Rapid generation of probabilistic inundation forecasts by utilizing cloud computing and deep learning

HydroLogic

UNIVERSITY OF TWENTE.

Author: Fedde Hop
Date: January 2023

Supervisors: Ralf Linneman (HL)
Bram Schnitzler (HL)
Anouk Bomers (UT)
Martijn Booij (UT)

Preface

During my studies I have developed an interest for programming and machine learning. I was very happy to have the opportunity to use both of these interests during my master thesis, in combination with a newly developed interest for cloud computing. Together with the University and HydroLogic a research objective was set up that combines cloud computing and machine learning for inundation predictions, a very relevant topic during times of climate change.

The research presented in this thesis was conducted over a period of six months. Firstly, I am grateful for the guidance and support of my supervisors at HydroLogic during my master thesis. Their daily help and dedication kept me on track, and their expertise and wisdom helped me navigate my research. I appreciate the time and effort they invested in me and the valuable contributions they have brought to this research. Thank you Ralf Linneman and Bram Schnitzler. Secondly I would like to thank my supervisors from the university of Twente for their invaluable feedback and helpful meetings throughout my master thesis journey. Their guidance greatly helped shape my research and their insights and suggestions greatly improved the final outcome. Thank you, Anouk Bomers and Martijn Booij. Lastly I would like to express my gratitude to my family and friends for their support and companionship. Thank you for being there for me and for the good times we shared together.

I hope you will enjoy this report,

Fedde Hop

January 2023

Summary

Heavy rainfall events are occurring more frequently due to climate change, and can lead to inundation which poses risks for society. Inundation forecasts are crucial to alleviate these risks. The forecasts are used as a warning system and can help making decisions on the measures to take to reduce the inundation hazards. The most important variable for predicting inundation is the rainfall, however the uncertainty in rainfall forecasts is usually high. To include the uncertainty of the rainfall forecasts in the inundation forecasts, probabilistic inundation forecasts are made. These forecasts consider an ensemble of rainfall forecasts, and predict the inundation for each ensemble member. Based on the number of ensemble members where inundation occurs the probabilities of inundating can be calculated, resulting in a probabilistic inundation forecast.

Conventional hydraulic models are too slow to generate probabilistic inundation forecasts, with total computation times generally exceeding one hour. This is because inundation for each of the ensemble members of the rainfall forecast has to be simulated. This study applies and analyses two methods of making probabilistic inundation forecasts fast enough to be used operationally. The study area is polder de Tol, located north-west of Utrecht in the Netherlands. The size of the area is about 12.5 km^2, mostly consisting of rural landscape. For this area, a 1D2D hydraulic model is setup and calibrated such that inundation depths during pluvial flooding can be predicted.

The first method to rapidly generate probabilistic inundation forecasts utilises cloud computing such that many simulations can be executed simultaneously. This significantly reduces the computational time required for making probabilistic inundation forecasts, since simulations for multiple ensemble members can be executed in parallel. In this study, cloud computing has proven to be a feasible solution, reducing the time required for probabilistic inundation forecasts to the computation time of a single hydraulic simulation. The costs of utilising cloud computing are dependent on the model complexity and the number simulations that are required. For the model used in this study, the costs of generating a probabilistic inundation forecast consisting of 50 ensemble members at a 10 meter resolution is 0.40 euros. The methodology used for this study can also be applied to other study areas, and is scalable when more simultaneous simulations are required.

The second method is utilising deep learning by training a neural network to make inundation forecasts. The neural network trained for this study can predict inundation depths with a 10 meter resolution at 12 time steps. The network is trained on 1600 hydraulic simulations, and can accurately predict inundation depth progression over time. The accuracy of the neural network's inundation forecasts is assessed by comparing the neural network predictions to hydraulic model forecasts for 200 different rainfall events. For each of these 200 events the mean absolute error is calculated, which is between $5.7 * 10^{-5}$ and 0.07 cm, with an average of 0.01 cm. The network also performs very well in generating probabilistic inundation forecasts: For 99.6% of the predictions the neural network inundation probability is within 2% of the probability predicted by the hydraulic model. The neural network can generate a probabilistic inundation forecast within seconds. The neural network architecture used for this study is also expected to perform well in other study areas as long as the network is trained on data for that study area.

Both methods of providing probabilistic inundation forecasts within an operational time frame have proven successful. Which method is preferred depends on the application. The neural network can make forecasts much faster, and at a negligible cost. However to do so a large number of training simulations have to be performed beforehand, and this has to be re-done when changes in the hydraulic model are required. Utilising cloud computing is more expensive and slower, but the hydraulic model used to make the inundation predictions can be changed whenever required.

There are several suggestions for future research to improve the accuracy and applicability of the neural network. These suggestions include: studying the impact of additional input variables on the network, evaluating the network's performance in areas with diverse topography and urban environments, and creating a generalised neural network that can be applied to any area without specific training. Also, it is recommended to evaluate different network architectures and to study how the amount of training data affects the accuracy of predictions.

Future research opportunities to improve the accuracy and applicability of the neural network have been identified. These include: investigating the effects of additional input variables on the network, assessing the network's performance in diverse topographical and urban environments, developing a generalizable neural network that can be applied without training for a specific area, evaluating various network architectures, and examining the impact of the quantity of training data on prediction accuracy.

Contents

1 Introduction

1.1 Background

Inundation poses a significant risk and is becoming an even greater threat due to climate change and urbanization (Trenberth, 2011). As an example, in 2011 a cloudburst brought 150 mm of rain to Copenhagen within three hours, which resulted in flooding in cellars, streets, and major roads (CPH Post, 2011). This event serves as a demonstration of the potential damage and destruction that can occur from heavy rainfall, with the damage estimated at 1 billion euros (Gerdes, 2012). Coupled with the increasing likelihood of such events due to climate change, such an example highlights the importance of being able to accurately predict inundation for these types of events.

It is not feasible to prevent all inundation due to heavy rainfall. However, one way to minimize the damage caused by such floods is through timely forecasting of inundation, allowing for effective measures to be taken before and during the event (Moel et al., 2014). Examples of such measures are the installation of emergency pumps, changes to weir levels and evacuation of inhabitants. One of the difficulties in making these inundation forecasts is the uncertainty in the rainfall forecasts. To include these uncertainties in the inundation forecasts, probabilistic inundation forecasts can be performed (Zarzar et al., 2018). The study focusses on making rapid probabilistic inundation forecasts for pluvial flooding, such that these forecasts can be used operationally.

1.2 State of the art

Numerical simulations play a key role in inundation forecasts due to heavy rainfall (Mei et al., 2020; Goodarzi et al., 2019). Many studies have shown the effectiveness of physically based hydraulic models to predict inundation depths (Shahapure et al., 2010; Seyoum et al., 2012). To make these forecasts, the models numerically solve a set of partial differential equations to simulate the flow of water (Brutsaert, 2006). Factors that contribute to the accuracy of the forecasts are the quality of input data, and on the spatial and temporal scale of the hydraulic model: with smaller steps in time and space, the accuracy increases. However, these fine steps come with large computational costs, and this computational cost increases as the demand for accuracy increases (Wang et al., 2019). For this reason, operational inundation forecasting has yet to become the standard (Wu et al., 2020).

Using probabilistic inundation maps based on multiple ensemble members of the weather forecast increases the accuracy of modelling inundation by considering uncertainties in the forecasts (Georgas et al., 2016; Gomez et al., 2019; Duan et al., 2007; Di Baldassarre et al., 2010). However this comes at a cost of the computation time, since more simulations have to be performed to get to a probabilistic inundation forecast. This makes operational probabilistic inundation forecasting with numerical models even more of a challenge because of the associated computational costs.

By utilising cloud computing, multiple model runs can be performed simultaneously on different computing devices. This way, probabilistic forecasts can be made in the same computational time as single model forecasts. With cloud computing, the hardware is hired from cloud providers. Utilising cloud computing will reduce computational time since not all calculations have to be performed at the local machine, but it comes with the cost of hiring external machines (Jadeja and Modi, 2012).

Another method to reduce computational time is to apply machine learning methods to inundation forecasts. The advantage of machine learning methods is that they reduce the computational cost of making forecasts, since no physically based equations have to be solved. Machine learning algorithms build a model based on training data, in order to make predictions or decisions without being explicitly programmed to do so. Deep learning is a subset of machine learning in which artificial neural networks, algorithms inspired by the structure and function of the human brain, are used to model and solve complex problems. Real world data about inundation is often not available in large quantities, therefore hydraulic simulations are used to function as the training data. Methods that have been applied for inundation forecasts are random forest (Hou et al., 2021), support vector machines (Bermúdez et al., 2019) and neural networks (Bentivoglio et al., 2021). But there is a lack of comparisons between methods. In the work of Kabir et al. (2020) it has been shown how a neural network outperforms a support vector machine for their case study. In recent years, the application of neural networks has been most popular in research and is seen as a promising tool to speed up inundation modelling, due to the ability to learn complex non-linear patterns (Bentivoglio et al., 2021). However neural networks have only been applied in a handful of cases, with a few of them being for pluvial inundation. Some of these predict maximum inundation depths (Berkhahn et al., 2019; Guo et al., 2021), while the others are also capable of predicting inundation at different time steps (Chang et al., 2018; Chang et al., 2014; Kilsdonk et al., 2022). The accuracy of the inundation forecasts is promising for all five studies, but there is significant room

for improvement. Different network configurations are used in literature, but no comparisons have been made between different setups to find which setup is most effective. In the reviewed literature, neural networks have never been applied for making probabilistic inundation forecasts.

1.3 Research gap

Two research gaps were identified through the literature review that will be addressed in this study. The first is in making probabilistic inundation forecasts fast enough to be used in an operational setting. Since probabilistic forecasts require significantly heavier computation than deterministic forecasts, numerical models are often too slow and therefore cannot be used. Two potential solutions are running numerical models parallel with the help of cloud computing, and applying deep learning methods to speed up the forecasting process. These two methods have never been compared in the sense of accuracy, computational time and costs. The second research gap exists in the application of deep learning to pluvial inundation. While some research in this field has been done, the number of applications is small. Current applications in literature provide promising results, but there are still systematic errors in the inundation forecasts.

1.4 Objective and research questions

The objective of this research is to test and compare two different methods of making probabilistic inundation forecasts accurately and fast enough for operational use. The maximum computation time for an operational probabilistic inundation forecast is set at 1 hour for this study. The first method is utilising cloud computing to execute simulations in parallel such that the full probabilistic forecast can be made in a much shorter time. The second is training a neural network (deep learning model) that can predict inundation depths, which can then be used to generate probabilistic inundation forecasts. For both of these methods the goal is to speed up the probabilistic inundation forecasts while maintaining accuracy. This can then be used to aid local water managers when extreme rainfall is expected. This will be investigated for pluvial inundation in polder de Tol (see section 1.5 for more information).

Which of the two options is best depends on multiple factors. Most important is the accuracy of the neural network. If accuracy cannot be guaranteed, the results cannot reliably be used in making decisions. Another important factor is the computation time. For the neural network, forecasts can be made in less than a second, while cloud computing will be dependent on the computation time of a single deterministic simulation. Lastly, the costs of making forecasts are important. Cloud computing will cost a fixed amount per ensemble forecasts, while the neural network needs an initial investment for training data generation. These are all important considerations to make, and for each of them quantifications need to be made in order to make an educated decision on which method is favoured.

To test and compare the two different methods of making probabilistic inundation forecasts, multiple research questions are formulated. The first step is to set up a hydraulic model that can predict inundation levels due to extreme precipitation events. This model needs to be calibrated and validated for predicting water levels and arrival times in the study area.

RQ1: How fast and accurately can a hydraulic model replicate historic water levels and discharges during extreme precipitation events in polder de Tol?

With this hydraulic model, inundation forecasts for single events can be made. Together, multiple simulations based on ensemble members of the weather forecast can be converted to a probabilistic inundation map. Performing each simulation on local hardware would be too slow to provide results during an extreme event, and therefore cloud computing will be utilized to be able to automatically perform simulations for each ensemble member in parallel. Interesting to know is how long it takes to make a probabilistic inundation forecast, and what costs would be associated with this. To answer these questions, an automated cloud workflow for making probabilistic inundation forecasts based on ensemble weather forecasts has to be made. When this is done, the right resource setup for efficient computations needs to be determined.

RQ2: What are the cost and computation times for generating probabilistic inundation forecasts using cloud computing?

The next step is to set up a neural network to predict the results of the hydraulic model. The network will be trained and validated on the generated data from the hydraulic model. It is interesting to know how similar the outcome of the neural network will be to the inundation forecasts of the hydraulic model.

RQ3: How accurate can a neural network replicate the deterministic inundation depths and inundation probabilities of the hydraulic model?

With both the cloud computing workflow and the neural network finished, a comparison between the two methods can be made. The two methods will be compared in their performance on generating probabilistic inundation forecasts. Their performance is based on the accuracy of the forecasts, the computation time, and the costs associated with making forecasts.

RQ4: How do inundation predictions made with cloud computing and the neural network compare in terms of computation time, accuracy and costs for generating probabilistic inundation forecasts?

With the answers to these research questions, conclusions can be drawn on the advantages and disadvantages for both methods of achieving operational probabilistic inundation forecasts. This information can help deciding which method is most suitable based on the use case.

1.5 Study area

The study area is polder de Tol (figure 1.1), located north-west of Utrecht in the Netherlands. The size of the area is about 12.5 km^2, mostly consisting of rural landscape. The area is characterised by meadows, a high density of ditches, and soil mostly consisting of peat. Polder de Tol is mainly used for agricultural purposes. In the north-west some urban area is present, which is the village of Kockengen. In total, around 3.200 inhabitants live in the study area. Polder de Tol is interesting for multiple reasons. In terms of water management, the study area is a relatively closed system that runs off to pumping station de Tol. The area experiences little influences by processes outside of the model domain. This is helpful when validating the model, since most of the water that exits the study area discharges through the pumping station, for which discharge data is available. The study area is well-known and understood by experts as a result of numerous studies that have been conducted by waterboard Hoogheemraadschap de Stichtse Rijnlanden. Lastly the area is representative for polders in west and north of the Netherlands.

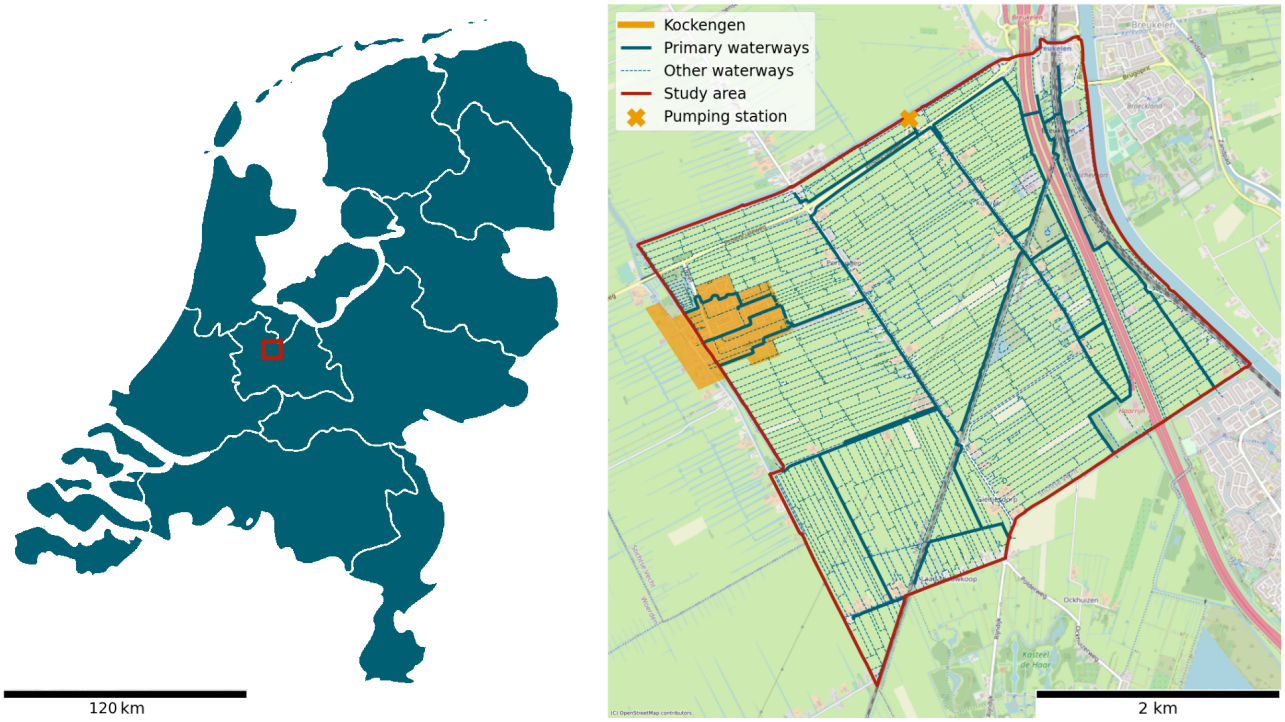


Figure 1.1: *The study area of polder de Tol. The left figure shows the location of polder de Tol within the Netherlands, the right figure shows the extent of the study area, the waterways, the pumping station and the village of Kockengen.*

2 Methodology

In this chapter, the methods that will be applied to answer each of the research question are addressed. The methodology for setting up, calibrating, and validating the hydraulic model is detailed in section 2.1. This model is used to make probabilistic inundation forecasts by simulating 50 ensemble members of the rainfall forecasts. To facilitate the heavy computation required, cloud computing is utilized to speed up the total computation time for creating an ensemble inundation forecast, and the process is explained in section 2.2. The methodology for training and testing a neural network to predict inundation depths and probabilities is outlined in section 2.3.

2.1 Hydraulic model

A hydraulic model is set up to simulate inundation resulting from heavy rainfall. In this section the methodology for setting up and calibrating the model is explained. The resulting model is used to answer research question 1: How fast and accurately can a hydraulic model replicate historic water levels and discharges during extreme precipitation events in polder de Tol?".

For this study, a 1D-2D hydraulic model with a coupled rainfall runoff module is set up. Each of the model components will be explained below. The exchange of information between the components is visualized in figure 2.1. The model is created with the D-HyDAMO Python package which is used to set up a model for the D-HYDRO software using Python scripts (De Graaff et al., 2022). Because of this, all model parameters can be easily varied, which makes calibrating the model easier. Also, a model can be automatically recreated for a different study area, provided that the correct input data is available.

A 1D2D coupled model is used such that the waterways can be modelled in 1D by using cross sectional information. It is possible to do this since the water has a predetermined flow path in the waterways (Henonin et al., 2013). If the waterways would have to be included in the 2D model, a very fine grid size in the waterways would be required, which would significantly increase computational costs. The rainfall runoff component is included such that the rainfall runoff processes are taken into account in the model (Deltares, 2022c). As input for the model, data about the study area is required. The data used can be seen in table 2.1.

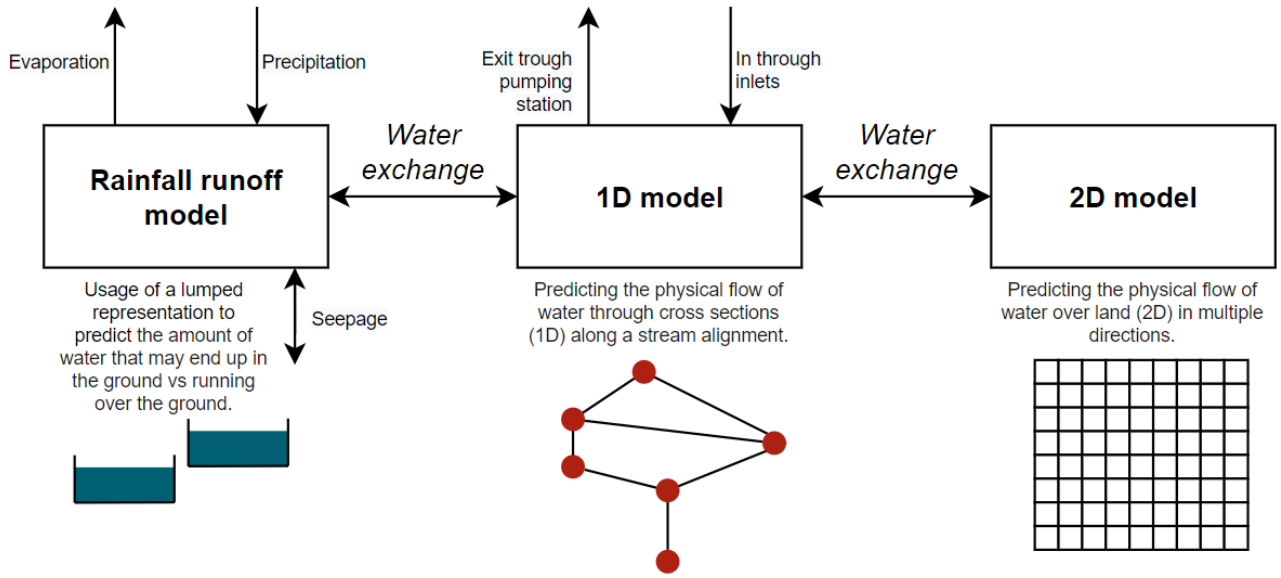


Figure 2.1: *Visualisation of the different model components and their interaction.*

2.1.1 Rainfall runoff model

The rainfall runoff model (RR model) is used for the simulation of the rainfall-runoff processes. It is capable of simulating the hydrological processes in rural and urban areas during wet and dry conditions (Deltares, 2022c). The entire area is represented by 16 catchments, which are based on the fixed drainage level areas (in Dutch: Peilgebieden). Each catchment has four associated "buckets" with it: paved, unpaved, open water and

Table 2.1: *Input data for the hydraulic model.*

Data	Source	Dates	Notes
Digital elevation map	(Actueel Hoogtebestand Nederland, 2020)	2020	0.5 meter resolution
Land use	(Stichting Toegepast Onderzoek Waterbeheer, 2020)	2020	0.5 meters resolution
Soil data	(BRO, 2018)	2018	soil type assumed uniform over study area
Seepage	OWASIS	2018	Seepage assumed uniform over study area
Rainfall	(MeteoBase, 2022)	2010-2022	1000 meters resolution
Evaporation	(MeteoBase, 2022)	2010-2022	1000 meters resolution
Structure data	(HDSR, 2022)	2022	-
Waterway data	(HDSR, 2022)	2022	-

greenhouses. The catchments and buckets can be seen in figure 2.3a. Each bucket contains a certain amount of water, which is calculated as the balance of all the ingoing and outgoing flows. The characteristics of the catchments are used to model the hydrology within a catchment. The processes that are considered are shown in figure 2.2. Rainfall is based on external forcing data, which can be both historical data or hypothetical events. The soil type is assumed to be peat for the entire study area. Seepage is based on the predictions of the OWASIS model, which predicts little to no seepage for the study area (KNW, 2020). The rainfall runoff model exchanges water with the 1D component of the model, water is exchanged at the RR links represented in figure 2.3a. Depending on the ground water level and the water level in the waterways, water can be exchanged with the 1D model in both directions as illustrated in figure 2.1.

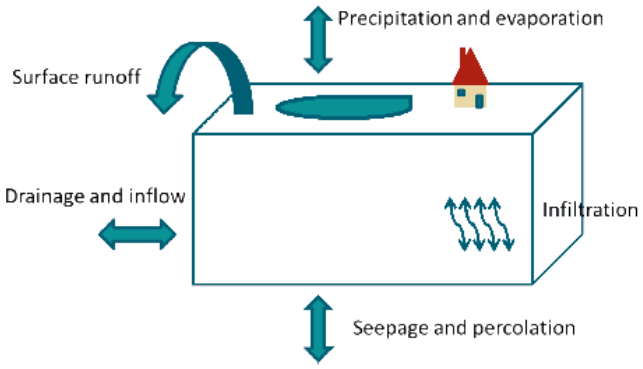
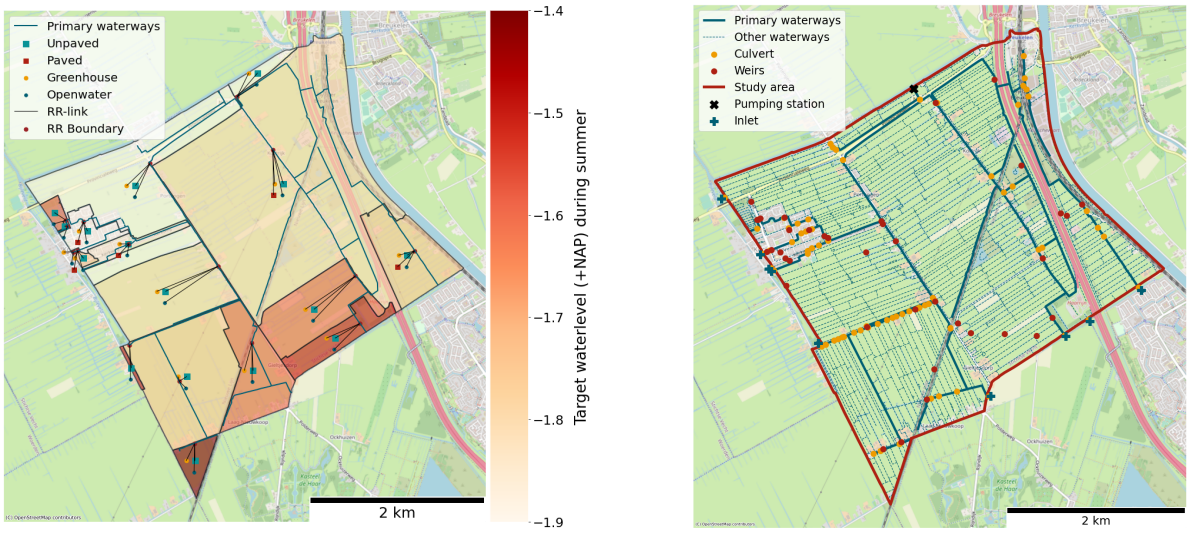


Figure 2.2: *Visualisation of processes considered in the rainfall runoff model (Deltares, 2022c).*

2.1.2 1D model

The 1D component of the model simulates the channel flow in the study area (Deltares, 2022b). For the primary waterways (see figure 1.1) profile measurements are available. For the non-primary waterways (figure 1.1) no data is available, and therefore for these waterways a default cross section is applied. The dimensions of these waterways is determined in the calibration. The roughness of the 1D waterways is also determined in the calibration in section 3.1. Structures that are included in the 1D schematization are culverts, weirs, dams, inlets and the pumping station. Data about the exact operation of the weirs is not available and the operation during heavy rainfall is often done manually by local water managers. In the model, the target water levels for the associated fixed drainage level area are used as operating levels for the weirs. There is a distinction between summer and winter operating water levels. An illustration of the model representation of the area can be seen in figure 2.3. 1D calculation nodes are placed every 20 meters, and 1 meter upstream and downstream of each structure.



(a) RR nodes and target water level (m + NAP) *(b) Structures and waterways in study area*

Figure 2.3: *Model representation of the study area*

Water inlets are located at the boundary of the study area. These inlets are used to compensate for the water loss due to evaporation during low rainfall periods, by letting in water from nearby areas. In the model, these are also included. They let in water when the water levels are below the target water levels and are closed when the water levels are higher then the target water levels. Since the area upstream of the study area is not considered, the water storage upstream is assumed to be infinite, so there always is water available for the inlets. During heavy rainfall events, the inlets will be closed most of the time because the water levels will be higher than the target water levels.

Downstream of the pumping station there is a fixed water level boundary condition. This way, all water that is discharged through the pumping station will exit the hydraulic model. Because of this fixed water level boundary condition, the full discharge capacity of the pumping station is always assumed to be available. This might not be realistic in all situations of extreme rainfall, since it could be that the downstream water levels are higher due to the rainfall and thus the pump would have to be shutdown. This is an important limitation of the current model.

2.1.3 2D model

The 2D model is used to simulate overland flow, where there is no predetermined flow path (Deltares, 2022b). A structured rectangular grid with a resolution of 10 meters is used. A structured rectangular grid is used since there are many 1D waterways in the model. Refining around each 1D waterways would result in a large part of the grid having to be refined. Using a rectangular grid at a higher resolution is simpler to work with, and provides a similar computation time and resolution.

The roughness is dependent on the land use per grid cell according to relations between land use and Manning roughness from literature (Papaioannou et al., 2018). The 2D roughness values are not considered in the calibration since there is no data available that can be used to calibrate the overland flow.

The 1D and 2D model can exchange water at 1D2D links. At every computational point of a 1D waterway, a link is placed between the waterway and the nearest grid cell. When the water level in the waterway is higher than that of the 2D grid cell, the water will flow into the 2D model at this grid cell (or the other way around). The number of links present is a trade off between accuracy and computation time. More links require more 1D calculation points and a finer 2D resolution. This increases accuracy, however also increases computational cost. A visualisation of the 1D2D links can be seen in figure 2.4.

Storage of water in waterways is handled by the 1D model. For this reason, the bed level of the 2D model uses the average bed level outside of the waterway, such that the water storage inside waterways is not present in both the 1D and 2D component. When the water level exceeds the surface level, there is a slight overlap in water storage. In that case, the water level is stored both inside the 1D model and in the 2D model which results in an underestimation of the inundation depths, however the impact is small (Deltares, 2022b).

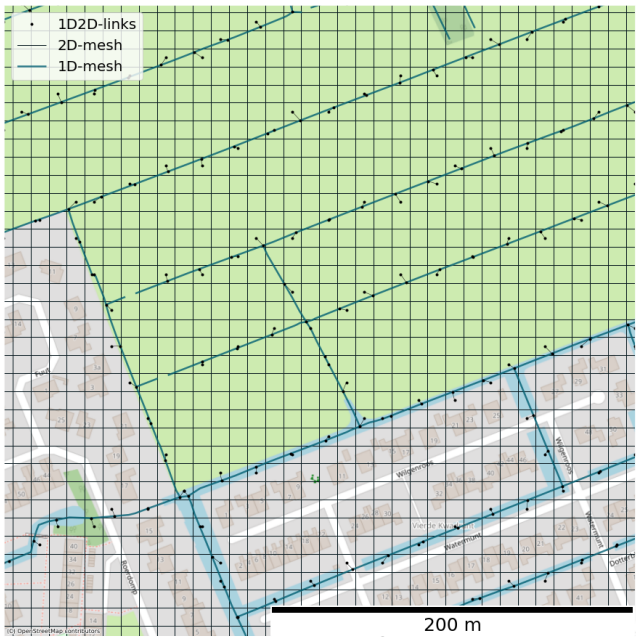


Figure 2.4: *Illustration of the coupling between the 1D and 2D model. In this example the grid size is 10 meters, and the 1D nodes are placed every 20 meters. Each 1D computation node is linked to the closest grid cell.*

2.1.4 Calibration

Within the model calibration, the goal is to determine values for uncertain model parameters such that the model output is as similar as possible to that of real life events. Observations are available for the discharge through the pumping station, and for water levels at various locations.

The first objective is ensuring the water balance of the model is correct. There is only one location where water can leave the 1D and 2D model, which is at the pumping station. There are also other ways for water to leave the system in the RR model (e.g. seepage), but for these processes no validation data is available. The calibration of the water balance is therefore done by comparing the predicted discharge through the pumping station to observations. The second objective is ensuring the water levels in the waterways are in line with observations. 16 measurement stations are located in the study area (figure 3.3b), and the observed water levels at these locations are compared to the water levels predicted by the model.

Six uncertain model parameters are modified to get the model results closer to the observations, while parameter values remain within realistic ranges. These six parameters have been chosen based on an exploratory sensitivity analysis and expert judgement. The parameter values are the same for the entire study area, there are no local deviations. The model parameters that are modified during the calibration are: channel roughness, drainage resistance, surface runoff resistance, land storage and the dimensions of the channels for which no cross sectional data is available. More information on these calibration parameters can be found in table2.2. Ideally, the model would also be calibrated with inundation patterns that occurred, however since there is no data available this is not possible.

To reach the calibration objectives, the mean absolute error (MAE, equation 1 and 2) for the water levels and the pumping station discharge are minimized, and the Nash–Sutcliffe model efficiency coefficient (NSE, equation 3) for the pumping station discharge is maximized. The mean absolute error is good to evaluate the overall fit, and the NSE is popular for evaluating the performance of hydrological models (Jackson et al., 2019). The plotted water level and discharge over time for the best performing setups are manually checked, as advised by Jackson et al. (2019).

Table 2.2: *Model parameters that are changed during the calibration. Min and max value indicate the ranges of values that the parameters could take during the Monte Carlo calibration.*

Parameter	Description	Min value	Max value	Unit	Source
Channel width	Width of the channels where no cross section data is available	1.5	4	m	Satellite imagery
Channel height	Height of the channels where no cross section data is available	1	1.5	m	Expert judgement
1D channel roughness	The resistance against flow of water in channels (Strickler)	8	40	$\frac{m^{1/3}}{s}$	(Veldman, 2006)
Drainage resistances	The ability to resist flow of water underground. Multiplication factor with values of 30, 200 and 10000 days for soil layer 1, 2 and 3 respectively	0.3	2	–	(Deltares, 2022c)
Land storage	Amount of water that can be stored on unpaved land	5	20	mm	Expert judgement
Surface runoff resistance	Resistance against surface flow of water	0.3	2	day	Expert judgement

$$MAE_{dis} = \frac{1}{n}\sum_{i=1}^{n} \mid dis_{pred,t} - dis_{obs,t} \mid \qquad (1) \qquad MAE_{wl} = \frac{1}{n}\sum_{i=1}^{n} \mid wl_{pred,t} - wl_{obs,t} \mid \qquad (2)$$

$$NSE = 1 - \frac{\sum_{i=1}^{n}(dis_{obs,t} - x_{pred,t})^2}{\sum_{i=1}^{n}(dis_{obs,t} - dis_{mean})^2} \qquad (3)$$

Where

$dis_{pred,t}$ is the cumulative discharge at time step t according to the model.

$dis_{obs,t}$ is the cumulative discharge at time step t according to the observations.

dis_{mean} is the mean of the cumulative discharge.

$wl_{pred,t}$ is the water level at time step t according to the model.

$wl_{obs,t}$ is the water level at time step t according to the observations.

n is the total number of time steps that are considered.

To perform the calibration, 5 historic events are chosen. These events are listed in table 2.3 and the rainfall during these events is shown in figure 2.5. Daily potential evapotranspiration data and hourly grid-based data for the rainfall between 2010 and 2022 is available (MeteoBase, 2022). The ten heaviest events are selected by sorting on total rainfall in 24 hours. Pumping station and water level data are available through Hoogheemraadschap De Stichtse Rijnlanden (HDSR, 2022). There are quite some data gaps, meaning that only five of the ten heaviest rainfall events can be used for the calibration and validation. For the events that are used during the calibration, the available data is not the same for each event. For this reason, some events are only used for calibrating the water balance, and some are only used for calibrating the water levels. The locations for which water level observations are available also differ between the events.

Of the ten heaviest rainfall events, 1 occurred in June, 4 in July, 2 in August, 2 in September and 1 in October. So most events occurred during summer for which conditions are more dry than during winter. Because of this, the decision is made to use a dry period as a warm up period, since this is most representative for the initial conditions of these events. The warm up period is 4 weeks, since after 4 weeks the model reaches a stable state. The simulation periods for the events are 13 days, starting 7 days before the day the heavy rainfall occurred, in order to include the effect of rainfall in the days leading to the event.

To find a well performing set of model parameters, a Monte Carlo simulation is carried out. For the six

Table 2.3: *Events used for calibration and validation of water balance and water levels. Total rainfall is the total amount of rain that fell within a 5 day period. Events are also visualised in figure 2.5.*

	Date (Y-M-D)	Total rainfall	Used for	Note
1	2013-9-9	55 mm	Calibration water balance	Rainfall relatively spread out
2	2016-6-22	50 mm	Calibration water balance and water levels	Majority rainfall within 6 hours
3	2010-8-26	115 mm	Calibration water balance and water levels	Rainfall is more spread out, with heavy rainfall in first 12 hours
4	2014-7-27	113 mm	Calibration water levels	All rain falls within 12 hours, no pumping data available
5	2013-10-12	91 mm	Validation water balance and water levels	Majority of rainfall within 24 hours

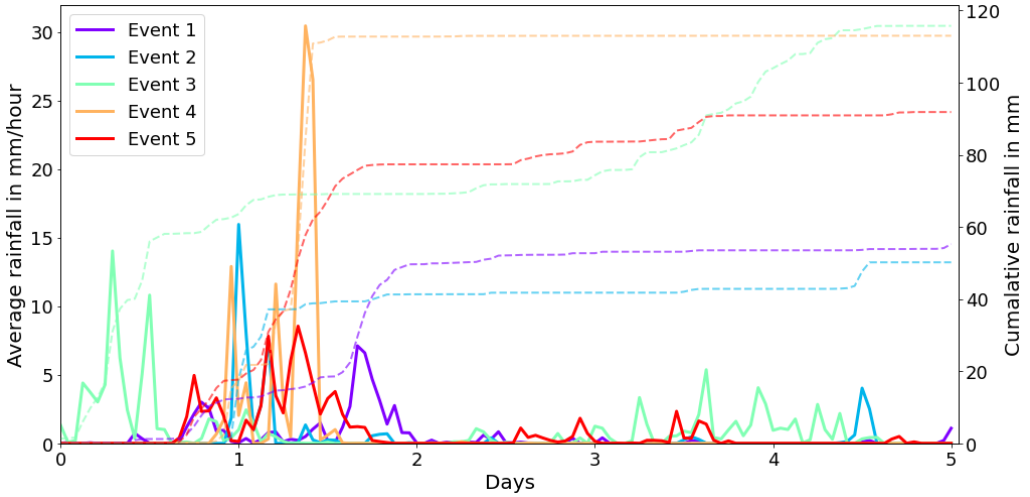


Figure 2.5: *Average rainfall per hour (solid lines) and cumulative rainfall (dashed lines) in the study area during the calibration vents.*

model parameters selected corresponding minimum and maximum values are determined. Parameter sets are randomly drawn from a uniform distribution within the given range for each parameter, shown in table 2.2 in section 3.1. For each parameter set, all 4 calibration events (event 1 to 4 in table 2.3) are simulated, and the resulting MAE and NSE are calculated.

In total, 2470 setups are tested, for which 9880 simulations are carried out (4 calibration events per setup). From these, the best performing parameter set has to be extracted. This is done based on a multi criteria analysis, the criteria and their respective weights are shown in table 2.4. First, the MAE and NSE values are normalised between 0 and 1, and then the score is calculated by multiplying each value with the respective weight from table 2.4. In principle, the weights are ordered such that water level and pumping discharge data have the same weight. Since for the pumping discharge both the MAE and NSE are used, they are both weighted at 0.25 instead of 5. Event 1 has no water level data, and event 4 has no pumping station data, so therefore their weights are set to 0. Because of this, not all events are weighted equally in the calibration.

2.1.5 Model resolution

In the calibration, only the water levels in waterways and pumping station discharges are compared to observations. These are all part of the 1D model, and therefore it is assumed that the 2D grid size is not important for the calibration phase and thus a coarse grid can be used to speed up simulations for the calibration. The main reason for the spacing between 1D computational nodes (1D resolution) is the exchange of water with the 2D grid. Therefore if the 2D grid is coarsened, it is assumed the 1D resolution can also be coarsened. To test these

Table 2.4: *Criteria used for the multi criteria analysis and their respective weights.*

Event	Pumping station discharge: MAE	Pumping station discharge: NSE	Water level stations: Average MAE
Event 1	0.25	0.25	0
Event 2	0.25	0.25	0.5
Event 3	0.25	0.25	0.5
Event 4	0	0	0.5

assumptions, the same event has been run for multiple 2D grid sizes and with different 1D resolutions. The test is done for event 4, since this event has the most rainfall in a short period, resulting in the most inundation. Five different setups are tested, with 2D grid sizes ranging from 100 to 10 meters, and 1D resolutions ranging from 50 to 10 meters. The maximum difference in water level between all simulations is 7 cm, which only occurs momentary. More than 95% of the time the difference is less than 1 cm, therefore the conclusion is that the 1D and 2D resolution have little influence on the outcome of the calibration. For this reason, a low resolution can be used to speed up the calibration process. From table A.2 in appendix A.6 it can be seen that especially the 2D resolution has a large influence on the computation time, this is because for this event there is some inundation, and thus flow over the 2D grid which requires heavy computation. The water levels and pumping station discharges for the model resolution test can be seen in appendix A.6, figure A.2b.

2.2 Cloud computing

To create probabilistic inundation forecasts, the inundation for each of the 50 ensemble members of the rainfall forecasts is predicted. By combining the inundation of all ensemble members, a probabilistic inundation forecast can be made. Predicting inundation for all ensemble members requires a significant amount of computing power. Running all simulations on a regular computer would take hours, which makes it infeasible to use the forecasts operationally. Cloud computing is utilised to run many simulations simultaneously, which drastically reduces the total computation time. This section explains the methodology for research question 2: "What are the cost and computation times for generating probabilistic inundation forecasts using cloud computing?".

2.2.1 Parallel execution

To be able to perform many simulations without being limited to computing power of a single computer, cloud computing will be utilized. This research is part of the TKI project of Deltares, aimed at developing the ability to perform D-HYDRO simulations in the cloud (Deltares, 2022a). For this research, cloud computing serves two goals. Firstly, it is used to make probabilistic inundation forecasts based on ensemble rainfall forecasts. Secondly, cloud computing is used to generate training and testing data for the neural network. For both goals, it is important that many simulations are performed automatically and simultaneously. In cloud computing, this is achieved by running the simulation simultaneously on virtual machines, that are hired by a cloud provider. For this study, the provider is Microsoft Azure.

Broadly speaking, there are two ways of setting up an infrastructure that can achieve this goal. The first method is hosting a Virtual Machine (VM) that is capable of running D-HYDRO simulations. A virtual machine can be seen as a computer that uses software instead of a physical computer to run programs and deploy apps. This software runs on top of a physical computer. This way, multiple virtual (guest) machines can be set up on a single physical (host) machine. Each virtual machine has its own operating system.

The second method is by using containers. A container differs from a virtual machine in that it only includes the necessary components to perform its intended task, rather than replicating an entire computer with its own operating system and applications. As an example, suppose you want to have a script that prints the time every hour. A VM would need a full operating system and the correct software installations to run the script you create for this task. A container on the other hand, would only need a very small base installation, a Python installation, and a script on it. Another advantage of the container is that because of the small size, it can easily be transferred between machines, which is a great benefit for cloud computing. In figure 2.6 an illustration of both virtual machines and containers can be seen.

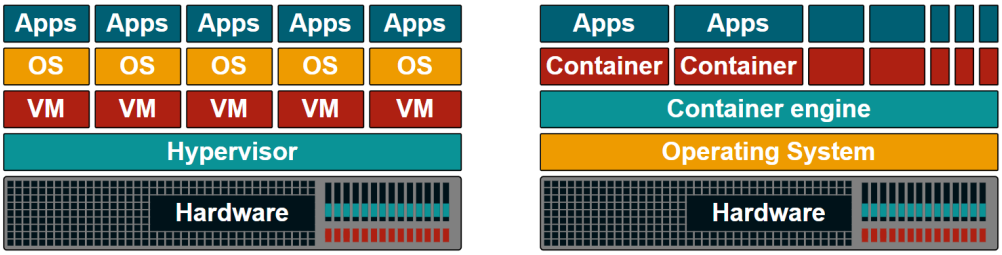


Figure 2.6: *Illustration of the architecture of Virtual Machines (left) and Containers (right).*

Containers can be transferred with all their required applications pre installed, and their tasks pre defined. Because they function as an isolated unit, the container will work on any machine with a container engine. This makes it easy to deploy containers in the cloud, since once the container functions as defined, it will always operate in the same way. Containers can run their applications almost immediately after deployment, while virtual machines require more initialisation steps. Because of these reasons, the decision is made to use a container-based approach for the cloud simulations.

2.2.2 Software

To be able to work with containers in the cloud, multiple steps need to be taken, each requiring existing software. In this section the most important software that has been used is explained.

Overview It might be overwhelming and slightly confusing since the different software packages have some overlap. To help understand the collaboration between the different components they are explained with a metaphor. Imagine a captain preparing for a naval battle. The captain has a fleet of 30 ships available to him. These ships can be interpreted as the hardware, and the combination of all hardware (so the entire fleet of ships) represents the cluster, hosted by Microsoft Azure. The captain has 200 crew members available, each with a different specialisation, some trained to operate canons, some trained to steer ships, etc. Each crew member can be seen as a container, where each specialisation represents a different container image. The container images are made with Docker, so Docker is responsible for having the right images (or crew member specialisations) available. Now, the captain has to divide the crew members over all the ships. This represents the containers that have to be assigned to the right hardware. If ships are damaged (hardware crashes), crew members (containers) have to be allocated to different ships (hardware). All of this is the job of Kubernetes, which assigns the containers to the right hardware, such that all containers can be used effectively. Kubernetes can be seen as the captain of the battle. Now lastly, a plan was made for the captain to follow during battle. The captain tries to follow this plan he has received. This plan can be seen as the workflow, created in Argo Workflow. The captain (Kubernetes), tries to follow this plan (workflow), but sometimes has to make sudden changes when ships are damaged (hardware crashes).

Building containers The first step towards deploying a container in the cloud, is building a container. To do this, an image is constructed. The image can be seen as a blue print for the container. It includes the tasks the container needs to execute, and the required software to execute these tasks. The Docker Desktop software is used to build and test these container images. With Docker a baseline image is used as a basis for the container, and on top of this functionality can be added. As an example, one can start with a baseline image, install Python on it, install some Python packages, and copy certain files and scripts to the container. Also the tasks for the container can be specified, for example to run a script. Docker will then build this container, such that all the installations are ready for use. When you run the container, it can immediately start operating.

Orchestrating containers Once all containers that are needed are build, they need to be executed in the right order, with the correct flow of information between them. This might be simple when containers need to be executed one after another, however can become troublesome when many need to be run simultaneously. All containers need hardware to be able to work, and therefore containers need to be allocated to available hardware without intervening each other. To manage all this, Kubernetes is used. With Kubernetes, you organize a cluster of virtual machines and plan the containers to run on the virtual machines based on the available computing

resources and the requirements for each container. Containers are grouped into pods, the basic operational unit for Kubernetes. These pods can be scaled based on the desired state. There is one master node, and many worker nodes. The master nodes arranges the orchestration, while the worker nodes are running the containers.

Planning orchestration Kubernetes can deploy containers according to the users needs. To simplify and organise the process of deploying containers, Argo Workflows is used. With Argo Workflow, deployment plans can be written, which can be used as an input for Kubernetes. These workflows consist of steps, where each step is executed by a container.

Computing resources As mentioned, Microsoft Azure is the computing resource provider for this study. The computing resources are called nodes. These nodes are virtual machines hired from Microsoft Azure. One virtual machine can run multiple containers, the amount depending on the required resources for a container, and the specifications of the virtual machine. A more powerful virtual machine can run more containers, but will also be more expensive to hire. It is important to analyse the required resources for the containers, such that the correct virtual machines can be deployed.

2.2.3 Probabilistic inundation forecasts

To make a probabilistic inundation forecasts, the hydraulic model predicts the inundation depths for each of the 50 ensemble members of the rainfall forecasts. A threshold of 5 cm is then used to determine whether a cell is inundated or dry. The inundation probabilities for each grid cell are then calculated according to equation 4.

$$P(I)_i = \frac{n_{I,i}}{N_{ens}} \tag{4}$$

Where

$P(I)_i$ is the inundation probability at grid cell i.

$n_{I,i}$ is the number of ensemble members where inundation occurs at grid cell i. The threshold for determining if inundation occurs is 5 cm.

N_{ens} is the total number of ensemble members considered.

2.2.4 Workflow probabilistic inundation forecasts

For this study, a cloud workflow that is able to make probabilistic inundation forecasts based on the KNMI ensemble rainfall forecasts is made. This workflow needs to be able to fully run in the cloud, and thus all operations have to be performed within containers. The workflow is responsible for downloading the ensemble rainfall forecasts, extracting the right information from the forecasts, run D-HYDRO simulations with these forecasts as external forcing, and post process the output. Executing these steps on a regular computer would result in long computation times, because all simulations have to be performed one after another. With cloud computing, all simulations can be performed simultaneously, and thus the computation time will be much shorter.

The workflow is shown in figure 2.7. The figure gives an overview of the steps taken. The workflow is based on three container images. The first container is responsible for gathering the model input data, downloading ensemble forecasts, and processing all this to be suitable to use as the hydraulic model input. The second container (or actually many containers simultaneously) is responsible for running all the simulations in D-HYDRO. The third container is responsible for processing the output from the simulations.

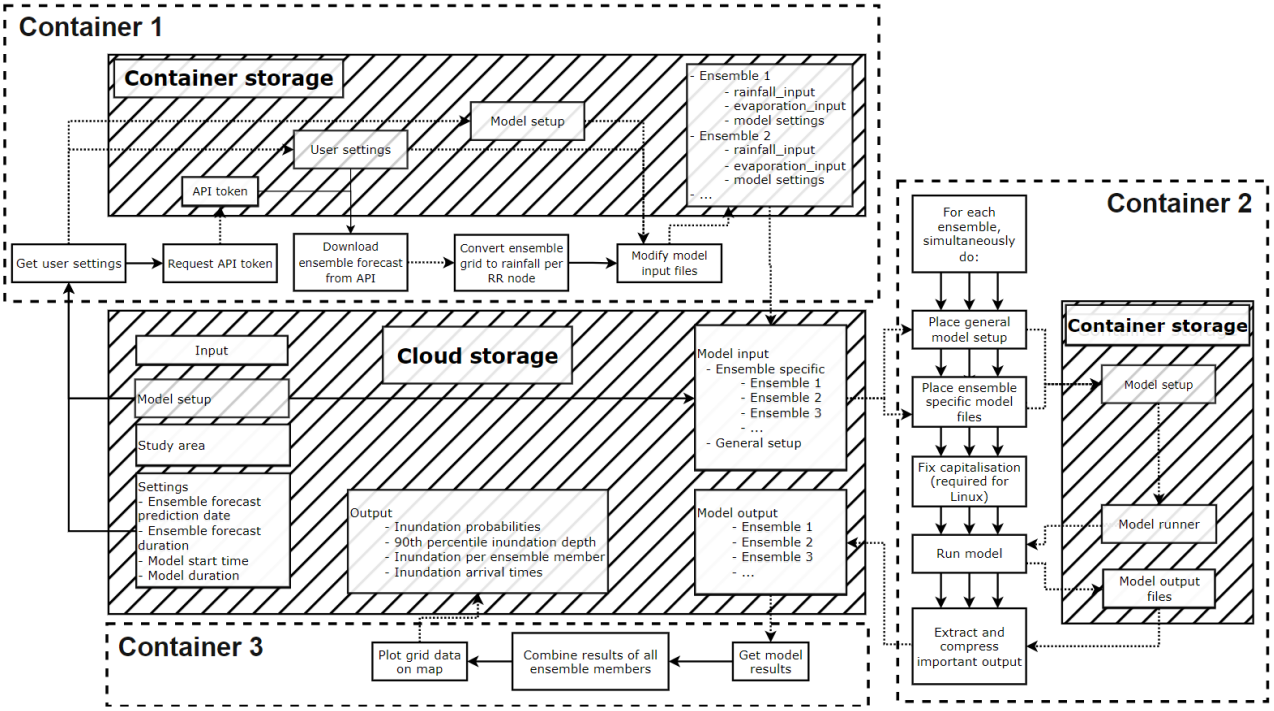


Figure 2.7: *Workflow to create probabilistic inundation forecasts in the cloud.*

2.2.5 Analysing costs and computation time

Interesting to know are the computation time for generating probabilistic inundation forecasts using cloud computing, and the associated costs. Multiple containers can run on a single machine. At Microsoft Azure, these machines are hired as virtual machines. From now on, virtual machines will refer to the computers that run the containers on top of them. These virtual machines can also be called nodes. Microsoft Azure has many different types of VMs available (currently over 700) that can be used. Each VM has different hardware allocated to it, such as the number of CPU cores, the amount of RAM memory, the type of CPU and cost per hour. Using the right VM is crucial for being able to efficiently generate probabilistic inundation forecasts.

Hardware usage To be able to make decisions on which hardware is required, it is important to first analyse the hardware usage of the hydraulic simulations. This is done by tracking both the CPU and RAM usage over the course of a simulation. This is investigated for multiple model resolutions, as the resolution is expected to impact the hardware usage.

Virtual machine setup To analyse which VM setup performs best, a series of tests is done. The tests are organised in two stages. In the first stage, different VM categories are tested. Azure organizes the VMs in different categories (e.g. optimised for memory usage, optimised for computational speed). Four different categories of machines are tested to see which VM performs best in terms of both costs and computation times. In the first stage, the number of CPU cores is kept constant over all tests. The virtual machines that are tested are shown in table 2.5. For each test, all 50 ensemble members are simulated. This is done for the model with a 1D resolution of 20 meters and 2D resolution of 10 meters. The ensemble forecast of event 1 from figure 2.13 is used for these tests. In section 2.3.6 the methodology for generating this ensemble forecast is explained.

In the second stage, within one category, different sizes of the VM are tested. Larger VMs (more cores and RAM) are able to host more containers, however it comes with less control on exactly how many cores are used. Smaller VMs can only host a few containers, but the number of cores is more flexible. Interesting to see is how many simulations can be run in parallel on each of the VM sizes, how long the entire workflow will take, and what the costs per probabilistic forecast are for the different configurations. Costs are determined according to $cost = t * s * c.\ t$, where t is the total time in hours required to make a probabilistic inundation forecast, s is the number of VMs that are used and c is the price per hour of hiring the VM used for this forecast.

The tests performed are all done with the 10 meter resolution model, forecasting inundation 12 hours ahead. To apply the outcomes of these tests to another model, the required resources for the model have to be known. With that information, the results from the performed tests can be used to help make a decision on what setup would suit best to minimize either computational time or the costs.

Table 2.5: *Virtual machines used during testing.*

VM name	Category	CPU cores	RAM (GB)	Storage (GB)	Compute benchmark score	Cost ($/hour)
Stage 1						
A8 v2	General purpose	8	16	80	52.295	0.38
F8s v2	Compute optimized	8	16	64	136.027	0.39
D8 ads v5	General purpose	8	32	300	153.951	0.50
E8 ads v5	Memory optimized	8	64	300	153.765	0.63
Stage 2						
D2 ads v5	General purpose	2	8	75	38.919	0.13
D4 ads v5	General purpose	4	16	150	72.644	0.25
D8 ads v5	General purpose	8	32	300	153.951	0.50
D16 ads v5	General purpose	16	64	600	306.800	1.00

2.3 Neural network

Another method to reduce the computation time of inundation forecasts is by training a neural network to predict inundation depths. The network is set up such that it can predict inundation over time based on the rainfall per hour. The network can predict inundation patterns twelve hours ahead, at one hour intervals. In this section, the methodology for setting up the neural network is explained. The resulting neural network is used to answer research question 3: "How accurate can a neural network replicate the deterministic inundation depths and inundation probabilities of the hydraulic model?".

The neural network is trained on data generated by the calibrated hydraulic model, where the network aims to predict the inundation depth at each grid cell of the hydraulic model. Ideally, the neural network would be trained on real world data rather than on model predictions, however this is not possible since this data is not available. The input of the network is the rainfall over time. The output represents the inundation depths for each grid cell. This means that the neural network has as many outputs as there are grid cells (53.381 in the 10 meter resolution model).

2.3.1 Data

With the calibrated hydraulic model (section 2.1), the resulting inundation of 2000 different rainfall events is simulated. Each simulation outputs the water depths of all grid cells at twelve time steps at one hour intervals. These 2000 simulations have been performed using cloud computing with a cost of around 25 euros.

The input for the neural network is the rainfall over the entire domain at each time step. With a forecasting time of twelve hours, this means that the network has a total of twelve inputs, each representing the rainfall intensity during a one hour time step. A one hour time step is chosen because the KNMI ensemble forecast predicts the rainfall with a temporal resolution of one hour. Uniform rainfall over the area is assumed because the study area is relativity small, and it reduces the complexity of the neural network. The training dataset needs to be well balanced, to prevent the network from having a bias towards a certain type of event. Historically, there have been many events with light to moderate rainfall, and only a few with very heavy rainfall. To get a balanced dataset, more heavy events need to be generated in a realistic way. Furthermore, using a neural network for extrapolation beyond the extent of the training data is not reliable (Sajikumar and Thandaveswara, 1999), so it is important to train the network on heavier rainfall events than those expected in reality. To get sufficient events that meet these requirements, events are generated according to the steps described in figure

2.8. Rainfall events used during the training are based on historic events between 1900 and 2020 adjusted for the 2050 climate change scenario with the largest change in both temperature and air flow patterns (Beersma et al., 2019; KNMI, 2014). From this time series, all events with more than 50 mm of rainfall in 12 hours are selected. For each simulation, a random event is selected. This event is shifted in time randomly such that the peak rainfall can occur at any time step. Then, the rainfall per hour is scaled according to the desired total rainfall for each event. The desired total rainfall in twelve hours ranges from 10 to 180 mm. 180 mm is far beyond what would be expected in 12 hours, but this is done such that the neural network would never have to extrapolate during any of the time steps. As a reference: the heaviest rainfall in 12 hours in Beersma et al. (2019) is 139 mm which has a recurrence time of once every 1000 years. All other model input parameters remain unchanged, with values according to the model calibration (section 3.1).

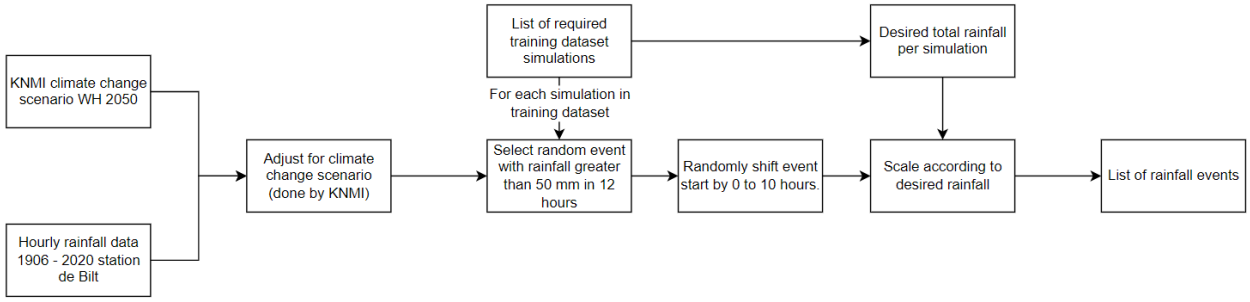


Figure 2.8: *Flowchart of the steps taken to generate the input rainfall data for generating training data.*

The samples (simulations) are then split up in training, testing and validation datasets. Training data is used to train the neural network. Validation data is used to asses the performance of the neural network during training and hyper parameter optimisation. Testing data is used to perform the final analysis on the performance of the network. This is done according to a stratified split of 80% training, 10% validation and 10% testing (similar to (Zhou et al., 2021; Zanchetta and Coulibaly, 2022)). This results in 1600 simulations for training, 200 simulations for validation, and 200 simulations for testing. All simulations are sorted based on total input rainfall. Testing and validation samples are then picked at fixed intervals, such that the testing and validation set are not biased towards samples of higher or lower rainfall. This way it is also made sure that the events with the highest and lowest rainfall are not used for testing and validation, since the neural networks ability to extrapolate beyond the training data cannot be guaranteed (Sajikumar and Thandaveswara, 1999).

2.3.2 Data preprocessing

Normalisation Before the output of the hydraulic model can be used for the neural network, it first has to be preprocessed. Neural networks are known to perform best when the input data is normalised between -1 and 1 (Rafiq et al., 2001), therefore both the input and output data have to be normalised. Normalisation is done between 0 and 1, since both the input (rainfall) and output (water depth) can physically not be below 0. This is useful in combination with the activation function of the last layer, which will be explained in section 2.3.4. Normalisation of the rainfall is done by scaling linearly based on the maximum value (equation 5). For the inundation depths, a similar scaling procedure is used. However there are a few cells that have a significantly higher inundation depths than almost all other cells. If these ”outliers” in inundation depth are used as normalisation boundaries, this would result in the majority of data not being normalised between 0 and 1, but between 0 and 0.4. As an example, see the illustration in figure 2.9. Excluding outliers in determining the normalisation boundaries has shown to improve the accuracy of the inundation predictions. The network has been tested with different values for the outlier exclusion (ranging from 0.1% until 10^{-8}%). Excluding the largest 0.00001% of the inundation depths resulted in the smallest MSE on the validation dataset. So the smallest 99.99999% of inundation depths is used to determine the normalisation boundaries.

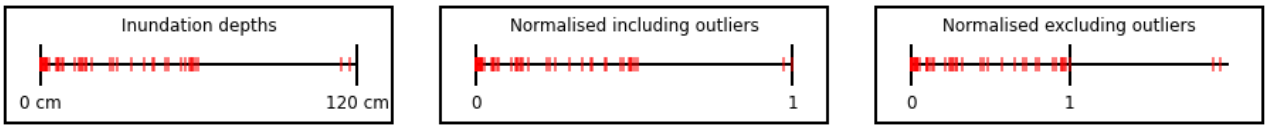


Figure 2.9: *Illustration of normalisation with and without including outliers in determining the normalisation boundaries.*

$$x = \frac{x_{in}}{x_{max}} \tag{5}$$

$$y = \frac{y_{in}}{y_{max_{99.99999\%}}} \tag{6}$$

Where

x is the neural network input, which represents the rainfall.

y is the desired neural network output, which represents the inundation depths.

x_{in} is the rainfall in millimetres per hour at each time step.

y_{in} is the inundation depth in meters for each grid cell and each time step.

Dry cells The hydraulic model outputs inundation depth per grid cell for each time step. In the study area, certain grid cells remain dry during all simulations. For these specific locations, a neural network prediction of inundation is not required, as these cells are never inundated in the training data. For this reason, grid cells that are never inundated are removed from the training, testing and validation data, and will not be further considered inside the neural network. This reduces the number of outputs of the neural network by about 50% for this model. By having less outputs for the network, the network will have to train less parameters, which makes training faster and less memory intensive.

2.3.3 General network architecture

In literature, there are no direct comparisons between different network architectures, and thus no knowledge on which network architecture is best for inundation predictions. The Long Short Term Memory (LSTM) network has shown promising results in literature, and thus was chosen to further analyse and optimise for this research. The initial network architecture is inspired by the research of Besseling (2022). Besseling (2022) has applied a LSTM neural network to predict water depths as a result of a dike breach.

LSTM A LSTM network is a special kind of Reccurent Neural Network (RNN), that solves the issue that regular RNN's cannot deal with long term dependencies (Hochreiter and Urgen Schmidhuber, 1997). This is done by replacing the traditional RNN cell by an LSTM cell which stores both the long and short term information. The way it does this is explained by Olah (2015), which will be summarised here.

The key to a LSTM is the cell state (from now on called long term memory), the horizontal line running through the top of the diagram in figure 2.10. The long term memory runs straight down the entire cell, with only some small modifications being made.

There are two operations that are performed on the long term memory. The first is the forget operation (f_t^l in figure 2.10), which forgets information that is no longer useful. The second is the update operation (i_t^l and $\tilde{C}_t^l$), which adds new information to the long term memory.

But the long term information is not directly used as an output. For the output, also the short term information has to be included. The output is used by the rest of the neural network. This is a combination of the long term memory and the current state (short term memory). The information from the long and short term memory are combined, then filtered, and outputted as $h_t^{\cdot}$. The long term memory is carried forward to the next LSTM time step.

An LSTM cell uses three information sources. The long term memory (cell state C_{t-1}^l), the short term memory (hidden state h_{t-1}^l), and new input data (h_t^{l-1}). It outputs an updated long term memory (C_t^l), and

a new hidden state (h_t^l). In figure 2.10 the hidden state is outputted twice, once for the LSTM cell at the next time step, and once for the rest of the network to use.

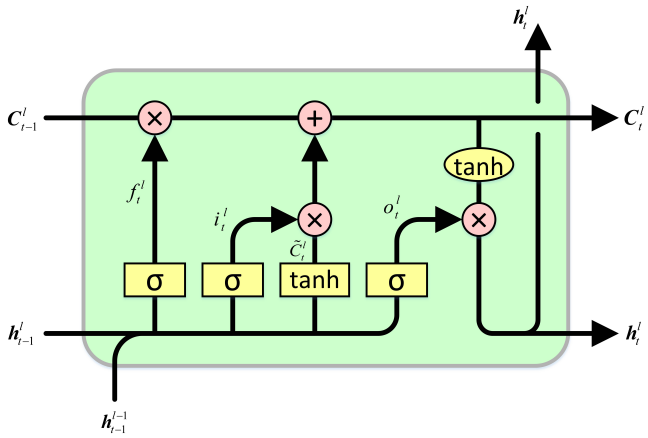


Figure 2.10: *Graphical representation of an LSTM cell.*

2.3.4 Hyper Parameter Optimisation

The next step is optimising the neural network architecture and the training settings for the objective of this research. To do this, a hyper parameter optimisation is performed. Hyper parameter optimisation is the process of choosing a set of optimal parameters for a learning algorithm. There are multiple techniques to perform this optimisation. Some of the most used algorithms are listed below. Hyper optimisation is a topic of ongoing research, and many more types of optimisation algorithms are available.

- Grid search: An exhaustive searching through a manually specified subset of the hyper parameter space of a learning algorithm (Bergstra et al., 2011).
- Random search: Random Search replaces the exhaustive enumeration of all combinations by selecting combinations of parameters randomly (Bergstra et al., 2011).
- Bayesian optimisation: Bayesian optimization is a global optimization method for noisy black-box functions. Applied to hyperparameter optimization, Bayesian optimization builds a probabilistic model of the function mapping from hyperparameter values to the objective evaluated on a validation set. By iteratively evaluating a promising hyperparameter configuration based on the current model, and then updating it, Bayesian optimization aims to gather observations revealing as much information as possible about this function and, in particular, the location of the optimum (Snoek et al., 2012).

The network can take minutes to hours of training time, depending on the setup. For this reason, a full grid or random search is not feasible since it would cost too much computational time for this study. A Bayesian optimisation might be possible, but provides less insight about what parameters have an effect on the output, and might require many iterations to provide optimal results. The Bayesian optimiser will minimize the objective function, with no possibility to manually consider trade-offs between accuracy and number of trainable parameters.

Because of this, a variation of the grid search is used. Starting with the most influential parameters, grid searches are performed to find the optimal setup for this parameters. The base network is updated with this new information, and the next parameter set is tested using a grid search. This way, an insight in how different parameters affect model convergence and number of trainable parameters is gathered, and the final model will be both accurate and efficient. The steps taken during the optimisation are listed below:

1. Number of LSTM layers and number of units per layer are optimised with a grid search.
2. Number of dense layers and number of neurons per layer are optimised with a grid search.
3. Number of dropout per layer is optimised with a grid search.

4. Learning rate and batch size are optimised with a grid search.

Two model variants are considered for this research, a 20 meter 2D resolution and a 10 meter 2D resolution. Generating training data for the 20 meter resolution model requires less computational power, and the training of the neural network is faster because there are less grid cells. The hyper parameter optimisation is first performed for the 20 meter resolution training data because this requires less training time. Later it is redone for the 10 meter resolution training data, with a more narrowed down parameter search space. The results for the 20 meter resolution optimization can be seen in figure 2.12, in the figure captions the decisions made based on the results are explained. The final setup can be seen in table 2.6, and a graphical representation is shown in figure 2.11. The final setup for the 10 meter resolution neural network turned out to be the same as that of the 20 meter resolution setup. In general, this neural network structure is expected to also be suitable for inundation predictions in other study areas, provided that there is a similar amount of grid cells. The setup resulted in accurate inundation predictions for both the 20 meter resolution inundation depths (14.291 cells) and 10 meter resolution (53.381 cells).

So far, the activation functions have not been mentioned yet. For the LSTM layers, a hyperbolic tangent activation function is used. Changing to other activation functions did not show significant improvements, and the hyperbolic tangent is most popular since it allows for GPU acceleration. For the dense layer, a rectified linear unit activation function (ReLU) is used. From testing, this turned out to be the most suitable. It also makes sense physically, since the output (water depths) can never be below zero, just like the rectified linear unit activation function ($ReLU : f(x) = max(0, x)$). Other activation functions that were tested are the sigmoid, linear and hyperbolic tangent activation function.

An important note is that within the final network setup there is no spatial coherence between adjacent cells. For the neural network, there is no difference between cells laying next to each other, and cells laying on the opposite side of the domain. Instead, the network will learn patterns for each cell, and if it learns them well enough, the spatial coherence of the original dataset will be replicated.

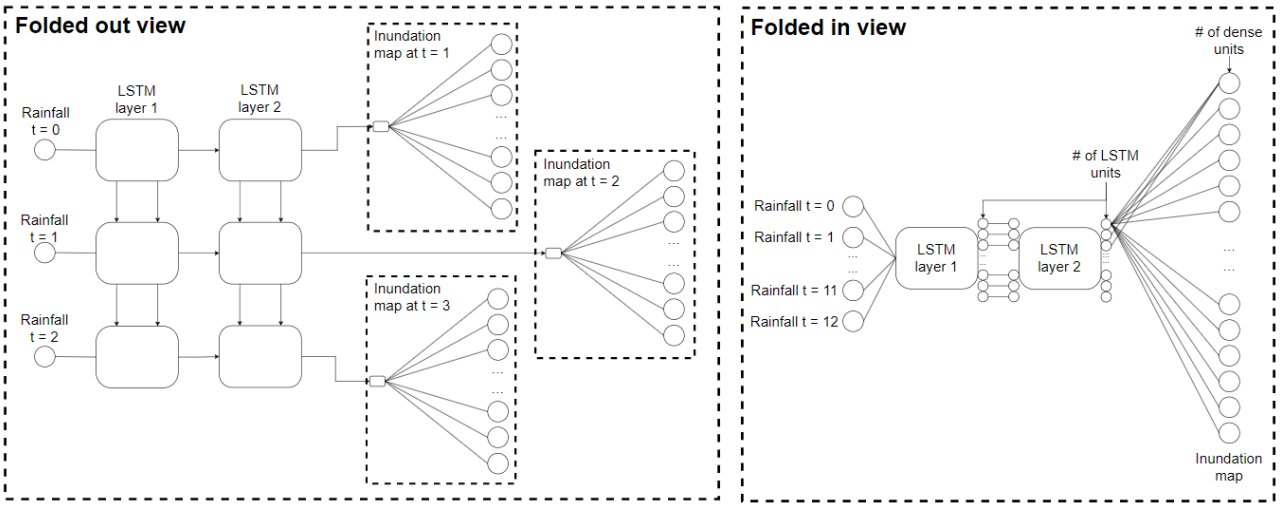
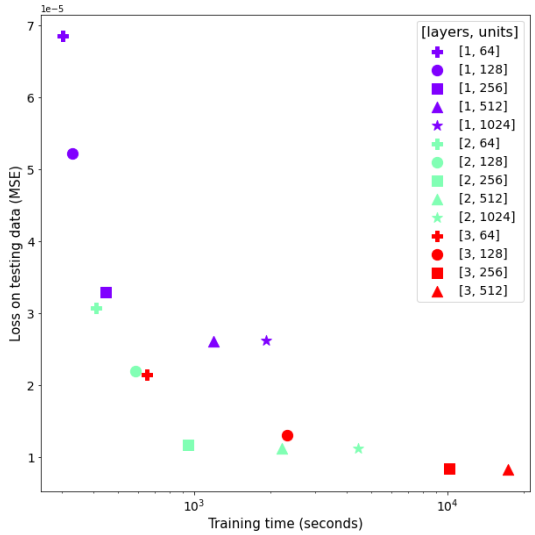


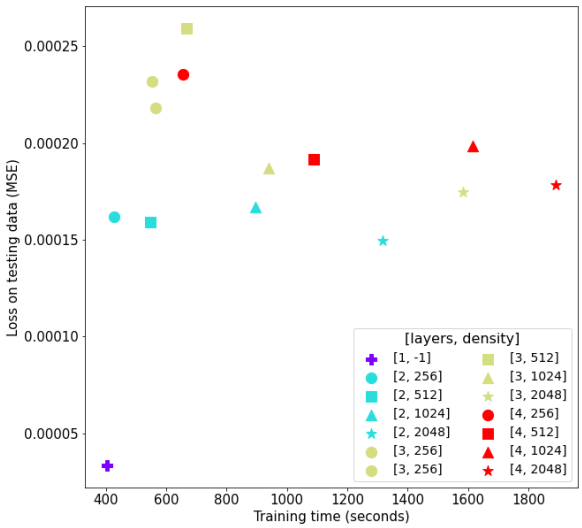
Figure 2.11: *Graphical representation of the neural network architecture.*

Table 2.6: *Neural network architecture resulting from the hyper parameter optimisation.*

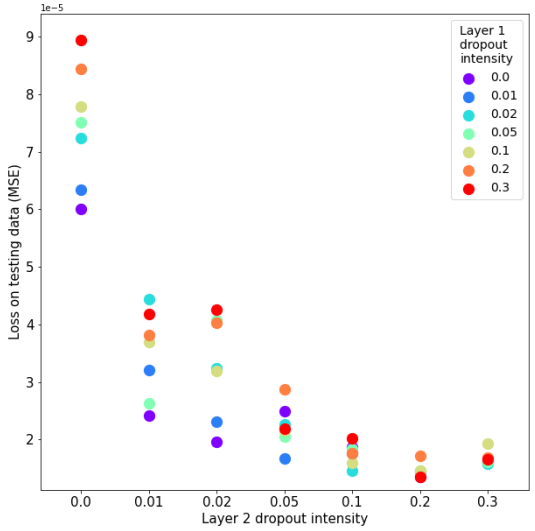
LSTM layer 1 units	256	**Dropout after layer 1**	0.02
LSTM layer 2 units	256	**Dropout after layer 2**	0.2
LSTM layer activation function	hyperbolic tangent	**Dense layer activation function**	rectified linear unit
Learning rate	0.001	**Batch size**	8
Dense layer units	53.381 (grid cells)		



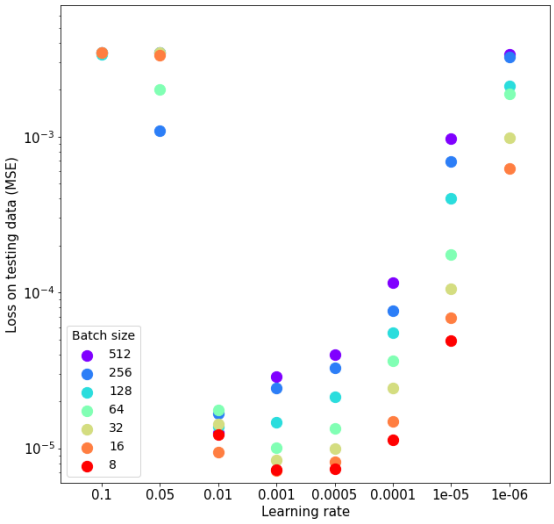
(a) Number of LSTM layers and number of units per LSTM layer. 2 layers with density of 256 are chosen since accuracy does not improve significantly with more complexity. Using different numbers of units per layer is also tested, but did not improve MSE and are therefore not shown.



(b) Number of dense layers and number of neurons per layer. Last layer always has the density of output size (number of grid cells), -1 density stands for default output size. Adding additional dense layers does not improve MSE, but it does increase training time. Therefore, no additional dense layers are used.



(c) Dropout after LSTM layer 1 and after LSTM layer 2. Dropout after LSTM layer 2 is more influential than after layer 1, with a dropout of 0.2 after layer 2 being the best option tested. Layer 1 dropout is chosen at 0.02, but influence of this is small. Note that small variations in mse can also be due to stochastic nature of the training, not necessarily caused by changing parameters.



(d) Learning rate and batch size. Final decision is to use learning rate of 0.001, since it results in the lowest MSE. For the batch size, smaller batches perform better, as expected from literature (Martinez and Wilson, 2003). However, larger batches can significantly improve training times especially when calculations are performed on a GPU. A batch size of 8 will be used.

Figure 2.12: *Results of the hyper parameter optimisation.*

2.3.5 Accuracy assessment

The accuracy of the neural network is assessed by comparing the neural network predictions to those of the hydraulic model. The neural network can never perform better than the hydraulic model, since it is trained on the output of the hydraulic model. The performance metrics that are used to asses the performance metrics will be explained in this paragraph.

Mean absolute error and mean squared error (MAE and MSE) The MAE (equation 7) and MSE (equation 8) are both used to asses the difference between the neural network and the hydraulic model. The MSE is used as a minimisation function for the neural network, since it provided the best results during testing. The MSE is the only error function used during the training process, all others are only used for testing. The lower the MSE and MAE, the better the neural network's performance. The MSE places more emphasis on larger errors than the MAE.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} \mid y_{pred,i} - y_{obs,i} \mid \qquad (7) \qquad\qquad MSE = \frac{1}{n}\sum_{i=1}^{n}(y_{pred,i} - y_{obs,i})^2 \qquad (8)$$

Where

$y_{pred,i}$ is the inundation according to the neural network at grid cell i.

$y_{obs,i}$ is the inundation according to the hydraulic model at grid cell i.

n is the total number of grid cells that are considered.

Nash–Sutcliffe model efficiency coefficient (NSE) The NSE is a normalized statistic that determines the relative magnitude of the residual variance compared to the measured data variance. An NSE of 1 corresponds to a perfect match between model and data. NSE = 0 indicates model predictions are as accurate as predicting the mean. NSE between 0 and $-\infty$ indicates that the mean would be a better predictor than the model. The formula for the NSE is shown in equation 9 (Krause et al., 2005).

$$NSE = 1 - \frac{\sum_{i=1}^{n}(x_{obs,i} - x_{pred,i})^2}{\sum_{i=1}^{n}(x_{obs,i} - x_{mean})^2} \qquad (9)$$

Where

$x_{pred,i}$ is the inundation according to the neural network at grid cell i.

$x_{obs,i}$ is the inundation according to the hydraulic model at grid cell i.

x_{mean} is the mean of the hydraulic model at grid cell i.

n is the total number of grid cells that is considered.

Critical succes index (CSI) The CSI measures the model's ability to correctly predict flooding. Some cells are infrequently flooded, so determining the accuracy of these cells would typically yield a high accuracy, even if the model always predicts no flooding. To counter this problem, the CSI is used, which does not consider true negatives (or in this case: predicting a dry cell when it is actually dry). To be able to use the CSI, an inundation threshold must be set. Below this threshold, a cell is considered to be dry (or non inundated), and above the threshold it is considered wet (or inundated). This threshold is set at 5 centimetres, which was also used in the work of Zanchetta and Coulibaly (2022). The CSI is calculated according to equation 10 (Schaefer, 1990).

$$CSI = \frac{\text{True Positives}}{\text{False Negatives} + \text{False Positives} + \text{True Positives}} \qquad (10)$$

2.3.6 Probabilistic inundation forecasts

The neural network is trained to replicate single simulations, however this research is aimed on predicting probabilistic inundation maps based on ensemble weather forecasts. To make a probabilistic prediction, the neural

network predicts the inundation patterns for each of the 50 ensemble members. The inundation probabilities for each grid cell are than calculated according to equation 4 in section 2.2.3.

To asses the performance, the probabilistic predictions of the network are compared to those of the hydraulic model. The quality is compared by plotting a reliability diagram and calculating the Brier score. The Brier score is equivalent to the mean squared error as applied to predicted probabilities, and can be seen in equation 11 (Brier, 1907). A lower Brier score indicates better performance.

$$\text{Brier score} = \frac{1}{T}\frac{1}{n}\sum_{t=1}^{T}\sum_{i=1}^{n}(f_{i,t} - o_{i,t})^2 \tag{11}$$

Where

$f_{i,t}$ is the neural network probability of inundation of grid cell i at time step t.

$o_{i,t}$ is the hydraulic model probability of inundation of grid cell i at time step t.

n is the total number of grid cells that are considered.

T is the total number of time steps that are considered.

To test the performance, input ensemble forecasts are required. For this study, the historical ensemble forecasts for the Netherlands between 15th of April to the 15th of October in 2022 are available. The ensemble forecast is published every 6 hours, forecasting 2 days ahead. It has a spatial resolution of 7.5*7.5 km, meaning that the entirety of the Netherlands is represented in 758 grid cells. To properly test the model, heavy rainfall predictions are required, since this is what the model is intended for. To get these, the following steps are taken:

1. Select days with rainfall greater than 5 millimetres, and download the ensemble forecasts for these days.
2. From the ensemble forecasts, select the three events (combination of time and location) with the highest average rainfall over all the ensemble members.
3. Scale events to get a total rainfall in 12 hours corresponding to a T1000 event (Beersma et al., 2019) for the ensemble member with the highest rainfall in 12 hours. A T1000 event is an extreme event that is expected to occur once every 1000 years.

The resulting scaled rainfall patterns for the three ensemble forecasts can be seen in figure 2.13.

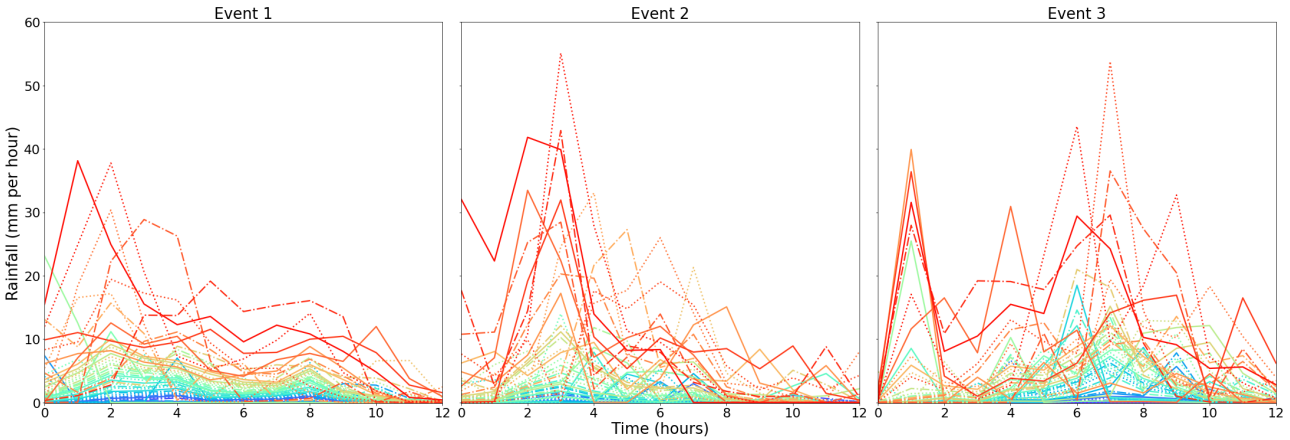


Figure 2.13: *Rainfall over time during each of the three ensemble forecasts. The different coloured lines represent the different ensemble members, there is a total of 50 ensemble members per event.*

3 Results

In this chapter the results that will help to answer the research questions will be presented and discussed. Section 3.1 contains the results of the model calibration, which will help to answer research question 1. The resulting model is then used to make probabilistic inundation forecasts. In section 3.2 the costs and computation time for generating probabilistic inundation forecasts with cloud computing are investigated, to answer research question 2. Cloud computing is then used to generate a training dataset for a neural network. In section 3.3 the performance of the neural network on both deterministic and probabilistic inundation forecasts is analysed to answer research question 3. By combining the results of section 3.2 and 3.3, the answer to research question 4 can be formulated.

3.1 Hydraulic model calibration

In this section the model calibration will be discussed. The performance of the calibrated model is analysed to help answer research question 1. Six model parameters are calibrated by performing a Monte Carlo simulation where 2470 different parameter setups are tested (section 2.1.4). With the help of the multi criteria analysis the best performing model is chosen. The final values for the calibrated model parameters can be seen in table 3.1. The performance of the final setup on the relevant metrics can be seen is shown in 3.1. The final setup performs near the optimum of the tested setups on all metrics for event 2 and 3. On both the pumping station discharge for event 1 and the water levels for event 4 the final setup performs averagely in comparison to all model setups tested.

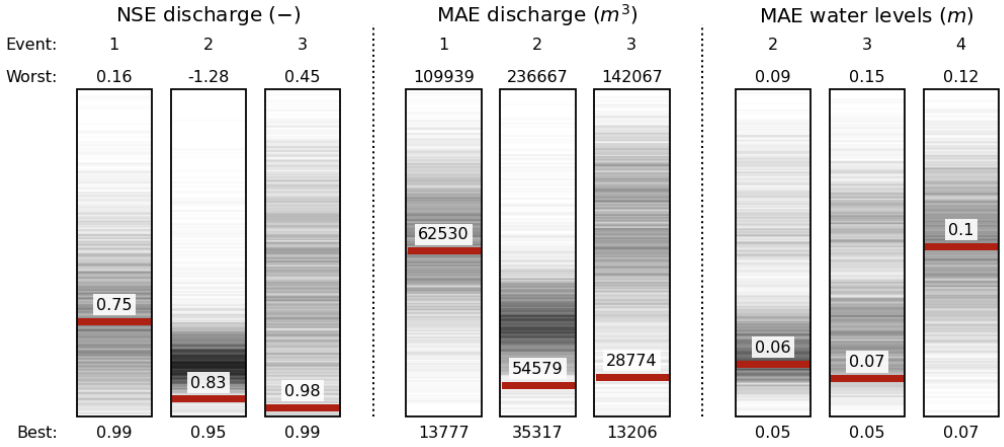


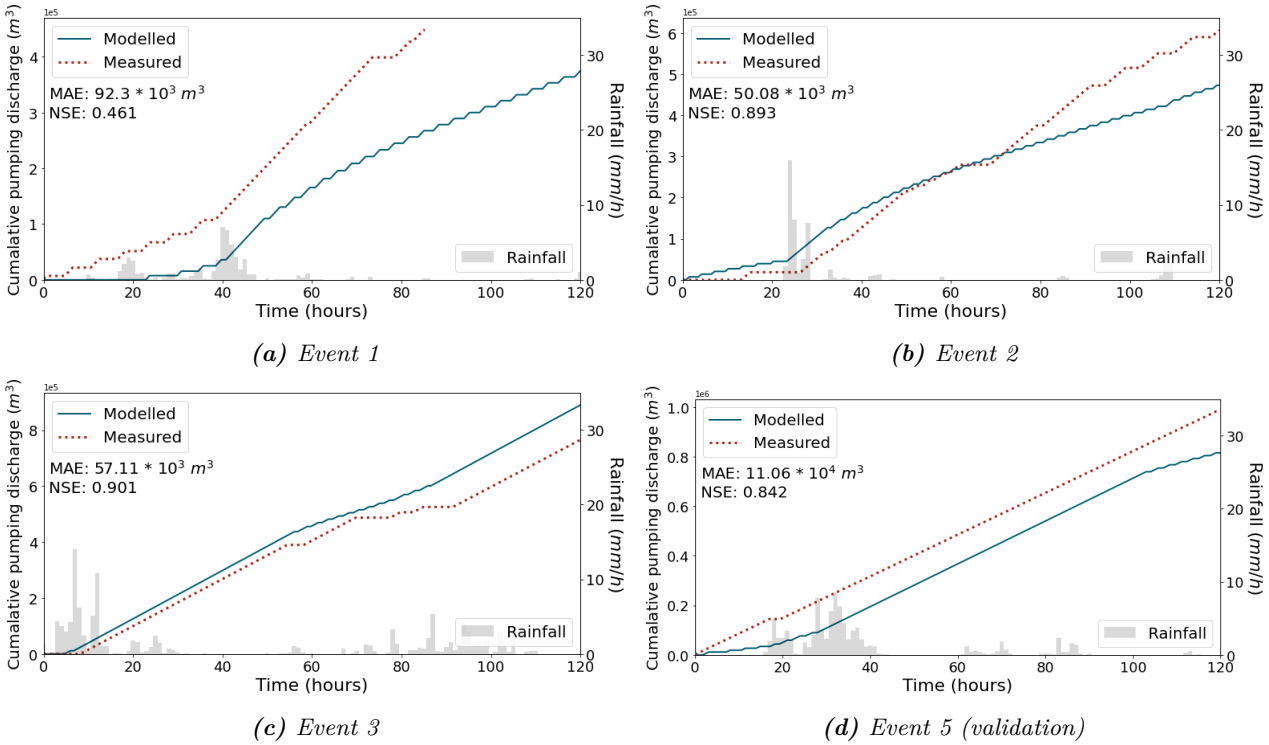
Figure 3.1: *Results of the Monte Carlo parameter calibration, the black lines indicate the individual performances of the simulations, the coloured lines represent the performance of the final setup. NSE and MAE discharge both relate to the error on the cumulative discharge predictions through the pumping station. MAE water levels relates to the error in water level predictions at different water level stations.*

In figure 3.2, the discharge through pumping station de Tol is shown for the events where discharge data is available. It can be seen that the model performs quite well in predicting the discharge with an average NSE of 0.77, but performance is not equally well for all events. For event 1 (figure 3.2a) the model performs significantly worse with an NSE of only 0.461. There is a good resemblance of the general discharge pattern, however in the model predictions it occurs later than in reality. Because of this offset in time, the NSE value is significantly lower than for the other events, despite predicting the discharge pattern well. The offset in time is likely due to the way the model handles the rainfall that occurs before the event. September 2013 was a month with above average rainfall. In reality, this leads to the pumping station already discharging a significant amount of water before the heavy rainfall event. In the model the pumping station is barely active before the event, which leads to the offset in the cumulative discharge pattern. This is because a dry warm up period is used for all simulations, while for event 1 a wet warm up period would have suited better. The additional warm up of 7 days before each event is insufficient to accurately represent the initial state of the system for event 1.

For event 2 (figure 3.2b) and event 3 (figure 3.2c) the model performs very well with an NSE around 0.9. For event 2 there is an overestimation of the cumulative discharge at the start, and an underestimation at the

Table 3.1: *Resulting values for model parameters after the calibration. Min and max value indicate the ranges of values that the parameters could take during the Monte Carlo calibration.*

Parameter	Description	Min value	Max value	Final value	Unit
Channel width	Width of the channels where no cross section data is available	1.5	4	2.75	m
Channel height	Height of the channels where no cross section data is available	1	1.5	1.2	m
1D channel roughness	The resistance against flow of water in channels (Strickler)	8	40	12.6	$\frac{m^{1/3}}{s}$
Drainage resistances	The ability to resist flow of water underground. Multiplication factor with values of 30, 200 and 10000 days for soil layer 1, 2 and 3 respectively	0.3	2	1.57	–
Land storage	Amount of water that can be stored on unpaved land	5	20	18	mm
Surface runoff resistance	Resistance against surface flow of water	0.3	2	1.21	day



(a) *Event 1* ***(b)*** *Event 2*

(c) *Event 3* ***(d)*** *Event 5 (validation)*

Figure 3.2: *Modelled and measured discharge through pumping station de Tol during heavy rainfall events.*

end of the simulation. In contrast, for event 3 there is a slight overestimation in pumping station discharge at the end. Lastly, the validation event shown in figure 3.2 shows good predictive skill with an NSE of 0.842, but again with some delay just like for event 1. For the validation event the pump runs at full capacity during almost the entire event. The model approximates this behaviour well, with a slight underestimation at the start and end of the event.

In figure 3.3 the modelled and measured water levels during event 3, 4 and 5 can be seen. Event 4 is the event with the heaviest rainfall in a short time, which causes significant inundation in the hydraulic model (figure A.4 in appendix A.7). For this event inundation also occurred in reality (Blekemolen and Schwarz, 2014; Kleinveld et al., 2016; Smorenburg, 2014), however no data about the exact extent of this inundation is available.

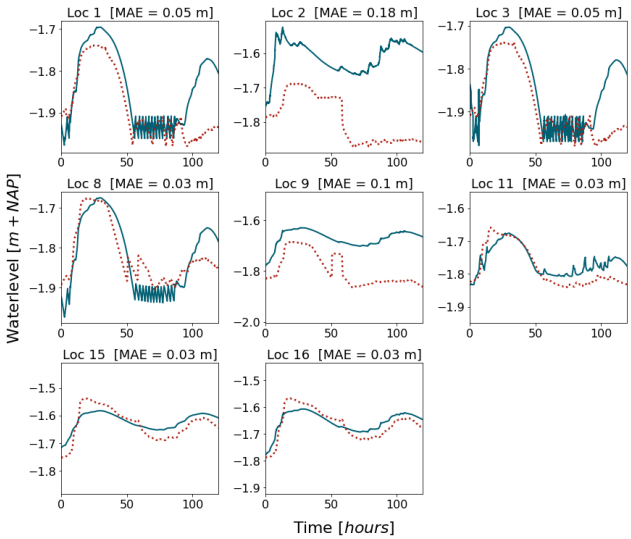
The model demonstrates the best accuracy in predicting water levels during event 3. During event 3 there was a similar amount of rainfall as during event 4, but it was distributed over a longer period of time. For the events considered during the calibration and validation, the model is better suited for predicting water levels in situations where rainfall is more evenly spread out over time, as opposed to concentrated bursts of rainfall.

The duration of the water level peak is overestimated significantly for event 4. The model overestimates the time it takes to discharge the water surplus resulting from the heavy rainfall. From figure 3.1 it can be seen that the final setup resulting from the calibration did perform worse for event 4 than it did for the other events. The reason for the difficulty in predicting water levels during event 4 could be because of human interference. During event 4, there was significant inundation, and therefore measures were taken to reduce water levels as soon as possible (Blekemolen and Schwarz, 2014). The exact actions are not reported, but it is likely that the measures have led to a faster reduction in the water levels. Examples of such measures are the installation of emergency pumps, changes to weir levels and water storage in retention areas (Smorenburg, 2014; Kleinveld et al., 2016). The measures are not included in the model, and therefore the model does overestimate the duration of high water levels. This would help explain why the hydraulic model is very accurate in predicting the duration of the water level peak for event 3, and inaccurate for event 4. Also this would help explain why the final parameter setup performs relativity well for all metrics, apart from the water levels during event 4.

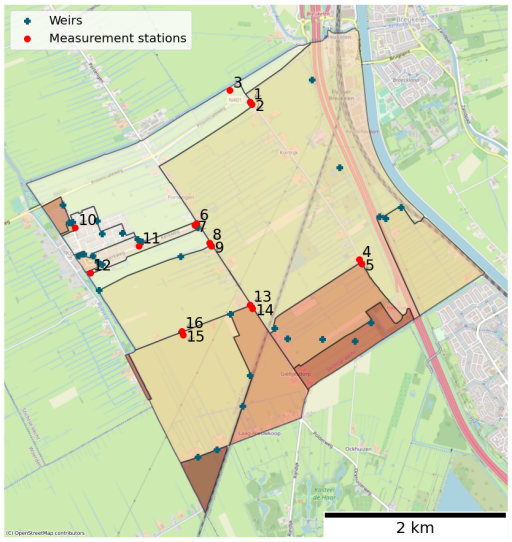
The maximum water level is approximated well with an error of a few centimetres at most of the locations. There are some overestimations of the maximum water levels during various events. One reason for this is the 2D schematisation of the model. Water starts inundating from the waterways to the surface when the surface level of the grid cell is lower than the water level in the connected waterway. With the model, the surface level is averaged over square grid cells. This means that local variations in the surface get evened out over grid cells. In reality, water would overflow at the point where the surface level is lowest. However due to the averaging out over grid cells, the lowest grid cell in the model is higher than the actual lowest point in reality. Because of this, water levels in the waterways are slightly higher before inundating nearby grid cells.

In general, the predictions of the model are in accordance with the observations. In some events there is a small overestimation in pumping discharge and peak water levels, while in others there is an underestimation. Apart from model inaccuracy there are different potential reasons for this, that can work together to cause this uncertainty. These will be explained in the discussion in section 4.1.

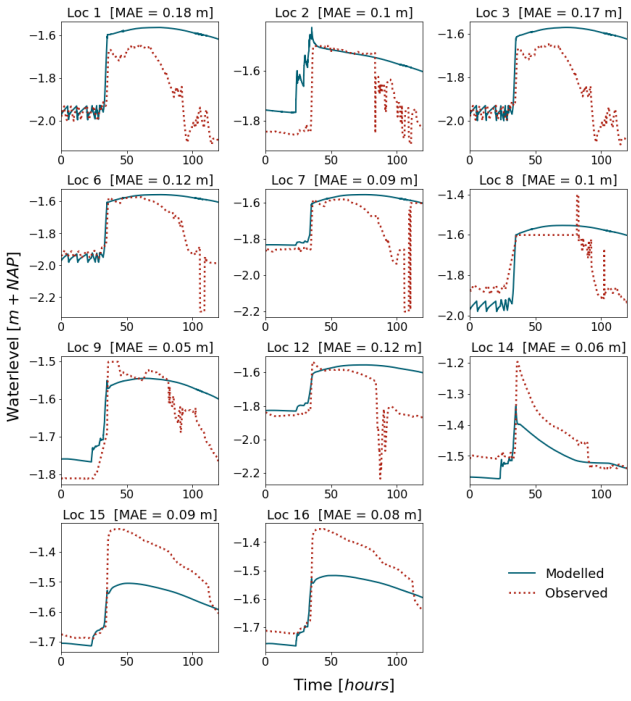
To conclude, the cumulative discharge through the pumping station is approximated reasonably well by the hydraulic model, with an average NSE of 0.77 over all calibration events. The water levels are also approximated well, with an average error of 0.07 metres over all calibration events. There are no significant systematic errors in the calibration events. For some events there is an overestimation, and for some an underestimation. The model also performs well on the validation event, with an NSE of 0.842 for the pumping station discharge, and an average error of 0.06 metres on the water levels. Because of this good performance on the validation event, the model is considered valid to be used for inundation predictions in the study area for the continuation of this study. The hydraulic model can produce 12 hour forecasts at a 10 meter resolution within 3 to 15 minutes on a computer with specifications according to A.5. The exact computation time depends on the amount of rainfall, with more rainfall leading to longer computation times. With more rainfall the amount of water in the hydraulic model increases, which generally leads to higher flow velocities. When flow velocities are higher, the time step of the model needs to be lowered to satisfy the Courant condition. Also with less rainfall there are fewer inundated 2D grid cells and therefore fewer cells that are actively considered in the calculations (Deltares, 2022b).



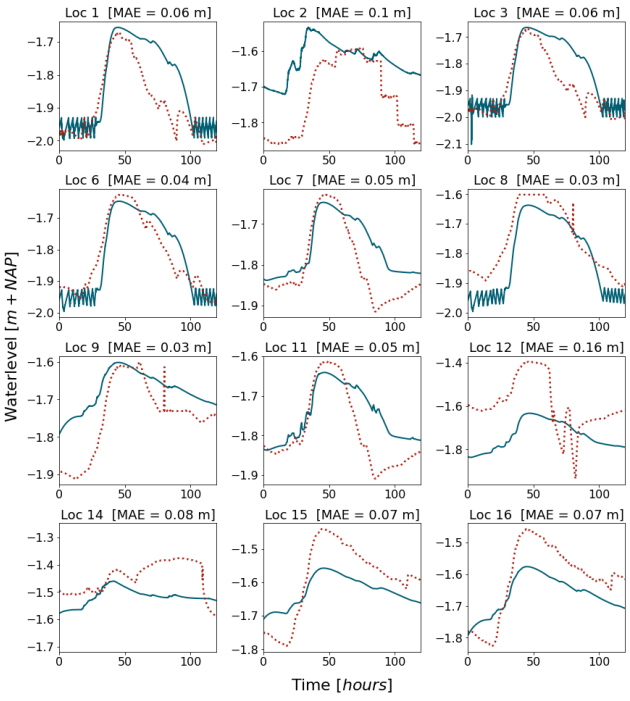
(a) Event 3 (heavy rainfall spread out over time)



(b) Locations of the water level measurement stations.



(c) Event 4 (heaviest rainfall)



(d) Event 5 (validation event)

Figure 3.3: *(a), (c) and (d) show the modelled (solid blue) and measured (dashed red) water levels, only stations where data is available are shown. (b) shows the locations of the measurement stations in the study area. Results of the other events can be seen in appendix A.7.*

3.2 Cloud computing

The model setup resulting from the model calibration is used to generate probabilistic inundation forecasts. Simulations are performed simultaneously by utilizing cloud computing, which reduces the total required computation time. This section will discuss the results of the probabilistic inundation forecasts generated with cloud computing. These results will help to answer research question 2. First, the required hardware for single simulations is analysed. With this information, a series of tests is done to find the best performing configuration of virtual machines.

3.2.1 Hardware resource requirements

The hardware requirements of a model need to be known such that appropriately sized VMs can be chosen to minimize the cost and computation time (section 2.2.5). To see which hardware characteristics are required for different model setups, the model resulting from section 3.1 is executed with different resolutions. For each resolution the CPU and RAM memory usage are tracked during the course of a simulation (figure 3.4).

The CPU usage is around 100% for all simulations, which was to be expected since a model is allowed to use one CPU. The RAM usage does differ depending on the model setup, with higher resolutions leading to more RAM usage. For the remainder of this section, the model with a 1D resolution of 20 metres and 2D resolution of 10 metres will be used. This is because this resolution provides sufficiently detailed output within reasonable computation time. For this setup a simulation uses 1 CPU core, and around 1 GB of RAM. Limiting the available RAM can be beneficial (appendix B), but will not be further investigated for this research.

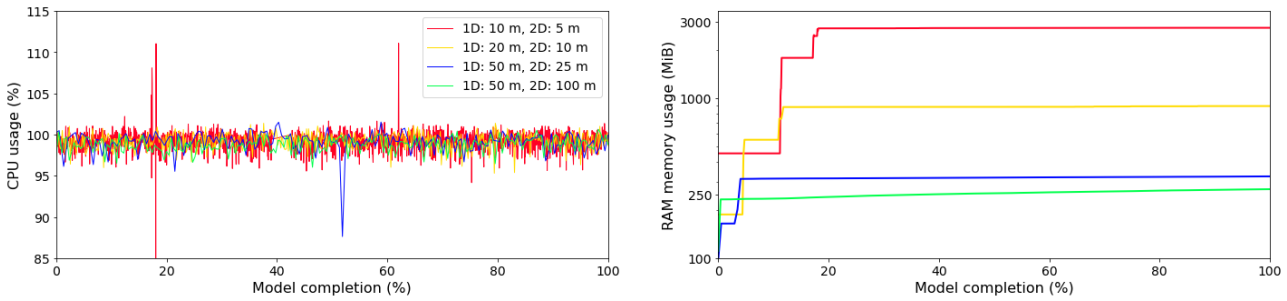


Figure 3.4: *CPU utilisation (100% means it uses one core) and RAM memory usage during a simulation. Greater than 100% means the computer is trying to do more work than it has capacity for, which can slow down computation slightly.*

3.2.2 Cost and computation time

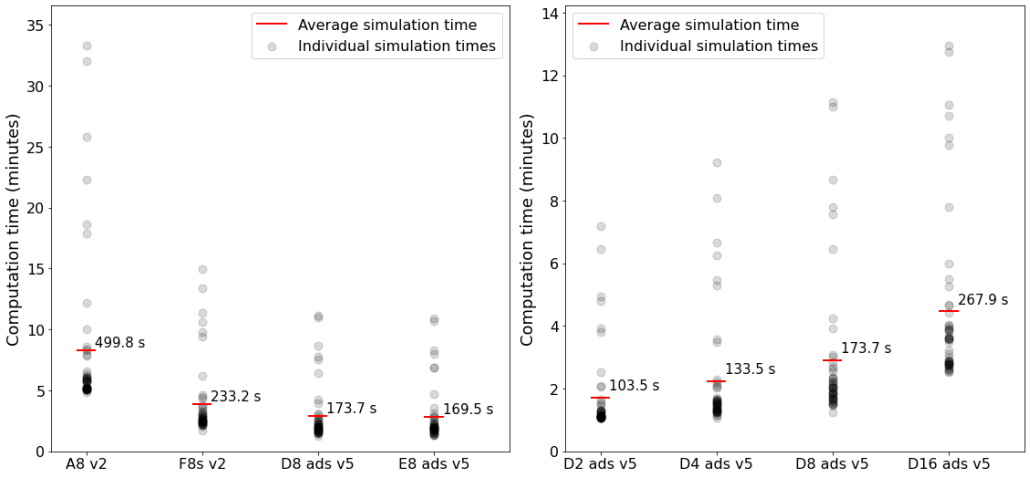
Tests are performed to find the best performing virtual machine setup (section 2.2). For each VM configuration, 50 ensembles members are simulated to obtain a probabilistic inundation forecast. Interesting are both the computation time and the costs of the different VM setups. In table 3.2 the number of simultaneous simulations per VM size can be seen. On each virtual machine, one simulation per CPU core is performed, with one core needing to remain unused. Ideally, all 50 simulations would be performed at the same time, however within an ensemble forecast there are large differences in rainfall per simulation, which causes signifiant differences in computation time. Since all simulations are required to make the probabilistic forecast, it is beneficial to run multiple fast simulations after one-another while the slower simulation is busy computing. When simulation times are more similar (for example when there is less rain), more simulations can be executed simultaneously to speed up the computation time.

Virtual machine types In the first testing stage different VM types are tested. The results can be seen in table 3.3, and simulations times are visualised in figure 3.5a. It is clear that the A8 v2 machine performs the worst in all aspects. Simulations on this VM are slow, causing the total workflow time to be long and the costs to be higher than the other setups. The F8s machine is significantly faster than the A8, however still slightly slower than the D8 and E8. The difference in computation time is due to the different CPUs that are present in the machines. The D8 and E8 both have the same CPU, and also show similar computation times. While the F8 machine is cheaper than both the D8 and E8, the total workflow cost for this machine is higher because of the

Table 3.2: *Simultaneous simulations per machine size. This assumes that sufficient RAM is available on the VM to host all simulations.*

CPU cores	Simultaneous simulations	% of CPUs used for simulations
2	1	50%
4	3	75%
8	7	87.5%
16	15	93.4%

higher computation times. The D8 and E8 show very similar performance, but the D8 is cheaper than the E8 causing a lower total cost. The D8 and E8 both have very similar configurations, with the main difference being the higher RAM for the E8. This does not affect computation time, however it makes this machine capable of running models that require more RAM. From the machines tested, the D series is considered to be the best for generating probabilistic inundation forecasts for the model setup used. This is because it is the most cost efficient, and provides low computation times.



(a) *Stage 1: different types of VMs.* ***(b)*** *Stage 2: different sizes of VMs.*

Figure 3.5: *Computation time of the simulations for the different VM setups.*

Virtual machine sizes For the D series, there are different VM sizes available. Larger VMs have more CPUs, RAM and storage, and thus allow for more simultaneous computations. Four different setups are tested, each with different VM sizes. The results can be seen in table 3.4, and simulation times are visualised in figure 3.5b. All simulations are limited to using 1 CPU core, and 1 GB of RAM, and therefore it was expected that regardless of the VM, the computation time would always be the same. This clearly is not the case, and this is further investigated in section 3.2.3. Both the fastest computation time for the full workflow and the lowest costs are achieved by using the four core machine, running three simultaneous simulations on each machine. While the two core simulations are fastest, the total workflow time is slower because only twelve simulations can be run simultaneously. The sixteen core machine is by far the worst performing. Despite having more cores available (32 instead of 24 for all other setups) it is the slowest out of all setups. Because more CPUs are used, the setup is significantly more expensive per hour than the others.

Table 3.3: *Results for different types of virtual machines. In each setup, three virtual machines of the given name are used, this means a total of 24 cores were available for each test.*

Name	Type	CPU's	RAM (GB)	Azure compute score	Cost ($ per hour)	Average simulation time (s)	Simultaneous simulations	Total workflow time (mm:ss)	Total cost to complete workflow
A8 v2	General purpose	8	16	52.295	0.38	500	21	43:28	$ 0,83
F8s v2	Compute optimized	8	16	136.027	0.39	233	21	20:51	$ 0,41
D8 ads v5	General purpose	8	32	153.951	0.50	174	21	16:08	$ 0,40
E8 ads v5	Memory optimized	8	64	153.765	0.63	170	21	15:22	$ 0,49

Table 3.4: *Results for different sizes of virtual machines. Setup 1, 2 and 3 all have 24 cores in total. Setup 4 has 32 cores in total.*

Name	CPU's	RAM (GB)	Azure compute score	Cost ($ per hour)	Machines hired	Average simulation time (s)	Simultaneous simulations	Total workflow time (mm:ss)	Total cost to complete workflow
D2 ads v5	2	8	38.919	0.13	12	104	12	17:25	$ 0,45
D4 ads v5	4	16	72.644	0.25	6	134	18	15:14	$ 0,38
D8 ads v5	8	32	153.951	0.50	3	165	21	16:08	$ 0,40
D16 ads v5	16	64	306.800	1.00	2	268	30	18:09	$ 0.60

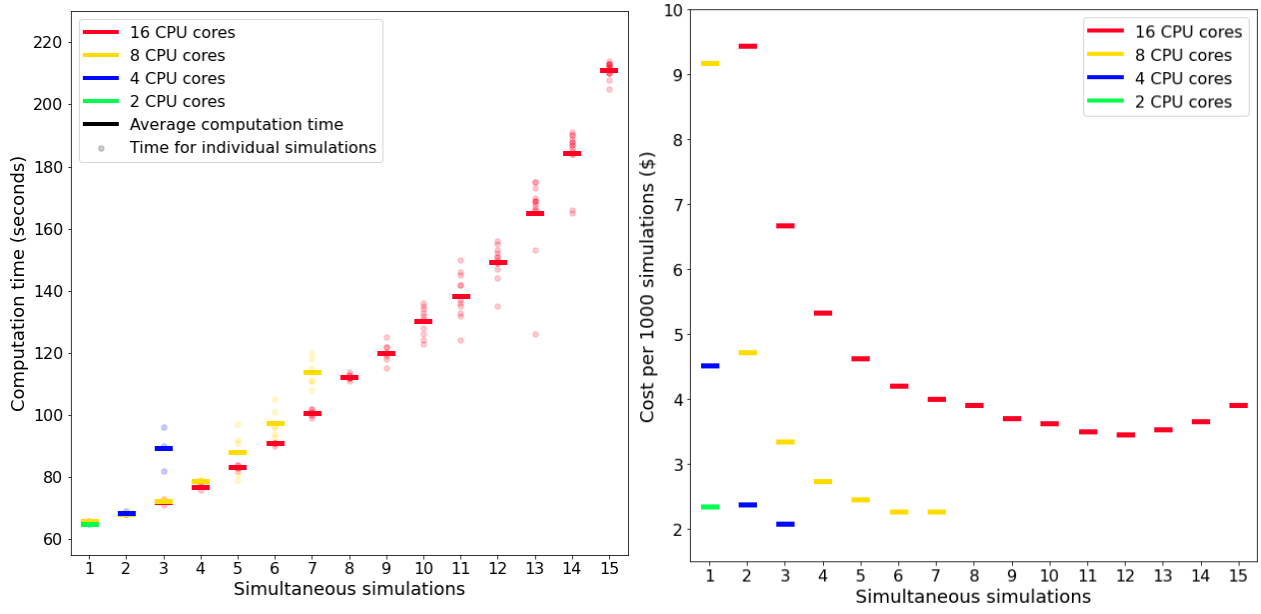
3.2.3 Individual simulation times

As mentioned in section 3.2.2 the simulation times are dependent not only on the type of VM, but also on the VM size. Larger VMs which use a larger percentage of their cores for simulations have slower computation times. To see how the number of simultaneous simulations affects the computation time, a series of tests is performed. For each of the VM sizes from section 3.2.2, 1 up to 15 simultaneous simulations are performed to compare computation times. This is done for same model setup and rainfall pattern each time. The resulting simulation times and costs per 1000 simulations for the different setups can be seen in figure 3.6.

Clearly, the simulation time increases when more simulations are run simultaneously. Interestingly, this does not scale directly with the percentage of the CPUs used for simulations. For example: 8 simulations on 16 cores is almost two times slower than 1 simulation on 2 cores, or 2 simulations on 4 cores.

Within the tests, it is made sure that only the actual simulation times are measured, and that no other processes (e.g. writing to shared storage) are executed during the simulations. For this reason, only the simulations themselves can be limiting the computation time. Since there is significantly more RAM available on the VMs than required by the simulations, the RAM cannot be causing the reduction in computation times. It is possible that differences in computation times occur due to the scheduling of threads on the CPU cores during a simulation. The vCPUs on a VM are not all linked to separate physical cores, and therefore allocation of time on the physical CPU cores could be the reason for the increased simulation times.

In figure 3.6b it can be seen that the most cost efficient option is performing 3 simulations on the 4 core machine. This was also the most cost efficient option for the full probabilistic inundation forecast, which can be seen in table 3.4, and therefore it will be used for the remainder of this research. There is no general best setup, as the best setup depends on the simulations that need to be carried out (required hardware), their computation times, and user preference (time vs cost efficiency). That being said, using the 4 core VM with 3 simulations is generally a good option for low cost D-HYDRO simulations. Simulation times are not optimal, but the simulations are most cost efficient for the tests performed for this research. When computation times are most important then the 2 core VM with 1 simulation is the best option. This setup achieves the lowest computation time, and does so at a low cost. Furthermore it provides flexibility when scaling the number of machines, since the VMs are relatively small. For example if you need to run one additional simulation, needing to hire an 8 core VM would not be efficient since it only uses a small part of its capacity.



(a) *Simulation times*

(b) *Cost per 1000 simulations*

Figure 3.6: *Comparison between different setups, varying the VM sizes and the number of simultaneous simulations. Only considering actual simulation time, ignoring time for initialisation and data saving.*

3.3 Neural network

In this section the accuracy of the neural network inundation predictions are discussed. First the performance on single simulations is analysed. Secondly, 50 ensemble members are combined to create a probabilistic inundation forecast. For both forecasts, the neural network performance is compared to that of the hydraulic model. Together, these results will help to answer research question 3.

3.3.1 Performance on single simulations

The network used in this section and section 3.3.2 has been trained for 2000 epochs. The network sees the full training dataset once per epoch, so in the optimisation the network saw each training sample 2000 times. 200 simulations are selected that are not used during the training and hyper parameter optimisation (section 2.3). This is the test dataset, and this dataset will be used to analyse the performance of the neural network on unseen data. The performance of the network on one of these testing events can be seen in figure 3.9. This event has been selected because it is the worst performing event out of all the 200 simulations in the test dataset. The rainfall during this event can be seen in figure 3.7. Even at this worst event, the network performs very well, and no differences stand out by visually comparing the two inundation maps. A map of the differences in maximum inundation depths for this event is shown in figure 3.10. The MAE for each of the testing events can be seen in figure 3.8, there generally is a higher MAE for events with more rainfall, and events where the rainfall starts at an early time step.

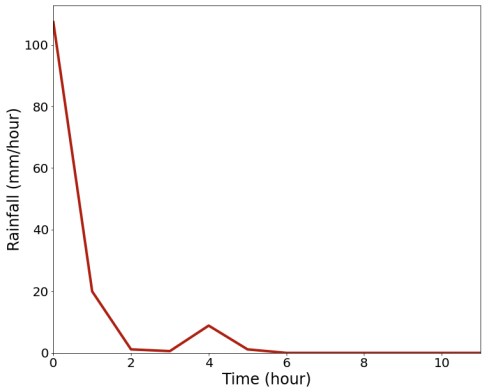


Figure 3.7: *Rainfall for the event shown in figure 3.9 and 3.10.*

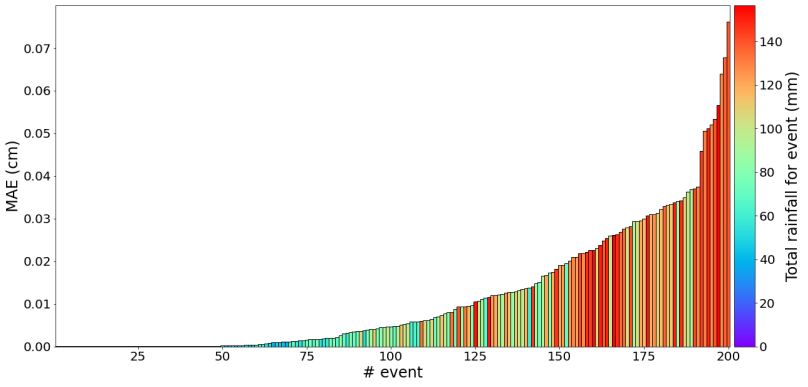
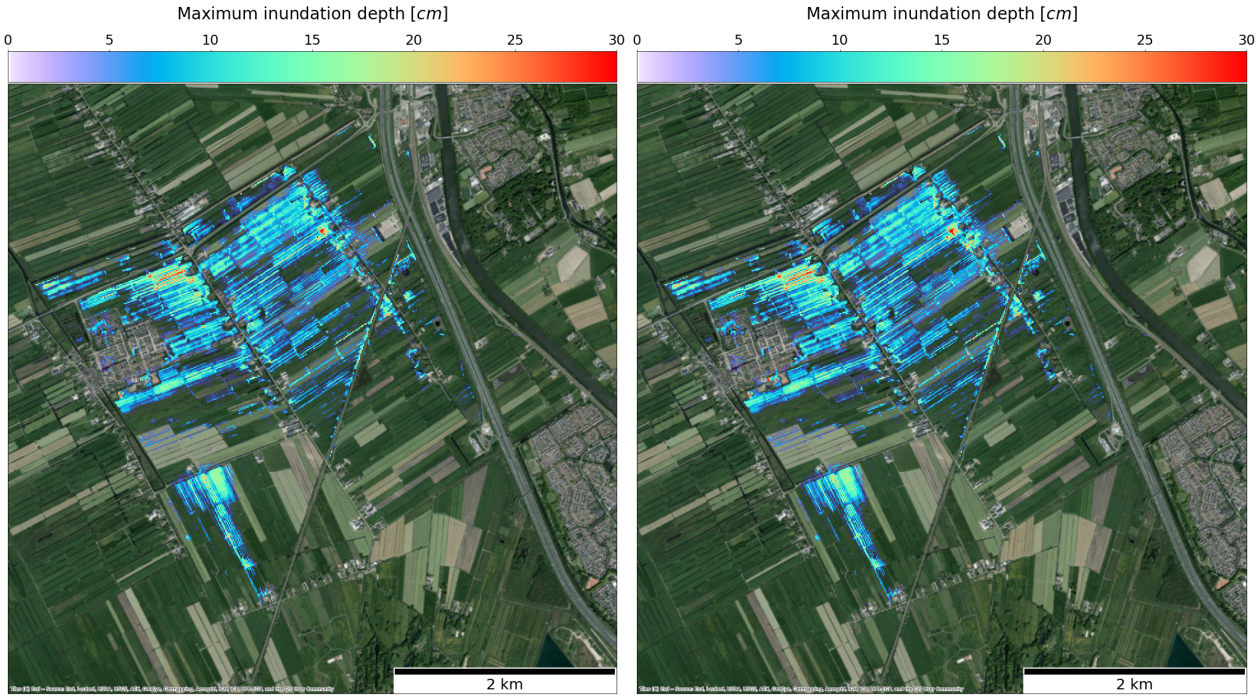


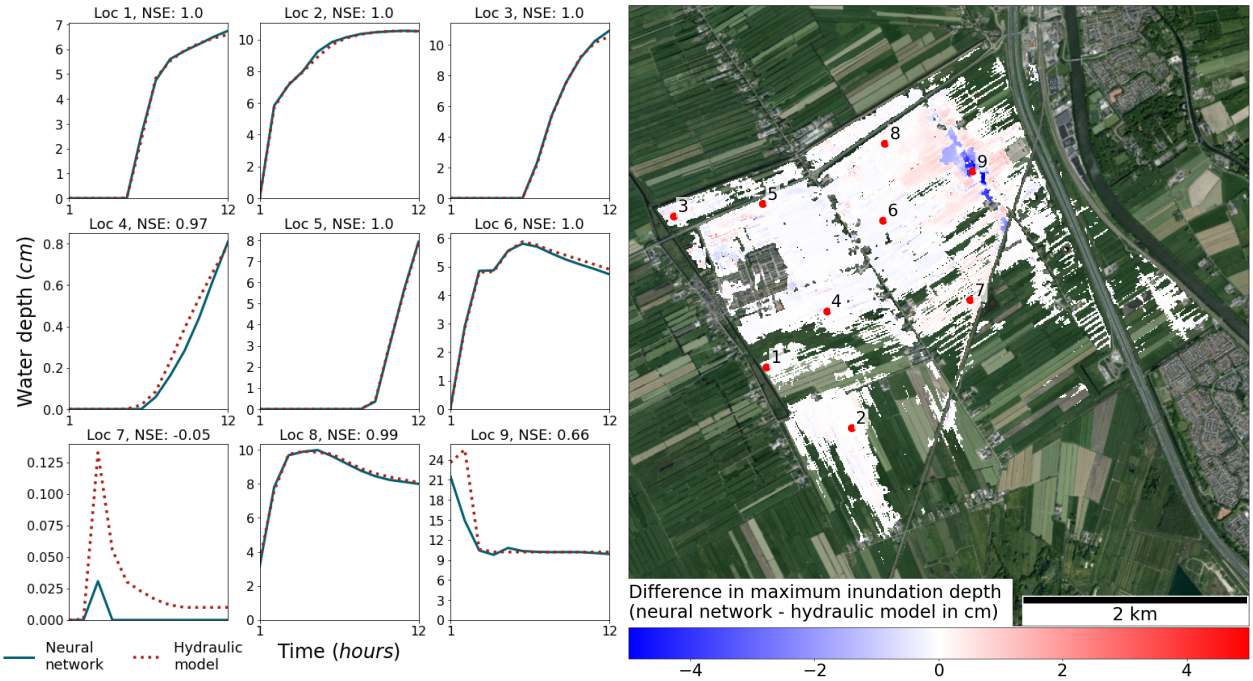
Figure 3.8: *Performance on all 200 testing events.*

In figure 3.10 a map of the difference in maximum water depth, and the predicted inundation depths over time for nine locations can be seen. From this figure, it can be seen that the progression over time is predicted very accurately, except at locations 7 and 9. The inundation depth at location 7 is very small (0.1 cm), so this is no significant error, and the reason for this error will be explained later with figure 3.12a. At location 9, inundation depth is predicted well, except at the first and second time step where the inundation depth is under predicted by a few centimetres. The same error occurred for grid cells nearby, resulting in significant errors in the maximum water depth prediction in this area. The differences between the neural network and hydraulic model for the rest of the domain are small. The error in maximum water depth predictions near location 9 is only present for simulations with very high intensity rainfall peaks.



(a) Neural network *(b) Hydraulic model*

Figure 3.9: *Inundation depths during a heavy rainfall event with total rainfall of 139 mm in 12 hours (figure 3.7). The differences in maximum water depth can be seen in figure 3.10b. This is the worst performing event of the test dataset, with an MAE of 0.076 cm, and an RMSE of 0.25 cm.*



(a) Inundation depth over time for the neural network (solid blue line) and the hydraulic model (dotted red line). *(b) Locations for which inundation depth over time is plotted, and the difference in maximum inundation depths.*

Figure 3.10: *Inundation depths over time during the same heavy rainfall event as in figure 3.9 with total rainfall of 139 mm in 12 hours (figure 3.7). This is the worst performing event out of the entire testing dataset.*

The scores on relevant performance metrics for all testing events are shown in table 3.5. In figure 3.11 a scatter plot of the inundation depth forecasts can be seen. A comparison is made for each grid cell, each time step, and each testing sample. Because there is a total of 128.114.400 data points to compare, a density plot rather than a scatter plot is used. The colour represents the density of points. Keep in mind that at some points in the scatter plot, point density is higher than in the colour bar. For example, the density is very high at (0,0), because this points represents the occasions where a cell is not inundated.

Table 3.5: *Performance on different metrics for the entire test dataset of 200 events. CSI are averages over all grid cells.*

Metric	MSE (m^2)	RMSE (m)	MAE (m)	CSI
Score	$6.98 * 10^{-7}$	$8.36 * 10^{-4}$	$1.15 * 10^{-4}$	0.964

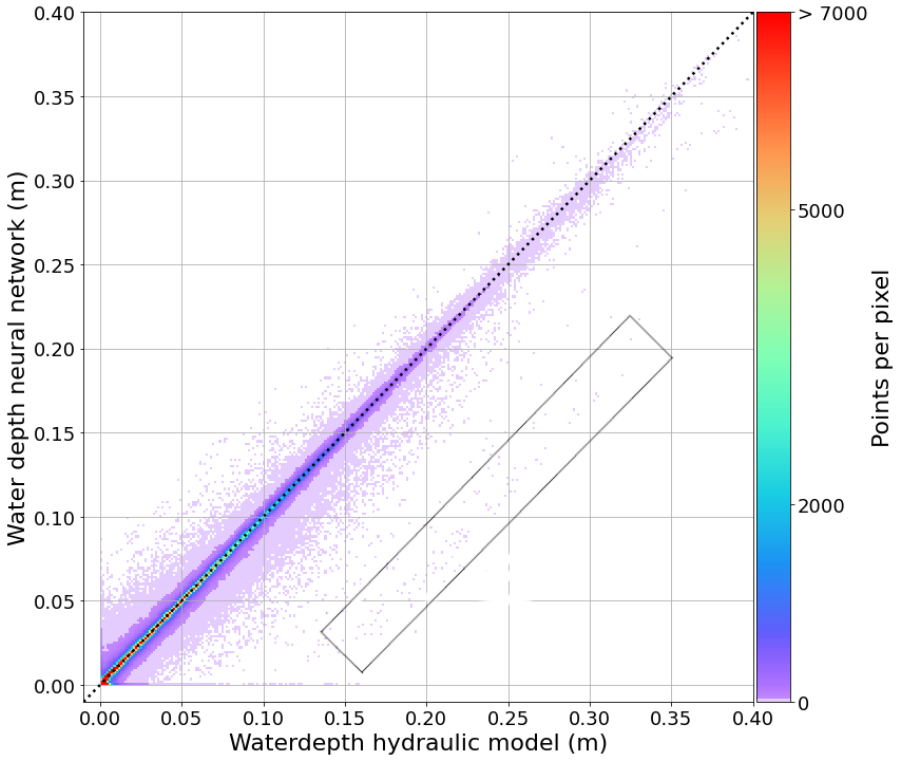


Figure 3.11: *Inundation predictions of the neural network versus the inundation predictions of the hydraulic model (testing data).*

From figure 3.11 it can be seen that the neural network performs very well at predicting the inundation depths across all events. Because of the choice of the ReLU activation function for the last dense layer, predictions are never below 0, as can be seen in the figure. The density around the x=y prediction line is high, and going further from this line the density drastically reduces. From the figure, there is no large bias towards either under or over prediction of inundation depths.

However, when zooming in towards (0,0), and changing the colour scale, a clear pattern is present (figure 3.12a). The neural network often tends to predict inundation depths of 0, while in the hydraulic model there was a small amount of inundation (generally < 1 cm). This is caused by the combination of the ReLU activation function and the MSE as an optimisation objective. Because of the ReLU activation, predicting 0 for the network is "easy", in the sense that everything below 0 will be set to zero in the last layer. When the network is optimising, it will quickly learn to not predict 0 when inundation is large, because this yields high MSE values. But when inundation is very small (below 1 cm), predicting 0 is not punished as harshly. When the inundation depth for a specific cell is frequently 0 and occasionally very small, the network does not have much motivation to learn this pattern since predicting 0 already results in a low MSE. As a result the network does not always

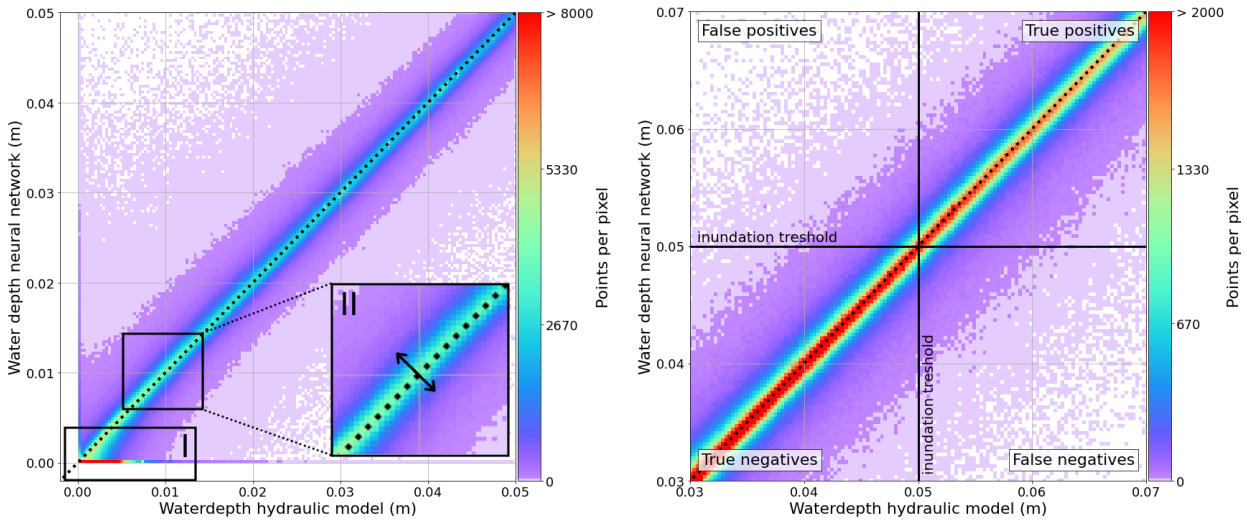
improve these errors, but since the errors are so small this is not necessary. The MSE puts more emphasis on larger errors, which is desired for achieving accurate inundation forecasts.

Another pattern which is present in the scatter plot is highlighted with the rectangle in figure 3.11. Within this rectangle there is a trend line in the predictions, which indicates the network is making a systematic prediction error. More of these "trend lines" are present when inspecting figure 3.11 closely, however the others are closer to the x=y line and therefore are less apparent. Such a trend line is caused by an occurrence where the inundation depths are estimated inaccurately in a particular area. This happened for some events where there was a large peak in rainfall, and right after this peak the network underestimates the increase in inundation depths the rainfall causes. A similar systematic underestimation in inundation depths is present in figure 3.10 near location 9, where the inundation depth at the second time step was systematically under predicted in a region around location 9. The relative prediction error is similar in the entire region, and therefore these "trend lines" emerge in figure 3.11.

The neural network is trained to predict inundation depths at 12 different time steps for various types of rainfall events. The performance of the network is evaluated at each time step and for different types of events in appendix C. The analysis shows that the network performs consistently well across all time steps. The neural network is able to accurately predict inundation depths regardless of the characteristics of the rainfall event, such as the distribution of the rainfall (e.g. high peak or spread out) and the timing of the peak.

Table 3.6: *Confusion matrix for the neural network predictions.*

		Neural network	
		Wet	Dry
Hydraulic model	Wet	2.850.161	59.596
	Dry	46.736	125.157.907



(a) *Zoomed in near 0* ***(b)*** *Zoomed in near inundation threshold*

Figure 3.12: *Zoomed in views of the scatter plot shown in figure 3.11. Note the difference in the color scalebar for the figures.*

Inundation classification To be able to make probabilistic inundation forecasts, it is important to be able to classify cells as either dry or wet. This is done by setting an inundation threshold at 5 cm. When the water depth is larger than 5 cm, a cell is classified as inundated. When it is smaller, the cell is considered dry. With this classification, the confusion matrix is made, which can be seen in table 3.6.

When excluding the correctly predicted dry cells, 96.4% (CSI) of the test data is correctly classified. The main source of error in classifying inundation is near the inundation threshold value. For instance, if the neural

network predicts an inundation depth of 4.99 cm and the hydraulic model predicts 5.01 cm, the classification would be incorrect, even though the network's prediction is very accurate. To address this issue, a margin of safety can be implemented around the threshold value. If both the hydraulic model and neural network predict an inundation depth within the margin of safety, then the classification of this grid cell will not be taken into account in calculating the CSI. The resulting CSI for different margins of safety can be seen in table 3.7. It can be seen that by taking a margin around the threshold, the CSI improves significantly. This proves the classification error indeed mainly comes from the predictions that are very close to the inundation threshold.

Table 3.7: *CSI for different margins around the inundation threshold.*

Margin	0 cm	0.1 cm	0.5 cm	1 cm
CSI	96.4%	98.6%	99.7%	99.9%

There is a very slight bias towards predicting false negatives (predicted dry but is wet) over predicting false positives (predicted wet but is dry). When looking closely, the reason for this can be seen in figure 3.12b. At the 5 cm inundation threshold there is a very slight bias towards predicting lower inundation depths. When the classification between wet/dry is made, predictions close to the threshold are biased to be slightly lower, and therefore a larger part of them are classified as dry when they should be wet.

The inundation threshold has an influence on this classification bias. For example, if the inundation threshold was set at 1 cm instead of 5 cm, the bias would likely be the other way around, which can be seen in figure 3.12a at II. Inundation thresholds below 1 cm are unreliable, because below this threshold the tendency for the network to predict 0 at small inundation depths becomes a problem. Inundation predictions below 1 cm are often not even considered by experts (Kabir et al., 2020; Zanchetta and Coulibaly, 2022; Zhou et al., 2021), and therefore this prediction error at very low inundation depths is not very relevant for inundation forecasts.

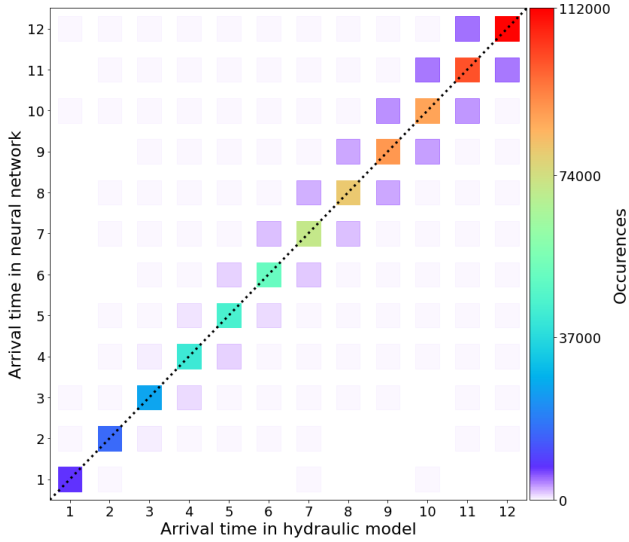


Figure 3.13: *Scatter plot of the arrival times (in hours) of the neural network and hydraulic model.*

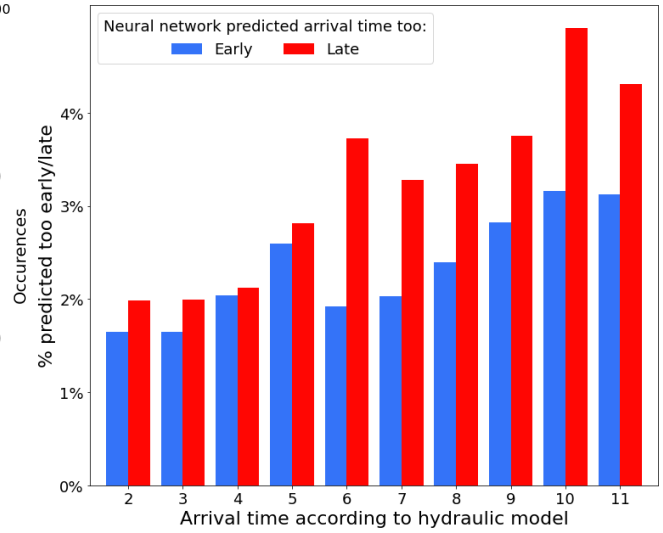


Figure 3.14: *The number of occurrences where the neural network predicts arrival times either too late or too early.*

The arrival time is the first time step at which a cell is inundated. In general, the network predicts the arrival times very well. For 94% of all grid cells the arrival time is predicted correctly. Important to remember is that the small errors near the inundation threshold are the main cause of the errors in the arrival time. The slight bias towards under prediction of inundation depths around the inundation threshold of 5 cm leads to arrival times being biased to be predicted a bit too late, however this effect is so small it is barely visible in figure 3.13. Therefore the prediction errors in the arrival time are plotted in figure 3.14. The network is slightly more likely to predict arrival times too late, however the difference is small. Mostly, the error is either one hour too late or one hour too early. Prediction errors of more than one hour are rare, only occurring 0.2% of the time.

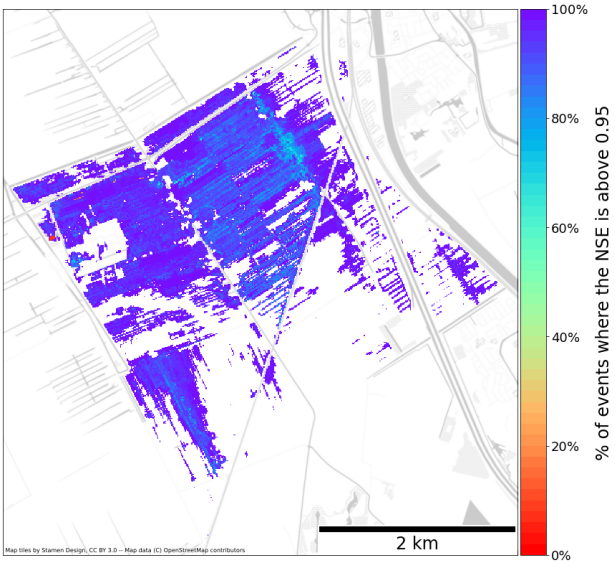


Figure 3.15: *The percentage of events for each grid cell where the NSE was higher than 0.95. Grid cells that are never inundated in the training dataset are not shown in this figure.*

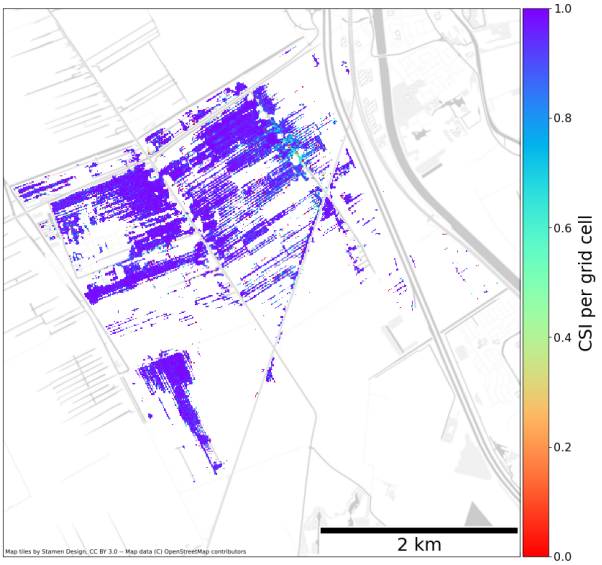


Figure 3.16: *The CSI for all grid cells in the study area. Grid cells that never have more inundation than 5 cm in the testing data are not shown in this figure.*

Spatial distribution of errors For each of the grid cells and for every testing event, the NSE is calculated. An NSE of 1 indicates perfect forecasting skill. The percentage of events where the NSE of a grid cell is above 0.95 is shown in figure 3.15. For most of the grid cells, the NSE is above 0.95 for the far majority of the events. Over the entire domain, the NSE is larger than 0.95 in 94.5% of the cases. This indicates near perfect forecasting skill in these cases. Here, a case refers to the NSE of a grid cell in one testing event.

In 2.3% of the cases, the NSE is between 0 and 0.95. This generally is caused by low inundation (on average 0.4 cm), which means the mean would already be a good predictor. This is difficult to improve upon by the neural network, as the network has little incentive to improve in areas where the objective function (mse) is small. At these locations the network performs better than the mean, however it does not perfectly predict the inundation depths.

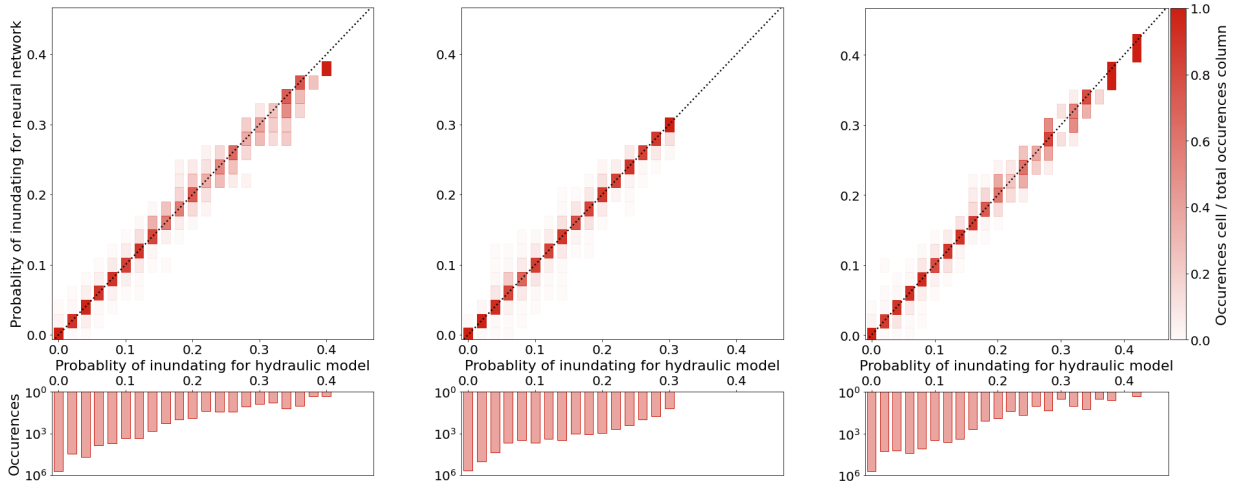
In 3% of the cases the NSE is lower than 0. These are locations with very low inundation depths (on average 0.1 cm), where the network has a tendency to always predict zero, as shown in figure 3.12a at I. Because of this tendency, the mean is sometimes a better predictor than the neural network, despite that the actual prediction error is often very small at these locations.

The resulting CSI for all grid cells can be seen in figure 3.16. In general, the network performs well. At most locations, the CSI is near 1, indicating that the network almost always classifies these cells correctly. There are some cells with lower CSI values. This mainly is at locations where inundation larger than the inundation threshold occurs rarely, meaning a few errors can lead to a low CSI. When this is the case, there are only a few cases of inundation, meaning that a small number of wrong classifications can drastically lower the CSI value.

3.3.2 Performance on probabilistic inundation forecasts

The goal of the neural network is to be utilised for generating probabilistic inundation forecasts, based on ensemble rainfall forecasts. To do so the inundation classification for all 50 ensemble members are combined into a probabilistic inundation forecast. The reliability diagram is a 2D density plot of the neural network inundation probabilities and the hydraulic model inundation probabilities for all 53.381 grid cells at all 12 time steps. The resulting points are coloured based on how often they occur. The reliability diagrams for each of the three testing events can be seen in figure 3.17. Furthermore, the Brier score for each of the events is calculated and can be seen in table 3.8.

From both the Brier scores and the reliability diagram it can be seen that the neural network performs really well in generating probabilistic inundation forecasts. This was also expected, since the network already has shown a good performance on inundation classification for deterministic simulations in section 3.3.1.



(a) Event 1 *(b) Event 2* *(c) Event 3*

Figure 3.17: *Reliability diagram for each of the events, each event consists of 50 ensemble members. The occurrences show how many times a certain probability occurs according to the hydraulic model.*

In general the performance of the neural network on probabilistic inundation forecasts is mainly limited by the accuracy around the inundation threshold, shown in figure 3.12b. When a significant number of rainfall events in a given ensemble forecast result in inundation near the threshold value, there is an increase in classification errors. When inundation is either significantly higher than the threshold, or significantly lower, the classification error reduces significantly, making the neural network more accurate in generating probabilistic inundation forecasts.

Performance is good for all three testing events, but especially the diagram of event 2 stands out by displaying a very high reliability. The reason for this high reliability is due to rainfall patterns of event 2. The ensemble forecast shows a few ensemble members with a high rainfall, and many members with quite a low rainfall. This results in a few simulations where many grid cells are inundated, and many simulations where there is little to no inundation. This makes it easier to predict the probabilities, since for the ensemble members with low rainfall it is easy to predict no inundation, which contributes for a large part to the probabilities shown in the diagram. For the ensemble members with very high rainfall, more grid cells will have an inundation significantly larger than the inundation threshold, which also reduces the classification error. The main reason for errors in the inundation classification are caused by inundation depths that are close to the threshold, because a small error in the predicted inundation depth can lead to a wrong classification. In this case the inundation depth is very close to the threshold anyway, so it is not such a relevant issue that the classification is wrong.

The network predicts the exact same probability as the hydraulic model for 91% of the grid cells. For 99.6% of grid cells the neural network inundation probability is within 2% of the probability predicted by the hydraulic model. These numbers exclude the grid cells where both the neural network and hydraulic model predicted a 0% chance of inundation. From these percentages it is clear that the neural network performs very well at making probabilistic inundation forecasts.

Table 3.8: *Brier score for each of the ensemble forecast testing events.*

Event	1	2	3
Brier score	$4.9 * 10^{-6}$	$7.4 * 10^{-6}$	$7.3 * 10^{-6}$

4 Discussion

4.1 Hydraulic model

4.1.1 Validation of inundation depths

The primary objective of the hydraulic model is to predict inundation depths. However, as there is no observational data available, the model is calibrated to reproduce discharge at the pumping station de Tol and water levels at 16 different observation stations, rather than being calibrated to inundation observations. If inundation data would be available, the hydraulic model could be further calibrated and validated for predicting inundation depths. However since the main purpose of this study is to explore methods of reducing the computation time of inundation predictions, this will not influence the main findings of this study.

4.1.2 Model schematisation

During the calibration, target water levels of 2022 were used to manage structures (weirs, pump and inlets). However, when extreme rainfall is expected water managers try their best to prepare the water system such that impact of the rainfall is reduced. These decisions are not always reported and therefore they are not included in the model, despite that they might have affected the observational data that the model is calibrated on. Including the measures in the hydraulic simulations could have improved the quality of the calibration, however since this data was not available this was not possible.

The catchments of the rainfall runoff model are currently based on the fixed drainage level areas. During additional testing it was found that having smaller catchments in the rainfall runoff model had a significant impact on the resulting inundation. Each catchment is connected to one location in the 1D model where it can exchange water. With more catchments, there are more locations where water is exchanged between the 1D model and the rainfall runoff model. When the hydraulic model would be applied in practice, it is recommended to further investigate the influence of the rainfall runoff catchment size on the resulting inundation.

4.1.3 Rainfall forecasts

Another reason for differences between model predictions and observations are the temporal and spatial resolution of the rainfall data. The rainfall data used during the model calibration is based on radar with a spatial resolution of 1 km^2. At this resolution it could be that some local differences in rainfall are not accurately represented in the data (Ochoa-Rodriguez et al., 2015). Furthermore rainfall based on radar data is not always consistent with actual rainfall (Di Curzio et al., 2022). This may lead to discrepancies between the actual rainfall and the radar-based rainfall data used in the calibration, potentially resulting in inaccuracies in the simulation as the rainfall in the model may not match the real-life events. Furthermore the rainfall data has a temporal resolution of 1 hour, which is significantly coarser than the 5 minutes advised by Ochoa-Rodriguez et al. (2015). Despite that data is available at 1 hour intervals, the measurements which the data is based upon are done multiple times per hour, so this error is somewhat limited. Still, averaging per hour can cause signifiant errors in the arrival times of rainfall events. For example: an event could either occur at 00:01 or 00:51, however in the data there would be no difference, and therefore the hydraulic model cannot predict inundation arrival times at finer temporal scales then one hour.

4.1.4 Initial conditions

The initial conditions for a simulation can play a significant role in the outcome. To represent the initial conditions, a dry warm up period of 1 month was used. Next to that, before each calibration event an additional 7 days were simulated to expose the simulation to the same external forcing as in reality. Despite this the initial conditions that occurred in the real world can still be quite different to those simulated in the model. The model is calibrated for extreme events, thus the performance during low rainfall periods cannot be guaranteed. During low rainfall periods the inflow to the system is dependent on the operation of the pumping station, inlets, and the weir levels. As mentioned above, the real-world operation of these is not always consistent with the provided data because of human interference. This causes the initial conditions of the model to be different than those of the real world, which can have a significant impact on the resulting water level increase and inundation. To improve the accuracy of the initial conditions, a separate model that is calibrated during periods of regular rainfall could be used. Additionally, information about how water managers prepare a study area for heavy rainfall would need to be included in the models initial conditions.

4.1.5 Generalisation

The method used to establish the model schematization can be adapted to other similar study areas. The entire process of model schematization is carried out through scripting, making it easy to replicate the process with the availability of appropriate input data. However, if the study area has distinct hydraulic characteristics compared to Polder de Tol, certain modifications to the scripting may be necessary. For example, in the case of a study area with a large river flowing through it, it would be important to consider upstream river discharge to ensure an accurate representation of the study area in the model.

The Monte Carlo calibration methodology can also be applied to other study areas. In this case, relevant observational data in the study area need to be acquired. Ideally this includes inundation data. If the study area has different hydraulic characteristics there might be different calibration parameters to consider, and additional input data might need to be used (e.g. upstream river discharge). A disadvantage of the Monte Carlo calibration is that is computationally expensive, which could cause issues with a more complex model schematisation.

4.2 Cloud computing

4.2.1 Practical applicability

With the developed workflow, probabilistic inundation forecasts can be generated automatically as soon as expected rainfall is above a certain threshold. The results can be visualised spatially as inundation probabilities, maximum water depths, and arrival times. With such a system, the predictions can be sent directly to crisis management teams, that can immediately use the information to minimize the damage of heavy rainfall events. Further post processing steps could be applied to aid the crisis team, such as expected damages and the number of inhabitants that are in danger.

4.2.2 Generalisation

The cloud workflow that is made for this study can be applied to any study area, given that that a hydraulic model is available. With the current workflow there is one limitation to this. The time to make a probabilistic inundation forecast is limited to the slowest simulation of the ensemble forecast. When a more complex model would be used, computation times of single simulations could be significantly longer than they have been for this study. When the computation time of the slowest single simulation is too long to be used operationally, the current cloud workflow will not be able to generate probabilistic inundation forecasts fast enough to be used operationally.

During this research the D-HYDRO modelling software was used. The methodology for the cloud workflow can be applied to other software, under the condition that the modelling software can be containerized. An example of a popular software package that would be suitable for this is HEC-RAS. When a different modelling software is used, the cloud workflow would have to be modified to satisfy the input and output of the specific modelling software.

4.2.3 Potential and further research

With the developed cloud workflow it is not possible to significantly speed up single simulations. Running the model on multiple CPUs does provide a small speed up, however the speed up is not always sufficient and cannot be scaled over multiple machines. A potential solution for this is model partitioning. By dividing the model domain in several smaller domains, the models for each of the domains can be executed simultaneously and exchange information at the boundaries. This would allow for more complex models to be used in an operational setting. But the required infrastructure comes with additional challenges. Different simulations depend on each other and need to exchange information. This provides new challenges for the orchestration and communication between containers, especially when combined with running multiple ensemble members in parallel.

The developed workflow could be improved by having a further consideration of individual computation times. Currently, simulations are sorted by the total rainfall, such that the simulations with longer computation times (more rainfall) are executed first. If computation times could be estimated accurately before carrying out the simulations, more efficient orchestration of containers can be made. For example, when all computation times are similar, more simulations would be run in parallel. When a few simulations have much higher computation times, less simulations would be run in parallel, such that multiple fast simulations are run sequentially in the same time of a slower simulation.

In the current workflow, a fixed warm up simulation was used for all new simulations. When used operationally, this does not always accurately reflect the initial state of the system. This could be improved by updating the warm up simulation at fixed intervals, based on the real life external forcing. This way, the initial state of the system would be more accurately reflected in the model, yielding better inundation predictions. By updating the warm up simulation at fixed intervals, this does not need to happen at the moment a probabilistic inundation forecast is required. Simulations of the initial conditions are longer than those of the heavy rainfall events, so simulating the initial conditions beforehand has a significant impact on the computation time for a probabilistic inundation forecast.

4.3 Neural network

4.3.1 Limitations

A disadvantage of the neural network that has not been mentioned yet is the rigidity. With a hydraulic model, changes in the study area can easily be implemented by changing the model schematisation. This can be used to analyse the effect of certain measures on the inundation. The neural network does not provide this flexibility. The network is trained on a large number of hydraulic simulations, that all need to be performed beforehand. When there is an important change in the study area, a new training dataset would have to be generated. This would require all hydraulic simulations to be performed again, with the new model schematisation. Afterwards, the network has to be trained again on the new dataset. When predictions need to be made often and few changes to the model schematisation are required this is no immediate problem, but for heavy rainfall events predictions are needed rarely. This could make it more efficient to perform the hydraulic simulations when predictions are needed, instead of training a neural network to do so.

There are a few adjustments that could potentially enhance the performance of the neural network. One such improvement would be to select the best-performing network on the validation dataset instead of the final network after the last epoch. Another potential enhancement would be to vary the learning rate throughout the training process. Currently, a fixed learning rate is used throughout the entire training process. At the beginning of the training, the network needs to make large adjustments to fit the data and escape local minima, while later on, the adjustments need to be more subtle. By implementing a decaying learning rate, the network will be able to make these larger adjustments at the start and smaller adjustments later on. This has proven to be able to improve network performance (You et al., 2019), but it is not tested if this improves performance for this case specifically.

Currently, the network is trained on inundation simulations as real-world inundation data is not readily available. In an ideal scenario where the network had access to real-world inundation data, its performance may not be as strong. This is because in reality, there are many other factors beyond rainfall that can impact inundation. While some of these variables can be incorporated as inputs for the network, additional variables would require more training data to make accurate predictions. Additionally, there may be variables that influence inundation but cannot be included as inputs for the neural network, resulting in a level of unpredictability in the inundation that cannot be captured by the network, potentially leading to lower accuracy than obtained this study.

4.3.2 Generalisation

The neural network in this study has been applied to polder de Tol. It is expected that the network would perform similarly well if trained for other study areas, when it is trained on hydraulic simulations for that specific area. Further research should be conducted to evaluate the performance of the neural network in areas with greater topographical variations, urban environments, or regions that are heavily impacted by external factors such as a river flowing through the study area.

For this study, the inundation is predicted for every grid cell in the study area at a 10 meter resolution. This provides a good insight over the entire study area, and also shows the capabilities of a neural network. For study areas with a larger domain, predicting inundation depths at a 10 meter resolution is expected to also be possible, however it does come with additional challenges. Firstly the training data would be more expensive to generate, since the hydraulic simulations at a higher resolution have to be performed. Secondly the neural network has to feature more parameters which will influence the training time. If a network would be used operationally it might not be necessary to predict inundation for every grid cell. Locations of interest can be selected beforehand, and the network could be trained to only predict inundation at these locations. This would allow for less complex network structures (e.g. Zhou et al., 2021), which reduces the training time.

A challenge for further research lies in researching the possibilities of a generalised neural network for inundation predictions, which can be applied to many areas (without needing to be trained specifically for new areas). The input of this network would need to include geospatial information. Training a generalized network is significantly more difficult than training a network for a specific area. Rather then remembering relations between a grid cell and predetermined input variables, the network would need to have an understanding of how the geography of an area affects inundation. It is unlikely that such a network would reach anywhere near the same accuracy as a network that is specifically trained for a certain area. However it would allow for rapid inundation predictions without needing to setup a hydraulic model for an area. Possibly a convolutional neural network would be better suited to such an application, because of its ability in interpreting rasters (Bentivoglio et al., 2021; Guo et al., 2021). Also physics-informed neural networks could be well suited to such a task, as they leverage governing physical equations in neural network training (Cai et al., 2021). The network in this study relies only on the training data to discover patterns between rainfall and inundation in a certain grid cell. A physics-informed neural network could be trained to satisfy both the training data as well as the governing physical equations.

4.3.3 Potential

The neural network has proven to be very effective in predicting the inundation patterns as a result of rainfall. In reality there are more variables that influence the inundation than just the rainfall. For example, an important contributor would be the initial ground water level, since this has an impact on the ability to store water in the study area. The neural network used for this study treats the initial ground water level as a constant, that does not change in any of the simulations in the training dataset. It would be relevant to investigate how well the network would perform if important variables like the ground water level would be variable within the training dataset, and would also be included in the network as inputs. Since the network has performed so well with only rainfall as an input variable, it is expected that it also would be able to learn additional patterns. The number of extra variables that can be considered should be limited to only the most important variables. When too many variables would be considered, the network would either not be able to learn all patterns, or it would require too much training data to do so.

The initial state of hydraulic system also is an important source of uncertainty. With the low computation times of the neural network, the possibility arises of including the uncertainty in the initial conditions within the probabilistic forecast. In this case the network would need to be able to predict inundation based on rainfall and the initial state. When considering an ensemble of initial states, together with the ensemble of rainfall forecasts, the neural network could create probabilistic inundation forecasts that do include both the uncertainty in the initial state of the system, and the uncertainty in the rainfall forecasts.

The neural network performance is dependent on both the quality and quantity of the training data. For this study, 1600 training samples were used. This number was chosen arbitrarily, since the effect of the number of training samples on the accuracy of the network was not known. For this reason, a series of tests has been done by training the same network with a different number of unique training samples. The results of this test is presented in appendix C.3, and the scatter plots of the inundation depths can be seen in figure C.5. Clearly, the number of training samples affects the performance. More training data leads to higher accuracy. Especially when the network is trained for a longer period of time, it is able keep improving when there is sufficient training data. When there is less training data, the network is prone to over-fitting. That being said, training on 200 samples already yields a good accuracy, especially in comparison to the error of the hydraulic model itself. It would be interesting to see how the number of required training samples would change when more input variables would be considered.

Including the time component can provide useful insight about arrival times of inundation. Additionally it can help the neural network understand how the rainfall distribution over time can influence the inundation. But by including the time component and using an LSTM, the neural network requires more training data to learn all patterns (Lu et al., 2021). In some cases, it might be more efficient to use a neural network that can only predict maximum inundation depths.

4.3.4 Practical applicability

The ability of the neural network to produce high-resolution inundation forecasts in real-time has numerous practical applications. Firstly, there is the application to operational inundation (probabilistic) forecasts, as was the goal of this study. The neural network is capable of producing highly detailed probabilistic forecasts within a second, which is orders of magnitude faster than what is achievable through traditional hydraulic

modelling methods. A key benefit of this ability is that it allows for the instant communication of expected inundation based on rainfall forecasts to water managers. Additionally, the low cost associated with generating new forecasts allows for frequent updates, incorporating the most recent rainfall forecasts.

Another potential use case of the neural network could be in education. Typically, the process of making inundation predictions is limited to experts in the field. However, by integrating the neural network with a user-friendly interface, it could be used to educate individuals who may not have prior experience with modelling, such as students or water managers. By allowing users to interact with model inputs, they can gain a deeper understanding of how variables such as rainfall can affect inundation and explore the associated risks under different conditions.

4.3.5 Other studies

It is difficult to directly compare metrics with other studies. This is because there are more factors than just the performance of the neural network that affect these metrics. The reviewed studies include both pluvial and fluvial studies. Especially for studies that concern fluvial floods, inundation depths and flow velocities are higher. This leads to significantly larger inundation depths and rising speeds than in this study, which causes larger errors in inundation depth predictions. That being said, compared to other studies where the metrics are reported the neural network in this study has provided the best critical success index (Besseling, 2022; Zanchetta and Coulibaly, 2022). This study also provides a significantly lower mean absolute error and root mean squared error than all other reviewed studies (Berkhahn et al., 2019; Zhou et al., 2021; Chang et al., 2018; Chang et al., 2014; Zanchetta and Coulibaly, 2022; Guo et al., 2021; Kabir et al., 2020; Besseling, 2022; Kilsdonk et al., 2022). This can partly be attributed to the fact that this study features generally small inundation depths, but also proves how well the network can predict these. Furthermore the Nash–Sutcliffe model efficiency coefficient is performing well compared to other studies (Kilsdonk et al., 2022; Besseling, 2022; Kabir et al., 2020), but it is difficult to directly compare the average value for this metric since it is sensitive to outliers. This is the first study where a neural network is used to generate probabilistic inundation forecasts, and therefore the performance on probabilistic forecasts cannot be compared to other studies. Compared to the studies mentioned, the network in this study is trained on a significantly larger training dataset. Partly because of this, the network has shown great ability in generalising the information of different rainfall events. It uses this generalisation to accurately predict inundation patterns for new rainfall events, regardless of their distribution over time.

5 Conclusion & Recommendations

5.1 Conclusion

The objective of this study is to test and compare two different methods of making probabilistic inundation forecasts accurately and fast enough for operational use. Four research questions were set up to guide the research and reach the objective of the study. Each of these will be answered in this chapter.

RQ1: How fast and accurately can a hydraulic model replicate historic water levels and discharges during extreme rainfall events in polder de Tol?

The cumulative discharge through the pumping station is approximated well by the hydraulic model, with an NSE of 0.84 on the validation event. Within the calibration and validation events there are some inaccuracies on certain events, however there are no systematic errors. For some events there is an overestimation, and for some an underestimation. The causes for these errors can partly be attributed to the quality of the observed discharge and water level data used during the calibration. The water levels are also approximated well, with an average error of 0.09 metres on the validation event. The hydraulic model can produce 12 hour forecasts at a 10 meter resolution within 3 to 15 minutes. The exact computation time depends on the computer specifications and the amount of rainfall, where more rainfall leads to longer computation times.

RQ2: What are the cost and computation times for generating probabilistic inundation forecasts using cloud computing?

With the cloud workflow that is designed for this study, probabilistic inundation forecasts based on 50 ensemble members are generated in the computation time of the slowest ensemble member simulation. The total computation time is around 15 minutes for the forecast used during this study. Generating a full probabilistic inundation forecasts based on 50 ensemble members costs around 0.40 euros. When the same probabilistic forecast would be made by running each simulation sequentially on a regular computer, it would take several hours. The same methodology can be applied to other study areas, where the costs and computation time depend on the model characteristics. The key factors that have the greatest impact on computation time and cost are the size of the study area, the spatial resolution, and the amount of rainfall. Forecasts only have to be made a limited number of times per year, when very heavy rainfall is expected. With each forecast costing around 0.40 euros the total costs for keeping up a forecasting system are low.

RQ3: How accurate can a neural network replicate the deterministic inundation depths and inundation probabilities of the hydraulic model?

The neural network is able to very accurately predict both the inundation depths and inundation probabilities of the hydraulic model. The neural network can accurately predict inundation depths for 53.381 grid cells, at 12 time steps. It has the ability to accurately predict inundation regardless of when rainfall starts, which is important for making probabilistic inundation forecasts since different ensemble members can have rainfall starting at different times. The mean absolute error on 200 testing events is only 0.0115 cm. This makes that error of the neural network inundation predictions is small compared to the error in the water level predictions in the hydraulic model itself, which was 9 cm. These two metrics can not directly be compared as the latter concerns water levels within the channels, however there is a significant difference in order of magnitude. The neural network performs very well in making probabilistic inundation forecasts: For 99.6% of grid cells the neural network inundation probability is within 2% of the probability predicted by the hydraulic model. The neural network is able to predict all 50 ensemble members in less than 1 second. The network underwent training using 1600 hydraulic simulations over a period of 2000 epochs. It is also possible to train the network faster on less samples, but this will affect the accuracy.

RQ4: How do inundation predictions made with cloud computing and the neural network compare in terms of computation time, accuracy and costs for generating probabilistic inundation forecasts?

The neural network is significantly faster than the cloud computing workflow, with the neural network being able to make a probabilistic inundation forecasts within 1 second. The cloud workflow takes around 16 minutes. Both methods provide a similar accuracy. The neural network can never be more accurate than the hydraulic model, since the network is trained on the hydraulic simulations. But since the neural network is so accurate in predicting the inundation depths of the hydraulic model, the error in the hydraulic model compared to reality

is far greater than the error of the neural network compared to the hydraulic simulations.

Then lastly there are the costs. These are difficult to compare since the associated costs for both methods are coming from different sources. The cloud workflow requires less investment, and costs are based on the number of forecasts made. The neural network needs a larger initial investment to train the neural network, but after this making probabilistic inundation forecasts comes at a negligible cost. For the study area, training the network on 200 samples would reach sufficient accuracy (possibly less depending on the criteria). In that case the cost of generating the training data would be around 4 times the cost of a single probabilistic inundation forecast with the cloud workflow. While there are other costs associated with setting up a neural network, it is clear that training a neural network would be cheaper than the cloud workflow when more than a handful of predictions with the same model schematisation need to be made.

Research objective Which method is preferred mainly depends on the application. In table 5.1 the key differences between both methods can be seen. The neural network is suitable for situations where predictions need to be made often, or when there are few changes required to the model schematisation. In these cases, it is cheaper, much faster, and provides similar accuracy to cloud computing. However when frequent changes to the model schematisation are required or when predictions are only needed rarely, using cloud computing to perform the hydraulic simulations is the cheaper option. Using cloud computing is infeasible when the simulation time of a single model prediction is too slow to be used operationally. Using the neural network is difficult when there are too many variables that affect the inundation, as this means the required training data would increase rapidly.

Table 5.1: *Key differences between both methods for generating probabilistic inundation forecasts.*

Hydraulic model with cloud computing	**Neural network**
Prediction time depends on computation time of hydraulic model	Predictions can be made instantly
Flexible, hydraulic model can be changed whenever required	Rigid, additional costs required when changes need to be made to the hydraulic model
All parameters of the hydraulic model can be varied	Limited number of input variables
Repetitive cost for each probabilistic inundation forecast	Negligible cost per probabilistic inundation forecast
No initial costs before predictions can be made	High initial cost of generating training data for neural network

5.2 Recommendations

This study has yielded important findings that not only have practical implications, but also provide direction for future research. The recommendations stemming from this study can be broadly divided into two categories: those for the practical application in an operational context, and those that suggest avenues for further research.

5.2.1 Application in an operational setting

- **Inundation records** An improvement to the model could be made by being able to calibrate and validate the model with actual inundation data. For this research, this data was not available. By gathering satellite imagery when inundation occurs, hydraulic models that predict inundation can be validated on inundation records. This can be beneficial to water mangers when setting up hydraulic models for inundation predictions.
- **Automated cloud infrastructure hiring** With the current setup, the cloud infrastructure needs to be manually turned on/off. Inundation predictions are only required rarely. If an inundation forecasting system is used in practice, it is recommended to further automatize the process of hiring infrastructure. This needs to happen in a save way, since hiring the infrastructure unnecessarily is expensive.
- **Model partitioning** With the current setup total computation time can be improved by running multiple models in parallel, but it cannot significantly improve the computation time of a single simulation. A single simulation could be sped up by enabling computation on multiple CPUs, but this provides only a limited improvement. Interesting for further research at HydroLogic is how the process of model partitioning and execution in the cloud can be automatised. This could significantly reduce computation time of single simulations, especially when the model can be distributed over multiple VMs. This knowledge can be applied in future projects to speed up simulations, and to be able to run more complex models within reasonable computation times.
- **Operational probabilistic inundation forecasts** For future projects where operational probabilistic inundation forecasts are required, it is recommended to consider the options of both cloud computing and neural networks. Based on the results of this research and client preferences, the best method can be selected and implemented using the findings of this research. The two methods can also be used in combination: the neural network can predict inundation depths for every new ensemble weather forecasts. When a certain inundation probability is exceeded the cloud workflow can be triggered to simulate the inundation with the hydraulic model. By implementing this approach, cloud computing resources are only utilized when truly necessary, thus reducing costs associated with unnecessary predictions.
- **Interactive inundation predictions** The neural network is capable of predicting inundation in real-time. By providing a user-friendly interface, this technology allows water managers who may not typically conduct modelling studies to gain valuable insights into how different variables impact inundation. The ability to interactively adjust parameters and immediately observe the resulting inundation can provide valuable information for decision-making. This type of service could also be beneficial for other organizations, and potentially even for individual citizens.

5.2.2 Further research

- **Input variables neural network** At present, the neural network is able to predict inundation using a 12-hour rainfall pattern as an input. However, it should be noted that there are many other variables that can impact inundation. Currently these are treated as constants. It is recommended to further research how adding additional input variables affects the accuracy of the neural network, and how it affects the required training data. Examples of such an input variable are the initial ground water level, seepage, evaporation and pumping station capacity.
- **Uncertainties in initial conditions** If the network can predict inundation based on both the rainfall and the initial state, as mentioned in the previous point, then his network could be used to make probabilistic inundation forecasts that includes both the uncertainty in rainfall and initial conditions. Because the neural network can provide inundation predictions within seconds, the inundation can be predicted for multiple initial states, for each ensemble member of the rainfall forecast. As an example: if 10 initial states are considered, and 50 ensemble members of the rainfall forecast, 500 inundation predictions would

be needed. This would be infeasible with the hydraulic model, however the neural network could make these 500 predictions within seconds.

- **Application to other areas** The network has proven very effective for polder de Tol. Further research should be conducted to evaluate the performance of the neural network in areas with greater topographical variations, urban environments, or regions that are heavily impacted by external factors such as a river flowing through the study area.
- **Generalised inundation predictions** A challenge for further research lies in researching the possibilities of a generalised neural network for inundation predictions which can be applied to areas without needing to be trained specifically for them. The input of this network would need to include geospatial information. It is recommended to first explore the capabilities of such a network without predicting inundation progression over time to reduce complexity. Possibly a convolutional neural network would be well suited to such an application, because of its ability in interpreting rasters (Bentivoglio et al., 2021; Guo et al., 2021). Also physics-informed neural networks might perform well at this task, as they leverage governing physical equations in neural network training (Cai et al., 2021). A physics-informed neural network could be trained to satisfy both the training data as well as the governing physical equations.
- **Network architecture** Several studies on neural network for inundation predictions have been done, however none of them have directly compared the performance of different network architectures for inundation forecasts. For further research it is interesting to compare different types of architecture (e.g. convolutional neural network, physically informed neural network, long short term memory neural network), such that information is gathered on which network architectures perform best.
- **Amount of training data** There currently is little knowledge on how the amount of training data affects the accuracy of inundation predictions by a neural network. For further research it would be interesting to find relations between the number of unique training samples and the accuracy of inundation predictions. In combination with the previous point, it would be interesting to see how this differs between different network architectures.

6 References

Journal articles

Bentivoglio, R., E. Isufi, S. N. Jonkman, and R. Taormina (2021). "Deep Learning Methods for Flood Mapping: A Review of Existing Applications and Future Research Directions". en. In: *Hydrology and Earth System Sciences Discussions* 2021.December, pp. 1–43. DOI: 10.5194/hess-2021-614.

Bergstra, J., R. Bardenet, Y. Bengio, and B. Kégl (2011). "Algorithms for hyper-parameter optimization". In: *Advances in Neural Information Processing Systems 24 (NIPS 2011)*, pp. 2546–2554.

Berkhahn, S., L. Fuchs, and I. Neuweiler (2019). "An ensemble neural network model for real-time prediction of urban floods". en. In: *Journal of Hydrology* 575, pp. 743–754. DOI: 10.1016/j.jhydrol.2019.05.066.

Bermúdez, M., L. Cea, and J. Puertas (2019). "A rapid flood inundation model for hazard mapping based on least squares support vector machine regression". en. In: *Journal of Flood Risk Management* 12.S1. DOI: 10.1111/jfr3.12522.

Brier, G. (1907). "Verification of Forecasts Expressed in Terms of Probability". In: *Monthly Weather Review* 26.653, pp. 25–27.

Cai, S., Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis (2021). "Physics-informed neural networks (PINNs) for fluid mechanics: a review". In: *Acta Mechanica Sinica/Lixue Xuebao* 37.12, pp. 1727–1738. DOI: 10.1007/s10409-021-01148-1.

Chang, L. C., M. Z. M. Amin, S. N. Yang, and F. J. Chang (2018). "Building ANN-based regional multi-step-ahead flood inundation forecast models". en. In: *Water (Switzerland)* 10.9, p. 1283. DOI: 10.3390/W10091283.

Chang, L. C., H. Y. Shen, and F. J. Chang (2014). "Regional flood inundation nowcast using hybrid SOM and dynamic neural networks". en. In: *Journal of Hydrology* 519.PA, pp. 476–489. DOI: 10.1016/j.jhydrol.2014.07.036.

Di Baldassarre, G., G. Schumann, P. D. Bates, J. E. Freer, and K. J. Beven (2010). "Cartographie de zone inondable: Un examen critique d'approches déterministe et probabiliste". fr. In: *Hydrological Sciences Journal* 55.3, pp. 364–376. DOI: 10.1080/02626661003683389.

Di Curzio, D., A. Di Giovanni, R. Lidori, M. Montopoli, and S. Rusi (2022). "Comparing Rain Gauge and Weather RaDAR Data in the Estimation of the Pluviometric Inflow from the Apennine Ridge to the Adriatic Coast (Abruzzo Region, Central Italy)". In: *Hydrology* 9.12. DOI: 10.3390/hydrology9120225.

Duan, Q., N. K. Ajami, X. Gao, and S. Sorooshian (2007). "Multi-model ensemble hydrologic prediction using Bayesian model averaging". en. In: *Advances in Water Resources* 30.5, pp. 1371–1386. DOI: 10.1016/j.advwatres.2006.11.014.

Georgas, N, A Blumberg, T Herrington, T Wakeman, F Saleh, D Runnels, A. Jordi, K. Ying, L. Yin, V. Ramaswamy, A. Yakubovskiy, O. Lopez, J. Mcnally, J. Schulte, and Y. Wang (2016). "The stevens flood advisory system: Operational H3E flood forecasts for the greater New York/New Jersey metropolitan region". en. In: *International Journal of Safety and Security Engineering* 6.3, pp. 648–662. DOI: 10.2495/SAFE-V6-N3-648-662.

Gomez, M., S. Sharma, S. Reed, and A. Mejia (2019). "Skill of ensemble flood inundation forecasts at short- to medium-range timescales". da. In: *Journal of Hydrology* 568, pp. 207–220. DOI: 10.1016/j.jhydrol.2018.10.063.

Goodarzi, L., M. E. Banihabib, and A. Roozbahani (2019). "A decision-making model for flood warning system based on ensemble forecasts". en. In: *Journal of Hydrology* 573, pp. 207–219. DOI: 10.1016/j.jhydrol.2019.03.040.

Guo, Z., J. P. Leitão, N. E. Simões, and V. Moosavi (2021). "Data-driven flood emulation: Speeding up urban flood predictions by deep convolutional neural networks". en. In: *Journal of Flood Risk Management* 14.1, pp. 287–315. DOI: 10.1111/jfr3.12684.

Henonin, J., B. Russo, O. Mark, and P. Gourbesville (2013). "Real-time urban flood forecasting and modelling - A state of the art". en. In: *Journal of Hydroinformatics* 15.3, pp. 717–736. DOI: 10.2166/hydro.2013.132.

Hochreiter, S. and J Urgen Schmidhuber (1997). "Long Shortterm Memory". In: *Neural Computation* 9.8, pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

Hou, J., N. Zhou, G. Chen, M. Huang, and G. Bai (2021). "Rapid forecasting of urban flood inundation using multiple machine learning models". en. In: *Natural Hazards* 108.2, pp. 2335–2356. DOI: 10.1007/s11069-021-04782-x.

Jackson, E. K., W. Roberts, B. Nelsen, G. P. Williams, E. J. Nelson, and D. P. Ames (2019). "Introductory overview: Error metrics for hydrologic modelling – A review of common practices and an open source library to facilitate use and adoption". In: *Environmental Modelling and Software* 119, pp. 32–48. DOI: 10.1016/j.envsoft.2019.05.001.

Jadeja, Y. and K. Modi (2012). "Cloud computing - Concepts, architecture and challenges". In: *2012 International Conference on Computing, Electronics and Electrical Technologies, ICCEET 2012*, pp. 877–880. DOI: 10.1109/ICCEET.2012.6203873.

Kabir, S., S. Patidar, X. Xia, Q. Liang, J. Neal, and G. Pender (2020). "A deep convolutional neural network model for rapid prediction of fluvial flood inundation". en. In: *Journal of Hydrology* 590, p. 125481. DOI: 10.1016/j.jhydrol.2020.125481.

Kilsdonk, R. A., A. Bomers, and K. M. Wijnberg (2022). "Predicting Urban Flooding Due to Extreme Precipitation Using a Long Short-Term Memory Neural Network". In: *Hydrology* 9.6. DOI: 10.3390/hydrology9060105.

Krause, P., D. P. Boyle, and F. Bäse (2005). "Comparison of different efficiency criteria for hydrological model assessment". In: *Advances in Geosciences* 5, pp. 89–97. DOI: 10.5194/adgeo-5-89-2005.

Lu, D., G. Konapala, S. L. Painter, S. C. Kao, and S. Gangrade (2021). "Streamflow simulation in data-scarce basins using bayesian and physics-informed machine learning models". In: *Journal of Hydrometeorology* 22.6, pp. 1421–1438. DOI: 10.1175/JHM-D-20-0082.1.

Martinez, T. R. and D. R. Wilson (2003). "The general inefficiency of batch training for gradient descent learning". In: *Neural Networks* 16.10, pp. 1429–1451.

Mei, C., J. H. Liu, H. Wang, Z. J. Li, Z. Y. Yang, W. W. Shao, X. Y. Ding, B. S. Weng, Y. D. Yu, and D. Y. Yan (2020). "Urban flood inundation and damage assessment based on numerical simulations of design rainstorms with different characteristics". en. In: *Science China Technological Sciences* 63.11, pp. 2292–2304. DOI: 10.1007/s11431-019-1523-2.

Moel, H. de, M. van Vliet, and J. C. Aerts (2014). "Evaluating the effect of flood damage-reducing measures: A case study of the unembanked area of Rotterdam, the Netherlands". In: *Regional Environmental Change* 14.3, pp. 895–908. DOI: 10.1007/s10113-013-0420-z.

Obroślak, R. and O. Dorozhynskyy (2017). "Selection of a semivariogram model in the study of spatial distribution of soil moisture". In: *Journal of Water and Land Development* 35.1, pp. 161–166. DOI: 10.1515/jwld-2017-0080.

Ochoa-Rodriguez, S., L. P. Wang, A. Gires, R. D. Pina, R. Reinoso-Rondinel, G. Bruni, A. Ichiba, S. Gaitan, E. Cristiano, J. Van Assel, S. Kroll, D. Murlà-Tuyls, B. Tisserand, D. Schertzer, I. Tchiguirinskaia, C. Onof, P. Willems, and M. C. Ten Veldhuis (2015). "Impact of spatial and temporal resolution of rainfall inputs on urban hydrodynamic modelling outputs: A multi-catchment investigation". In: *Journal of Hydrology* 531, pp. 389–407. DOI: 10.1016/j.jhydrol.2015.05.035.

Papaioannou, G., A. Efstratiadis, L. Vasiliades, A. Loukas, S. M. Papalexiou, A. Koukouvinos, I. Tsoukalas, and P. Kossieris (2018). "An operational method for Flood Directive implementation in ungauged urban areas". In: *Hydrology* 5.2. DOI: 10.3390/hydrology5020024.

Rafiq, M. Y., G. Bugmann, and D. J. Easterbrook (2001). "Neural network design for engineering applications". In: *Computers and Structures* 79.17, pp. 1541–1552. DOI: 10.1016/S0045-7949(01)00039-6.

Sajikumar, N. and B. S. Thandaveswara (1999). "A non-linear rainfall-runoff model using an artificial neural network". In: *Journal of Hydrology* 216.1-2, pp. 32–55. DOI: 10.1016/S0022-1694(98)00273-X.

Schaefer, J. T. (1990). "The Critical Success Index as an Indicator of Warning Skill". In: *Weather and Forecasting* 5.4, pp. 570–575. DOI: 10.1175/1520-0434(1990)005<0570:tcsiaa>2.0.co;2.

Seyoum, S. D., Z. Vojinovic, R. K. Price, and S. Weesakul (2012). "Coupled 1D and Noninertia 2D Flood Inundation Model for Simulation of Urban Flooding". et. In: *Journal of Hydraulic Engineering* 138.1, pp. 23–34. DOI: 10.1061/(asce)hy.1943-7900.0000485.

Shahapure, S. S., A. T. Kulkarni, K. S. Bharat, T. I. Eldho, and E. P. Rao (2010). "Coastal urban flood simulation using fem-gis based model". et. In: *ISH Journal of Hydraulic Engineering* 16, pp. 74–88. DOI: 10.1080/09715010.2010.10515017.

Snoek, J., H. Larochelle, and R. P. Adams (2012). "Practical Bayesian optimization of machine learning algorithms". In: *Advances in Neural Information Processing Systems* 4, pp. 2951–2959.

Trenberth, K. E. (2011). "Changes in precipitation with climate change". In: *Climate Research* 47.1-2, pp. 123–138. DOI: 10.3354/cr00953.

Wang, X., G. Kingsland, D. Poudel, and A. Fenech (2019). "Urban flood prediction under heavy precipitation". en. In: *Journal of Hydrology* 577. DOI: 10.1016/j.jhydrol.2019.123984.

Wu, W., R. Emerton, Q. Duan, A. W. Wood, F. Wetterhall, and D. E. Robertson (2020). "Ensemble flood forecasting: Current status and future opportunities". en. In: *WIREs Water* 7.3. DOI: 10.1002/wat2.1432.

Zanchetta, A. D. L. and P. Coulibaly (2022). "Hybrid Surrogate Model for Timely Prediction of Flash Flood Inundation Maps Caused by Rapid River Overflow". en. In: *Forecasting* 4.1, pp. 126–148. DOI: 10.3390/forecast4010007.

Zarzar, C. M., H. Hosseiny, R. Siddique, M. Gomez, V. Smith, A. Mejia, and J. Dyer (2018). "A Hydraulic MultiModel Ensemble Framework for Visualizing Flood Inundation Uncertainty". en. In: *Journal of the American Water Resources Association* 54.4, pp. 807–819. DOI: 10.1111/1752-1688.12656.

Zhou, Y., W. Wu, R. Nathan, and Q. J. Wang (2021). "A rapid flood inundation modelling framework using deep learning with spatial reduction and reconstruction". In: *Environmental Modelling and Software* 143. DOI: 10.1016/j.envsoft.2021.105112.

Other sources

Actueel Hoogtebestand Nederland (2020). *AHN4*. URL: https://www.ahn.nl/ahn-4 (visited on 07/12/2022).

Beersma, J., H. Hakvoort, R. Jilderda, A. Overeem, and R. Versteeg (2019). *Neerslagstatistiek en -reeksen voor het waterbeheer 2019*. Tech. rep.

Besseling, L. (2022). "Dike breach flood prediction of an LSTM compared to the HAND.FLOW model for real-time flood forecasting". MSc thesis. Univeristy of Twente. URL: https://www.utwente.nl/en/et/cem/research/wem/education/msc-thesis/2022/besseling.pdf.

Blekemolen, M. and G. Schwarz (2014). *Evaluatie wateroverlast Kockengen juli 2014*. Tech. rep. URL: https://www.hdsr.nl/buurt/watergebiedsplannen/kamerik-kockengen/raamwaterplan/wateroverlast/.

BRO (2018). *BRO Bodemkaart*. URL: https://app.pdok.nl/viewer/?origin=pdoknl (visited on 07/16/2022).

Brutsaert, W (2006). *Hydrology: an introduction*. en. Vol. 43. 07. Cambridge: Cambridge University. DOI: 10.5860/choice.43-4036.

CPH Post (2011). *Drenched*. URL: https://cphpost.dk/2011-07-04/news/local-news/drenched/ (visited on 01/20/2023).

De Graaff, B., G. Rongen, and R. Hurkmans (2022). *D-HYDAMO Modelgenerator voor D-HYDRO*. URL: https://www.hkv.nl/product/d-hydamo/ (visited on 07/13/2022).

Deltares (2022a). *D-HYDRO GUI Visualisatie en Cloud*. URL: https://publicwiki.deltares.nl/pages/viewpage.action?pageId=185140218 (visited on 01/10/2023).

— (2022b). *D-HYDRO Suite 1D2D User Manual*. Tech. rep. Deltares.

Deltares (2022c). *D-Rainfall runoff, user manual*. Tech. rep. Deltares.

Gerdes, J. (2012). *What Copenhagen Can Teach Cities About Adapting To Climate Change*. URL: https://www.forbes.com/sites/justingerdes/2012/10/31/what-copenhagen-can-teach-cities-about-adapting-to-climate-change/?sh=192f13ac1e89 (visited on 01/20/2023).

HDSR, W. (2022). *Leggers van watergangen en keringen*. URL: https://www.hdsr.nl/werk/leggers-watergangen/ (visited on 07/14/2022).

Kleinveld, E., P. Smorenburg, E. van Dijk, and M. van Bergen (2016). *Geleerde lessen en verkenning naar geschikte maatregelen maalgebied De Tol*. Tech. rep. URL: https://www.hdsr.nl/buurt/watergebiedsplannen/kamerik-kockengen/raamwaterplan/wateroverlast/.

KNMI (2014). *KNMI'14: Climate Change scenarios for the 21st Century – A Netherlands perspective*. URL: https://www.knmi.nl/kennis-en-datacentrum/publicatie/knmi-14-climate-change-scenarios-for-the-21st-century-a-netherlands-perspective (visited on 08/26/2022).

KNW (2020). *OWASIS biedt actueel inzicht in water in bodem*. URL: https://www.h2owaternetwerk.nl/h2o-actueel/owasis-biedt-actueel-inzicht-in-water-in-bodem (visited on 12/06/2022).

MeteoBase (2022). *Het online archief van historische neerslag en verdamping in Nederland*. URL: https://www.meteobase.nl/?tb=rasterdata&dp=rasterdata&dp_sub=introductie (visited on 08/09/2022).

Olah, C. (2015). *Understanding LSTM Networks*. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (visited on 10/15/2022).

Smorenburg, P. (2014). *Noodvoorziening wateroverlast Kockengen*. Tech. rep. Hogeschool van Amsterdam. URL: https://www.hdsr.nl/buurt/watergebiedsplannen/kamerik-kockengen/raamwaterplan/wateroverlast/.

Stichting Toegepast Onderzoek Waterbeheer (2020). *Landgebruikkaart*. URL: https://www.waterschadeschatter.nl/damage/ (visited on 07/14/2022).

Veldman, W. (2006). "Effect van de Flora- en Fauna wet op het peilbeheer". Msc. Thesis. University of Twente. URL: http://essay.utwente.nl/57353/1/scriptie_Veldman.pdf.

You, K., M. Long, J. Wang, and M. I. Jordan (2019). *How Does Learning Rate Decay Help Modern Neural Networks?* Tech. rep. URL: http://arxiv.org/abs/1908.01878.

Appendix A Model details

In this section of the appendix, more elaboration is given on the modelling process.

A.1 D-HyDAMO

As mentioned, the model is created using the D-HyDAMO Python package. The following steps were executed to be able to create the model with D-HyDAMO.

1. Get data for the hydraulic structures, waterways, land use, seepage and elevation data.
2. Transform the data to the HyDAMO standard, this is done using Python scripts.
3. Correct errors in the input data. (incorrect waterway connections, missing cross sectional data)
4. Place 1D model components in model according to the input data
5. Generate rectangular 2D grid in study area
6. Calculate and apply 2D roughness based on land use
7. Generate RR nodes from data
8. For the RR parameters that can be estimated from data, calculate their values. These are based on elevation data, soil data and land use.
9. Place the correct precipitation and evaporation data for the period of reference.

With the resulting script, model setups can be generated within a single click. Model parameters are customisable, and the model is generated in around 2 minutes. Once setup, this functions as a great tool for building models quickly. However it can be difficult to customize certain things (for example adding a single weir or pump), since this means multiple input data sources have to be modified. Also, because D-HyDAMO is still in development, there are some bugs, and certain features are not compatible yet.

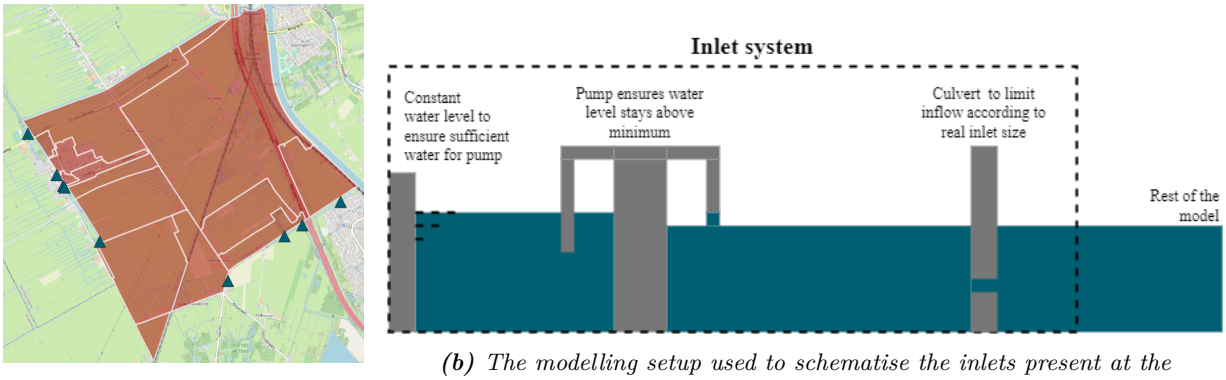
A.2 Cross sections

Not all waterways have available profile measurements. The primary waterways do have measurements for most waterways. For the primary waterways where no profile is available, the profile of the nearest primary waterway is used. For the secondary and tertiary waterways, there are no profile measurements. Because D-HyDAMO is not yet very flexible in assigning default cross sections, it is decided that all non primary waterways have the same cross section. The dimensions of this default cross section are determined with satellite imagery, and the validity of the assumption is tested during the calibration phase. All default profiles have a rectangular shape, and they are placed on surface level at the location of the waterway.

A.3 Inlets

As noted in section 2.1.4, there are inlets present at the boundary of the study area (A.1a). These are used to supply water to the area when the water level is lower than the target water level. At first it was tried to model the area without these inlets, however the results showed that when precipitation is low, the system would dry out indefinitely. To compensate this, simulations were done without evaporation, however the results were not yet satisfactory, and therefore inlets are included. In the D-HyDAMO python package, not all features of D-Hydro are available yet. This makes modelling the inlets quite challenging. To still be able to include the inlets, the inlet setup shown in figure A.1b is used.

The constant water level boundary condition ensures that there will always be enough water available outside of the study area. This assumption is deemed valid since the goal of the model is to predict inundation as a result of heavy precipitation, the goal is not to predict water levels in times of drought. The pump ensures water is transported in only one direction. The inlets do not function as outlets, and thus water cannot exit the system through them. The culvert ensures that the discharge trough the inlet is limited. With this setup, the inlets will let in water when the water level is below the target water level of the area. There are 9 inlets in total, all with different dimensions.



(a) *Locations of the inlets.*

(b) *The modelling setup used to schematise the inlets present at the boundaries of the study area.*

Figure A.1: *Schematisation of the inlets present at the boundaries of the study area.*

A.4 Water management

Pumps, culverts and weirs often have components that can be set up according to the strategy of the water managers in the area. Since these set ups do not always follow predefined rules, it can be difficult to predict the setup for historic events. Because of this, the model does not always have the same setup as in real life. The following decisions are made to get as close as possible to the real world conditions.

- Weir operating levels are taken as the target water levels for their respective fixed drainage level area. Their is a difference between summer and winter target water levels.
- Culverts are assumed to be always opened, except those that would cause a fixed drainage level area to stay in direct connection with a fixed drainage level area that has another target water level. In these cases, the culvert is replaced with a weir levelled at the target water level.
- During calibration pump discharge and operating levels are decided based on the historical data, such that for each event they are the same as in the real world.
- Inlets are operated to match up the target water level of the fixed drainage level area. Their discharge is limited by the real world opening size.
- Outside of the study area, there is assumed to be infinite supply and storage of water. In other words: the pumping station can always pump water out of the study area, and the inlets can always let water in the study area.

A.5 Computer specifications

Table A.1: *Specifications of the laptop used for making the hydraulic model*

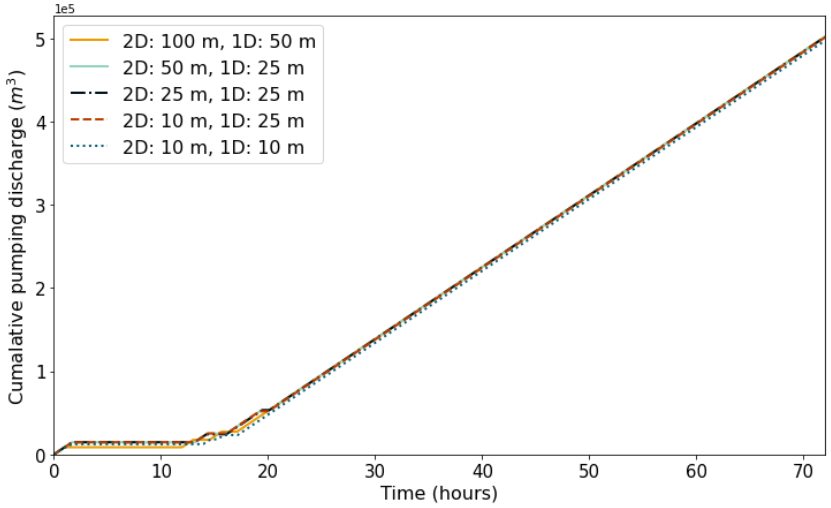
Processor	Intel(R) Core(TM) i7-6820HQ CPU @ 2.70GHz
RAM	16.0 GB

A.6 Sensitivity of calibration event to resolution

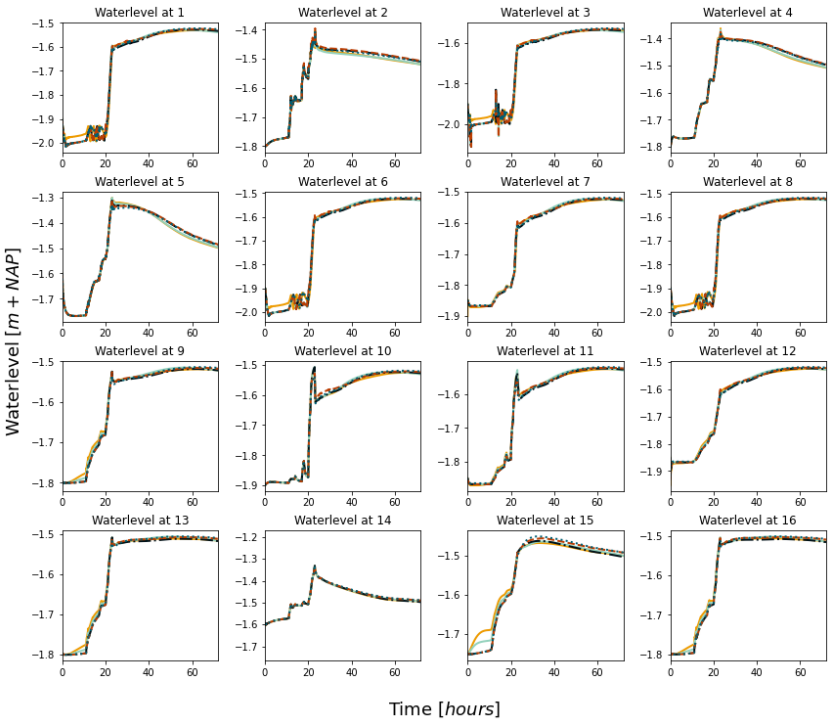
The sensitivity of calibration event 4 to different 2D and 1D resolution has been tested. For this, 3 days are simulated, with no warm up period. The results show that for this event, the resolution has little influence. This is because water stays in the waterways, and thus does not flow towards the 2D rectangular grid. The computation times for each of the models can be seen in A.2.

Table A.2: *Computation times for different model setups, total simulation duration is 3 days.*

1D resolution	2D resolution	Computation time	1D computational nodes	2D grid cells
50 meters	100 meters	3 minutes 18 seconds	6.252	1.417
25 meters	50 meters	7 minutes 27 seconds	10.956	4.795
25 meters	25 meters	10 minutes 13 seconds	10.956	19.576
25 meters	10 meters	40 minutes 6 seconds	10.956	123.993
10 meters	10 meters	49 minutes 31 seconds	26.302	123.993



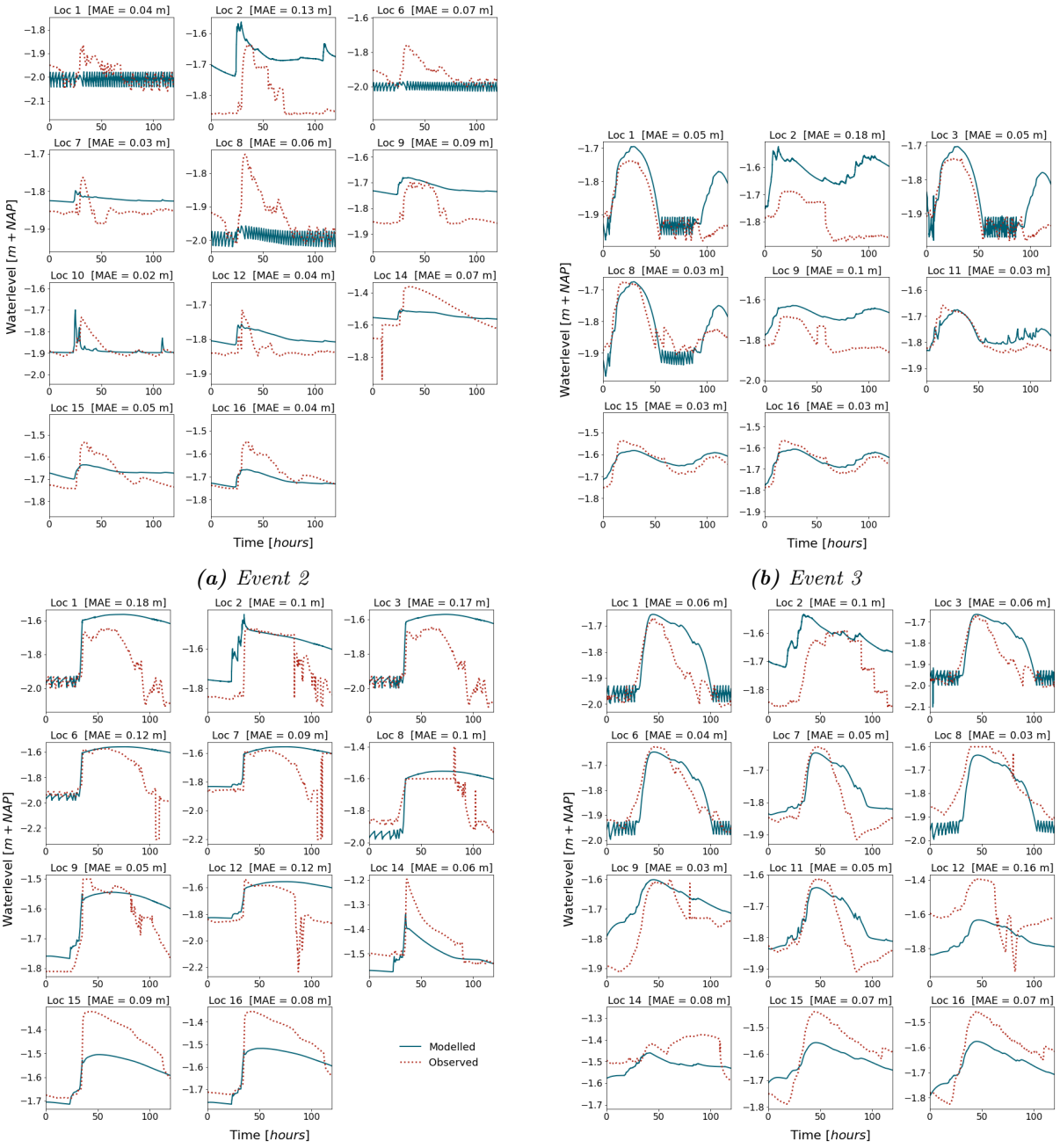
(a) *Influence on discharge trough pumping station.*



(b) *Influence on water levels (same legend as top figure).*

Figure A.2: *Influence of 1D and 2D resolution on the model output.*

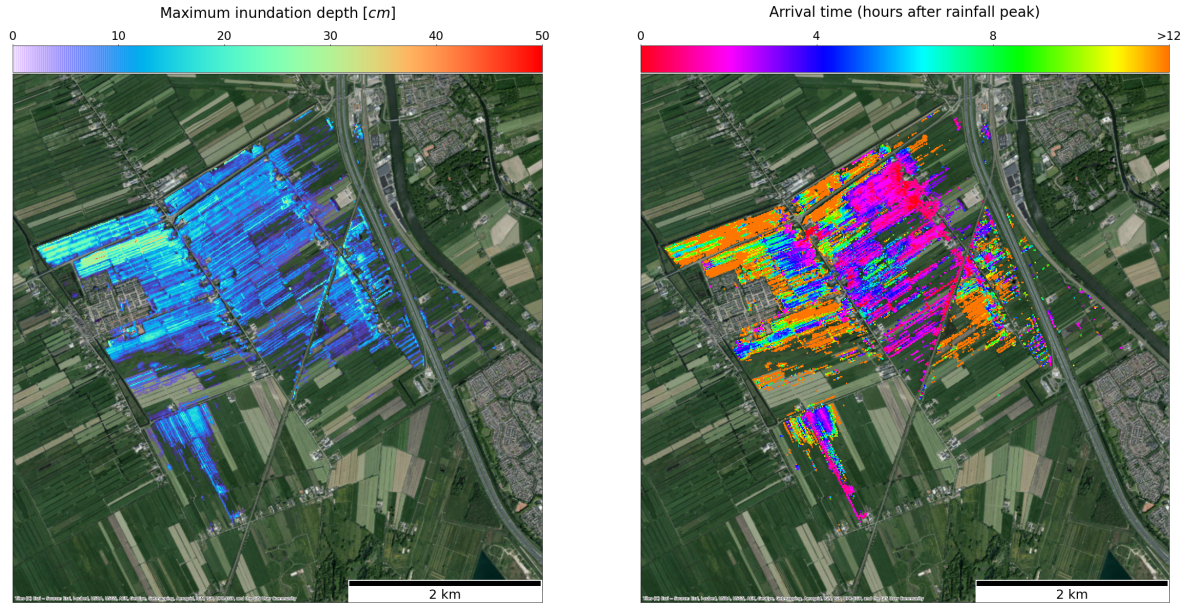
A.7 Water level results



(a) Event 2 *(b) Event 3*

(c) Event 4 *(d) Event 5*

Figure A.3: *Modelled (blue) and measured (red) water levels during heavy precipitation at different measuring stations in the study area.*



(a) Maximum inundation depths. *(b) Arrival times of inundation greater than 5 cm.*

Figure A.4: *Example of resulting inundation patterns from the hydraulic model. This figure displays inundation during event 4, with resolution 1D: 10 meters, and 2D: 10 meters.*

Appendix B Limiting hardware resources during a simulation

To see how limiting the available hardware affects the computation time and model stability, several tests have been done. For these tests, the model with a 1D resolution of 20 meters and a 2D resolution of 10 meters is used. It can be concluded that the model will run with limited CPU availability, however this increases computation time. Allowing more than 1 CPU to be used does not decrease computation time any further, because using multiple CPU's for a single simulation is not considered for this research. If this where allowed, more CPUs would decrease the computation time, however it would not be utilising the CPU to its full efficiency (Deltares, 2022b).

Surprisingly, reducing the available RAM does not affect the computation time. Actually, from the tests it can be concluded that reducing the RAM yields the exact same results. However, if the RAM is reduced below 500 MiB the simulation crashes. This is an interesting result, however for the rest of this research the default RAM usage will be allowed (RAM usage when the RAM availability is not limited). Limiting the available RAM is an interesting topic for further investigation to improve efficiency.

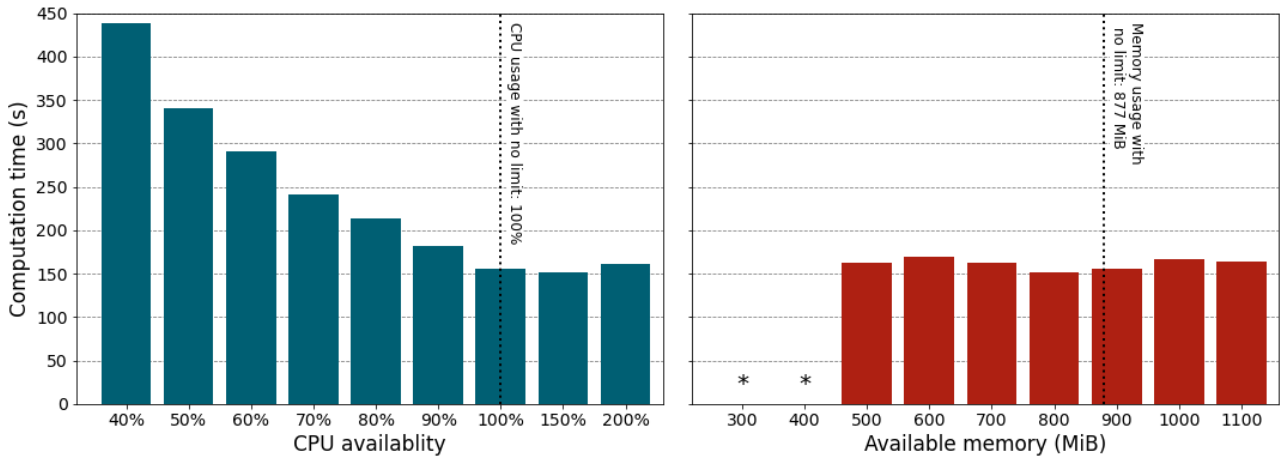


Figure B.1: *Model computation time when the CPU availability is limited (left) and the available memory is limited (right)*

Appendix C Neural network performance

C.1 Performance over time

The neural network predicts the inundation patterns at 12 time steps. It is important to analyse if the network performs equally well for all time steps. This is done by calculating the MAE and relative error for all time steps. Figure C.1 shows the results at each time step. Clearly, the MAE is higher for later time steps. This is because as time progresses, there is more time for rain to fall in the study area, which will increase inundation depths. With higher inundation depths the prediction errors will also increase, leading to a higher MAE. For this reason, the relative error is also calculated, which is the MAE as a fraction of the average inundation. It can be seen that the relative error is decreasing over time. The relative error is significantly higher at time step 1, and after time step 1 it slightly decreases each time step.

The higher relative errors at the earlier time steps have to do with the optimisation objective of the neural network. The network tries to minimize the MSE. When there is more inundation, the MSE is larger. Therefore by minimizing the MSE, the network will have a tendency to first focus on improving the locations with the larger MSE. Because at these locations, the most improvement in the MSE can be made. At earlier time steps, there is less inundation, meaning lower MSE values, and thus improving the network for these events will not yield as big of an improvement in MSE as for later time steps.

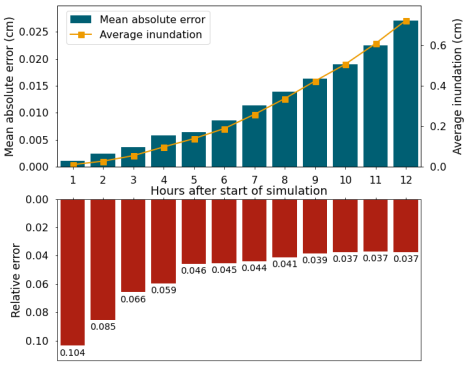


Figure C.1: *The MAE and relative error of the inundation predictions at different time steps, for the entire testing dataset.*

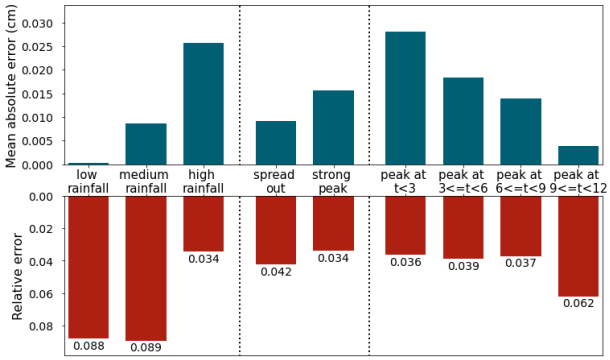


Figure C.2: *The MAE and relative error of the inundation predictions for different event types.*

C.2 Performance for different types of events

The neural network is used for predicting precipitation events that can vary significantly in size and distribution over time. To check how well the network performs on different types of events, the events are categorized and for each event type the MAE and relative error is calculated. The different categories and their performance can be seen in figure C.2.

The MAE is higher for events with more rainfall, which is expected since in these cases there is more inundation and thus larger errors. The relative error is similar for low and medium rainfall events, and is significantly lower for events with high rainfall. This is because the high rainfall events lead to more inundation and therefore higher MSE values. These are the events that are most optimised for, as explained in the previous paragraph.

When differentiating between rainfall with strong peaks and more spread out rainfall the difference in error is small. MAE is slightly higher for rainfall with stronger peaks, but for these events there also is more inundation. The relative error is slightly higher for spread out rainfall, but the overall performance is similar indicating that the network can predict inundation well no matter the type of event.

Lastly there is the arrival time of the peak rainfall. Generally, later peaks cause lower MAE. This especially is clear when the peak occurs later than 9 hours after the start of the simulation. When the rainfall occurs so late, less inundation occurs since there is no simulation time left to overflow the waterways and spread over the domain. In practice, such simulations need to be avoided because the event of interest is partly happening outside of the simulation time domain. In general, it can be said that the network performs well regardless of the arrival time of the rainfall, because the relative error is almost identical for arrival times between $0<t<9$.

Spatial correlation There is no spatial coherence between adjacent cells in the chosen network setup (section 2.3.4). In reality, water depths of adjacent cells are highly correlated, and therefore it is interesting to see if the neural network also shows this high correlation between cells located closely to each other. To do this, a spatial semivariogram is made according to the methodology outlined in literature (Obroślak and Dorozhynskyy, 2017). For each data pair, the semi variance is calculated. The semi variance is then averaged over bins of similar distances. The semi variance is calculated according to equation 12. The semi variance is calculated for 1000 random data points, since calculating it over all data points is too computationally expensive. The resulting semivariograms of the hydraulic model inundation predictions and the neural network predictions can then be compared to each other, to see if the neural network is able to capture the spatial correlation without them being explicitly included in the network architecture.

$$\gamma(h) = \frac{\sum_{i=1}^{n} (x_i - x_{i+h})^2}{2n} \tag{12}$$

Where

h represents a distance bin, for example, data pairs need to be between 75 and 100 meters apart.

x_i is the inundation at grid cell i.

x_{i+h} is the inundation within h meters away from grid cell i.

n is the total number of data pairs that are within distance bin h away from each other.

$\gamma(h)$ is the semi variance for distance bin h.

The resulting semivariogram can be seen in figure C.3. The semivariogram of the testing data and the neural network predictions are very similar for all distance bins. However the predictions of the neural network show a slightly higher semi variance then the testing data.

Since there is no spatial coherence in the neural network itself, the only way it replicates the spatial coherence is trough replicating the training data patterns. The network does a very good job at replicating the inundation patterns, but there is a small error. In principal, these errors have no spatial correlation in them (because the only spatial correlation in the neural network can come from replicating the data) and thus the error causes the spatial correlation to be reduced. Since the error is small, the reduction in spatial correlation also is small, as can be seen in figure C.3.

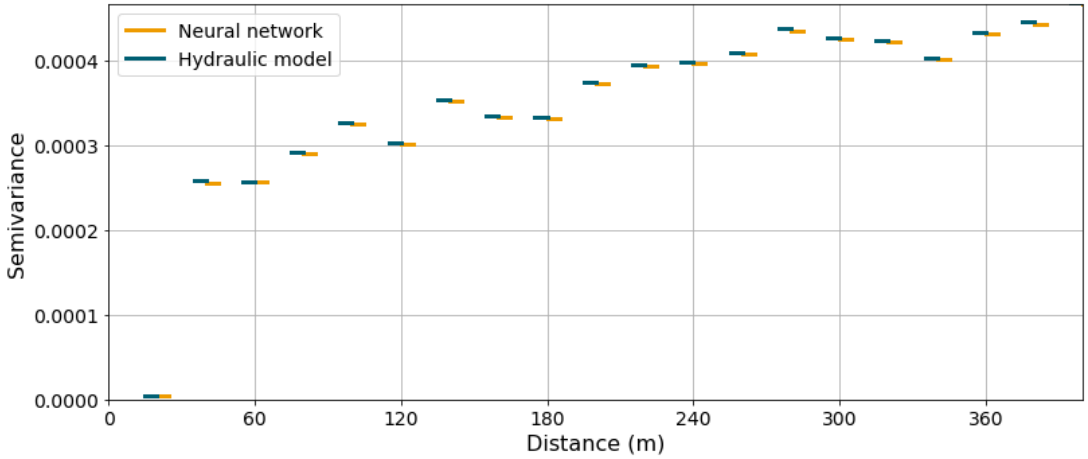


Figure C.3: *The semivariogram for both the validation data and the neural network predictions. The semivariogram includes data of 1000 random grid cells, since using all grid cells would require to much computation.*

C.3 Required training data

For the neural network presented in this study, 1600 simulations are used for training the neural network. Generating the training data either costs money (when generated with cloud computing) or takes a long time,

and therefore limiting the amount of training data would reduce the costs of developing the neural network. To see how the amount of training data influences the performance of the neural network, a series of tests is done. Throughout the tests, the validation and testing data is kept the exact same, however the number of samples used for training the neural network is limited. The MSE is plotted for different amounts of available training data, and the other performance metrics are analysed. With this, insight will be generated on how much training data is required, and how the amount of training samples affects the performance of the neural network.

The model is trained with 100, 200, 400, 800 and 1600 training samples. For each of the tests, the total amount of training iterations is kept constant. To achieve this, the number of epochs is higher when there are less samples, meaning that the model will see the same data more often. If this was not done, difference in performance can be caused by both the sample count and the training duration.

In figure C.4 the progression of the MSE throughout the training process can be seen for each subset of samples. The number of samples used clearly affects the training process significantly. When 100 samples are used, the model already starts over fitting after 10% of the training process. As a result of the over fitting, the performance on the testing data significantly reduces when training continues, because it starts learning the 100 training samples in the training dataset by heart. With 200 samples, the same phenomenon can be seen, however it occurs only after around 30% of the training process. If these network would be used in practice, the training process would have to be stopped before the over fitting starts.

From figure C.4 it is clear that using more samples during training improves the performance. However, as the number of samples increases, the performance increases less quickly. Actually, the 800 samples and 1600 samples show very similar performance, and due to the stochastic nature of the training process they barely can be differentiated in terms of performance. Important to remark here is the number of epochs. As can be seen from the networks with less training samples, at some point the network either does not improve any more, or starts over fitting. The network with 400 samples does not improve any more after 60% of the training process. Both the 800 and 1600 samples network do still improve up until 100% of the training process, and show very similar performance up until this point. When the networks are trained for longer, the network trained on 1600 samples does outperform the 800 samples network.

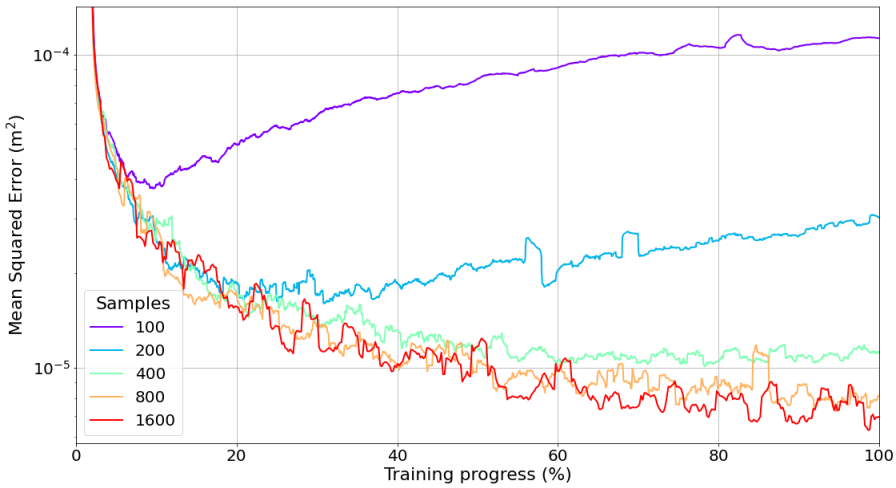


Figure C.4: *The MSE throughout the training process for differently sized training datasets. The x axis shows the training progress. In total, each model will see 800.000 samples, but for models with less training data there will be more repetition of the same samples.*

In figure C.5 the scatter plot of the inundation predictions can be seen for each of the networks trained with a different number of samples. As could be expected from figure C.4, the network with 100 samples clearly performs worst. Performance is significantly better with 200 samples, showing how using only 100 samples really limits the performance of the neural network. With 400, 800 and 1600 samples the density around the 1:1 line increases further, indicating more accurate predictions.

The network with 400 samples shows a slight bias towards over prediction, and the 1600 samples network towards under prediction. The 800 samples network has less of a bias. While a bias could be caused by an imbalanced training dataset, this is not the reason for the different biases in this figure, since samples are

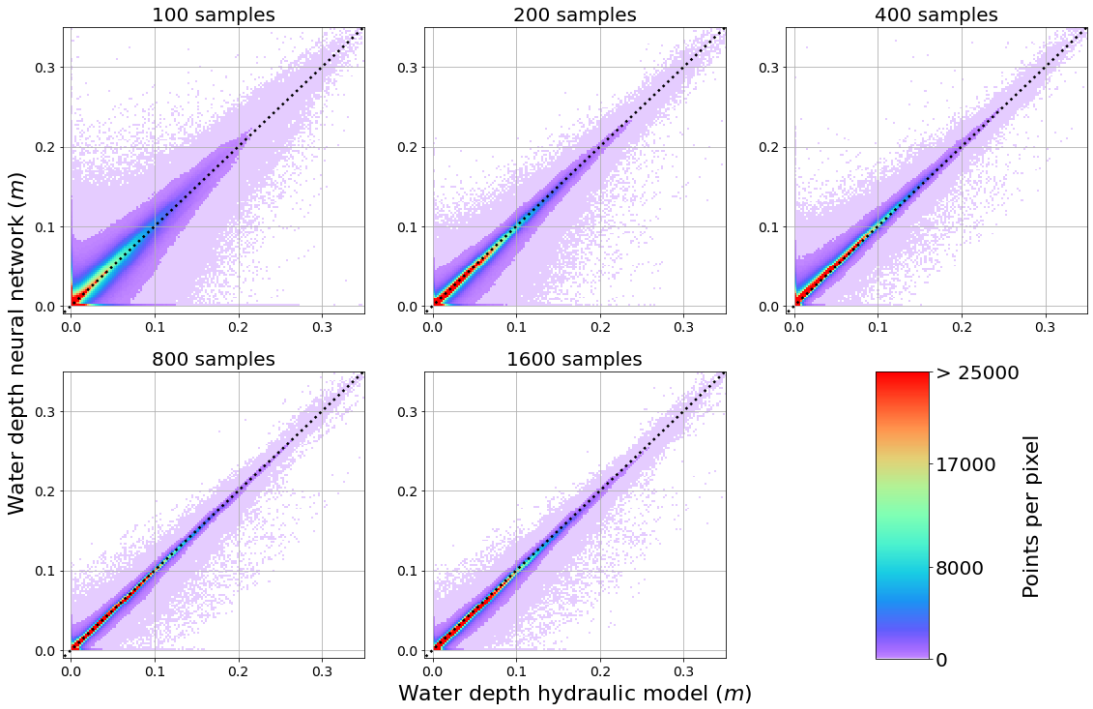


Figure C.5: *Scatter plots of the inundation depths for networks training with limited number of unique samples. All scatter plots show the results after the network has seen 800.000 samples. Equivalent to 500 epochs on the full training dataset.*

selected in a way that avoids having biased data. The reason for the different biases in figure C.5 is the stochastic nature of the training process. One network will see different samples, affecting the optimisation algorithm and therefore the outcome of the optimisation. For this reason each network is slightly different, and because of this some might have a slight bias towards over prediction while other under predict. Berkhahn et al. (2019) identified this and suggests training multiple networks and combining their output. A disadvantage of this is that it significantly increases the total required training time.

The amount of training data required depends on multiple factors. Firstly, if additional parameters would be considered (e.g. initial groundwater level), more training data would be required to be able to learn all patterns. Secondly, the ideal amount of training data depends on the costs of generating data. If this cost is small compared to other costs of setting up the network, then using more training data (e.g. 800 or 1600 samples) would be recommended. However if the costs of generating the training data is high, the network with only 200 samples could be sufficient. Lastly there also is the required accuracy of a network. With the accuracy of the neural network in this study, the training data inaccuracy is far greater than that of the neural network. Because this is the case, a slightly lower accuracy of the neural network is insignificant compared to the inaccuracy of the training data. Even when using only 100 samples, the neural network would not be the largest source of error when compared to reality.