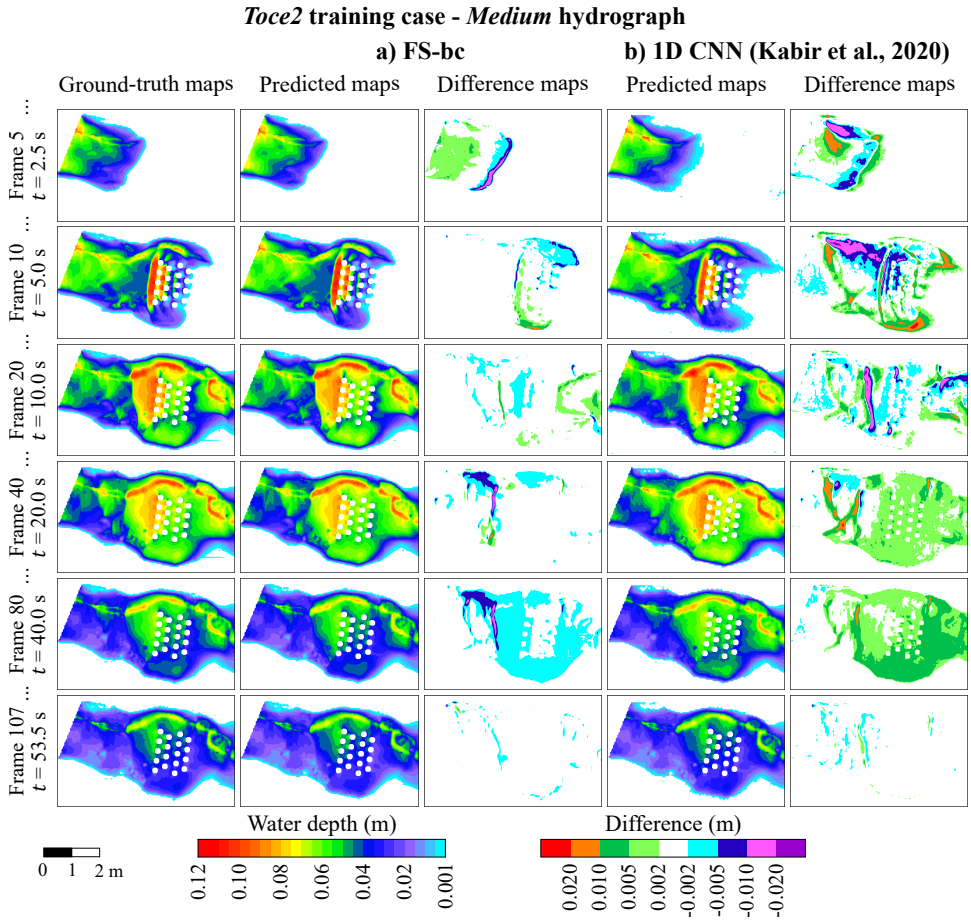

FIGURE 6.37 – Toce River case study: comparison between maps predicted by the FS-bc model (a) and the 1D CNN model (b) for the *Medium* hydrograph. The first column presents the ground-truth maps obtained from the hydrodynamic model, while the second and fourth columns display the maps predicted by the FS-bc and 1D CNN models, respectively. The third and fifth columns show the differences between predicted and target maps for each model. Only selected representative time steps are shown.

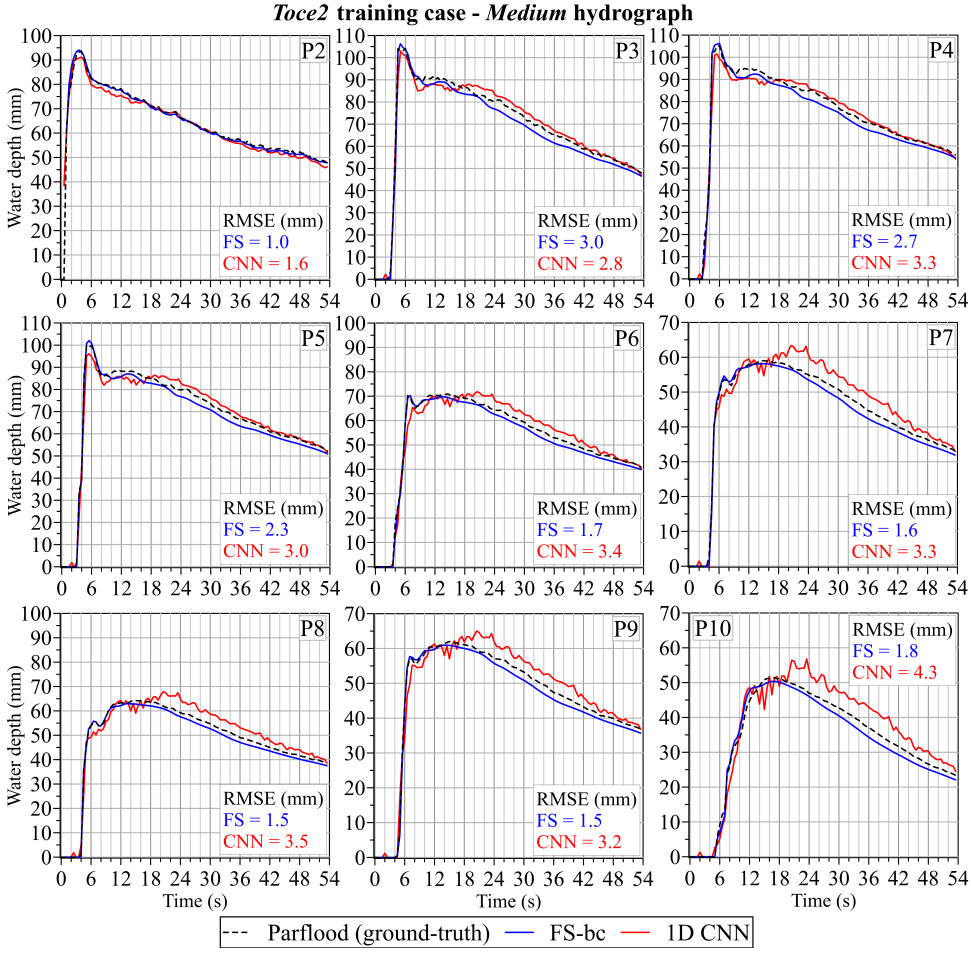

FIGURE 6.38 – Toce River case study: comparison of water depths at control points for the *Medium* hydrograph predicted using the FS-bc and 1D CNN models, both trained using the *Toce2* configuration.

TABLE 6.9 – Comparison of average performance metrics for the FS-bc and 1D CNN models. Results correspond to the *Toce2* and *Po2* training configurations.

Testing event	RMSE [m]		RMSE_ND [-]		F1 [-]	
	FS	1D CNN	FS	1D CNN	FS	1D CNN
Toce River						
Low	0.0018	0.0032	0.056	0.098	0.995	0.916
Medium	0.0026	0.0038	0.060	0.087	0.995	0.906
High	0.0032	0.0057	0.061	0.108	0.991	0.874
Gradual	0.0036	0.0064	0.083	0.157	0.991	0.868
Po River						
Nov 2011	0.19	0.36	0.060	0.120	0.973	0.966
Nov 2014	0.15	0.66	0.036	0.174	0.988	0.964
Jun 2020	0.16	0.66	0.041	0.181	0.982	0.908

Po River case study

The FS-bc model also exhibited superior performance in predicting flood events also for the Po River case study. For the November 2014 flood, Fig. 6.39 shows the water depth maps generated by the 1D CNN. This model shows low accuracy in capturing the spatiotemporal variations of water depths in defended floodplains, where errors exceeding 2 m are observed at certain time steps. This limitation is largely due to the 1D CNN's inability to account for spatiotemporal correlations across consecutive inundation maps, which is critical for accurately modeling complex flood dynamics. Furthermore, the comparison of these predictions with the results from the FS-bc model (Fig. 6.30) further confirms the lower performance of the 1D CNN.

Figs. 6.40–6.42 compare water depths extracted at the control points in the Po River region (see Fig. 6.21 for locations). In general, the performance of the 1D CNN model is acceptable in the main channel (i.e., at G1 and G2), but still inferior to that of the FS-bc model, especially for the November 2014 flood event (Fig. 6.41). Differently, the 1D CNN model performs poorly in defended floodplain areas (i.e.,

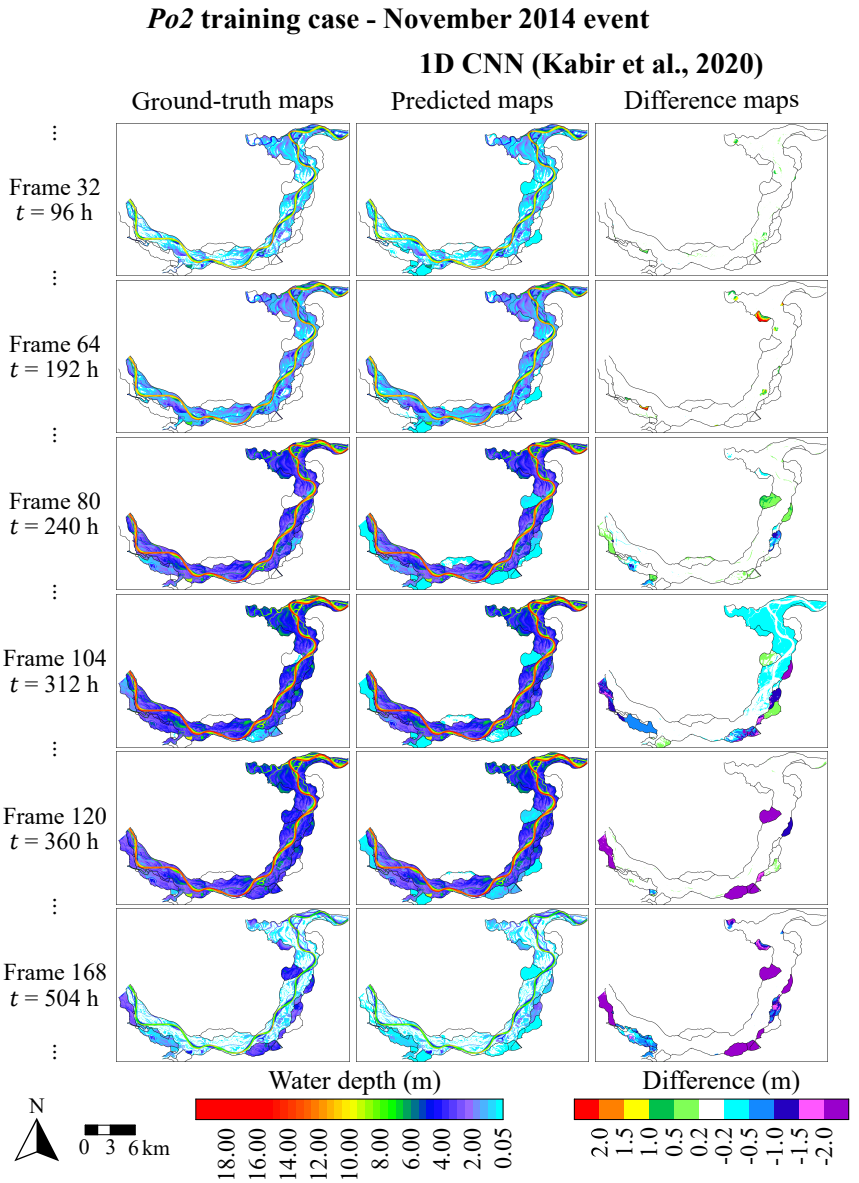

FIGURE 6.39 – Po River case study: November 2014 flood prediction using the 1D CNN model trained with the *Po2* configuration. The columns represent the ground-truth maps (first), the maps predicted by the 1D CNN model (second), and the difference between predicted and target maps (third). Only selected representative instants are shown. This Figure can be compared with Fig. 6.30, which shows the results for the FS-bc model for the same flood event.

G3-G6), where its predictions significantly deviate from the target data.

For the November 2014 flood event (Fig. 6.41), control points G4 and G5 exhibit a nonphysical trend in the water depths predicted by the 1D CNN model, which sudden oscillations in the range of 3-4 m. Differently, at control points G3 and G6, the convolutional model predicts the water depths with higher accuracy. However, during the recession phase of the flood, the 1D CNN model empties these floodplains, failing to capture the physics of the phenomena.

For the June 2020 flood event (Fig. 6.42), the 1D CNN model predicts the onset of the inundation in the defended floodplains approximately 360 hours after the event began. However, this forecast is completely nonphysical, as this flood event was characterised by discharge values at the Casalmaggiore section below 2,500 m^3/s,

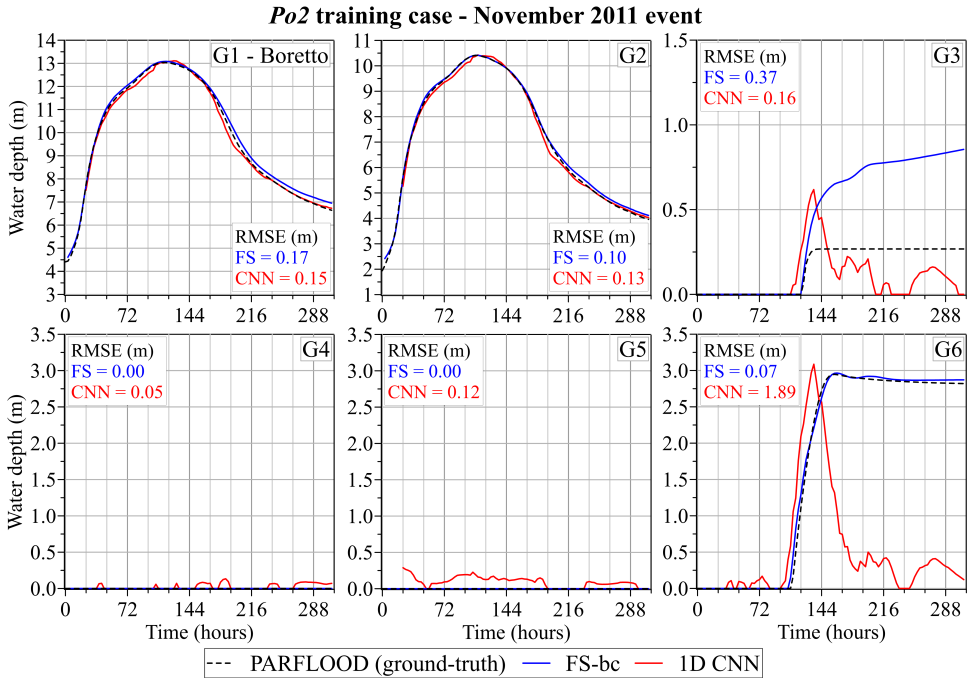

FIGURE 6.40 – Po River case study: comparison of simulated water depths at control points for the November 2011 flood event, using the FS-bc and 1D CNN models trained with the *Po2* configuration.

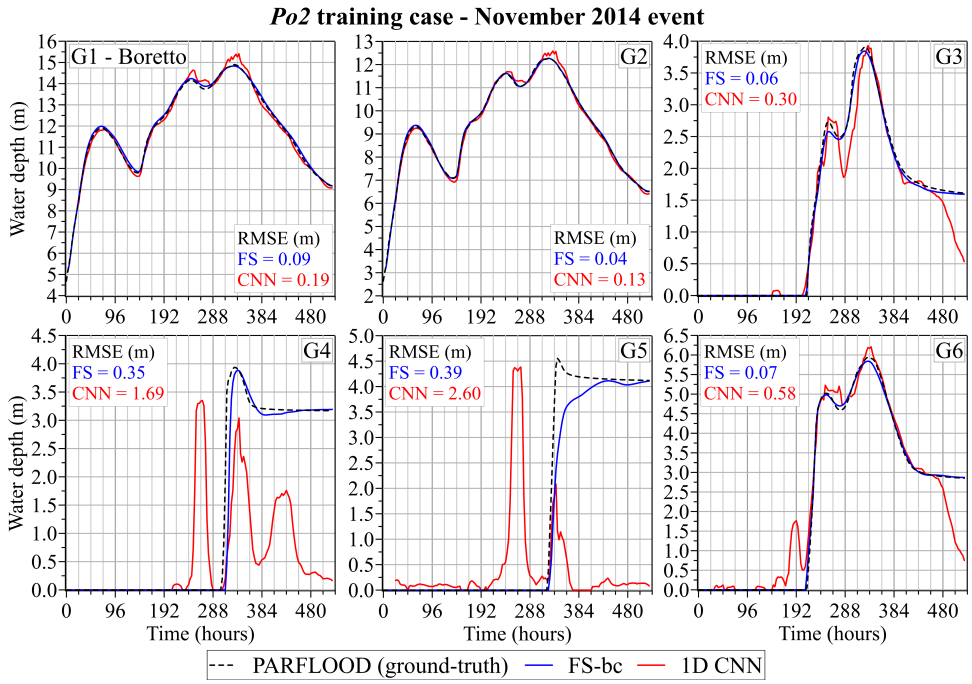

FIGURE 6.41 – Po River case study: comparison of simulated water depths at control points for the November 2014 flood event, using the FS-bc and 1D CNN models trained with the *Po2* configuration.

meaning that only the main river channel was affected by the inundation, as confirmed by the ground-truth maps in Fig. 6.31. In contrast, the FS-bc model correctly predicts that the defended floodplains remained completely dry for the entire duration of the June 2020 flood event.

Figs. 6.43–6.45 provide the scatter plots and histograms comparing target and predicted water depths for the three testing events obtained with the 1D CNN model. These figures, when compared to the results from the FS-bc model (Figs. 6.32–6.34), clearly demonstrate the superior performance of the transformer-based model. Notably, the FS-bc model exhibits a higher frequency of data points closely aligned with the dashed black line, indicating more accurate predictions. Furthermore, the 1D CNN model shows a significantly higher tendency to produce FP, with wrongly

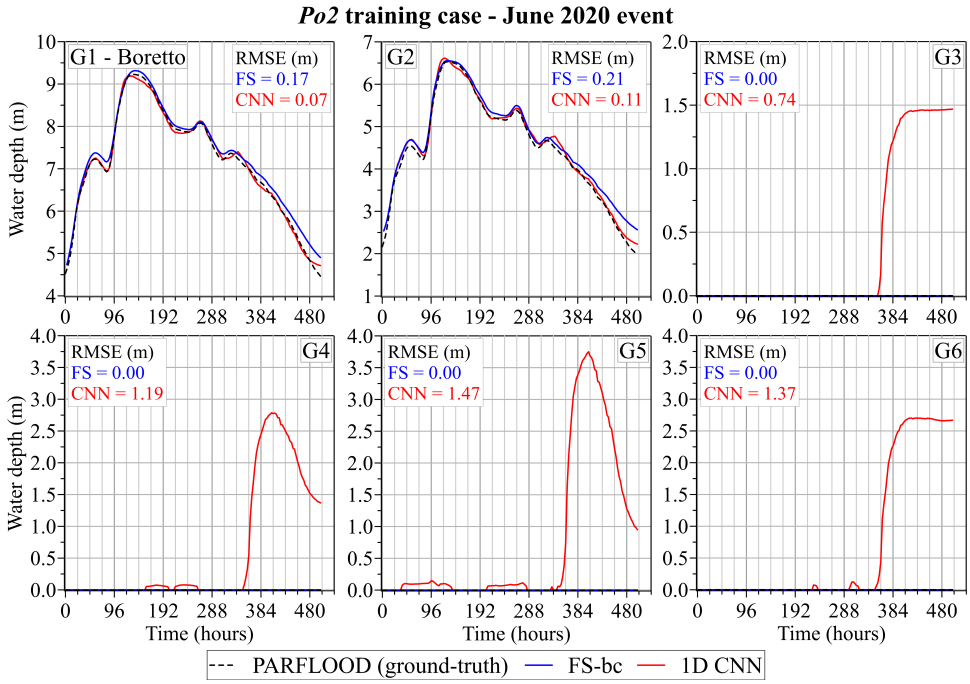

FIGURE 6.42 – Po River case study: comparison of simulated water depths at control points for the June 2020 flood event, using the FS-bc and 1D CNN models trained with the *Po2* configuration.

predicted water depths reaching up to 6-7 m.

The histograms in Figs. 6.43b, 6.44b, and 6.45b highlight the substantial percentage of cells with high errors (i.e., greater than 0.5 m) for the 1D CNN model, contrasting with the more accurate results generated by the FS-bc model (Figs. 6.32b, 6.33b, and 6.34b).

This benchmark analysis underscores the FS-bc model's superior performance compared to the state-of-the-art 1D CNN architecture. The superior accuracy of the FS-bc model is attributable to its ability to incorporate previous inundation conditions (i.e., water depth maps from prior time steps) into its predictions. The dynamics of a flood event in a river or floodplain at any given moment depend not only on the upstream boundary conditions but also on preceding flood dynamics.

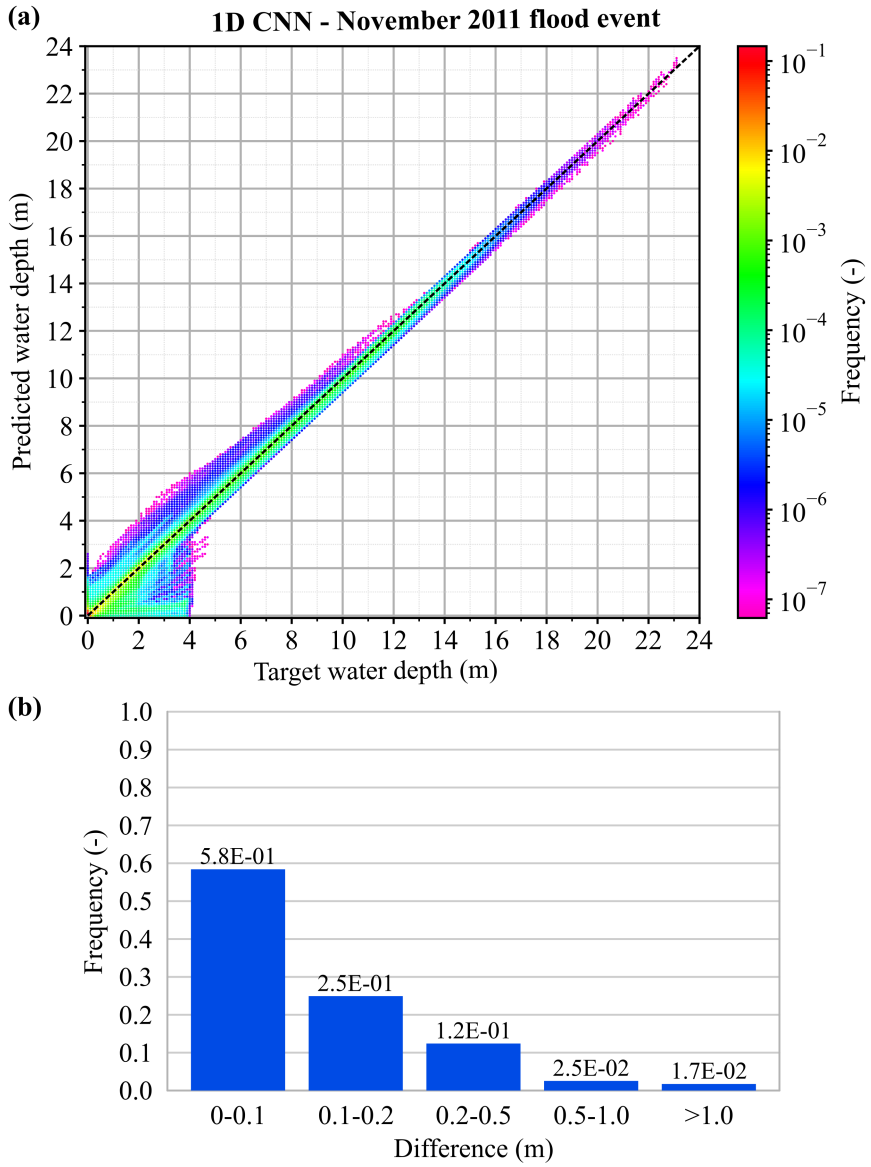

FIGURE 6.43 – Po River case study: comparison of predicted and target water depths for the November 2011 flood event, using the 1D CNN model trained with the *Po2* configuration. (a) Scatter plot comparing predicted and target water depths across all grid cells and time steps of the November 2011 flood event. The color scale indicates the frequency, normalized by the total number of wet cells (about 16.1M). The dashed black line represents the correct forecast. (b) Histogram showing the distribution of the differences between target and predicted water depths.

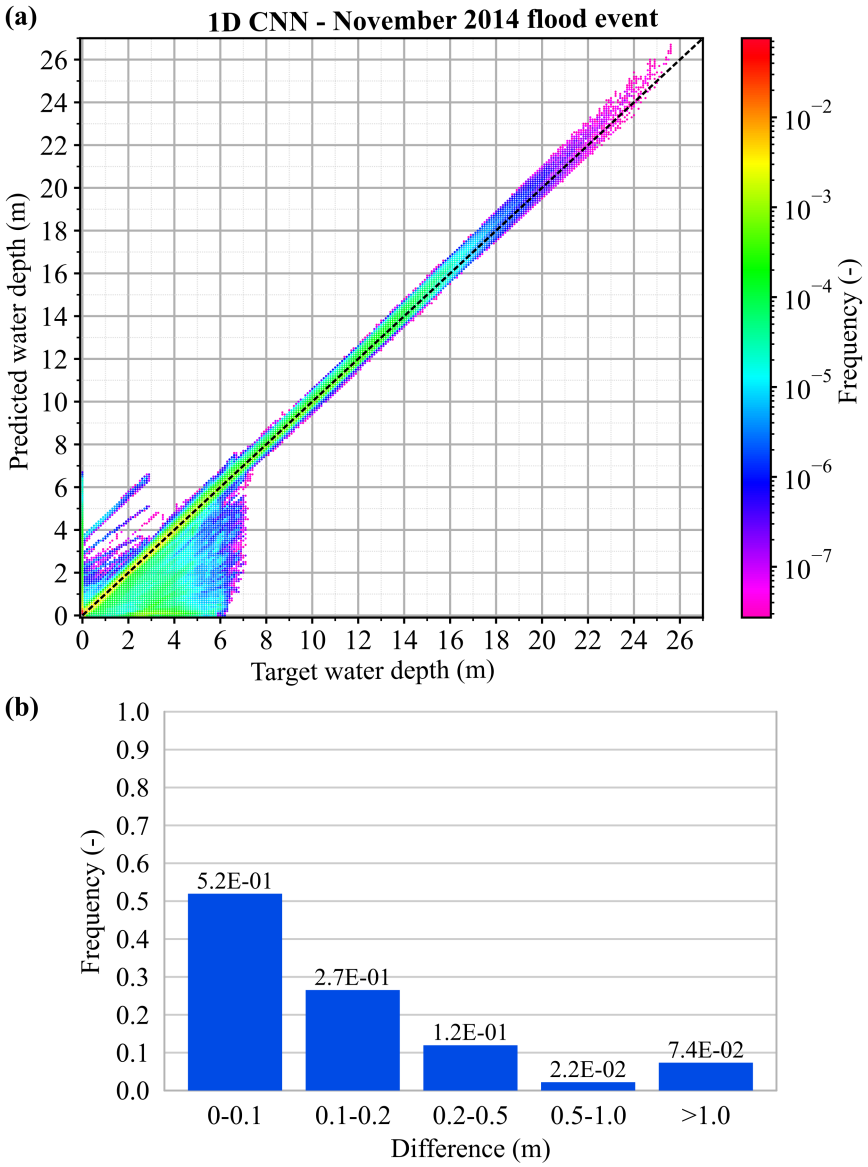

FIGURE 6.44 – Po River case study: comparison of predicted and target water depths for the November 2014 flood event, using the 1D CNN model trained with the *Po2* configuration. (a) Scatter plot comparing predicted and target water depths across all grid cells and time steps of the November 2014 flood event. The color scale indicates the frequency, normalized by the total number of wet cells (about 36.7M). The dashed black line represents the correct forecast. (b) Histogram showing the distribution of the differences between target and predicted water depths.

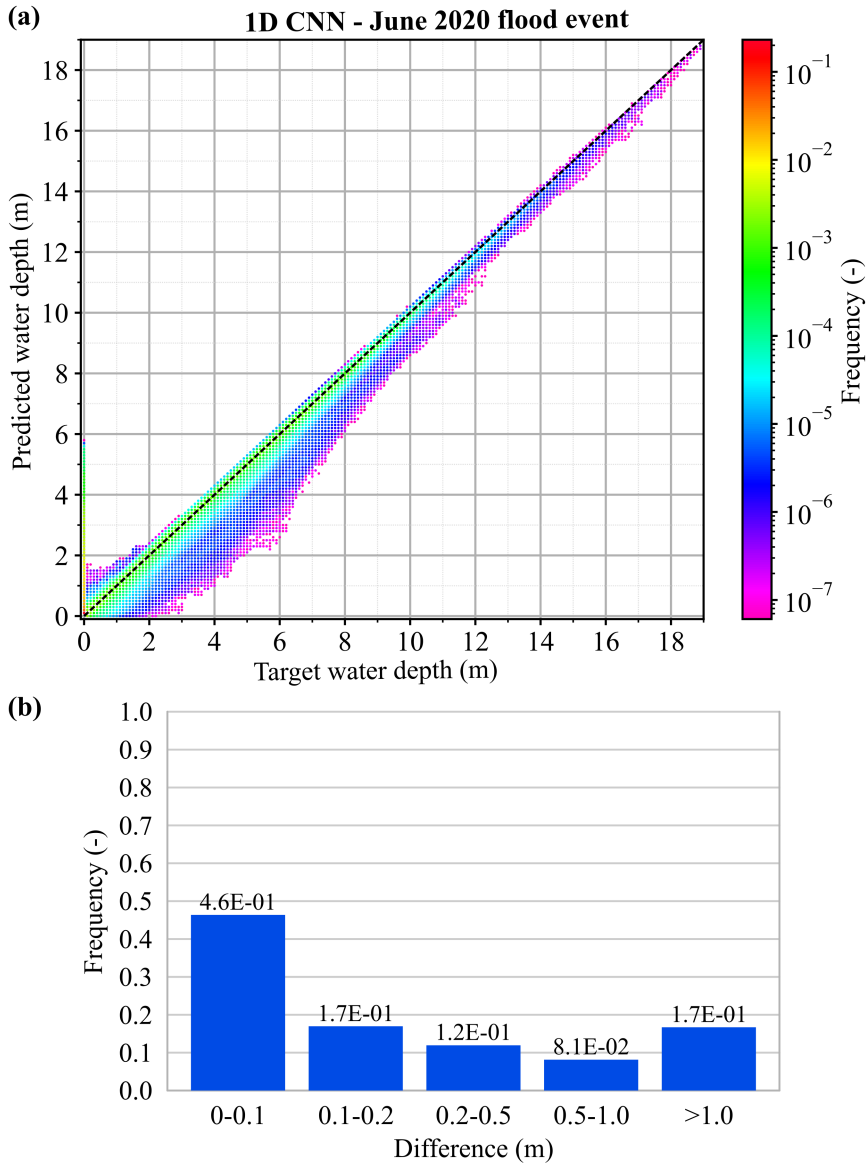

FIGURE 6.45 – Po River case study: comparison of predicted and target water depths for the June 2020 flood event, using the 1D CNN model trained with the *Po2* configuration. (a) scatter plot comparing predicted and target water depths across all grid cells and time steps of the June 2020 flood event. The color scale indicates the frequency, normalized by the total number of wet cells (about 16.5M). The dashed black line represents the correct forecast. (b) Histogram showing the distribution of the differences between target and predicted water depths.

This explains why the 1D CNN model struggles to predict the inundation in the defended floodplains of the Po River and displays a more oscillatory trend in water depths for the Toce River case.

6.6 Concluding remarks

In this Chapter, the FS-bc model was applied for real-time prediction of river flood events. Three case studies were conducted to assess the performance of the proposed surrogate model, evaluating its predictive capabilities under varying conditions of geometry, scale, and flow dynamic. The results consistently demonstrated the model's robustness in predicting the dynamics of river flood events across different conditions, achieving a high degree of accuracy in both simplified and realistic scenarios.

The first case study, focusing on the Baganza River, served as the initial application of the FS-bc model for river flood predictions. The findings highlighted the model's ability to replicate flood events for a river without levees and defended floodplains, achieving an average RMSE of approximately 3 cm.

The second case study centered on urban flash flood simulations in the Toce River valley, reproduced at a laboratory scale. Different types of flood events were used to generate the training dataset, enabling an analysis of the impact of hydrograph shape and intensity on the model's performance. The results emphasized that incorporating a diverse range of hydrographs in the training process enhances the model's predictive accuracy, yielding a reduction in the average RMSE by 10% to 30%.

Finally, the Po River case study assessed the model's performance in a real-world scenario characterized by particularly complex topography and a large number of cells. Different training configurations were tested, varying the number and intensity of flood events in the training dataset, as well as the spatial resolution of the maps. The results confirmed that a larger and more diverse dataset, encompass both real and synthetic hydrographs, improves the model's ability to predict unseen flood events, with a reduction in the average RMSE of up to 70%.

Additionally, a sensitivity analysis confirmed that the AR procedure of the FS-bc model is largely unaffected by the choice of initial condition maps. Consequently, for operational purposes, a small database of water depth maps corresponding to various steady flow conditions can be prepared prior to the emergency situation. This eliminates the need for time-consuming simulations with physically based models when real-time predictions are required, thus significantly reducing computational overhead.

The performance of the FS-bc model was also benchmarked against the 1D CNN model proposed by Kabir et al. (2020). The predictions of the FS-bc model exhibited an average RMSE that was 30% to 75% lower than that of the 1D CNN model. This result confirmed the superior accuracy of the FS-bc model, particularly attributed to its capacity to incorporate previous conditions (i.e., the water depth maps from prior time steps) into its predictions, as opposed to the 1D CNN, which relies solely on a sequence of inflow discharge values.

Chapter 7

Discussion

In this Chapter, the computational times of the FS model are presented in Section 7.1. Then, a critical analysis of the main results, along with the advantages and limitations of the proposed model, is provided in Section 7.2. Finally, potential directions for future research are discussed in Section 7.3.

7.1 Computational times

In the context of real-time flood forecasting, computational efficiency becomes a critical factor. During emergency situations, such as flood events, minimizing computation time is essential as it extends the time available for emergency services to implement protective measures, thereby mitigating loss of life and economic damage.

For practical applications, the primary concern is the surrogate model's inference time during emergency events, as this is when timely predictions are needed. The time and cost required to generate the training dataset and to train the surrogate model are less critical since these tasks can be performed before the emergency occurs.

In this Section, the computational times for both the FS-db and FS-bc models

are presented and compared to the runtimes of the PARFLOOD code as well as the 1D-CNN model (only for the cases 5 and 6). The computational times for the case studies on real topographies (i.e., cases 3, 4 and 6) are reported in Table 7.1.

It is worth noting that the PARFLOOD code exploits the computational power of GPUs and thus is far more efficient than other physically based serial codes (Vacondio et al., 2014).

The speed-up achieved by the FS model relative to the hydrodynamic model varies depending on the type of flood event being simulated. While the runtime of the PARFLOOD code is influenced by factors such as the CFL condition (see Section 2.2), the duration of the event, and the spatial resolution used, the surrogate model's inference time is driven primarily by the size of the model and the number of predicted inundation maps. Please note that the number of forecasted maps is influenced by the type of flood event. For instance, DB scenarios, which are characterized by rapid evolution times (typically spanning a few hours), necessitate a fine temporal resolution (in the order of minutes) to accurately capture the flow dynamics, resulting in high-frequency maps. In contrast, the slower and more gradual dynamics of river flood events allow for coarser temporal resolution (in the order of few hours), although these events can last much longer, from days to weeks. Therefore, while both the hydrodynamic model and the surrogate model have their respective computational constraints, the nature and dynamics of the flood event play a critical role in determining the speed-up achievable by the surrogate model.

It is important to note that the inference time of the surrogate model is unaffected by variations in the number of past frames (P), while the number of input frames (I) influences the training time (see Section 5.2.3).

All numerical simulations, as well as the training and inference processes, were conducted using NVIDIA A100 GPUs.

Table 7.1 – Computational times of the PARFLOOD code and inference times for the FS model using the AR procedure. The results presented concern exclusively to flood events on real topographies. The reported runtimes represent the mean of 3 identical simulations, each executed on a single GPU.

Case study	# of cells	Physical time [h]	# of maps	Inference time [min]	Physical/ inference time [-]	PARF. runtime [min]	PARF./ FS time [–]
3) Parma River flood-control dam							
Res. 20m	114,688	3	90	0.18	1000	0.13	0.72
Res. 10m	458,752			0.37	486	0.6	1.6
4) Baganza River							
2019 event	1,802,240	59	118	1.8	3933	4.8	2.7
6) Po River - November 2014 event							
Res. 20m	1,306,624	522	174	3.1	9788	29.3	9.5
Res. 10m	5,226,496			5.9	5308	139.3	23.6

Dam-break scenarios

For the DB scenario involving the Parma River flood-control dam with a spatial resolution of 20 m (case 3a), the total training time for the FS-db model is under 3.5 hours on a single GPU. The time required to recursively forecast 90 future maps is about 11 s. This runtime is comparable to the one of the highly efficient PARFLOOD code and it is negligible when considering the physical flood event duration of 3 hours, achieving a ratio of physical time to the FS model's computational time equal to 1,000.

Moreover, the current implementation of the FS-db model allows the inference procedure to be executed on a CPU. For instance, on an Intel Xeon CPU E5-2680v4 2.4GHz, the AR procedure takes approximately 2 minutes, despite the model not being optimized for CPU-based inference. This suggests that while training requires high-performance hardware, real-time forecasting could feasibly be conducted on standard laptops or workstations within a fraction of the event's actual duration.

For 10 m resolution maps (case 3b), the overall training time increases to approxi-

mately 6.5 hours, which is less than twice the time required for the coarser grid. This additional time is due to the larger FS-db model size, which requires fitting more parameters (see Table 5.1), and the increased dimensions of the maps. Nonetheless, the recursive forecasting of 90 future maps (equivalent to 3 hours of lead time) is completed in about 22 s.

The computational times for the synthetic DB case studies (cases 1 and 2), while less relevant for practical application, are on the order of a few seconds, aligning with PARFLOOD runtimes.

River flood events

For the Baganza River case study (case 4), the FS-bc model's training time is approximately 11 hours using 2 GPUs, while inference takes less than 2 minutes on a single GPU. This represents a computational time roughly 2.5 times faster than the PARFLOOD code, which takes about 5 minutes.

For the Toce River case study (case 5), the FS-bc model's training process takes approximately 2 hours on a single GPU, and the AR prediction of a flood event from the test dataset requires around 10 seconds. This computational time is comparable to that of the PARFLOOD code, which is highly efficient for small case studies like this one.

In the Po River case study (case 6), the computational time for the FS-bc model depends on the spatial resolution of the maps. At a 20 m resolution, training takes approximately 50 hours on 2 GPUs, while forecasting the 3-weeks flood event from November 2014 requires around 3 minutes on one GPU. In contrast, the PARFLOOD code takes roughly 30 minutes to simulate the same event under identical hardware conditions.

At a finer 10 m resolution, the training time for the FS-bc model increases to approximately 71 hours using 4 GPUs. Despite this longer training process, the model forecasts the entire November 2014 flood event in approximately 6 minutes on one GPU, while the PARFLOOD code takes roughly 140 minutes to simulate the

same event.

Overall, for the Po River case study, the FS-bc model achieves a ratio of physical time to computational time between 5,000 and 10,000, and, depending on the spatial resolution, it is approximately 10 to 20 times faster than the PARFLOOD code.

Comparison with the 1D CNN model

To evaluate the FS-bc model's performance in terms of computational time against state-of-the-art models, the runtimes of the 1D CNN model for the Toce River and Po River case studies are presented. It should be noted that both the training and inference of the 1D CNN model were performed on a single GPU.

For the Toce River case study, the 1D CNN model requires about 2 minutes for training and approximately 1.5 minutes for inference. For the Po River case study, training takes about 21 minutes, and prediction of the November 2014 flood event, with a 20 m resolution, takes about 3 minutes.

In conclusion, while the 1D CNN model's training time is shorter than the FS-bc model, the inference time is comparable.

7.2 Discussion of the surrogate model's results

The FS model represents an advancement in flood prediction, distinguishing itself from previously developed surrogate models by integrating autoencoder and transformer architectures. At the core of the transformer model lies the attention mechanism, which excels in capturing long-range dependencies and attending to various spatial and temporal aspects of the input data. This feature is essential to improve prediction accuracy, particularly for long-lasting flood events. In the context of simulating DB scenarios using data-driven models, the FS-db model is the first surrogate model capable of forecasting the temporal evolution of inundation maps based solely on the initial water level in the upstream reservoir, as shown in Section 5.4.3. In contrast, a large number of data-driven models has been developed

for river flood prediction (Karim et al., 2023). However, unlike other DL models that rely exclusively on upstream BCs data to generate inundation maps (e.g., Kabir et al., 2020), the FS-bc model incorporates not only upstream BC data but also inundation maps from previous time steps. By combining these data sources, the model captures the complex relationships between flow dynamics and topographical features more effectively, resulting in higher predictive accuracy than other simpler state-of-the-art models (see Section 6.5). The importance of incorporating spatiotemporal correlations in surrogate models for flood simulations is underscored by this improvement. This consideration is further validated by the interesting results obtained by recent studies employing various DL models designed to analyze these interdependencies (e.g., Bentivoglio et al., 2025).

Dataset creation

The generation of high-quality datasets is crucial for training data-driven models. The accuracy of flood predictions depends heavily on the reliability of the ground-truth data used during training. Unfortunately, observed flood maps with the required temporal and spatial resolution are rarely available. For example, maps derived from satellite imagery may have limitations due to low acquisition frequency and susceptibility to meteorological conditions (Bentivoglio et al., 2022). As a result, numerically generated datasets are the only viable option, underscoring the importance of using a validated hydrodynamic model capable of accurately simulating flood dynamics. In both DB and river flood studies, the well-established PARFLOOD code, rigorously validated in challenging real-world scenarios (see Section 2.2.2), has been adopted for this purpose.

The quality of the simulated maps and, consequently, the effectiveness of the training dataset, relies not only on the accuracy of the physically based model but also on the calibration of the roughness coefficient. For river flood predictions, historical records, such as gauged water levels at specific river sections (e.g., the water levels recorded at the Boretto gauging station in the Po River), are used for

calibrating the roughness coefficient for the river region. In the external lowland, if historical data on flood arrival times from previous inundations are available, they can be used as information for the calibration process. In the absence of such data, roughness coefficients from the literature can be used as an alternative, as was done for Toce River case. In contrast, for DB scenarios, where observed data are limited or nonexistent, reasonable roughness coefficients are adopted based on literature or calibration from other flood events (e.g., river floods) occurred in the same watercourse (see, for example, the case study 3). While hydrodynamic model calibration is crucial for ensuring the quality of training data, it is important to note that, in rapid flood events such as DB scenarios, the swift dynamics of flooding minimize the influence of inaccuracies in the roughness coefficient (Ferrari et al., 2023). However, in real-time forecasting applications, particularly for river floods, thorough calibration remains essential for ensuring reliable predictions.

The size and diversity of the training dataset are pivotal in determining the performance of data-driven models like the FS. These models tend to excel at interpolating within the range of the training data but struggle with extrapolation beyond it (Fraehr et al., 2024). To address this limitation, it is essential to construct datasets that cover a broad spectrum of potential scenarios to ensure robust generalization during inference. For DB predictions, this involves incorporating a wide range of scenarios, including extreme cases such as the maximum allowable water level in the dam. Similarly, for river flood predictions, the dataset should encompass diverse flood events, varying in hydrograph shapes, peak discharge levels, and intensities. Failure to include high-intensity or rare events can significantly reduce model accuracy, as observed in prior analyses (see Section 6.4.2). Consequently, ensuring a large and diverse dataset is fundamental for the model's success. Including a wide range of event types enhances the model's ability to generalize and reduces the risk of poor performance when encountering extreme or previously unseen flood scenarios.

Input data for the AR prediction

Once trained, the FS model leverages the AR procedure to predict long-lasting events using specific input data, depending on the model version. For DB scenarios, the FS-db model only requires the initial water depth map in the upstream reservoir (see Section 5.4.3). In view of practical application, the reservoir water level can be directly obtained from a gauging station inside the reservoir. This measure can be easily converted into a water depth map under lake-at-rest condition, which can be used as initial condition for the AR procedure.

For river flood predictions, the FS-bc model requires two types of input data. The first is the initial condition of the flood event. The methodology that can be adopted for the creation of this initial water depth map, along with the negligible influence that this information has on prediction results, have been already comprehensive discussed in Section 6.4.3. The second required input data for the AR procedure is the time series of upstream discharge throughout the flood's duration, typically provided by a hydrological model.

It is important to note that the computational cost of the hydrological model is generally much lower than that of a 2D hydrodynamic model, as the latter is governed by more complex equations and requires millions of cells to discretize large domains. Consequently, in a hydrological-hydraulic model chain, the hydrodynamic model often becomes the bottleneck in terms of computational cost. For this reason, this work focuses on developing a surrogate model for the hydrodynamic component, aiming to significantly reduce computational time without compromising accuracy. Furthermore, even if rainfall-runoff models generally have low computational demands, many data-driven surrogate models have already been developed in the literature to simulate these processes (e.g., Kratzert et al., 2018; Yin et al., 2022). Therefore, an alternative approach (not explored here) could involve using hydrographs generated by such surrogate models as input for the FS-bc model, further reducing the overall computational time required by the model chain.

In many cases, the lead time for predicting upstream inflow discharge may be

shorter than the total duration of a flood event, especially in rivers with long flood propagation times, such as the Po River. For this reason, the FS-bc model's AR procedure can be executed multiple times to update forecasts by incorporating the most up-to-date outputs from the rainfall-runoff model.

Benchmark comparison

In Section 6.5, the FS-bc model was benchmarked against the 1D-CNN model proposed by Kabir et al. (2020), demonstrating superior accuracy due to its ability to analyse spatiotemporal relationships between outputs. In contrast, the 1D-CNN model relies solely on upstream inflow discharge time series for its predictions. Furthermore, the two surrogate models exhibit comparable inference times, while the 1D-CNN architecture benefits from shorter training times.

The 1D-CNN model was among the first architectures applied to river flood forecasting using DL models. This model is characterized by its simplicity, computational efficiency, and ease of implementation, making it a suitable reference for diverse case studies. Moreover, its performance has been demonstrated on large domains with millions of cells (see Fraehr et al. (2024) and Kabir et al. (2020)), providing confidence in its applicability to the Po River case. However, it is important to acknowledge that more advanced architectures have been developed in recent years, combining CNNs with RNNs such as LSTM networks to capture both spatial and temporal dependencies in hydrological data, or using more complex DL frameworks such as GNN and GP models (see Section 1.4 and Bentivoglio et al. (2022), Bomers and Hulscher (2023), and Karim et al. (2023) for examples). These advanced models have shown superior performance particularly in scenarios where spatiotemporal dynamics play a critical role. However, they often come with increased complexity, computational cost, and tuning requirements. The decision to benchmark against the 1D-CNN model was driven by the need for a well-established baseline that is simple to train across different case studies and ensures comparability across domains.

Future work could explore benchmarking against other state-of-the-art approaches

to provide a more comprehensive evaluation of the FS-bc model's performance.

Limitations and advantages of the FS model

A common challenge associated with AR models, such as the FS, is the accumulation of errors over time, which can negatively impact the accuracy of long-term predictions. This issue arises because prediction at each step is based on the outputs from previous steps, leading to the potential compounding of small errors. One strategy to mitigate this problem is to adopt a non-AR model that predicts a fixed number of future frames (K) simultaneously, thereby reducing error accumulation and improving inference speed (Ye & Bilodeau, 2023). However, non-AR models present their own limitations, including more complex training processes due to the larger number of parameters, the need for sophisticated loss functions to capture spatiotemporal information, and increased GPU memory and time requirements (Ye & Bilodeau, 2023). These factors constrain the number of frames that can be predicted in a single step. As a result, if the desired lead time (i.e., F future frames) exceeds the number of maps (K) that the model can predict in one step, the forecasting process must recursively use the first predicted set of K maps to generate the next K maps, and so forth, until all F maps are forecasted. This recursive process can introduce error accumulation, similar to what is observed in AR models. Given these trade-offs, the adoption of a non-AR model was not pursued in this study. Further research could explore the potential of transformer-based non-AR models for predicting the temporal evolution of flood maps.

The results obtained for the Po River case study indicate that the FS model is only marginally susceptible to error accumulation when simulating river flood events. Specifically, the FS-bc model effectively predicts inundation maps for long-lasting flood events with minimal error accumulation (see, for example, the continuous blue lines in Figs. 6.24–6.26). The primary sources of error are observed during the recession limb of low-intensity events and at higher spatial resolution, as discussed in Section 6.4.2. Consequently, the forecasting lead time of the FS-bc model is

primarily constrained by the availability of upstream inflow data rather than by error accumulation. In contrast, for DB scenarios, the error accumulation has a more significant influence on the results (see Fig. 5.9). Nevertheless, despite this increased sensitivity, the model still delivers predictions that are accurate enough for practical applications.

One of the major limitation of DL models for flood prediction, including the proposed FS model, is their limited generalizability to changes in topography (Bentivoglio et al., 2022; Karim et al., 2023). The surrogate model is capable of producing accurate results only for the specific case study and topographical conditions on which it was trained. When applied to a different region with distinct topographical features, the model's performance can degrade significantly, as it struggles to capture the unique interactions between terrain and flood dynamics in untrained environments. Addressing this limitation often requires either retraining the model from scratch or fine-tuning it using additional data from the new region to maintain prediction accuracy. It is important to emphasize, however, that the primary goal of the proposed FS model is to serve as a ready-to-use tool for specific regions where accurate flood forecasting is urgently needed. Unlike generalized models designed to adapt to various topographies, which remain at a developmental stage and often require significant additional work to apply in practice, the FS model is already well-suited for operational use in targeted high-risk areas. This makes it particularly valuable for practical forecasting applications, where immediacy and reliability in specific contexts are critical.

Currently, only a handful of studies have investigated the generalizability of DL models in the context of pluvial floods (Löwe et al., 2021) and river flood events (Bentivoglio et al., 2023, 2025). These efforts focus on using innovative approaches, such as transfer learning or GNNs, to enhance model adaptability with minimal retraining. However, the application of these methods remains limited to simplified, low-resolution cases, and significant research is needed to transition them from experimental frameworks to practical tools. While further exploration

of generalizable models represents an exciting research direction, the ready-to-use nature of the proposed FS model offers a pragmatic and immediate solution for regions at risk.

Another limitation of the FS model, as highlighted by the results in Sections 5.4.2 and 6.4.2, is its sensitivity to the spatial resolution. This is evident from the increase in RMSE at higher resolutions (see, for example, Fig. 5.9). While finer spatial resolutions offer more detailed data, they also require a more complex model with an increased number of parameters, complicating the training process and requiring larger datasets. Striking an appropriate balance between spatial resolution and model complexity is essential for maintaining the model's reliability and robustness in real-time river flood forecasting. Despite this challenge, the results at finer resolutions remain suitable for real-time forecasting.

Finally, an inherent limitation of the FS model, which is common to many data-driven models that rely on numerically generated datasets, is the need to construct a large dataset for model training. This process involves running the physically based hydrodynamic model multiple times to simulate a wide range of scenarios, which can be computationally expensive. The repeated simulations required to generate sufficient data significantly increase the computational load, often necessitating access to HPC resources, such as GPU-based clusters, to ensure feasible execution times.

Despite these limitations, the results confirm the applicability of the FS model for real-time forecasting of flood events over real topographies, delivering acceptable performance even in scenarios involving complex flow dynamics and intricate terrain geometries. Moreover, the extremely favorable ratio of physical to computational time, combined with the substantial speed-up achieved compared to the hydrodynamic model, especially for river flood events (see Table 7.1), underscores the model's efficiency. These advantages position the FS model as a promising tool for operational flood forecasting, capable of balancing accuracy and computational feasibility in practical applications.

7.3 Future developments

Future developments of the FS model could involve the integration of additional input data, such as velocity fields and terrain elevation, to enhance the information available to the surrogate model for making predictions. Additionally, a promising avenue for advancement lies in extending the model's capabilities to simulate dike breach events along rivers. This type of flooding, which differs from both DB and river flood events, introduces complex dynamics driven by factors such as inflow hydrographs, breach failure mechanisms, breach opening times, and the evolution of breach geometry.

Another valuable extension could focus on enhancing the integration of boundary conditions, enabling the use of multiple input hydrographs (e.g., in scenarios where multiple rivers must be modeled simultaneously) and incorporating downstream boundary conditions. This enhancement would enable more accurate simulations in cases where downstream factors, such as tidal effects or fluctuations in lake water levels, significantly influence the flooding dynamics.

The proposed FS model, while primarily designed for forecasting DB scenarios and river floods, presents potential adaptability to other types of flood events, such as pluvial floods. These events are typically driven by localized and intense precipitation occurring directly within the modeled area. Adapting the FS model to handle such scenarios would require modifications to its input structure, particularly through the incorporation of rainfall data. For instance, integrating rainfall maps as additional input layers could provide the model with the meteorological context necessary to capture precipitation-driven flooding dynamics. Furthermore, the model's architecture could be extended to address combined flood scenarios, such as those arising from the interplay of pluvial and river flooding. Such extensions would enhance the versatility and applicability of the FS model across a broader range of flood events.

Addressing the current limitations related to model generalization across diverse topographies could be another key research avenue, aiming to improve the model's

performance when applied to different regions without the need for extensive retraining. Techniques such as transfer learning or geographically diverse training datasets could be explored to enhance the model's adaptability. Furthermore, incorporating the DTM as additional input information may play a pivotal role in overcoming this limitation.

A comprehensive analysis of the model's hyperparameters and configurations could be undertaken to identify optimal values, potentially leading to even more accurate and reliable predictions. Furthermore, further research could explore improved parallelization techniques and more efficient use of computational resources to further reduce computational time for large-scale simulations.

Finally, integrating the FS model in a hydrological modeling chain used by water management authorities could enhance its practical applicability. Such integration would allow flood forecasts to be directly incorporated into real-time decision-making processes for EWS and emergency response systems. To achieve this integration, several adaptations may be necessary. The existing model chain would need to be modified to replace the hydrodynamic model with the FS model, ensuring compatibility with the input and output data. This process may involve reconfiguring the input preprocessing steps to accommodate the specific data requirements of the FS model, such as formatting discharge data. Additionally, adjustments to the model chain's output handling may be required to ensure that the predictions generated by the FS model are correctly interpreted and visualized by subsequent systems, including decision support tools and visualization platforms. Interfacing the FS model with real-time observational data sources, such as gauge stations or remote sensing data, may also be necessary to facilitate accurate and timely predictions. These adaptations collectively aim to ensure an effective integration of the FS model into the operational framework of water management authorities.

Conclusions

The thesis focused on the development of a data-driven model capable of real-time simulation of real flood events over large domains. Achieving this objective requires a DL model that is both accurate and efficient, capable of generating time-evolving flood maps in a fraction of the physical time. These maps can assist disaster management agencies in taking timely countermeasures during emergencies, thus reducing the impact on affected populations.

To address these requirements, the proposed model, named FloodSformer, leverages state-of-the-art DL architectures (i.e., autoencoder and transformer architectures) to predict 2D inundation maps. The model uses inundation maps at previous times and, where necessary, time series data of upstream inflow discharges. The key component of the proposed model is the attention mechanism, which enables the extraction of spatiotemporal correlations and the learning of relationships governing flow dynamics in the study area. Additionally, an autoregressive procedure is integrated to forecast entire flood events.

The use of a transformer-based model for simulating spatiotemporal flood evolution, combined with input data from both inflow discharges and previous inundation maps, represents a novel approach compared to other data-driven models in the literature. This methodology significantly improves performance over a simpler state-of-the-art model (as shown in Section 6.5), resulting in greater accuracy during inference and enhancing the model's reliability and applicability in real-time flood management scenarios.

The FloodSformer model has been evaluated across two types of food events: (a) dam-break scenarios and (b) river flood events. To handle these events, two versions of the model were developed:

1. FS-db: designed for predicting dam-break scenarios, this version incorporates a multi-head self-attention mechanism within the transformer layers. The model uses water depth maps from previous time steps to predict subsequent maps.

2. FS-bc: optimized for river flood events, this version replaces the multi-head self-attention with the multi-head cross-attention mechanism, allowing the use of diverse input data. Both inflow discharges and previous inundation maps serve as inputs to predict subsequent inundation maps.

The two versions are integrated into a unified code framework, enabling the end user to easily select the type of event (dam-break or river flood) when running the model.

Both versions of the FloodSformer model have been validated using synthetic and real-world case studies. Datasets for each case study were generated numerically through the PARFLOOD hydrodynamic model.

For dam-break scenarios (discussed in Chapter 5), the FS-db model was tested on three case studies: two synthetic dam-break scenarios and a hypothetical dam collapse in a real-world domain. The results highlight the model's strong performance in forecasting the temporal evolution of inundation in both synthetic and real scenarios. In the case of the hypothetical collapse of the Parma River flood-control dam, the average RMSE ranged between 10 cm and 16 cm, depending on spatial resolution. Additionally, the model required few seconds of computational time to forecast a 3-hour flood event, achieving a physical-to-computational time ratio between 480 and 1,000.

For river flood events (detailed in Chapter 6), the FS-bc model was evaluated across three case studies of increasing complexity in topography and flood dynamics. The model demonstrated high accuracy in predicting long-duration flood events, with prediction errors comparable to the uncertainty of physically based models. In the Po

River case study, for example, the average RMSE and RMSE_ND were consistently below 20 cm and 0.06, respectively, across all testing events, while the F1 score is higher than 0.97. Furthermore, the FS-bc model outperformed the state-of-the-art 1D CNN model in predicting the Po River and Toce River flood events (see Section 6.5).

The computational efficiency of the FloodSformer model is especially notable for river flood simulations (see Section 7.1). For instance, a flood event on the Po River lasting approximately 3 weeks required only a few minutes of computational time, with a physical-to-computational time ratio as high as 10,000. The model also achieved a speedup of 10 to 20 times compared to the PARFLOOD code, even though the latter was optimized for parallel processing on GPU.

Extensive sensitivity analyses were conducted to evaluate the influence of various hyperparameters, the type and number of flood events used for training, and the spatial resolution of the maps. These analyses are critical for improving the model's ability to generalize to unseen flood events.

In conclusion, the FloodSformer model has proven to be a versatile and effective tool for predicting flood events. The results indicate that the proposed model is a valuable resource for flood risk management and emergency response planning.

Code and data availability

The source code for the FloodSformer model, along with the datasets utilized in this research, can be accessed through the following Zenodo repositories:

- Python code for the FloodSformer model: Pianforini et al. (2024c). https://doi.org/10.5281/zenodo.13221319
- Datasets and trained model weights for the dam-break case studies: Pianforini et al. (2024b). https://doi.org/10.5281/zenodo.10878385
- Datasets and trained model weights for the river flood case studies: Pianforini et al. (2024d). https://doi.org/10.5281/zenodo.11472228

References

Abrahart, R. J., See, L. M., & Solomatine, D. P. (2008). *Practical hydroinformatics: Computational intelligence and technological developments in water applications* (Vol. 68). Springer Science & Business Media.

Aggarwal, C. C. (2018). *Neural networks and deep learning*. Springer. https://doi.org/10.1007/978-3-319-94463-0

Alfieri, L., Burek, P., Dutra, E., Krzeminski, B., Muraro, D., Thielen, J., & Pappenberger, F. (2013). GloFAS – global ensemble streamflow forecasting and flood early warning. *Hydrology and Earth System Sciences*, *17*(3), 1161–1175. https://doi.org/10.5194/hess-17-1161-2013

Aureli, F., Maranzoni, A., Mignosa, P., & Ziveri, C. (2008a). A weighted surface-depth gradient method for the numerical integration of the 2D shallow water equations with topography. *Advances in Water Resources*, *31*(7), 962–974. https://doi.org/10.1016/j.advwatres.2008.03.005

Aureli, F., Maranzoni, A., Mignosa, P., & Ziveri, C. (2008b). Dam-break flows: Acquisition of experimental data through an imaging technique and 2D numerical modeling. *Journal of Hydraulic Engineering*, *134*, 1089–1101. https://doi.org/10.1061/(asce)0733-9429(2008)134:8(1089)

Aureli, F., Maranzoni, A., & Petaccia, G. (2021). Review of historical dam-break events and laboratory tests on real topography for the validation of numerical models. *Water (Switzerland)*, *13*, 1968. https://doi.org/10.3390/w13141968

Bates, P. D., & De Roo, A. P. (2000). A simple raster-based model for flood inundation simulation. *Journal of Hydrology*, *236*, 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), 33–45. https://doi.org/10.1016/j.jhydrol.2010.03.027

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

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

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

Bermúdez, A., Dervieux, A., Desideri, J.-A., & Vázquez, M. (1998). Upwind schemes for the two-dimensional shallow water equations with variable depth using unstructured meshes. *Computer Methods in Applied Mechanics and Engineering*, *155*(1), 49–72. https://doi.org/10.1016/S0045-7825(97)85625-3

Bermúdez, M., Cea, L., & Puertas, J. (2019). A rapid flood inundation model for hazard mapping based on least squares support vector machine regression.

Journal of Flood Risk Management, *12*(S1), e12522. https://doi.org/10.1111/jfr3.12522

Bertasius, G., Wang, H., & Torresani, L. (2021). Is space-time attention all you need for video understanding? *ICML*, 813–824.

Besseling, L. S., Bomers, A., & Hulscher, S. J. M. H. (2024). Predicting flood inundation after a dike breach using a Long Short-Term Memory (LSTM) neural network. *Hydrology*, *11*(9). https://doi.org/10.3390/hydrology11090152

Bladé, E., Gómez-Valentín, M., Dolz, J., Aragón-Hernández, J., Corestein, G., & Sánchez-Juny, M. (2012). Integration of 1D and 2D finite volume schemes for computations of water flow in natural channels. *Advances in Water Resources*, *42*, 17–29. https://doi.org/10.1016/j.advwatres.2012.03.021

Bomers, A. (2021). Predicting outflow hydrographs of potential dike breaches in a bifurcating river system using NARX neural networks. *Hydrology*, *8*, 87. https://doi.org/10.3390/hydrology8020087

Bomers, A., & Hulscher, S. J. M. H. (2023). Neural networks for fast fluvial flood predictions: Too good to be true? *River Research and Applications*, *39*, 1652–1658. https://doi.org/10.1002/RRA.4144

Boosari, S. (2019). Predicting the dynamic parameters of multiphase flow in CFD (dam-break simulation) using artificial intelligence-(cascading deployment). *Fluids*, *4*, 44. https://doi.org/10.3390/fluids4010044

Bouallègue, Z. B., Clare, M. C. A., Magnusson, L., Gascón, E., Maier-Gerber, M., Janoušek, M., Rodwell, M., Pinault, F., Dramsch, J. S., Lang, S. T. K., Raoult, B., Rabier, F., Chevallier, M., Sandu, I., Dueben, P., Chantry, M., & Pappenberger, F. (2024). The rise of data-driven weather forecasting: A first statistical assessment of machine learning–based weather forecasts in an operational-like context. *Bulletin of the American Meteorological Society*, *105*(6), E864–E883. https://doi.org/10.1175/BAMS-D-23-0162.1

Brunner, G. W. (2016). HEC-RAS river analysis system: Hydraulic reference manual, version 5.0. *US Army Corps of Engineers–Hydrologic Engineering Center*, *547*.

Burrichter, B., da Silva, J. K., Niemann, A., & Quirmbach, M. (2024). A temporal fusion transformer model to forecast overflow from sewer manholes during pluvial flash flood events. *Hydrology*, *11*, 41. https://doi.org/10.3390/HYDROLOGY11030041

Burrichter, B., Hofmann, J., da Silva, J. K., Niemann, A., & Quirmbach, M. (2023). A spatiotemporal deep learning approach for urban pluvial flood forecasting with multi-source data. *Water*, *15*, 1760. https://doi.org/10.3390/W15091760

Caleffi, V., Valiani, A., & Zanni, A. (2003). Finite volume method for simulating extreme flood events in natural channels. *Journal of Hydraulic Research*, *41*, 167–177. https://doi.org/10.1080/00221680309499959

Campolo, M., Andreussi, P., & Soldati, A. (1999). River flood forecasting with a neural network model. *Water Resources Research*, *35*(4), 1191–1197. https://doi.org/10.1029/1998WR900086

Castangia, M., Grajales, L. M. M., Aliberti, A., Rossi, C., Macii, A., Macii, E., & Patti, E. (2023). Transformer neural networks for interpretable flood forecasting. *Environmental Modelling and Software*, *160*, 105581. https://doi.org/10.1016/j.envsoft.2022.105581

Casulli, V. (1990). Semi-implicit finite difference methods for the two-dimensional shallow water equations. *Journal of Computational Physics*, *86*(1), 56–74. https://doi.org/10.1016/0021-9991(90)90091-E

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

Chaudhary, P., Leitão, J. P., Schindler, K., & Wegner, J. D. (2024). Flood water depth prediction with convolutional temporal attention networks. *Water*, *16*(9). https://doi.org/10.3390/w16091286

Choi, C., Kim, J., Han, H., Han, D., & Kim, H. S. (2020). Development of water level prediction models using machine learning in wetlands: A case study of Upo Wetland in South Korea. *Water*, *12*(1). https://doi.org/10.3390/w12010093

Chollet, F. (2021). *Deep learning with python*. Manning Publications.

Chow, V. T. (1959). *Open channel hydraulics*. McGraw-Hill Book Company, Inc.

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

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

Dawson, C. W., & Wilby, R. L. (2001). Hydrological modelling using artificial neural networks. *Progress in Physical Geography: Earth and Environment*, *25*(1), 80–108. https://doi.org/10.1177/030913330102500104

Dazzi, S., Mignosa, P., Pianforini, M., & Vacondio, R. (2024). Regional-scale 2D hydraulic modelling for the assessment of the residual flood hazard due to levee breaches. In *River flow 2022* (pp. 1049–1057). CRC Press.

Dazzi, S., Shustikova, I., Domeneghetti, A., Castellarin, A., & Vacondio, R. (2021a). Comparison of two modelling strategies for 2D large-scale flood simulations. *Environmental Modelling and Software*, *146*, 105225. https://doi.org/10.1016/j.envsoft.2021.105225

Dazzi, S., Vacondio, R., & Mignosa, P. (2019). Integration of a levee breach erosion model in a GPU-accelerated 2D shallow water equations code. *Water Resources Research*, *55*, 682–702. https://doi.org/10.1029/2018WR023826

Dazzi, S., Vacondio, R., & Mignosa, P. (2020). Internal boundary conditions for a GPU-accelerated 2D shallow water model: Implementation and applications.

Advances in Water Resources, *137*. https://doi.org/10.1016/j.advwatres.2020.103525

Dazzi, S., Vacondio, R., & Mignosa, P. (2021b). Flood stage forecasting using machine-learning methods: A case study on the Parma River (Italy). *Water (Switzerland)*, *13*, 1612. https://doi.org/10.3390/w13121612

Dazzi, S., Vacondio, R., Mignosa, P., & Aureli, F. (2022). Assessment of pre-simulated scenarios as a non-structural measure for flood management in case of levee-breach inundations. *International Journal of Disaster Risk Reduction*, *74*, 102926. https://doi.org/10.1016/j.ijdrr.2022.102926

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255. https://doi.org/10.1109/CVPR.2009.5206848

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of naacL-HLT*, *1*, 4171–4186.

DHI. (2021). Mike-11: A modelling system for rivers and channels, reference manual. *DHI–Water and Development, Horsholm, Denmark*.

do Lago, C. A. F., Brasil, J. A. T., Junior, M. N. G., 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

do Lago, C. A. F., 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*, *618*, 129276. https://doi.org/10.1016/j.jhydrol.2023.129276

Donnelly, J., Abolfathi, S., Pearson, J., Chatrabgoun, O., & Daneshkhah, A. (2022). Gaussian process emulation of spatio-temporal outputs of a 2D inland flood

model. *Water Research*, *225*, 119100. https://doi.org/10.1016/j.watres.2022.119100

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. http://arxiv.org/abs/2010.11929

Dottori, F., Mentaschi, L., Bianchi, A., Alfieri, L., & Feyen, L. (2023). Cost-effective adaptation strategies to rising river flood risk in Europe. *Nature Climate Change*, *13*(2), 196–202. https://doi.org/10.1038/s41558-022-01540-0

Douris, J., & Kim, G. (2021). Atlas of mortality and economic losses from weather, climate and water extremes (1970-2019).

Emerton, R. E., Stephens, E. M., Pappenberger, F., Pagano, T. C., Weerts, A. H., Wood, A. W., Salamon, P., Brown, J. D., Hjerdt, N., Donnelly, C., Baugh, C. A., & Cloke, H. L. (2016). Continental and global scale flood forecasting systems. *WIREs Water*, *3*(3), 391–418. https://doi.org/10.1002/wat2.1137

Fernández-Nóvoa, D., González-Cao, J., & García-Feal, O. (2024). Enhancing flood risk management: A comprehensive review on flood early warning systems with emphasis on numerical modeling. *Water*, *16*(10). https://doi.org/10.3390/w16101408

Ferrari, A. (2018). *2D Shallow Water modelling of flood propagation: GPU parallelization, non-uniform grids, porosity, reverse flow routing* [Doctoral dissertation, Università di Parma. Dipartimento di Ingegneria e Architettura]. https://hdl.handle.net/20.500.14242/153845

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*, 59–72. https://doi.org/10.5194/nhess-20-59-2020

Ferrari, A., Vacondio, R., & Mignosa, P. (2023). High-resolution 2D shallow water modelling of dam failure floods for emergency action plans. *Journal of Hydrology*, *618*, 129192. https://doi.org/10.1016/j.jhydrol.2023.129192

Ferrari, A., Viero, D. P., Vacondio, R., Defina, A., & Mignosa, P. (2019). Flood inundation modeling in urbanized areas: A mesh-independent porosity approach with anisotropic friction. *Advances in Water Resources*, *125*, 98–113. https://doi.org/10.1016/j.advwatres.2019.01.010

Fraehr, N., Wang, Q. J., Wu, W., & Nathan, R. (2022). Upskilling low-fidelity hydrodynamic models of flood inundation through spatial analysis and gaussian process learning. *Water Resources Research*, *58*, 1–21. https://doi.org/10.1029/2022WR032248

Fraehr, N., Wang, Q. J., Wu, W., & Nathan, R. (2023). Development of a fast and accurate hybrid model for floodplain inundation simulations. *Water Resources Research*, *59*, e2022WR033836. https://doi.org/10.1029/2022wr033836

Fraehr, N., Wang, Q. J., Wu, W., & Nathan, R. (2024). Assessment of surrogate models for flood inundation: The physics-guided lsg model vs. state-of-the-art machine learning models. *Water Research*, *252*, 121202. https://doi.org/10.1016/j.watres.2024.121202

García-Feal, O., González-Cao, J., Gómez-Gesteira, M., Cea, L., Domínguez, J. M., & Formella, A. (2018). An accelerated tool for flood modelling based on Iber. *Water*, *10*(10). https://doi.org/10.3390/w10101459

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Guo, Z., Leitão, J. P., Simões, N. E., & Moosavi, V. (2021). Data-driven flood emulation: Speeding up urban flood predictions by deep convolutional neural networks. *Journal of Flood Risk Management*, *14*(1), e12684. 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

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. https://doi.org/10.1109/CVPR.2016.90

He, K., Gan, C., Li, Z., Rekik, I., Yin, Z., Ji, W., Gao, Y., Wang, Q., Zhang, J., & Shen, D. (2023). Transformers in medical image analysis. *Intelligent Medicine*, *3*(1), 59–78. https://doi.org/10.1016/j.imed.2022.07.002

Hofmann, J., & Schüttrumpf, H. (2021). Floodgan: Using deep adversarial learning to predict pluvial flooding in real time. *Water (Switzerland)*, *13*. https://doi.org/10.3390/w13162255

Hop, F. J., Linneman, R., Schnitzler, B., Bomers, A., & Booij, M. J. (2024). Real time probabilistic inundation forecasts using a LSTM neural network. *Journal of Hydrology*, *635*, 131082. https://doi.org/10.1016/J.JHYDROL.2024.131082

Hu, C., Wu, Q., Li, H., Jian, S., Li, N., & Lou, Z. (2018). Deep learning with a long short-term memory networks approach for rainfall-runoff simulation. *Water (Switzerland)*, *10*, 1543. https://doi.org/10.3390/w10111543

Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1125–1134.

Ji, J., He, J., Lei, M., Wang, M., & Tang, W. (2024). Spatio-temporal transformer network for weather forecasting. *IEEE Transactions on Big Data*, 1–16. https://doi.org/10.1109/TBDATA.2024.3378061

Jin, H., Lu, H., Zhao, Y., Zhu, Z., Yan, W., Yang, Q., & Zhang, S. (2024). Integration of an improved transformer with physical models for the spatiotemporal simulation of urban flooding depths. *Journal of Hydrology: Regional Studies*, *51*, 101627. https://doi.org/10.1016/J.EJRH.2023.101627

Johnson, J. M., Munasinghe, D., Eyelade, D., & Cohen, S. (2019). An integrated evaluation of the National Water Model (NWM)–Height Above Nearest Drainage (HAND) flood mapping methodology. *Natural Hazards and Earth*

System Sciences, *19*(11), 2405–2420. https://doi.org/10.5194/nhess-19-2405-2019

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*, 125481. https://doi.org/10.1016/j.jhydrol.2020.125481

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*, *598*, 126371. https://doi.org/10.1016/j.jhydrol.2021.126371

Karim, F., Armin, M. A., Ahmedt-Aristizabal, D., Tychsen-Smith, L., & Petersson, L. (2023). A review of hydrodynamic and machine learning approaches for flood inundation modeling. *Water (Switzerland)*, *15*, 566. https://doi.org/10.3390/w15030566

Kasiviswanathan, K., He, J., Sudheer, K., & Tay, J.-H. (2016). Potential application of wavelet neural network ensemble to forecast streamflow for flood management. *Journal of Hydrology*, *536*, 161–173. https://doi.org/10.1016/j.jhydrol.2016.02.044

Khan, S., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., & Shah, M. (2022). Transformers in vision: A survey. *ACM Comput. Surv.*, *54*(10s). https://doi.org/10.1145/3505244

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations (ICLR)*.

Kordani, M., Nikoo, M. R., Fooladi, M., Ahmadianfar, I., Nazari, R., & Gandomi, A. H. (2024). Improving long-term flood forecasting accuracy using ensemble deep learning models and an attention mechanism. *Journal of Hydrologic Engineering*, *29*(6), 04024042. https://doi.org/10.1061/JHYEFF.HEENG-6262

Kratzert, F., Klotz, D., Brenner, C., Schulz, K., & Herrnegger, M. (2018). Rainfall-runoff modelling using Long Short-Term Memory (LSTM) networks. *Hydrology and Earth System Sciences*, *22*(11), 6005–6022. https://doi.org/10.5194/hess-22-6005-2018

Kratzert, F., Klotz, D., Shalev, G., Klambauer, G., Hochreiter, S., & Nearing, G. (2019). Towards learning universal, regional, and local hydrological behaviors via machine learning applied to large-sample datasets. *Hydrology and Earth System Sciences*, *23*(12), 5089–5110. https://doi.org/10.5194/hess-23-5089-2019

Kurganov, A., & Petrova, G. (2007). A second-order well-balanced positivity preserving central-upwind scheme for the Saint-Venant system. *Communications in Mathematical Sciences*, *5*(1), 133–160.

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. https://doi.org/10.1109/5.726791

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

Lhomme, J., Sayers, P., Gouldby, B., Samuels, P., Wills, M., & Mulet-Marti, J. (2008). Recent development and application of a rapid flood spreading method. *Proceedings of FLOODrisk 2008*.

Li, C., Han, Z., Li, Y., Li, M., Wang, W., Chen, N., & Hu, G. (2023). Data-driven and echo state network-based prediction of wave propagation behavior in dam-break flood. *Journal of Hydroinformatics*, *25*, 2235–2252. https://doi.org/10.2166/hydro.2023.035

Li, G., Zhu, L., Liu, P., & Yang, Y. (2019). Entangled transformer for image captioning. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 8928–8937.

Li, S., Yang, J., & Ansell, A. (2023). Data-driven reduced-order simulation of dam-break flows in a wetted channel with obstacles. *Ocean Engineering*, *287*, 115826. https://doi.org/10.1016/j.oceaneng.2023.115826

Li, W., Liu, C., Hu, C., Niu, C., Li, R., Li, M., Xu, Y., & Tian, L. (2024). Application of a hybrid algorithm of LSTM and transformer based on random search optimization for improving rainfall-runoff simulation. *Scientific Reports*, *14*(1), 11184. https://doi.org/10.1038/s41598-024-62127-7

Liang, Q., & Borthwick, A. G. (2009). Adaptive quadtree simulation of shallow flows with wet-dry fronts over complex topography. *Computers and Fluids*, *38*, 221–234. https://doi.org/10.1016/j.compfluid.2008.02.008

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

Liu, C., Liu, D., & Mu, L. (2022). Improved transformer model for enhanced monthly streamflow predictions of the Yangtze River. *IEEE Access*, *10*, 58240–58253. https://doi.org/10.1109/ACCESS.2022.3178521

Liu, Y. Y., Maidment, D. R., Tarboton, D. G., Zheng, X., & Wang, S. (2018). A CyberGIS integration and computation framework for high-resolution continental-scale flood inundation mapping. *JAWRA Journal of the American Water Resources Association*, *54*(4), 770–784. https://doi.org/10.1111/1752-1688.12660

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., & Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. *Proceedings of the IEEE International Conference on Computer Vision*, 9992–10002. https://doi.org/10.1109/ICCV48922.2021.00986

Liu, Z., Ning, J., Cao, Y., Wei, Y., Zhang, Z., Lin, S., & Hu, H. (2022). Video swin transformer. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2022-June*, 3192–3201. https://doi.org/10.1109/CVPR52688.2022.00320

Loshchilov, I., & Hutter, F. (2019). Decoupled weight decay regularization. https://arxiv.org/abs/1711.05101

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

Lu, J., Batra, D., Parikh, D., & Lee, S. (2019). ViLBERT: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Advances in Neural Information Processing Systems*, *32*.

Ma, H., & Fu, X. (2012). Real time prediction approach for floods caused by failure of natural dams due to overtopping. *Advances in Water Resources*, *35*, 10–19. https://doi.org/10.1016/j.advwatres.2011.08.013

Mignosa, P., Vacondio, R., Aureli, F., Dazzi, S., Ferrari, A., & Prost, F. (2018). High resolution 2D modelling of rapidly varying flows: Some case studies. *Italian Journal of Engineering Geology and Environment*, *2018*, 143–160. https://doi.org/10.4408/IJEGE.2018-01.S-13

Mitchell, T. M. (1997). *Machine learning*. McGraw-hill Science.

Morales-Hernández, M., García-Navarro, P., Burguete, J., & Brufau, P. (2013). A conservative strategy to couple 1D and 2D models for shallow water flow simulation. *Computers & Fluids*, *81*, 26–44. https://doi.org/10.1016/j.compfluid.2013.04.001

Morales-Hernández, M., Sharif, M. B., Kalyanapu, A., Ghafoor, S., Dullo, T., Gangrade, S., Kao, S.-C., Norman, M., & Evans, K. (2021). TRITON: A Multi-GPU open source 2D hydrodynamic flood model. *Environmental Modelling & Software*, *141*, 105034. https://doi.org/10.1016/j.envsoft.2021.105034

Morsy, M. M., Goodall, J. L., O'Neil, G. L., Sadler, J. M., Voce, D., Hassan, G., & Huxley, C. (2018). A cloud-based flood warning system for forecasting impacts to transportation infrastructure systems. *Environmental Modelling & Software*, *107*, 231–244. https://doi.org/10.1016/j.envsoft.2018.05.007

Mosavi, A., Ozturk, P., & Chau, K. W. (2018). Flood prediction using machine learning models: Literature review. *Water (Switzerland)*, *10*, 1536. https://doi.org/10.3390/w10111536

Nevo, S., Morin, E., Gerzi Rosenthal, A., Metzger, A., Barshai, C., Weitzner, D., Voloshin, D., Kratzert, F., Elidan, G., Dror, G., Begelman, G., Nearing, G., Shalev, G., Noga, H., Shavitt, I., Yuklea, L., Royz, M., Giladi, N., Peled Levi, N., ... Matias, Y. (2022). Flood forecasting with machine learning models in an operational framework. *Hydrology and Earth System Sciences*, *26*(15), 4013–4032. https://doi.org/10.5194/hess-26-4013-2022

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

Nourani, V., Kisi, Ö., & Komasi, M. (2011). Two hybrid artificial intelligence approaches for modeling rainfall–runoff process. *Journal of Hydrology*, *402*(1), 41–59. https://doi.org/10.1016/j.jhydrol.2011.03.002

NVIDIA. (2024). *CUDA C++ Programming Guide*.

OpenAI. (2023). *Gpt-4 technical report* (tech. rep.). arXiv preprint arXiv:2303.08774.

Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., & Tran, D. (2018). Image transformer. *Proceedings of the 35th International Conference on Machine Learning*, *80*, 4055–4064. https://proceedings.mlr.press/v80/parmar18a.html

Pianforini, M., Dazzi, S., Pilzer, A., & Vacondio, R. (2024a). A deep learning model for real-time forecasting of 2-D river flood inundation maps. *Hydrology and*

Earth System Sciences Discussions, *2024*, 1–44. https://doi.org/10.5194/hess-2024-176

Pianforini, M., Dazzi, S., Pilzer, A., & Vacondio, R. (2024b). *Floodsformer: Dam-break datasets&checkpoints* (Version v1). Zenodo. https://doi.org/10.5281/zenodo.10878385

Pianforini, M., Dazzi, S., Pilzer, A., & Vacondio, R. (2024c). *Floodsformer: Python code* (Version v2.0.0). Zenodo. https://doi.org/10.5281/zenodo.13221319

Pianforini, M., Dazzi, S., Pilzer, A., & Vacondio, R. (2024d). *Floodsformer: River flood datasets&checkpoints* (Version v1.0.1). Zenodo. https://doi.org/10.5281/zenodo.11472228

Pianforini, M., Dazzi, S., Pilzer, A., & Vacondio, R. (2024e). 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

Rasiya Koya, S., & Roy, T. (2024). Temporal fusion transformers for streamflow prediction: Value of combining attention with recurrence. *Journal of Hydrology*, *637*, 131301. https://doi.org/10.1016/j.jhydrol.2024.131301

Rizzo, C., Maranzoni, A., & D'Oria, M. (2023). Probabilistic mapping and sensitivity assessment of dam-break flood hazard. *Hydrological Sciences Journal*. https://doi.org/10.1080/02626667.2023.2174026

Robbins, H., & Monro, S. (1951). A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, *22*(3), 400–407. https://doi.org/10.1214/aoms/1177729586

Rogers, D., & Tsirkunov, V. (2011). *Costs and benefits of early warning systems: Global assessment report on disaster risk reduction* (tech. rep.). The World Bank.

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. https://doi.org/10.1007/978-3-319-24574-4_28

Sanders, B. F., Schubert, J. E., & Detwiler, R. L. (2010). ParBreZo: A parallel, unstructured grid, Godunov-type, shallow-water code for high-resolution flood inundation modeling at the regional scale. *Advances in Water Resources*, *33*(12), 1456–1467. https://doi.org/10.1016/j.advwatres.2010.07.007

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

Seleem, O., Ayzel, G., Bronstert, A., & Heistermann, M. (2023). Transferability of data-driven models to predict urban pluvial flood water depth in Berlin, Germany. *Natural Hazards and Earth System Sciences*, *23*(2), 809–822. https://doi.org/10.5194/nhess-23-809-2023

Sharifian, M. K., Kesserwani, G., Chowdhury, A. A., Neal, J., & Bates, P. (2023). LISFLOOD-FP 8.1: New GPU-accelerated solvers for faster fluvial/pluvial flood simulations. *Geoscientific Model Development*, *16*(9), 2391–2413. https://doi.org/10.5194/gmd-16-2391-2023

Smith, P., Pappenberger, F., Wetterhall, F., Thielen del Pozo, J., Krzeminski, B., Salamon, P., Muraro, D., Kalas, M., & Baugh, C. (2016). On the operational implementation of the European Flood Awareness System (EFAS). In T. E. Adams & T. C. Pagano (Eds.), *Flood forecasting* (pp. 313–348). Academic Press. https://doi.org/10.1016/B978-0-12-801884-2.00011-6

Sun, C., Myers, A., Vondrick, C., Murphy, K., & Schmid, C. (2019). Videobert: A joint model for video and language representation learning. *Proceedings of the IEEE International Conference on Computer Vision*, 7463–7472. https://doi.org/10.1109/ICCV.2019.00756

Talei, A., & Chua, L. H. (2012). Influence of lag time on event-based rainfall–runoff modeling using the data driven approach. *Journal of Hydrology*, *438-439*, 223–233. https://doi.org/10.1016/j.jhydrol.2012.03.027

Tang, Z., Zhang, J., Hu, M., Ning, Z., Shi, J., Zhai, R., Liu, C., Zhang, J., & Wang, G. (2024). Improving streamflow forecasting in semi-arid basins by combining data segmentation and attention-based deep learning. *Journal of Hydrology*, *643*, 131923. https://doi.org/10.1016/j.jhydrol.2024.131923

Tayfur, G., Singh, V. P., Moramarco, T., & Barbetta, S. (2018). Flood hydrograph prediction using machine learning methods. *Water*, *10*(8). https://doi.org/10.3390/w10080968

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. *Environmental Modelling and Software*, *90*, 201–216. https://doi.org/10.1016/j.envsoft.2017.01.006

Teng, J., Vaze, J., Dutta, D., & Marvanek, S. (2015). Rapid inundation modelling in large floodplains using LiDAR DEM. *Water Resources Management*, *29*, 2619–2636. https://doi.org/10.1007/s11269-015-0960-8

Testa, G., Zuccalà, D., Alcrudo, F., Mulet, J., & Soares-Frazão, S. (2007). Flash flood flow experiment in a simplified urban district. *Journal of Hydraulic Research*, *45*, 37–44. https://doi.org/10.1080/00221686.2007.9521831

Toro, E. F. (1999). *Riemann solvers and numerical methods for fluid dynamics*. Springer.

Toro, E. F. (2001). *Shock capturing methods for free surface shallow water flows*. Wiley.

Turchetto, M., Dal Palù, A., & Vacondio, R. (2020). A general design for a scalable MPI-GPU multi-resolution 2D numerical solver. *IEEE Transactions on Parallel and Distributed Systems*, *31*, 1036–1047. https://doi.org/10.1109/TPDS.2019.2961909

Vacondio, R., Dal Palù, A., & Mignosa, P. (2014). GPU-enhanced finite volume shallow water solver for fast flood simulations. *Environmental Modelling and Software*, *57*, 60–75. https://doi.org/10.1016/j.envsoft.2014.02.003

Vacondio, R., Aureli, F., Ferrari, A., Mignosa, P., & Dal Palù, A. (2016). Simulation of the January 2014 flood on the Secchia River using a fast and high-resolution 2D parallel shallow-water numerical scheme. *Natural Hazards*, *80*, 103–125. https://doi.org/10.1007/s11069-015-1959-4

Vacondio, R., Dal Palù, A., Ferrari, A., Mignosa, P., Aureli, F., & Dazzi, S. (2017). A non-uniform efficient grid type for GPU-parallel shallow water equations models. *Environmental Modelling and Software*, *88*, 119–137. https://doi.org/10.1016/j.envsoft.2016.11.012

Van, S. P., Le, H. M., Thanh, D. V., Dang, T. D., Loc, H. H., & Anh, D. T. (2020). Deep learning convolutional neural network in rainfall–runoff modelling. *Journal of Hydroinformatics*, *22*(3), 541–561. https://doi.org/10.2166/hydro.2020.095

Van Der Knijff, J. M., Younis, J., & De Roo, A. P. J. (2010). LISFLOOD: A GIS-based distributed model for river basin scale water balance and flood simulation. *International Journal of Geographical Information Science*, *24*(2), 189–212. https://doi.org/10.1080/13658810802549154

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*.

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). https://doi.org/10.1029/2009WR008475

Wallemacq, P., & House, R. (2018). *Economic losses, poverty & disasters: 1998-2017*. Centre for Research on the Epidemiology of Disasters (CRED).

Wang, H., Qin, H., Liu, G., Huang, S., Qu, Y., Qi, X., & Zhang, Y. (2024). Hierarchical attention network for short-term runoff forecasting. *Journal of Hydrology*, *638*, 131549. https://doi.org/10.1016/j.jhydrol.2024.131549

Wang, J.-H., Lin, G.-F., Chang, M.-J., Huang, I.-H., & Chen, Y.-R. (2019). Real-time water-level forecasting using dilated causal convolutional neural networks.

Water resources management, *33*, 3759–3780. https://doi.org/10.1007/s11269-019-02342-4

Wang, W.-c., Tian, W.-c., Hu, X.-x., Hong, Y.-h., Chai, F.-x., & Xu, D.-m. (2024). DTTR: encoding and decoding monthly runoff prediction model based on deep temporal attention convolution and multimodal fusion. *Journal of Hydrology*, *643*, 131996. https://doi.org/10.1016/j.jhydrol.2024.131996

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*, *634*, 130993. https://doi.org/10.1016/J.JHYDROL.2024.130993

Wing, O. E. J., Bates, P. D., Neal, J. C., Sampson, C. C., Smith, A. M., Quinn, N., Shustikova, I., Domeneghetti, A., Gilles, D. W., Goska, R., & Krajewski, W. F. (2019). A new automated method for improved flood defense representation in large-scale hydraulic models. *Water Resources Research*, *55*(12), 11007–11034. https://doi.org/10.1029/2019WR025957

Xia, X., Liang, Q., & Ming, X. (2019). A full-scale fluvial flood modelling framework based on a high-performance integrated hydrodynamic modelling system (HiPIMS). *Advances in Water Resources*, *132*, 103392. https://doi.org/10.1016/j.advwatres.2019.103392

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

Xu, Y., Lin, K., Hu, C., Wang, S., Wu, Q., Zhang, L., & Ran, G. (2023). Deep transfer learning based on transformer for flood forecasting in data-sparse basins. *Journal of Hydrology*, *625*, 129956. https://doi.org/10.1016/j.jhydrol.2023.129956

Ye, X., & Bilodeau, G. A. (2023). Video prediction by efficient transformers. *Image and Vision Computing*, *130*. https://doi.org/10.1016/j.imavis.2022.104612

Yin, H., Guo, Z., Zhang, X., Chen, J., & Zhang, Y. (2022). RR-Former: Rainfall-runoff modeling based on Transformer. *Journal of Hydrology*, *609*, 127781. https://doi.org/10.1016/j.jhydrol.2022.127781

Yin, H., Zhu, W., Zhang, X., Xing, Y., Xia, R., Liu, J., & Zhang, Y. (2023). Runoff predictions in new-gauged basins using two transformer-based models. *Journal of Hydrology*, *622*, 129684. https://doi.org/10.1016/j.jhydrol.2023.129684

Zheng, X., & Zheng, M. (2024). Prediction of urban flood inundation using bayesian convolutional neural networks. *Stochastic Environmental Research and Risk Assessment*, 1–16. https://doi.org/10.1007/s00477-024-02814-z

Zhou, Y., Wu, W., Nathan, R., & Wang, Q. J. (2022). Deep learning-based rapid flood inundation modeling for flat floodplains with complex flow paths. *Water Resources Research*, *58*. https://doi.org/10.1029/2022WR033214

Zounemat-Kermani, M., Matta, E., Cominola, A., Xia, X., Zhang, Q., Liang, Q., & Hinkelmann, R. (2020). Neurocomputing in surface water hydrology and hydraulics: A review of two decades retrospective, current status and future prospects. *Journal of Hydrology*, *588*, 125085. https://doi.org/10.1016/j.jhydrol.2020.125085