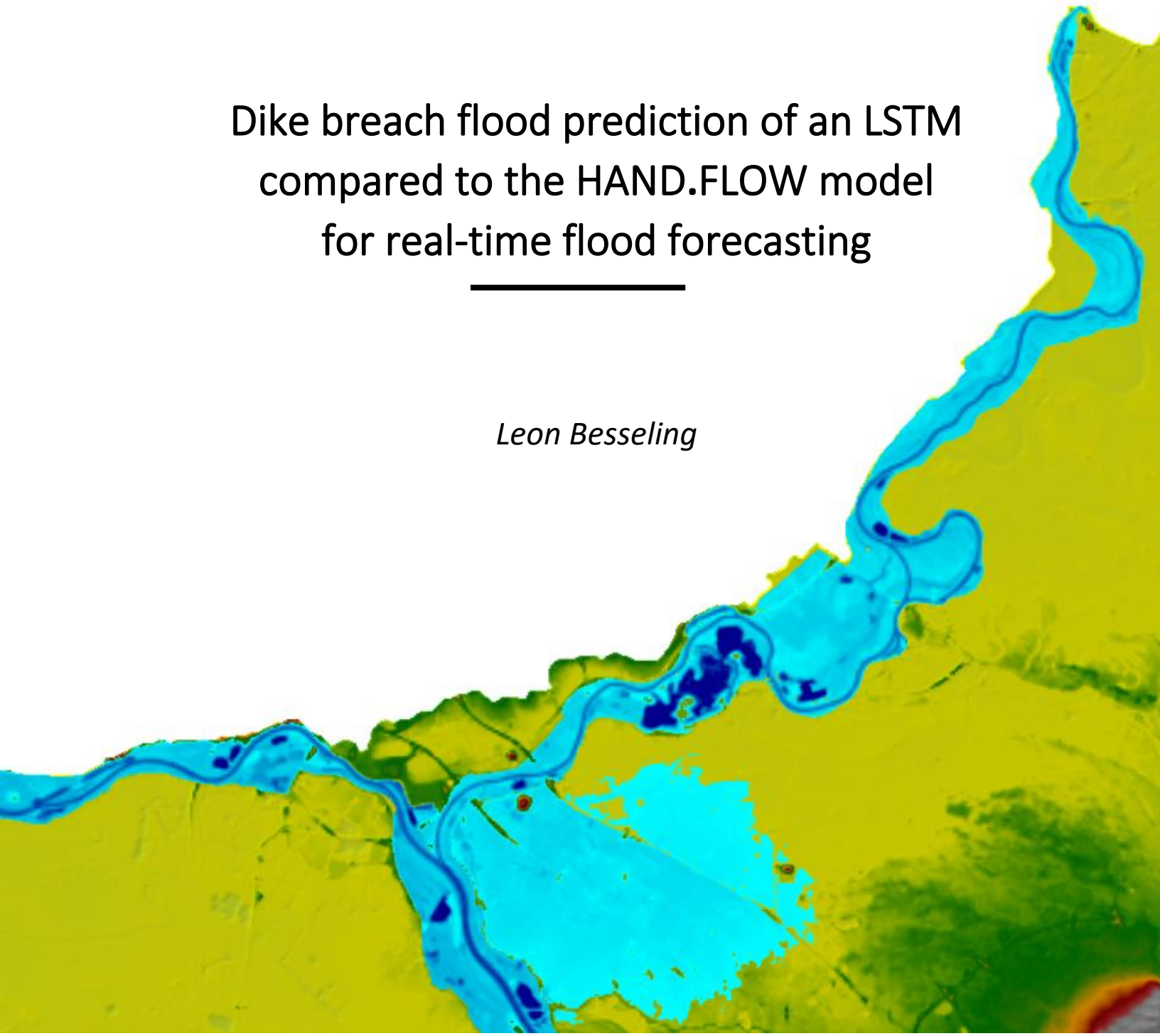
Dike breach flood prediction of an LSTM compared to the HAND.FLOW model for real-time flood forecasting

Leon Besseling



8th of July, 2022

Presented for the degree of MSc Civil Engineering & Management
Supervised by dr.ir. A. Bomers and prof.dr. S.J.M.H. Hulscher

UNIVERSITY
OF TWENTE.

PREFACE

In the final year of my high school, a group of friends taught a computer to play Super Mario Bros for their school research project. Up until this master thesis, however, I had never worked with machine learning myself. So when the opportunity came by to combine machine learning with my interest in flood modelling, I knew that this would be a topic I would want to dive into. Together with my supervisor, we quickly realized that machine learning for speeding up flood modelling is exciting, but it might not be the only or even the best way. That is how this research also lead to the creation of a conceptual model that does not require an awful lot of data and gives a reasonable first insight in how the flood develops, by utilizing the height map of the study area. I am proud to have created a first version of this model, and have learned a lot regarding machine learning and model development.

I would like to thank my supervisors Anouk Bomers and Suzanne Hulscher for their valuable feedback, guidance and flexibility, without which the end result of this project would have been much different. Finally, I would like to thank friends and family who listened to me when I was excited and supported me when I was in need.

I hope you have an interesting read,

Leon Besseling
Enschede, 8th of July 2022

ABSTRACT

The most common method to model flood dynamics is using two-dimensional depth-averaged (2DH) hydrodynamic models (Chu et al., 2020). However, these models generally have long computation times of many hours or even days. As a result, they cannot be used for scenario analysis in a real-time flood forecasting system after a warning for an incoming discharge is issued (Teng et al., 2017). The aim of this study is to identify which surrogate model is the most promising model for real-time flood forecasting in case of a dike breach: a new conceptual HAND.FLOW model or a data-driven neural network.

The neural network that was developed in this study is a Long Short Term Memory (LSTM) neural network, since it has been found suitable for predicting time series due to its ability to store information and to learn long-term dependencies in data (Le et al., 2019). In this study, data from 73 flood events modelled in a 1D2D-hydrodynamic model developed by Bomers (2021) was used to train and assess the LSTM. The outflow hydrograph of the dike breach functioned as the input, and the water depth in the hinterland was predicted per time step on every grid cell of the study area. The model architecture and hyperparameters such as dropout, number of neurons, activation function and learning rate were optimized using Bayesian optimization for the lowest value of the error function on water depth (Mean Absolute Error, MAE).

The original HAND model is a conceptual model that only requires the Digital Elevation Model (DEM) of the study area to be set up. It takes the river water level as its input and floods all cells along the river with a Height Above Nearest Drainage (HAND) value lower than this water level. A dike breach, on the other hand, is a point source of a flood. To model the flood propagation from the breach into the hinterland, the new HAND.FLOW model was created with a number of adaptations: a pathfinding algorithm for finding the steepest downstream path from the dike breach into the hinterland, a distance limit relationship limiting the pathfinding algorithm per time step to simulate flood propagation behaviour, and a volume component allowing the model to take the outflow hydrograph as input instead of the river water level.

Both models were tested on 15 flood events that were excluded from the LSTM training procedures. The LSTM performance was very accurate with a MAE of just 0.045 meters on an average water depth of 1.49 meters: an error of just 3% compared to HEC-RAS. The NSE values were close to 0.99 on nearly all grid cells in the study area, and the CSI metric for comparing the inundation areas was on average 0.94. The HAND.FLOW model was less similar to the water depths of HEC-RAS, due to a terrain feature not modelled in HEC-RAS. In a single corrected simulation, the MAE was 0.21 meters (error of 15%), the NSE was around 0.8 for large parts of the study area and the CSI averaged around 0.7. After a change in the hinterland, the HAND.FLOW model also correctly predicted the new flood pattern.

All in all, the data gathering for the LSTM requires a lot time (800 hours for Bomers (2021)). It has to be re-trained for a change in the hinterland or for another breach location, so it is not flexible. After the training procedure, however, it can predict the flood event near instantly and very accurately. The HAND.FLOW model requires a much shorter set-up time of around 30 minutes, so it is very flexible for changes in the hinterland, simulating other breach locations, or adapting spatial/temporal resolutions. The simulation time was 30 minutes on a detailed resolution of 10x10 meters, and only 1.5 minutes on the 150x150 meter resolution used by HEC-RAS and the LSTM. Therefore, the HAND.FLOW model offers in a relatively short simulation time a reasonable insight in how a dike breach flood will propagate in the hinterland, and could be the suitable and flexible model needed for a real-time flood forecasting system if it is further developed.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1 INTRODUCTION

Floods are terrible disasters with large consequences that affect more people than any other weather-related disaster (Verwey et al., 2017). For river flooding specifically, in 2030 the number of affected people is predicted to double compared to 2015 due to ongoing urbanization, population growth, inadequate maintenance of flood management infrastructure and climate change (Verwey et al., 2017). As such, there has always been and will always be a desire to assess flood risk and predict flood events (Teng et al., 2017).

Useful quantities associated with flood forecasting are flood extent, flood depth, flood arrival time and flood flow velocities, as these enable decision makers to make optimal decisions on measures such as evacuation (Verwey et al., 2017). If an incoming upstream discharge wave is noticed, these decisions have to be made in time and with sufficient certainty about the risks. To aid in this, a real-time flood forecasting system is desired, in which ensemble model predictions and uncertainty analysis of hundreds or even thousands of model runs allow for reviewing multiple scenarios and making these optimal decisions (Chu et al., 2020).

For the modelling of flood dynamics, and obtaining the quantities mentioned above, numerical simulation tools like hydrodynamic models are the most common method (Chu et al., 2020; Teng et al., 2017). These often have a one-dimensional (1D) part for describing the characteristics of the river and a two-dimensional horizontal (2DH) part for the hinterland. They are accurate, but require long computation times of many hours or even days, as a result of their complex descriptions of the physical system (e.g. in Bhola et al. (2018) and Bomers (2021) for flood modelling in the Rhine). Due to these long computation times, the discharge wave will have travelled further downstream once the model result is finally obtained, leaving little time for decision making. Depending on the characteristics of the river system and the modelling time, it could even happen that the model is still simulating while the actual flood is happening. This makes the use of hydrodynamic models in a real-time flood forecasting system unrealistic, despite increases in computation power (Bhola et al., 2018; Teng et al., 2017).

Surrogate models are specifically developed to be quicker to run, and come in two broad families: lower-fidelity and response surface surrogates (Razavi et al., 2012). In the lower-fidelity model family, conceptual models are a type of model that are still physically based, but that use a very much simplified description of the system (Teng et al., 2017). One of such models is the Height Above Nearest Drainage (HAND) model, which only requires the DEM of the area to calculate inundation extent and maximum water depths. Other types of conceptual models also use only DEM data, such as the Rapid Flood Spreading Model and the Planar or Bathtub method, but only the HAND model uses properties derived from the DEM such as slope and flow direction (McGrath et al., 2018). Perhaps this is why the HAND model is still relevant in literature, having been applied in various research fields: from modelling soil water conditions to landscape classification for distinguishing runoff characteristics in hydrological modelling (Speckhann et al., 2018). Similar to the purposes of this study, the original HAND model is also applied in multiple studies for the fast calculation of flood inundation extents and water depths. Web-based tools have been created that allow users to set the return period of a flood and see the expected flood inundation patterns, or even change the topography of the hinterland to immediately see the effects of their actions on potential floods (for example Chaudhuri et al. (2021) and Hu & Demir (2021)).

In the response surface surrogates family, the models are data driven and do not contain any physical descriptions of the system. Artificial neural networks (ANN) are the type of response surface surrogate most commonly used, and they are trained to find relations between the input and output of another model or field data (Mosavi et al., 2018). As such, they are black box models of which the internal relations remain largely unknown to the modeller. In the past years, ANN have become increasingly popular in the literature, gradually being applied to new problem sets (Chu et al., 2020). For flood modelling, they have mostly been used to model water levels for flow conditions in the river channel (such as Bomers, Meulen et al. (2019) who reconstructed the maximum discharge of the 1809 Rhine flood using ANN). That has quickly changed over the past years, with neural networks now being used for quickly modelling flood water depths (Xie et al., 2021). A popular neural network for these purposes is the Long Short-Term Memory (LSTM) neural network, since it is suitable for predicting time series due to its ability to store information and learn long-term dependencies in data (Le et al., 2019). However, the cases for which neural networks are mostly used consider relatively simple flooding events of rivers spilling into floodplains, while inundation due to dike breaches remains an unexplored field of study (Bentivoglio et al., 2021). Additionally, the number of events used to train the neural networks in the literature is fairly low, at 10 events in Chu et al. (2020) and 24 events in Kabir et al. (2020), for example. This raises the question if the LSTM neural network can be applied to a dike breach flood event, and if it is able to generalize and produce flood inundation results for any discharge wave input that is not similar to the few training events. If this is not the case, then their usefulness in a real-time flood forecasting system is questionable.

1.1 PROBLEM DEFINITION

The enthusiasm about neural networks as a tool for real-time flood forecasting in the research community is increasing. However, as was described in the introduction, neural networks find relations between the input and output data of training cases from hydrodynamic models. This means that they are only ever valid for the data and scenarios for which they were trained. Additionally, gathering the training data takes much time due to the many uncertainties that must be captured by the hydrodynamic models, such as the roughness of the main channel and floodplains, as well as determining when a dike section will fail.

If the river characteristics or hinterland properties like topography change, the neural network will most likely lose its validity. It will have to be retrained, which requires the large time investments for the training data to be made again for the new situation. This is not desirable for a real-time flood forecasting system, which requires to be fully operational at all times and not be subject to extensive periods of retraining. The updating of neural networks is in sharp contrast to that of conceptual models, which only rely on the Digital Elevation Model (DEM) and can be updated in a short while. However, the original HAND model predicts flood inundation extent and maximum water depth based on the water level in the river. It is not suitable for modelling flood propagation for a point source flood such as a dike breach, and especially not in flat delta regions such as the Netherlands. This is because of the way the HAND model inundates areas and that it has no time component for simulating flood propagation behaviour. In case of flood forecasting, it is important to obtain information on where the water flows first, since this knowledge allows decision makers to take appropriate measures such as evacuation for those areas that are at risk first.

As such, there is a need to know which surrogate model type is the most promising for a real-time flood forecasting system that is being used for a longer period of time, during which the hinterland can change.

1.2 OBJECTIVE

The objective of this study is to identify if an LSTM or the HAND model is most promising model for real-time flood forecasting after a dike breach, and investigate its capabilities for long-term use in case of changes in the hinterland. To achieve this, the original HAND model will be expanded with a module that enables the modelling of water depth time series in the hinterland after a dike breach, and its performance in reproducing water depths of a hydrodynamic model will be compared to an LSTM neural network.

The scope of the research is limited to only a part of a real-time flood forecasting system. Such a system consists of multiple models, namely a 1D model for the river discharge and the water level, a model for calculating the moment of dike failure, and finally a model for calculating the flood inundation in the hinterland. In this study, the water level in the river and the moment of dike failure are not considered. So only the flood propagation through the hinterland will be modelled with the dike breach outflow hydrograph as boundary condition. That means that this study does not aim to create a fully functioning real-time flood forecasting system, but rather contribute to knowledge about the best model type for the flood inundation part of such a system.

1.3 RESEARCH QUESTIONS

The main research question of this study is as follows:

What are the drawbacks and benefits of neural networks and conceptual models in the context of real-time flood inundation forecasting after a dike breach for current and future conditions of the hinterland?

In order to answer the main research question and structure the research, several sub-questions are set up. The first sub-question deals with the training performance of the LSTM neural network. The neural network is trained to find relations between the input and output of a known dataset of 1D2D-hydrodynamic simulations. The LSTM is known to be suitable for predicting time series, so if it is trained well it should perform accurately on this data set. However, in reality an unknown flood event can occur that was not specifically trained for. The first question therefore is:

1. *What is the performance of an LSTM network for an unknown set of dike breach flood events?*

The setup of the original HAND model is a well-documented procedure, and its outputs are flood inundation extent and maximum water depths along the riverine area. However, this study is aimed at modelling flooding after dike breaches, so the model will have to be adapted to model flood propagation from a single point source. The second research question is aimed at creating the new model:

2. *How can the original HAND model be modified to model a dike breach flood?*

After the creation of the new model, called HAND.FLOW, the performance will be evaluated and compared to the output of the same 1D2D-hydrodynamic model using the same set of flood events as for the LSTM neural network. The third research question thus concerns the performance of the new model:

3. *What is the performance of the HAND.FLOW model for a set of dike breach flood events?*

The fourth research question deals with the long-term applicability in case of changes in the study area. The Digital Elevation Model (DEM) of the hydrodynamic model will be altered to reflect a topographical change in the hinterland, and the extended HAND model will be updated to reflect this change. As was described in the introduction, neural networks are a black box model for which the internal workings after the training are unknown. This makes them only valid for the conditions they were tested, so even if the neural network performs well after the topographical change it is highly questionable if it is for the correct reasons. It cannot be known if this was a coincidental correct prediction, or if it will perform well for any change in the topography. Furthermore, gathering new training data for the changed situation would require all the hydrodynamic simulations to be conducted again, which would take too much time for this research. Therefore, the fourth question only assesses the performance of the adapted HAND model:

4. *What is the performance of the HAND.FLOW model after a change in the hinterland topography?*

1.4 READING GUIDE

The outline of this thesis is as follows. First, the workings of neural networks and the original HAND model will be described in the theoretical framework (chapter 2). This includes the development from basic neural networks up until the LSTM network, to build an understanding of how the network functions. Second, the methodology is presented (chapter 3), in which the methods for answering the four research question are outlined. The most important steps are the training of the LSTM from the hydrodynamic model (HEC-RAS) data, the development of the HAND.FLOW model, and the assessing of the performance before and after a change in the hinterland topography. A flow chart of the main research activities is shown in Figure 1. After the methodology, the results of the comparison between the hydrodynamic model (HEC-RAS) and the HAND.FLOW and LSTM models are presented in detail in chapter 4. Finally, the research and its implications are discussed, conclusions to the research questions are drawn, and several recommendations to researchers and policy makers are made in chapters 5, 6, and 7 respectively.

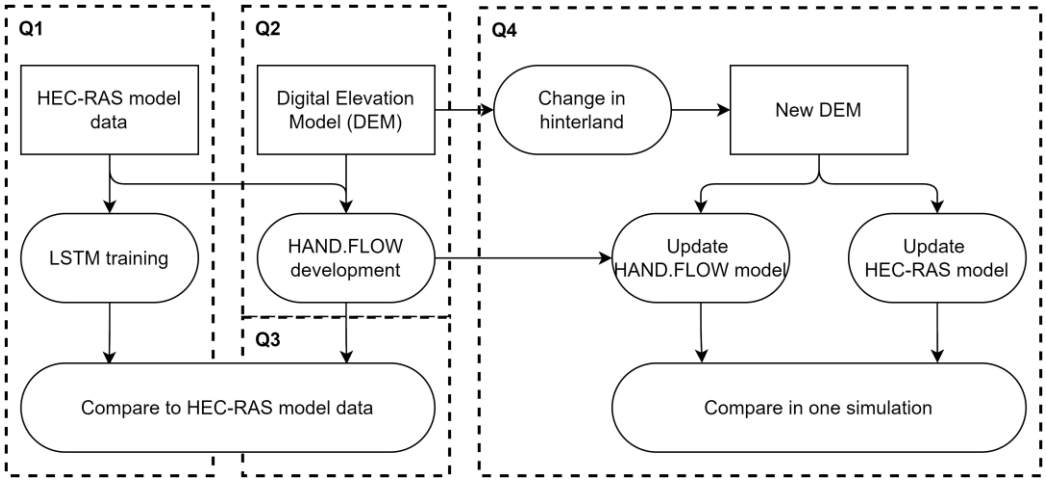


Figure 1 – Flow diagram of steps in this research and the research questions they help answer

2 THEORETICAL FRAMEWORK

2.1 ARTIFICIAL NEURAL NETWORKS (ANN)

Neural networks are data-driven models that generate an output from one or more input parameters. The most common form of neural networks has traditionally been the multi-layer perceptron (Razavi et al., 2012). In its most basic form, it consists of three layers that communicate values from the input to the output via one or more hidden layers (Figure 2). Hence, these networks are also called feed-forward neural networks. The input layer receives the input data for which a prediction is desired at several nodes in the network called neurons. These generate an output and pass it to the subsequent layers, which each consist of neurons as well. Each neuron in a layer is connected to all neurons in the preceding layer, in a so-called densely connected network. Finally, the output layer produces the desired output.

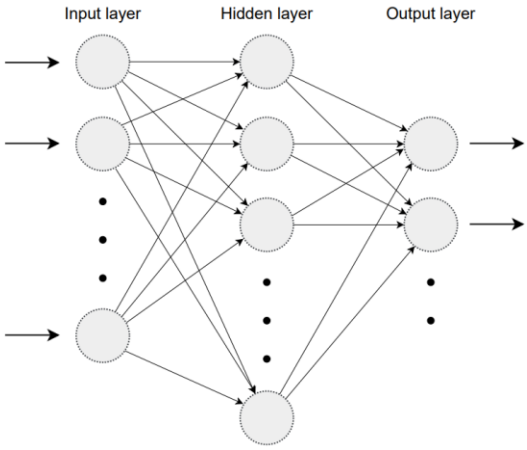


Figure 2 – Structure of three-layered feed-forward neural network with several neurons in each layer

Zooming in on a neuron of Figure 2, it can be seen that neurons output a value based on all the received input values (Figure 3). Each neuron receives inputs from previous neurons, assigns them an individual weight factor and sums the weighted values (Staudemeyer & Morris, 2019). Additionally, neurons have an internal input called a bias. The bias and the weighted sum are fed to the threshold or activation function, which determines the output value of the neuron.

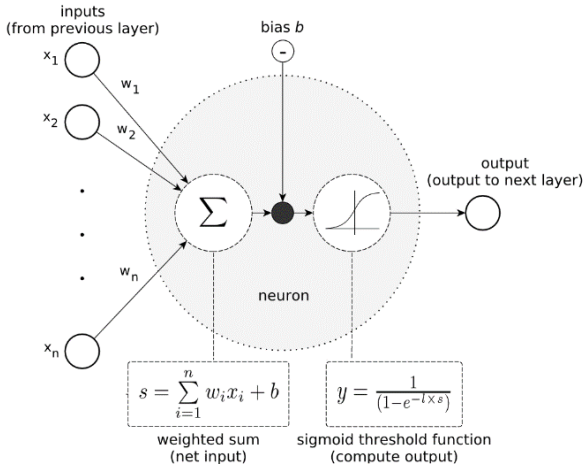


Figure 3 – Structure and workings of one neuron (Staudemeyer & Morris, 2019)

Training of neural networks is done by repeatedly presenting it with a set of inputs and corresponding desired outputs. The neural network initializes with randomized weight factors between all neurons, and computes its own output from the given input (Staudemeyer & Morris, 2019). Through comparison of the true output and its predicted output, the network calculates the error of the network using a loss function. Several loss functions exist for evaluating the performance of a neural network compared to the true outputs. For different problems, such as classification or regression, different loss functions might be more applicable (Le et al., 2019).

After the error of the network is calculated, the relative contribution of each neuron to this error is derived in a process called backpropagation, which allows for a small adjustment of the weights of the neuron (Le et al., 2019). The algorithm responsible for monitoring the change in the loss function due to weight adjustments is usually a stochastic gradient descent optimizer. Backpropagation starts from the output layer and proceeds towards the input layer, chasing the most optimal gradient of the loss function in every step (Staudemeyer & Morris, 2019). Eventually, all updates to the weight factors are made and the cycle is repeated. The network is again presented with the inputs and outputs and the weights are updated through backpropagation of the error. Each of these cycles during training is called an epoch, and every epoch results in a neural network that is slightly more accurate in reproducing the desired output from the given input.

A disadvantage of backpropagating the errors from the output layer towards the first layers is that neurons in these first hidden layers are left with only very small gradients for optimization, resulting in their weights being adapted very slowly. This problem is called the vanishing gradients problem, and it leads to very long training times (Le et al., 2019). Another problem associated with traditional neural networks is that they are not particularly well-suited to sequential data problems, such as sentence completion or time series problems. This is because the inputs and outputs are independent from each other, so the network has no information about its state in a previous computation step (Zhang et al., 2018).

2.2 Recurrent Neural Networks (RNN)

Recurrent neural networks (RNNs) were developed in the 1980s to have a chain-like structure that allows them to store information about previous computation steps (Le et al., 2019). This stored information is used again for the next time step. Due to such circular and self-feedback connections between neurons, RNNs are more dynamic and more effective in time series problems (Staudemeyer & Morris, 2019). The neurons can be connected in various ways to create an RNN, but the degree of connectivity varies. For purposes of understanding the general working, the Simple Recurrent Network developed by Elman in 1990 is explained.

This RNN is very similar to the standard three-layered ANN displayed in Figure 2, but there are additional neurons in a so-called *context layer* (Figure 4). Every neuron in the hidden layer is connected to a neuron in the context layer, and after a pass through the feed-forward network information is stored in the neurons of the context state (Staudemeyer & Morris, 2019). In the next pass through the network, the neurons in the hidden layer receive information not only from the input layer, but from the context layer as well. This structure allows the network to store information for use in a later phase, essentially giving it a kind of memory.

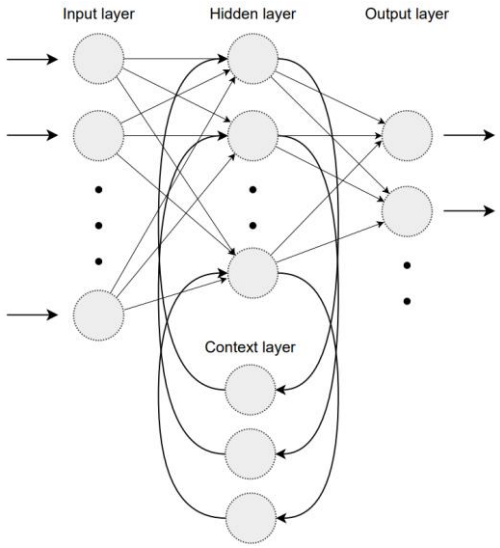


Figure 4 – Simple version of an RNN (Elman network, after Staudemeyer & Morris (2019))

During training, RNN utilize an error backpropagation algorithm similar to the one described for ANN. However, an important difference is that this algorithm works the error through the time steps, leading to its name of backpropagation through time (Staudemeyer & Morris, 2019). It works by considering that for a given a finite period of training time, an RNN can be considered as a traditional feed-forward neural network (Staudemeyer & Morris, 2019). The RNN is unfolded in time, and the errors are backpropagated for each time step independently. To update the weights in the original RNN, the difference in weights is summed over all individual time steps. Although it is a complicated procedure, recurrent neural networks are able to learn short-term dependencies in the data of the problems they are intended to solve. However, backpropagation through time does not eliminate the vanishing gradient problem, and training is not sufficiently efficient to learn long-term dependencies (Le et al., 2019).

Long Short-Term Memory (LSTM)

The Long Short Term Memory (LSTM) model was introduced in 1997 and addresses the vanishing gradient problem. It is a special type of RNN, which is able to store information for longer periods of time and to learn long-term dependencies (Le et al., 2019). The explanation of Le et al. (2019) will be summarized here. LSTM networks operate using memory blocks called cells (Figure 5). The main data flow happens through the cell state C_t, which is used to store information and pass it on to the next step. Data can be added to the cell state via a number of transformation functions that operate as gates in the network. Three gates are used in the LSTM network: the forget gate, the input gate and the output gate (Figure 5).

First, information is identified that should be used to update the cell state C_t. The available information is the new input x_t and the hidden state or output of the previous step h_{t-1}. A sigmoid transfer function identifies and excludes irrelevant data by creating a *forget vector* with values ranging from 0 to 1. Closer to 0 means to forget and closer to 1 means to keep. The forget vector is multiplied with the cell state of the previous step C_{t-1}, to either forget or to keep the information of the previous cell state. An example of how this gate operates is in the field of language modelling: suppose the neural network is tasked with predicting the next word in a sentence, based on the previous words. The hidden state h_{t-1} might have stored, among other things, the gender of the current subject of the sentence. For example, the sentence might be “John is hungry”. The next word could be a personal pronoun, which the neural network can then obtain from the hidden state h_{t-1} as “he”, to continue the sentence with “He goes to the supermarket”.

If the new input, however, is another subject called “Emma”, the forget gate comes into action to forget the gender of the previous subject John, since the sentence is no longer about him. Determining which data is important to keep or which data can be forgotten is part of the training process.

Next, the input gate is used to decide which information should be added to and stored in the cell state. The sigmoid function determines if the new information is relevant or not, and the hyperbolic tangent function decides the importance of each value. The results are multiplied, and added to the old cell state C_{t-1} to form C_t. This would result in the gender of Emma to be stored in the cell state. The final gate determines the output h_t of the LSTM. The output gate first filters the input via a sigmoid transfer function, and then multiplies it with the new cell state C_t processed by a hyperbolic tangent function.

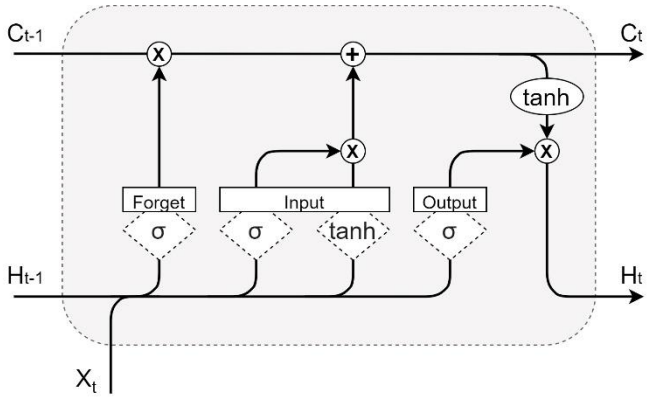


Figure 5 – Structure of an LSTM neural network (reproduced from Le et al. (2019))

In previous sections on ANN and RNN, it was described that the networks consist of interconnected neurons. Figure 5 displays an LSTM cell, but it is incorrect to think of this structure as the LSTM equivalent of a neuron (Karim, 2020). The neurons are actually inside the structure of the LSTM, in each of the gates. It is important to note that LSTM works with vectorized information internally, and that the dimensions of the input x_t can differ from the dimensions of the cell state C_t and hidden state h_t. The dimension of these latter two is actually a modelling choice that should be made, called the number of units. Figure 6 thus shows an LSTM with three input values at each time step x_t, and the number of units set to two (visible as C_t and h_t). The gates are shown as they operate internally, as a small and dense neural network that processes the concatenated input and hidden state to the cell state, using either the previously described sigmoid or hyperbolic tangent activation functions.

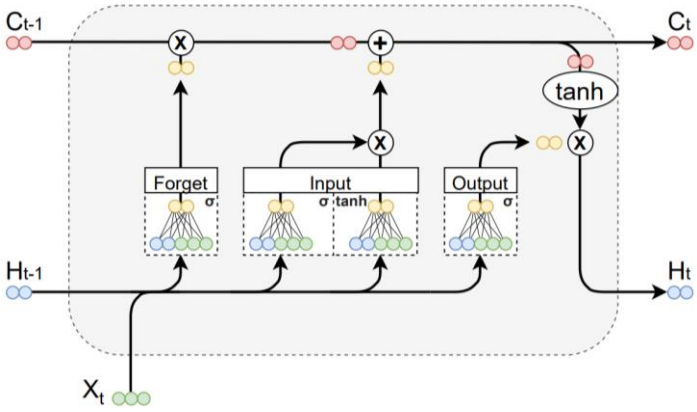


Figure 6 – Structure of an LSTN neural network with flow of dimensionalities (reproduced from Karim (2020))

2.3 Height Above Nearest Drainage (HAND) model

The original HAND model is a conceptual model that does not utilize physical descriptions of water flow, such as the shallow water equations. Instead, it uses characteristics of the Digital Elevation Model (DEM) of an area to derive the flooded area using the water level in the river (McGrath et al., 2018). The model was contrived by Nobre et al. (2011) and is constructed in a number of steps (Figure 7). First, from the DEM the Local Drainage Direction (LDD) is derived, which is made up of the downward paths that water flows into as it makes its way downstream (Nobre et al., 2011). This is done using the eight-direction pour point model, or D8 algorithm. For each grid cell, the local drainage direction is determined in a 3x3 cell window with the considered cell in the centre (dotted red square in Figure 7.1). One of the eight neighbouring grid cells has the steepest elevation gradient with respect to the centre cell in the DEM, so that neighbour will be the next in the local drainage direction path (cell towards which red arrow points in Figure 7.1).

Also visible in Figure 7.1 are five blue grid cells. These cells are what the model will consider the drainage cells of this small DEM. The most upstream cell of these drainage cells is determined using a threshold that has to be set manually. This threshold is the number of cells that drain into the most upstream drainage cell, and is equal to 9 in the figure. In an actual study area, it can be calibrated such that the identified drainage cells correspond well with the actual river in the area.

Then, every grid cell in the DEM is associated with the drainage cell that its water eventually drains into, which could be interpreted as the sub-catchment of each cell on the drainage path (Figure 7.2). Next, within each sub-catchment the topographic height of the drainage cell is subtracted from the height of its associated cells, which results in the Height Above Nearest Drainage (HAND) value for each cell (Figure 7.4).

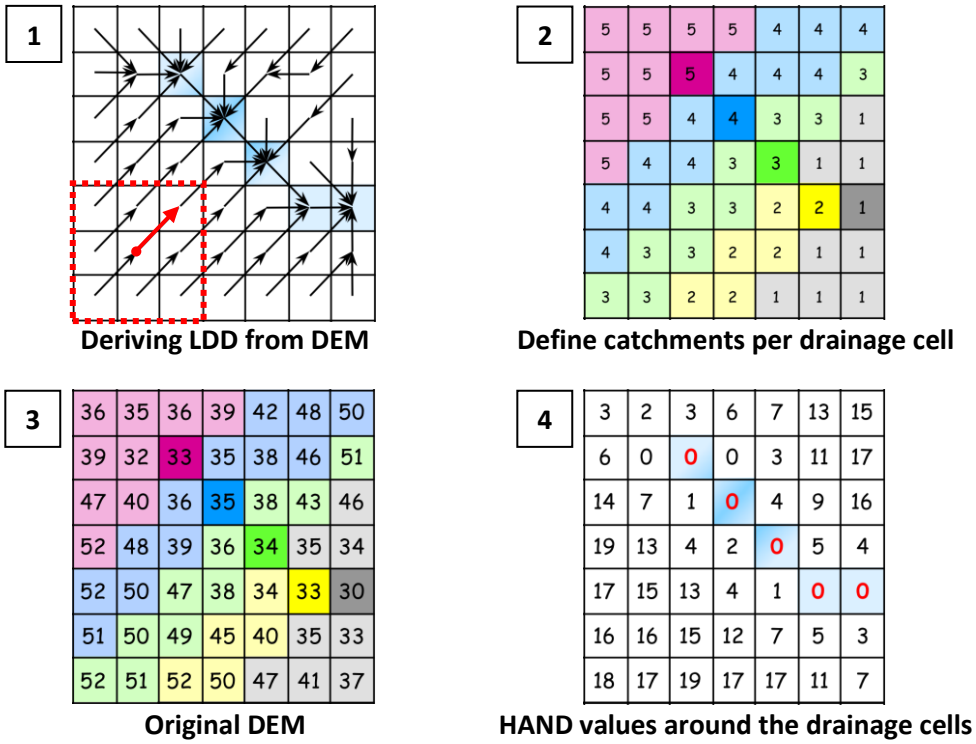


Figure 7 – Procedure to generate the HAND model (figure adapted from Nobre et al. (2011))

Nobre et al. (2011) initially developed the model for use in classifying soil environments for soil water, but the model was quickly applied in flood inundation modelling too. The inundation extent is determined by selecting the cells with HAND values less than the water level in the river channel (McGrath et al., 2018). In Figure 7.4 the river channel is represented by the blue drainage cells with a HAND value of 0. For a water level of 5 meters in the river, all surrounding grid cells with a HAND value below 5 will be flooded. The water depth is calculated by subtracting the HAND value from the water level: for a cell with HAND value 2, the water depth will be 3 meters.

The water level for the upcoming flood has to be determined using another modelling technique, such as a Q-H relationship or 1D model for the river in question. In any case, the HAND model does not spread the water according to total flood volume, and does not start its filling process at one specific location, but along the complete river as a whole. This is under the assumption that the water level in the river is constant as it flows downstream (Nobre et al., 2016). Usually, the HAND model is applied in riverine areas that are valley-like, such as Hu & Demir (2021) in the river valley of Cedar Rapids. These areas have somewhat clear topographical boundaries on both sides of the river (Figure 8). However, in the study area in the Netherlands and in delta regions in general, there is no hillside marking a clear boundary for the flood, as the hinterland is mostly very flat or even lower than the river. Thus, the method of considering cells to be flooded if their HAND value is lower than the water level in the river will result in an unrealistically large flood in such areas. Therefore, this research will adapt the model to work with volume and time, resulting in applicability in the Netherlands.

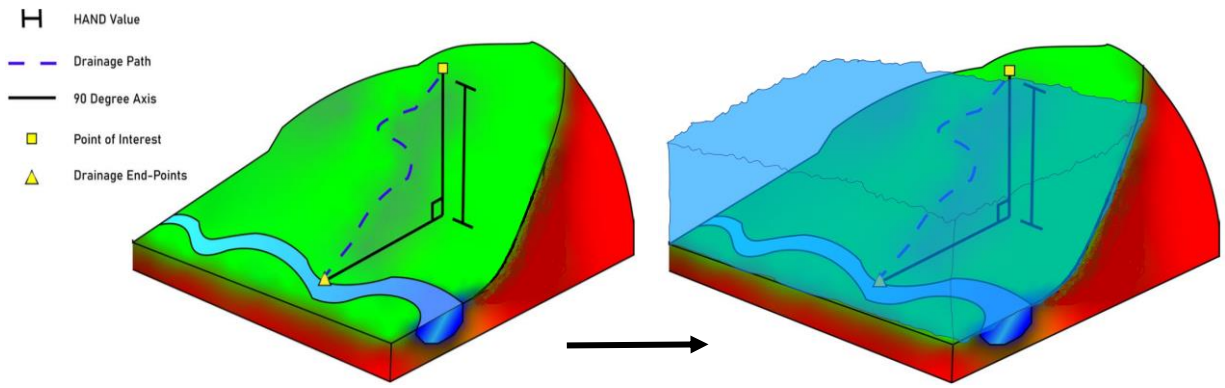


Figure 8 – HAND value on hillside (left) and corresponding flood level (right) (figure adapted from Scriven et al. (2021))

3 METHODOLOGY

3.1 DATA

The output data of the 1D2D hydrodynamic model constructed by Bomers et al. (2021) in HEC-RAS will be utilized. The model consists of the Dutch part of the Rhine River as it enters the Netherlands (Figure 9). The upstream boundary is located at Emmerich in Germany, and uses a discharge wave as a boundary condition. Normal water depths in the river function as downstream boundary conditions. The breach growth is modelled as immediate: as soon as the water in the river overtops the dike, the dike is reduced to the natural terrain level and a constant breach width of 150 meters occurs. Although several more complicated and growing breach models exist, Bomers et al. (2021) mention that the overland flows and inundation extents are not that sensitive to the breach model. Additionally, the objective of this current research is to create a model that takes the outflow hydrograph as input, so the accuracy of the HEC-RAS outflow hydrographs compared to reality is less relevant. Therefore it is assumed that the data from Bomers (2021) are applicable to this research. For the calculation of the inundation extent, the hinterland is implemented on a 150x150 meter grid, with realistic roughness values for the land use in the area.

The model results of 73 discharge waves and corresponding flood inundation patterns for two breach locations of the Rhine river in the Netherlands are available and suitable for this project (Figure 9). Note that Bomers (2021) were only interested in the outflow hydrographs of the dike breach, not in the flood inundation. Therefore, the water depth data and results have not been published, but were made available for this research through personal communication. Only the dike breach location in the IJssel river breached during all 73 simulations, which is why only this breach location will be considered in this study. The affected part of the hinterland is coloured darker green in Figure 9, which will be the only area modelled with the surrogate models. However, in some simulations both locations experienced a dike breach, which results in a lower outflow hydrograph peak at the IJssel river breach. Since the surrogate models are based on the outflow hydrograph only, it is expected that this will not greatly influence their performance accuracy. Output data of the HEC-RAS model that will be used are outflow hydrographs of the dike breach and the flood water depths in the hinterland throughout the flooding event.

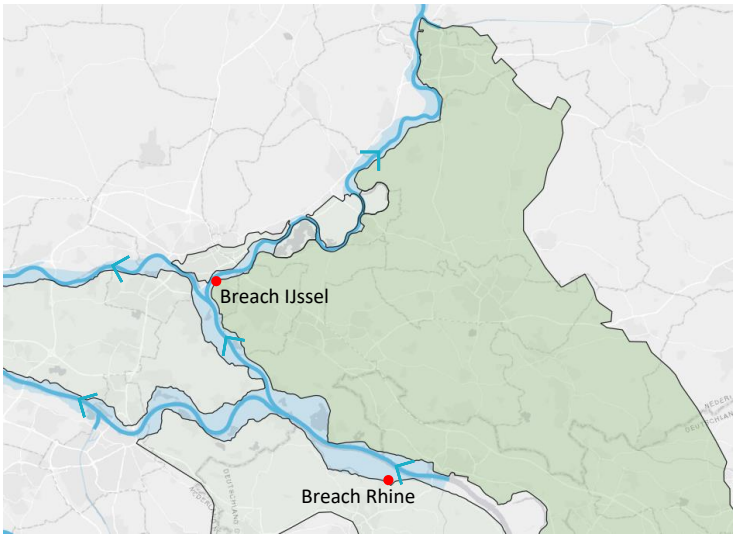


Figure 9 – Study area of hydrodynamic HEC-RAS model constructed by Bomers et al. (2021)

3.2 LSTM TRAINING

The training procedure of the LSTM neural network covered in this section is split into three steps. First, the pre-processing of data is important for the functioning of a neural network. Afterwards, the LSTM is programmed and its parameters then have to be optimized.

3.2.1 Data pre-processing

The input data to any LSTM layer must be three-dimensional, consisting of samples, time steps and features (Le et al., 2019). A sample is an instance of an input sequence, so the number of samples in this study is the number of available HEC-RAS simulations. The amount of time steps equals the number of moments of observation within each sample. For the neural network to function, the amount of time steps should be the same in each sample. Lastly, features are the values of the observations that are made at every time step. In this study, the number of features equals the number of grid cells in the hinterland (49,733 grid cells in the dark green area of Figure 9). Both the input variable of outflow hydrograph data and the target variable of flood water depth over time should conform to this three-dimensional format.

The data from the HEC-RAS simulations required processing to fit in this three-dimensional format. First, due to the way the study by Bomers (2021) was carried out, not all simulation samples are of the same length. While the interval for writing outputs is constant at three hours, some simulations have more time steps due to a longer simulation period. To fulfil the requirement that all samples have the same amount of time steps, the shorter samples were zero-padded to the length of the longest by adding zeroes in front.

The 73 samples were split to create separate datasets for training, validation and testing of the neural network. Training data is used to update the weights of relationships between nodes of the neural network during the training. Validation data is not used for weight updates, but only monitored to see if the neural network is overfitting and to terminate the training process if overfitting is happening (see for example the description of early stopping in the next section). Testing data is not seen or evaluated by the network during the training, but is used after completion of the training to review the neural network's performance in unseen scenarios. Conform the literature, a non-overlapping 60% – 20% – 20% split is used for these purposes respectively (Chu et al., 2020; Rajaee et al., 2019). This means that 43 simulations were used for training, 15 for validation and 15 for testing the model.

A last step was to normalize the data. Neural networks have been found to be sensitive to the size of input values, with large input values being able to result in instabilities during the network's convergence (Shao et al., 2020). Therefore, both the outflow hydrograph and the flood water depth data were normalized to an interval of 0 to 1 via Equation 1. The water depth data was normalized per feature, so per grid cell.

$$x_i' = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{1}$$

Where:

- x_i *is the feature at time step i, with* x_i' *its normalized counterpart*
- x_{min} *is the minimum value of the feature in the dataset*
- x_{max} *is the maximum value of the feature in the dataset*

Importantly, the scope of determining x_{min} and x_{max} was limited to the training and validation data sets only. If the testing data set was also included, then its information could leak to the normalized training data. Since it is the purpose of the testing data set to be completely unknown to the neural network, it was excluded from the determining of x_{min} and x_{max}.

3.2.2 Programming neural network

The neural network was programmed in Python using the Keras library, which acts as an interface for the TensorFlow machine learning and AI library developed by Google. In Keras, a neural network is constructed by first defining the layers of the model, then compiling the model, and then fitting the model to the data.

In the first step, the architecture of the neural network is decided. The most simple form is the vanilla LSTM model, which consists of only a single layer. Stacked LSTMs contain two or more layers, and there are even studies combining LSTM layers with other types of algorithms (such as Liu et al. (2020), who combine LSTM with the K-nearest neighbour algorithm for flood forecasting). Through initial testing, an appropriate architecture was determined (see section 4.1.1). Following the LSTM, a standard dense layer was added, which connects each neuron to each cell in the previous layer. This layer contains one neuron for each grid cell, enabling the model to make predictions for the complete study area per time step.

Within the LSTM and dense layer, three modelling choices had to be made. First, the number of units in the LSTM layer is explained as the number of neurons that make up the hidden state of the network, resulting in more units requiring more processing power. Second, the degree of dropout is a method for regularization of the network, by randomly excluding a fraction of the input during training. Its intended effect is to improve model performance and reduce overfitting (Le et al., 2019). The third choice consists of the activation functions, which define the output of a neuron based on its inputs. Both the LSTM and dense layers support various types of activation functions. However, it was chosen to make use of graphics card (GPU) computing capabilities of Keras, since training using the processor (CPU) could take upwards of an hour per network. Since the network needs to be trained many times to find the best set of parameters, computation times need to be sufficiently low. GPU computing was about 12 times faster, but to enable it in Keras the LSTM layer can only use the hyperbolic tangent and the sigmoid functions. These were thus kept at their default setting. For the dense layer, more options were available. The choices for the parameters described were made using an optimization process described in the next section (3.2.3).

The second step of the network construction is the compiling of the model. Here, the loss function is defined, which will be optimized by the optimization algorithm. In this study, loss functions for regression problems are relevant, including the Mean Square Error (MSE), its root (RMSE) and the Mean Absolute Error (MAE). Initial testing revealed that the MAE (Equation 2) resulted in more accurate predictions, which is why it will be used as the loss function. It is only calculated if a grid cell is flooded for at least one time step. Otherwise, lots of dry cells artificially decrease the MAE.

$$\mathrm{MAE} = \frac{1}{\mathrm{n}} \sum_{i=1}^{n} \left| y_{i,\ true} - y_{i,\ predicted} \right| \tag{2}$$

Where:

- y_i *is the feature at time step* i
- n *is the number of predictions*

For the optimization algorithm, a stochastic gradient descent method called Adam is often used in literature (e.g. Le et al. (2019) and Liu et al. (2020)). It has been found to be well-suited for problems that have large data sets and require many parameters (Kingma & Ba, 2017). Keras allows an important parameter of Adam to be changed: the learning rate. This rate defines the magnitude of the weight updates within each epoch of training, and thus determines the speed at which the network converges.

The third step is to fit the model to the data. The training and validation dataset are given as inputs to the model. To divide this into the 60% – 20% split that was mentioned earlier, a parameter called the validation split is set to 0.25. The remaining 20% of data is saved to test the model after training. The number of epochs is defined as 1000, and to prevent overfitting early stopping is used. This checks if the loss function on the validation dataset has not improved over a set number of epochs, which would indicate overfitting on the training dataset. It was chosen to stop the training after 100 epochs of no improvement.

3.2.3 Hyperparameter optimization

Four hyperparameters that were discussed in the previous section are optimized using KerasTuner: the number of units, the degree of dropout, the activation function of the dense layer and the learning rate of Adam. KerasTuner provides three types of search algorithms: random search, hyperband and Bayesian optimisation (Kumar, 2021). Random search is the most basic of the three, randomly sampling parameters from the complete hyperparameter space. This means that it can find a combination of hyperparameters that performs poorly, but continue sampling similar combinations from that region of the hyperparameter space. Hyperband search tries to overcome this problem, by only training the randomly sampled models for a few epochs. It keeps track of the most promising sets of parameters, and eventually only runs full training on the final candidates. However, even in hyperband search, the parameters are sampled randomly. Bayesian optimization instead samples only the first few sets at random. Based on their performance, it assesses the probability that another set of parameters achieves a better score. That way, it takes into account the past sets of hyperparameters that were tried to find even better sets. This is why Bayesian Optimization is chosen in this study.

The Bayesian optimization algorithm optimizes the loss function of the model, which in this study is the mean absolute error (MAE). It requires a range or set of options to choose from for optimizing the given hyperparameters. The number of units of the LSTM layer was set to a range between 16 and 1024, with steps of 16. The dropout of the LSTM layer was allowed to vary between 0 and 0.25, with steps of 0.05. For the learning rate of the Adam optimizer algorithm, three options were given: 0.01, 0.001 (default) or 0.0001. Finally, for the dense layer's activation function, the algorithm could choose from linear, rectifier or sigmoid functions. The rectifier function uses only the positive part of a linear function, so negative values are truncated to 0. The results of the hyperparameter optimization are found in Table 1 in section 4.1.2.

3.3 HAND.FLOW MODEL

The set-up of the new Height Above Nearest Drainage (HAND) model is described in this section. As was described in section 2.3, the original HAND model works with the Digital Elevation Model (DEM) of the study area as input. The resolution of the DEM used is 10x10 meters, which is much higher than the 150x150 meter resolution used by HEC-RAS and the neural network. This was chosen since HEC-RAS uses this same 10x10 meter resolution DEM as its base input, and utilizes it to calculate the 150x150 meter resolution map. In initial tests with the new HAND.FLOW model, the 10x10 meter resolution was sufficiently fast in terms of computation time that it was decided to continue the research with this high resolution. The effect of using the HAND.FLOW model on lower resolutions such as 150x150 meter is covered in section 5.3.1 of the discussion.

3.3.1 Data pre-processing

Normally, the DEM of the entire riverine area is used in a HAND model, including the river channel. However, in this study only the hinterland is of interest, since this is where the flood inundation will take place. Therefore, just as for the LSTM, the DEM used has the river dikes as its boundary.

To derive the Local Drainage Directions (LDD) from the DEM, a plugin called PCRasterTools is used in the open-source geographic information system QGIS. This plugin allows for mapping operations and calculations useful in environmental modelling disciplines such as geography, ecology and hydrology. The plugin is also available as a Python package. However, the function for the creation of the LDD could not be successfully run in Python, so QGIS is used for it. After the DEM is converted to the PCRaster .map format, the function *lddcreate* is used to carry out the D8 algorithm. As mentioned in section 2.3, this algorithm determines the steepest downslope neighbour of each grid cell, resulting in the LDD.

In the original HAND model procedure, the next step is to determine the drainage cells of the study area. Normally this is done by setting a threshold for the amount of cells that should drain in the most upstream drainage cell. The threshold can be calibrated such that the identified drainage cells correspond well with the actual rivers in the study area. Such an approach is valid for riverine floods in which the source of the flood are the drainage cells themselves, since a river water level is set on these cells and extrapolated towards the neighbouring cells. However, in this study, the source of the flood is a dike breach, which the original HAND model cannot model as it is a single point source of water. The next section discusses adaptations to the model that make it possible to model such an event.

3.3.2 HAND model extension

In order to make the new HAND model able to deal with a point source such as a dike breach in a flat and low-lying delta such as the Netherlands, an extended version of the HAND model is presented called the HAND point flow model (HAND.FLOW) model. It models not just the final inundation extent, but also the propagation of the flood through the hinterland starting at the dike breach. In order to accomplish this, two mechanisms are introduced. First, the flood water is allowed to travel only a certain distance from the dike breach per time step along a flow path. By setting such an increasing limit to the distance travelled, an increasingly large area is considered floodable every time step. This enables the HAND.FLOW model to take flood volume per time step as its input instead of a river water level like the original HAND model, which will be the second mechanism described in the coming sections. The basic step by step procedure of the model is as follows:

PATHFINDING
- Define the dike breach location
- Follow the flow of water downstream from the breach (via the LDD)
- Limit the distance travelled along the path per time step, as water can only travel so far in a time step
- Define the grid cells of the flow path in a time step as the “drainage cells”

VOLUME & WATER LEVEL
- Derive from the LDD the sub-catchments of the drainage cells (like in Figure 7.2)
- Calculate the Height Above Nearest Drainage (HAND) values of all cells in the hinterland
- Raise the water level on the drainage cells, and extrapolate to cells with lower HAND values
- Calculate the volume of flood water in the hinterland for this water level
- Find the water level for which the volume in the hinterland matches the volume that has entered through the breach

Pathfinding from breach

In order to let the water flow along a path from the dike breach, the first step is to define the row and column coordinates of the dike breach in the Digital Elevation Model (DEM). From this grid cell, the flow path along the Local Drainage Direction (LDD) is followed. As was explained in section 2.3, the LDD describes to which of the eight neighbouring cell water will flow downstream. Every cell travelled along the LDD adds to the total distance travelled depending on the resolution of the map, which in this study is 10x10 meters: adjacent neighbours add 10 meters to the distance travelled, diagonal neighbours add $\sqrt{2} \cdot 10$ meters.

In this study, the distance limit is imposed using data from the available HEC-RAS simulations. The arrival time of flood water on the cells of the flow path from the dike breach was analysed to determine the relationship between arrival time and distance to the dike breach. Figure 10 shows for 15 test flood events the arrival time of the flood water on the cells of the LDD flow path from the dike breach. Note that the figure shows the time step since the dike breach instead of the actual arrival time, to better explain the equation that will be derived for the distance limit. Every time step is 3 hours in real time. The individual data points are given a high transparency, so that a high concentration of data points reflects the general trend and a low concentration reveals outliers in the arrival times.

It can be seen that for flow path cells at a distance up to 20,000 meters from the breach, there are a lot of instances of the flood arriving immediately (0 time steps after the breach). Sometimes the flood also arrives at time step 1, but through the lower concentration of points it can be concluded that this happens less often. After 20,000 meters from the breach, the figure has a clear upward trend: for roughly every additional 5,500 meters distance, the flood arrives one time step later. This corresponds with the red trend line through the areas of highest concentration of data points in Figure 10. The distance limit for the HAND.FLOW model is therefore programmed as a simple equation depending on the time step t since the dike breach: $limit = 20000 + 5500t$.

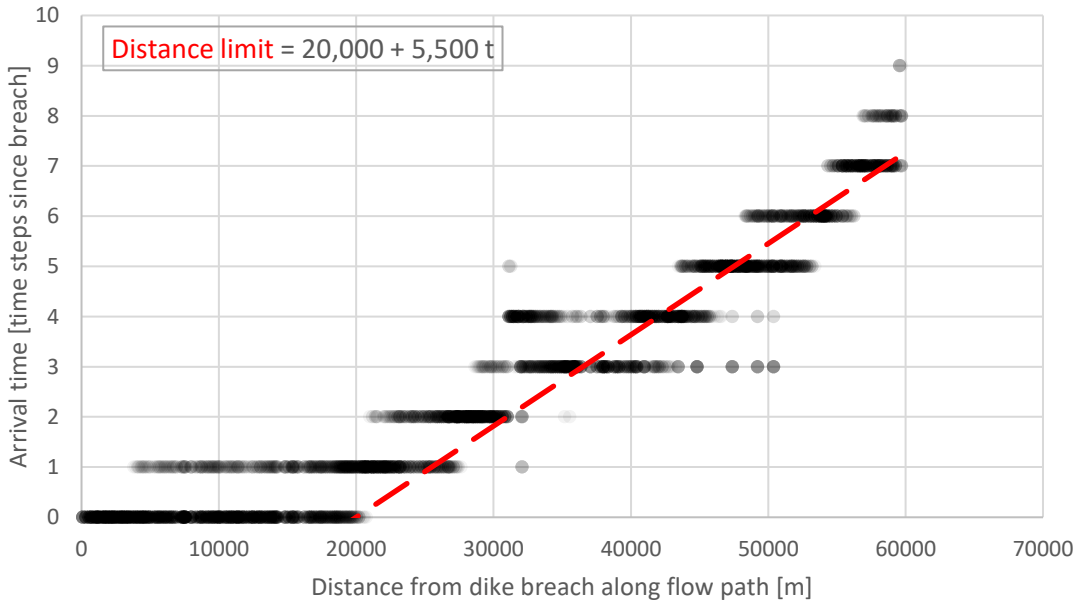


Figure 10 – Relationship between arrival time and distance from dike breach in 15 HEC-RAS test events

With the pathfinding algorithm and the distance limit relationship, a rough outline of where the flood can flow every time step can be plotted (Figure 11). Starting from the breach, the light blue flow path moves downstream through the hinterland, with the perpendicular darker lines indicating where the path is limited by the distance limit relationship throughout the flood duration. Note that these perpendicular lines strongly correlate with the flood extent per time step, but that other factors are also important. This has to do with the next step in the procedure: finding the sub-catchments for each grid cell of the flow path, and finding the water level in the area.

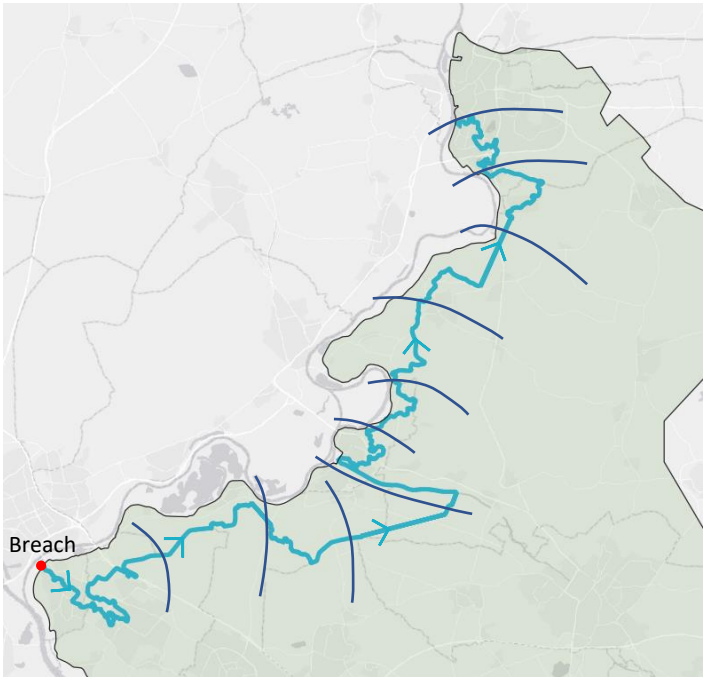


Figure 11 – Study area with path found by algorithm and distance limit indicated per time step

Flood volume and water level

As was described in the previous paragraph, the flow path from the dike breach downstream into the hinterland is found using the Local Drainage Direction (LDD) and limited every time step by the distance limit. The grid cells that make up the flow path are considered the drainage cells that were described in section 2.3. Next, the original HAND model procedure is followed: the sub-catchments of the drainage cells are derived from the LDD. In Figure 12, the section of the flow path from Figure 11 corresponding to the simulation up to the second time step is shown along with the sub-catchments. It shows the same principle as Figure 7.2 in section 2.3 on the theory behind the original HAND model. Each drainage cell making up the flow path has been assigned a random colour, and all the cells in the LDD that drain into that cell are coloured accordingly. To understand the draining patterns, a few arrows show how the water flows downstream from a sub-catchment into the flow-path.

As Figure 12 indicates, cells beyond the dotted lines do not drain into one of the flow path cells. These dotted lines are where a road and railroad lie approximately one meter above ground level. Grid cells on one side of this topographical obstacle can drain into the flow path cells, while grid cells on the other side of this cannot. This becomes important in the next section, on how the HAND.FLOW model is programmed to also flood the area beyond these topographical obstacles.

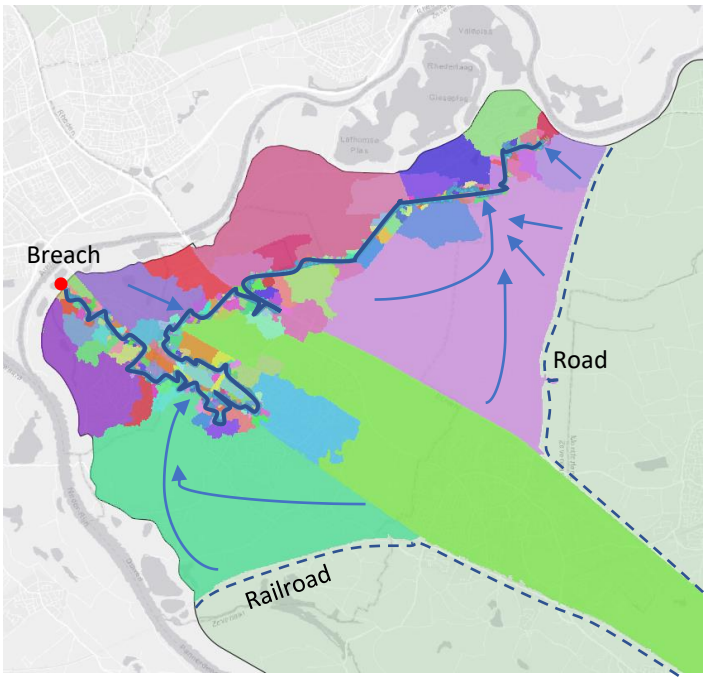


Figure 12 – Sub-catchments of flow path indicated by various colours. Blue arrows indicate that water flows from the sub-catchment into the flow path, due to the terrain height.

Within each sub-catchment, the topographic height of the drainage cells making up the flow path is subtracted from the height of the other cells, which results in the Height Above Nearest Drainage (HAND) value for each cell. To calculate the flood water depths in the hinterland, the volume that has entered the hinterland through the dike breach needs to be computed. Similar to the neural network constructed in section 3.2, the input of the HAND.FLOW model is the outflow hydrograph through the breach in m^3/s. The flood volume is calculated by multiplying this discharge with the duration of a time step (3 hours).

With the total flood volume known for each time step in the simulation, a water level can be found that matches this volume. To start the search procedure for this, a low water level is placed on the drainage cells making up the flow path. All surrounding cells with HAND values lower than the water level are flooded too, with the water depth of a cell being equal to the water level minus the HAND value of that cell. Using the water depth on all grid cells and the resolution of the DEM, the volume of water in the area is calculated. The water level is increased with small steps of 0.1 meters until the volume is approximately equal to the total flood volume at that time step.

Pathfinding out of pit

A complicating factor of letting the flood flow from the dike breach into the hinterland is that the water can end up in a so-called pit. This is a grid cell which has no downstream neighbours; it is the lowest point in an area enclosed by higher terrain features, such as a meadow surrounded by roads with a higher elevation. The location shown in Figure 12 contains such a pit in the top-right corner: the flow path ends here in this time step because it is at the lowest point in the area enclosed by the dotted lines (a road and a railroad). This situation (shown enlarged in Figure 13) is used to explain how the HAND.FLOW model deals with this situation for the next time step below.

In a grid cell that is a pit, the Local Drainage Direction (LDD) does not point to one of the eight neighbouring cells, but to the cell itself. Therefore, the pathfinding algorithm described above would get stuck, while in reality the flood can continue flowing if the water level in the area rises above the surrounding terrain feature. To programme this behaviour in the HAND.FLOW model, a dedicated function is developed that activates when the flood is stuck in the pit. Its goal is to find the moment when and the location where the water depth is high enough to flow over the enclosing terrain features. The algorithm makes three checks for all the grid cells that are flooded:

- Is there a neighbouring cell that is not flooded?
- Is this neighbour outside of the known HAND sub-catchments? It has to be outside of the known sub-catchments, so that the flow path from this neighbour does not lead back to the same pit.
- Is the water level of the grid cell higher than the elevation of this neighbour?

In case that the three checks are passed, the neighbouring grid cell lies outside of the enclosed area and following the LDD from it will not lead back to the pit (Figure 13). Therefore, this cell is added to the path and the flood continues downstream from there. However, it could be the case that multiple grid cells meet the requirements for continuing the path away from the pit. In this case, it is assumed that the grid cell closest to the pit will be most accurate compared to the location where the flood will flow over the enclosing high terrain feature in reality. This is assumed due to that in reality, water levels are highest close to the pit, as a result of water flowing towards and accumulating in this lowest point of the area.

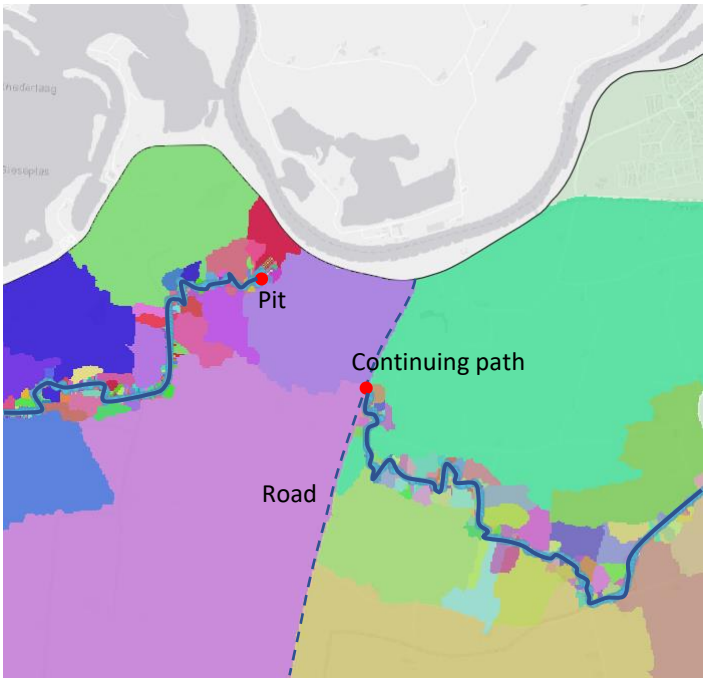


Figure 13 – Example of a pit and the flow path continuing downstream beyond the obstacle

3.4 UPDATING DEM

To research if the HAND.FLOW model is able to function in a real-time flood forecasting system for a longer period of time in a changing riverine area, a change in the hinterland topography is made. The location chosen for this experiment is between the service interchanges at Duiven and Westervoort of the A12 highway. The highway is raised some 2 meters above the surrounding area on both ends of these service interchanges, but lies at ground elevation in the section between them (Figure 14). Therefore, this middle section is changed in the DEM to be 2 meters above ground elevation as well, in order to compartmentalize the study area close to the dike breach (highway section in red circle in Figure 15).

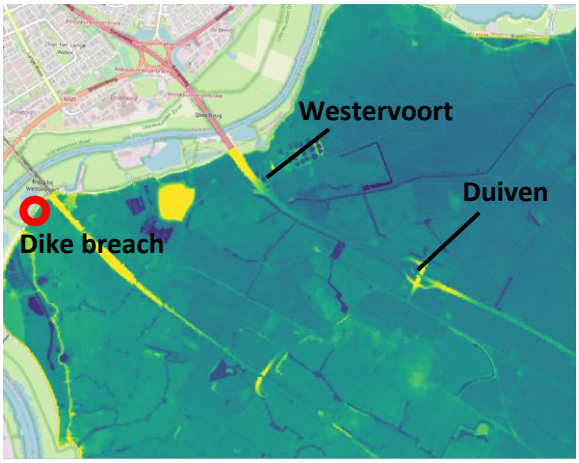


Figure 14 – Original DEM of section between A12 highway service interchanges at ground elevation

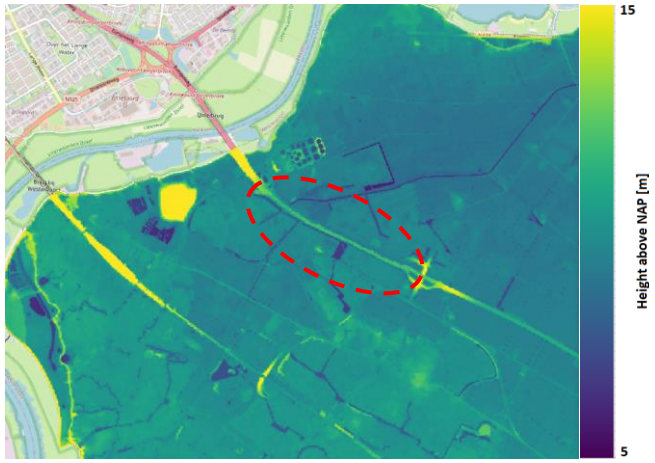


Figure 15 – Updated DEM with raised section of A12 highway between service interchanges

To utilize the updated DEM in the HAND.FLOW and HEC-RAS models, a few more steps need to be taken. For the HAND.FLOW model, the new DEM is used to create a new Local Drainage Direction (LDD) map. Due to the new section of raised highway, an additional compartment is created in the LDD map, which should change the route of the pathfinding algorithm. For HEC-RAS, the updated DEM is loaded into the mapping software and included in the geometry data. Then, a new simulation can be conducted using the updated DEM.

3.5 EVALUATING PERFORMANCE

As was mentioned in section 3.2.2, the Mean Absolute Error (MAE) is used as a loss function to train the neural network. Therefore, it is also used to evaluate the performance of the HAND.FLOW model. Note that the MAE will only be calculated for cells that are flooded for at least one time step in the simulation of either the LSTM/HAND.FLOW or HEC-RAS models. Otherwise, cells that remain dry the complete simulation period will have a perfect MAE of 0, and the average MAE will then be artificially lowered.

To determine which cells are flooded, a wet-dry threshold value is implemented. In the literature, quite a wide range of threshold values is used: from 0.01 meters by Chu et al. (2020) and 0.1 meters by Bermúdez et al. (2019), up to 0.3 meters used by Kabir et al. (2020). For this study, a threshold value of 0.1 meters was chosen, as in some initial testing cases the neural network generated water depths of up to 0.05 m while the flood had not yet started. Therefore, the threshold value was chosen above this value, at 0.1 meters.

An interesting performance indicator using this threshold is the Critical Succes Index (CSI), which concerns the accuracy of the prediction of inundation extent. It measures the agreement between two flood inundation maps considering the amount of cells that are wet and dry in each map. The CSI (sometimes called F1) is computed via Equation 3 (Lim & Brandt, 2019). It is defined as the area of true positives (hits) divided by the sum of the areas of hits, false positives (false alarms) and false negatives (misses). If the LSTM or HAND.FLOW model is 100% accurate with respect to HEC-RAS, the area of false alarms and misses is zero, so the CSI becomes 1. If the LSTM or HAND.FLOW model fail to predict a single flooded cell of HEC-RAS, then there are no hits at all and the CSI becomes 0. Any result in between, for example 0.7, can be interpreted as that 70% of all predictions are correct.

$$CSI = \frac{TP}{TP + FP + FN} \tag{3}$$

Where:

- *TP is the amount of true positives (hits): cells flooded in both models*
- *FP is the amount of false positives (false alarms): cells flooded in LSTM/HAND.FLOW but not in HEC-RAS*
- *FN is the amount of false negatives (misses): cells that are dry in LSTM/HAND.FLOW but flooded in HEC-RAS*

To compare how well the predicted water depth h_m perform against observed water depths h_o in HEC-RAS, the Nash-Sutcliffe Efficiency (NSE) is used (Equation 4). A perfect model achieves a score of 1. A bad model can reach values down to -∞. An NSE coefficient of 0, however, already indicates that the average observed water depth $\overline{h_o}$ is just as good of a predictor for the actual depths as the modelled values were (Y. B. Liu & Smedt, 2004).

$$NSE = 1 - \frac{\sum_{t=1}^{T}({h_m}^t - {h_o}^t)^2}{\sum_{t=1}^{T}({h_o}^t - \overline{h_o})^2} \tag{4}$$

Where:

- *h_m is the predicted depth at time t*
- *h_o is the observed depth at time t*
- *T is the total number of time steps*

4 RESULTS

4.1 NEURAL NETWORK TRAINING

The LSTM training process started with the selection of the most appropriate model architecture and best hyperparameter combination. The following two sections will discuss the findings of these steps. Afterwards, the performance of the best neural network on the testing data is assessed in more detail.

4.1.1 Determining LSTM architecture

Based on initial testing with the number of LSTM layers in the model architecture, it seemed that the amount of layers does not severely affect the overall performance of the model. Figure 16 and Figure 17 show the performance of networks with varying amounts of LSTM layers on the validation data set during the training. The performance does not seem to change significantly based on the number of layers, both for the small (Figure 16) and larger amounts (Figure 17) of units displayed. The water depth maps of the flooding event also showed no improvements to accuracy with more LSTM layers in the network. As adding more LSTM layers makes the training process more complicated and thus more computationally expensive, it was chosen to further develop the more simple architecture of a single-layered LSTM.

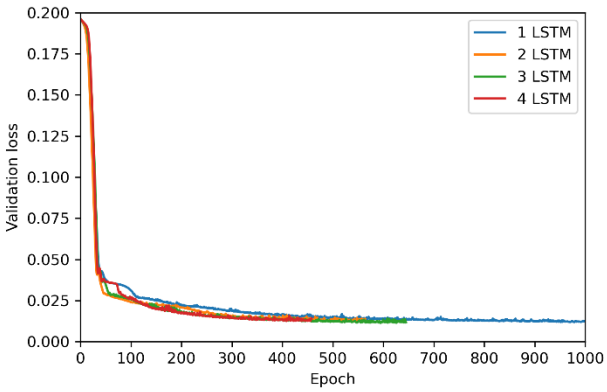


Figure 16 – Normalized loss on validation data for four models with 16 units and either 1, 2, 3 or 4 LSTM layers

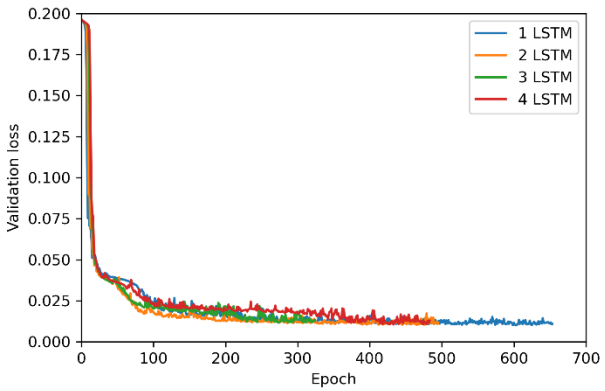


Figure 17 – Normalized loss on validation data for four models with 256 units and either 1, 2, 3 or 4 LSTM layers

4.1.2 Hyperparameter optimization

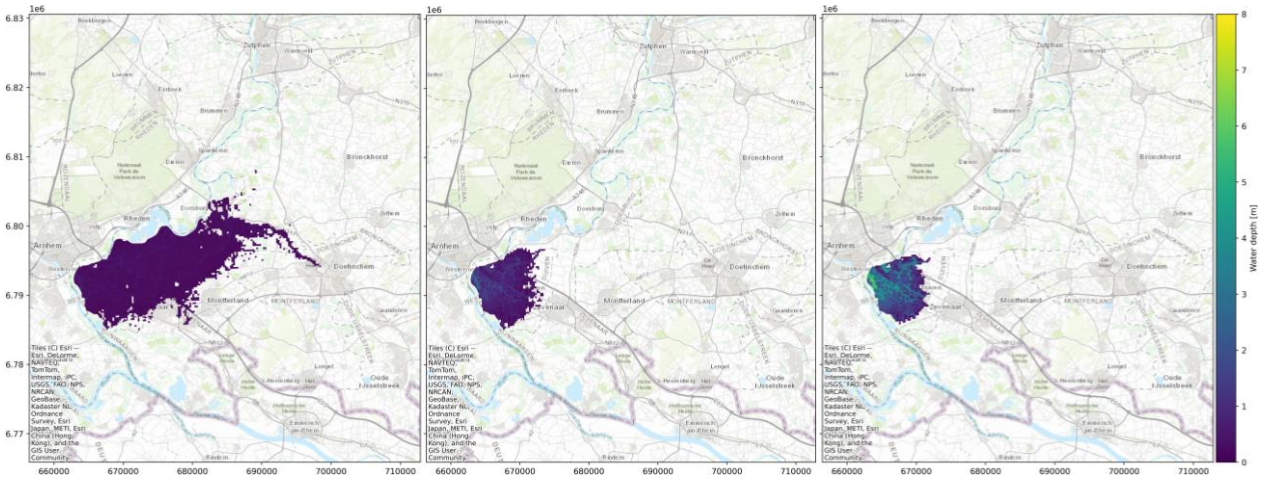
The Bayesian optimization algorithm described in section 3.2.3 was used to evaluate combinations of four hyperparameters and find the most accurate. In total, 150 hyperparameter combinations were tested, and the best combination is shown in Table 1. The time needed for training depended mainly on the number of units: 25 minutes for 800 units or more, or only 5 minutes for 100 units, for example. The total time spent training the 150 hyperparameter combinations time was in the order of 24 hours.

Table 1 – Hyperparameters after optimization

Hyperparameter	**Value**
LSTM units	912
LSTM dropout	0.25
Adam learning rate	0.0001
Dense layer activation function	rectifier

The mean absolute error (MAE) of the water depth predictions for all time steps of flooded cells on the unknown test data set was 0.045 meters, while the average water depth over all time steps of flooded cells is 1.49 meters in HEC-RAS. This is an average error of 3%. It should be mentioned that combinations of hyperparameters were found that resulted in lower mean absolute errors. The lowest MAE found for the test data set was 0.038 meters. However, the neural network that produced this value had a low predictive accuracy (Figure 18a). The figure shows that model A severely overpredicts the flooding in the first time step, although the water depths are small. Throughout most of the flooding event, this model floods an area just a few time steps earlier compared to the HEC-RAS simulation. The model from Table 1, on the other hand, predicts the flood event much more accurately throughout the event (Figure 18b). That is why it was decided to consider model B as the best model found by the hyperparameter optimization.

A possible reason that model A had a better MAE lies in the edges of the flooded area furthest away from the dike breach. In section 4.1.3, the performance of model B in these edge regions is shown to be quite poor. Model A performs better in this region, and has a slightly better MAE. Apparently, the bad predictive accuracy in the early time steps of model A is compensated by its better performance in the outer cells. A reason for this is that the large overpredicted area shown in Figure 18a consists of small water depth errors of 10-20 cm, and the prediction improves over the course of the flood. The errors made by model B in the edge region, however, are sometimes about a meter even at the end of the flood, which is also discussed in section 4.1.3. This is likely why model A has a better MAE overall, though its predictive accuracy in the beginning of the flood is worse.



(a) Model A with low predictive accuracy, but with best MAE of 0.038 m

(b) Model B with highest accuracy, but with slightly worse MAE of 0.045 m

(c) HEC-RAS simulation

Figure 18 – Water depth predictions of model with best MAE and model with slightly lower MAE, for first time step after dike breach in test data set

Lastly, the training of a neural network is a stochastic process, meaning that two training procedures with the same hyperparameter configuration lead to slightly different models and different model performance. Two causes of variation are the initialization of the network with random weights, which was described in section 2.1, and the fact that the network shuffles the training samples each epoch (Brownlee, 2020). These features ensure that the model can start each training cycle from a different point in the search space, and that the weight updates are not the same each time. However, this process also is likely to contribute to the relatively large differences between the models from Figure 18, though they had quite similar hyperparameter configurations.

4.1.3 Performance on test data

As was described earlier, 20% of the available data is used for testing the LSTM after it has been trained and optimized, which equals 15 simulations. Averaging these events, the most accurate LSTM model obtained through the hyperparameter optimization achieves high NSE-values near to 1 throughout most of the study area (Figure 19). Close to the dike breach, the prediction is slightly less accurate with NSE values around 0.95. The reason for this is an apparent delay of one time step by the LSTM flood prediction, compared to the flood of the HEC-RAS model; during the first time step the water depths are much smaller than they should be (see also Figure 18b compared to Figure 18c). The exact cause of this delay is unknown, though one time step delay in the prediction of sudden variable increases is also experienced by Kilsdonk et al. (2022) in their use of LSTM as well as by Bomers (2021) in the use of NARX neural networks. Perhaps, at the first time step of the breach, the hidden state neurons of the LSTM network still contain values associated with the near-zero water depths from before the breach. The hidden state neurons are updated when the flood starts, after which higher and more accurate water depths can be predicted starting from the second time step. Thus, if a more accurate prediction of the start of the flood is desired, a smaller time step than three hours could be used. This would result in the effect of the delay being smaller.

Figure 19 also shows that the edge of the area at risk of flooding is poorly predicted with NSE values of 0 and below. This is explained below at the discussion of the water depths for separate grid cells (Figure 22).

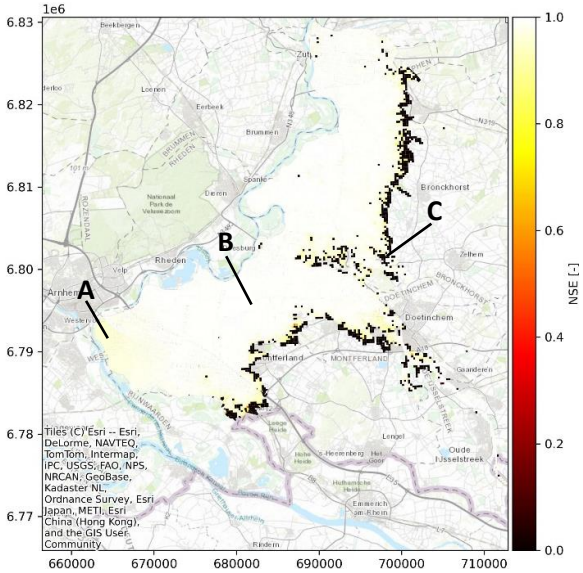


Figure 19 – Mean NSE for grid cells across all test data flood events (letters indicate locations of Figure 22a, b and c)

Visualizing the water depth accuracy for the complete study area, the water depths for all grid cells and all time steps predicted by the LSTM are plotted against the true values of the HEC-RAS simulation in Figure 20. The flood event chosen for this visualization is flood event 3, as it is representative of the average performance of the LSTM. The maximum water depths during this flood for all grid cells are also shown in Figure 21. The closer the predictions are to the red line of $y = x$, the better the overall performance of the network. As can be seen in Figure 20, the majority of points are located along this line, indicating relatively accurate predictions throughout the complete duration of the flood. There is an interesting trail of points in a less steep slope than the rest. These points are when the HEC-RAS water depths are high, but the LSTM predictor is lagging behind, which matches the behaviour for the first time step described above. Regarding the maximum water depths, Figure 21 shows that the prediction is highly accurate.

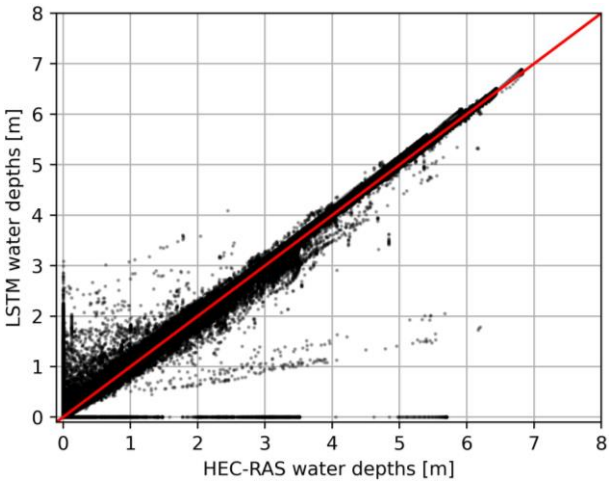


Figure 20 – Scatter plot of LSTM and HEC-RAS water depths for all grid cells and time steps in test flood event 3

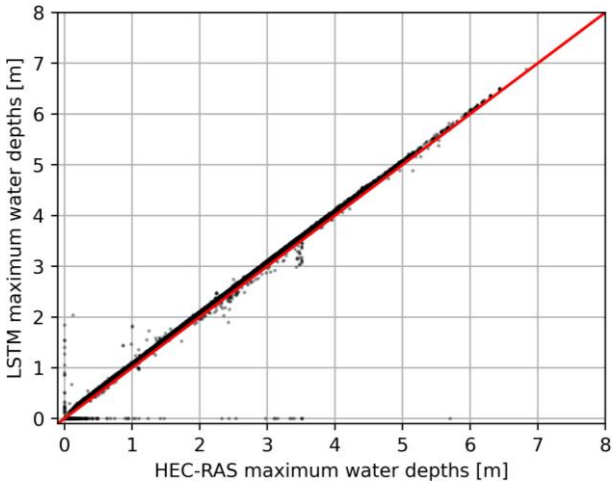
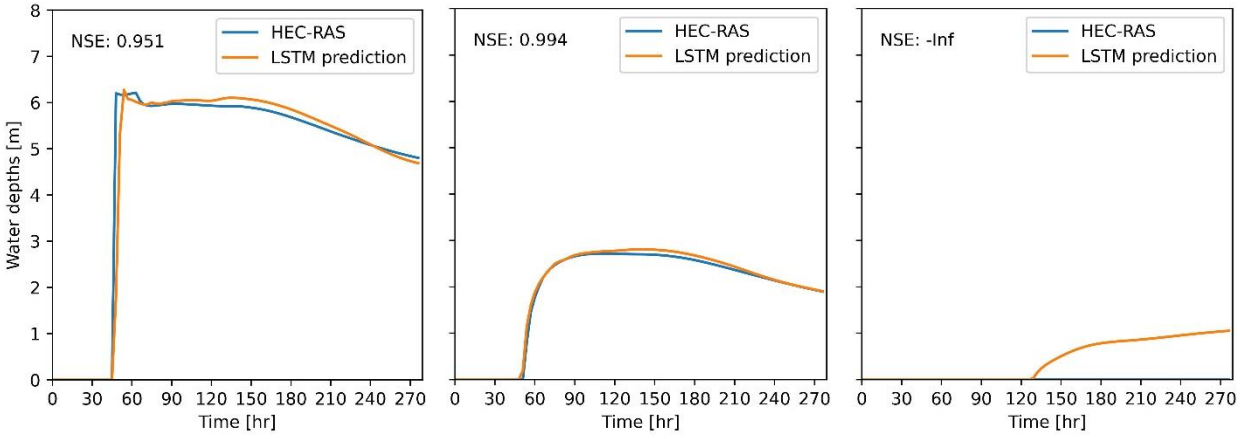


Figure 21 – Scatter plot of LSTM and HEC-RAS maximum water depths for all grid cells in test flood event 3

Looking at separate grid cells, Figure 22 displays the water depths for one of the test flood events for three locations in the study area. Both Figure 22a and Figure 22b show that the prediction is accurate with respect to the HEC-RAS simulation. For Figure 22a, located close to the dike breach, it can be seen that the LSTM lags slightly behind in predicting the rapidly increasing water depth. This behaviour was also visible in Figure 18b and c on page 28 in which the LSTM prediction and actual HEC-RAS flooding extents match quite closely, but the predicted water depths are lower than the actual water depths. In surrounding cells, this behaviour is also found, and it is the reason for the slightly lower NSE in this area close to the breach.

On the previous page, Figure 19 showed poor NSE values for cells in the edge region of the flood. In this area it often happens that either the LSTM predicts no flooding while there is a flood, or that the LSTM predicts a flood while there is no flood (Figure 22c). Since flood depths in these edge cells are generally smaller than in the rest of the study area, it seems that the LSTM has trouble determining the relationship between the outflow hydrograph discharge and the water depth for these cells. It was found that in some test flood events, the prediction in these areas can also be relatively accurate. However, the average NSE is calculated with the inclusion of near negative infinity NSE values of edge cells such as in Figure 22c, leading to the poor NSE values shown in Figure 19.



(a) Cell A near dike breach *(b) Cell B in centre of area* *(c) Cell C in edge region*

Figure 22 – LSTM and HEC-RAS water depths plotted for three grid cells in the study area for test flood event 3

For evacuation purposes, the arrival time of the flood is relevant as well. An optimal prediction is shown by the red line of $y = x$ (Figure 23), and it can be seen that the flood prediction of the LSTM is slightly earlier than that of HEC-RAS. This is interesting, as it was previously described that the LSTM is often one time step delayed. The reason for the behaviour shown in Figure 23 is that the LSTM is one time step delayed in the area close to the dike breach, which was explained to be likely caused by the updating of the hidden state neurons only after the first time step. For the remaining duration of the flood, the LSTM is seen to flood the hinterland slightly quicker than HEC-RAS, resulting in the arrival times being one or two time steps (3 or 6 hours) off in Figure 23.

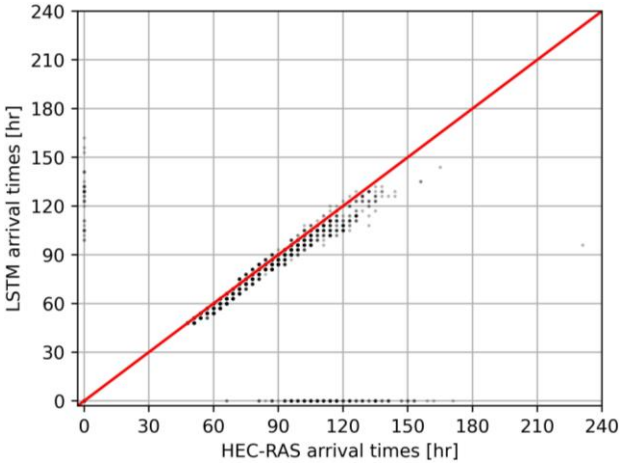


Figure 23 – Scatter plot of arrival times of LSTM compared to HEC-RAS for flood test event 3

Related to the accuracy of the arrival time is the Critical Success Index (CSI), which was described in section 3.5 and measures the degree of similarity in the area of the floods of the LSTM model compared to HEC-RAS, per simulation time step. Note that the accuracy of the water depth prediction is thus not relevant here. For test flood event 3, the CSI is very high throughout the simulation. Starting near 0.9 and maintaining close to 0.98 during the complete simulation means that the inundation extent is very well predicted by the LSTM (Figure 24). The average CSI of this simulation is 0.97, indicating great performance. Across all 15 test flood events, the average CSI of the simulations is 0.94, with only two test events having a score below 0.90: one of 0.83 and one of 0.88. Therefore, the LSTM model consistently predicts the area of the flood in HEC-RAS throughout the simulations with a high accuracy.

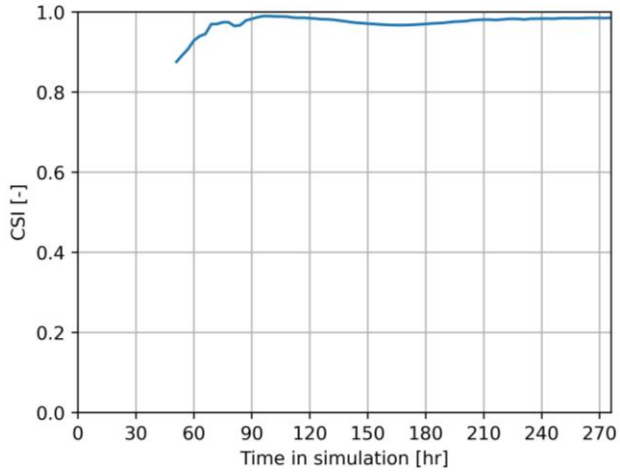


Figure 24 - CSI per time step for test flood event 3 comparing LSTM to HEC-RAS

4.2 HAND.FLOW MODEL

The following section discusses the performance of the HAND.FLOW model. The same 15 test flood events used for testing the LSTM neural network are used, to make a fair comparison.

4.2.1 Performance on test data

Unlike the LSTM neural network, the HAND.FLOW model does not match closely with HEC-RAS. It achieved a mean absolute error (MAE) of the water depth predictions for all time steps on flooded cells of 0.84 meters compared to HEC-RAS, while the average water depth over all time steps of flooded cells is 1.49 meters in HEC-RAS. The average error is thus 56%. For comparison, the LSTM model had a MAE of only 0.045 meters compared to HEC-RAS, which is an average error of only 3%. This section explains the results.

Evaluating the water depth predictions using the NSE value, it can be seen that the NSE is 0 or lower in most of the study area (Figure 25). The reason for this is that the HAND.FLOW model is not equipped with an emptying method, while HEC-RAS has a downstream boundary in the most northern part of the study area. As a result, HEC-RAS water depths start to decrease half-way through the simulation, even though water is still coming through the dike breach at a few hundred m^3/s. This is the main reason for the low NSE values in the complete study area, and also explains the large average error of 56% in the MAE. It also explains why there is a large flood around the hilly area of Montferland in Figure 25; the water levels in HAND.FLOW are much higher, leading to a much larger inundated area in some of the 15 simulations. Regarding the area of the flood and flood propagation accuracy measured with the Critical Success Index (CSI), which does not take into account water depth beyond the threshold of 0.1 meters, the HAND.FLOW model scores reasonably at an average of 0.70 for all 15 simulations. This indicates that while the water depth prediction is not accurate compared to HEC-RAS, the flood propagation patterns are.

As HEC-RAS reaches its maximum water levels half-way the simulation, evaluating the NSE at this time step renders better results (Figure 26). For example, the area around Montferland is not yet flooded half-way through any of the testing events, as the water levels are still relatively low then. The average MAE of this first half is also better: an error of 0.37 meters on the average water depth of 1.49 meters (25% error).

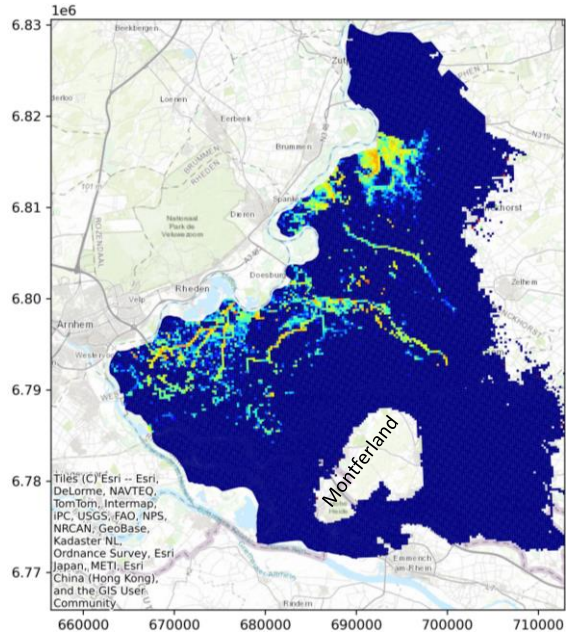


Figure 25 – Mean NSE for grid cells across all test data flood events (complete simulation period)

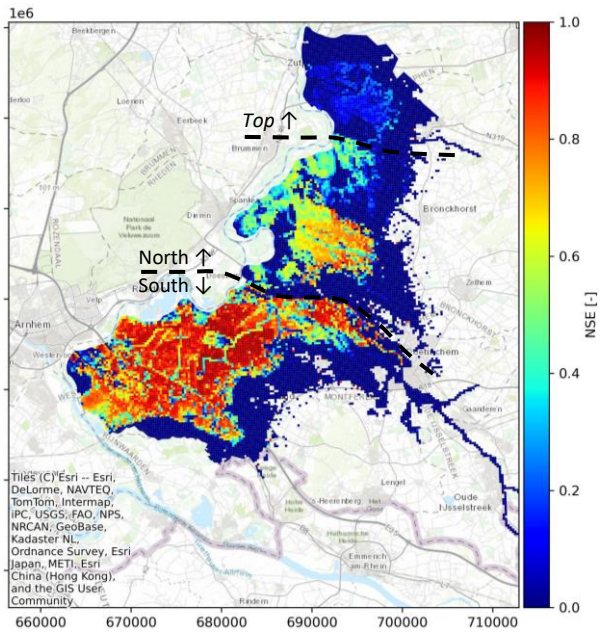


Figure 26 – Mean NSE for grid cells across all test data flood events (first half of simulation period)

For the evaluation of the first half of the simulation period, the best predictions with an NSE of about 0.8 to 0.9 are made in the south part of the flooded area closer to the dike breach, though it can be seen that this area is spotted with worse NSE values too (Figure 26). In the north the NSE values are lower, with the region closer to the middle line having NSE values of about 0.5, and the top region values of about 0.2. Additionally, there are still large areas with an NSE of 0 around the edges of the flood, where either HAND.FLOW or HEC-RAS is the only model predicting a flood.

From Figure 26 it seems that the HAND.FLOW model is losing accuracy in steps between the south and the north of the study area, and again in the most top region. However, it could actually be the HEC-RAS model that is inaccurate here. In the middle of the study area lies the Oude IJssel tributary of the main IJssel river (lower dotted line in Figure 26). In the northern half of the study area lies a drainage channel called Stroomkanaal van Hackfort, that also drains into the IJssel river (upper dotted line Figure 26). Both are flanked by dikes that are about 3 meters high. However, the width of these flow channels and their dikes are about the same as one grid cell in HEC-RAS, and the grid has not been adapted to take these dikes into account. As a result, the narrow dikes are lost in the height averaging performed by HEC-RAS and the water is free to flow across the dikes unhindered. The HAND.FLOW model, on the other hand, is able to see these dikes due to its resolution of 10x10 meters. It indeed stops flowing until it reaches 3 meters water depth, only then proceeding to flow across the dikes to the other side. To verify that this behaviour is accurate, a single HEC-RAS simulation is conducted with a grid adapted for the dikes. The simulation chosen is test flood event 2 (Figure 27), which is representative of the average performance of the HAND.FLOW model.

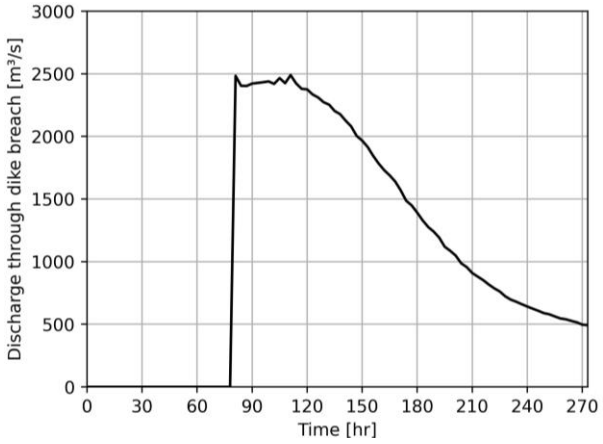


Figure 27 – Outflow hydrograph of test flood event 2

In test flood event 2 with the adapted HEC-RAS grid, the Oude IJssel dike halts the flood for two extra time steps (6 hours), and the Stroomkanaal van Hackfort halts it for one extra time step (3 hours). The adapted HEC-RAS flood thus arrives about 9 hours later in the northern parts of the hinterland compared to the initial simulation, which is more in line with the HAND.FLOW prediction. Visualizing the NSE values for the HAND.FLOW model for this specific flood event reveals a large increase in the NSE values in the northern part of the flood to NSE values of around 0.8 and 0.9, similar to the southern part of the flood (Figure 28). Still, there are large areas around the edge where only one of the models makes a prediction, leading to an NSE of 0 there. The MAE of this flood event is 0.21 cm (error of 14%). Interestingly, while in the original 15 flood events HEC-RAS never flooded near Montferland, it does happen in the simulation with the updated grid (Figure 28). Apparently, the blocking of the flood at the Oude IJssel results in higher water levels in the southern part of the flood, enabling water to flow in the direction of Montferland. All in all, the HAND.FLOW model is reasonable in predicting flood propagation patterns in this flood event.

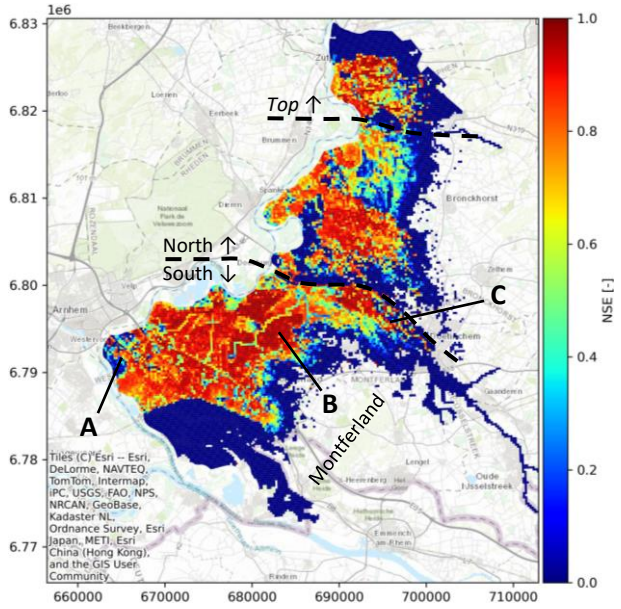


Figure 28 – NSE for grid cells in adapted test flood event 2 (first half of simulation period)

Similar to the NSE values, the Critical Success Index (CSI) shows that the prediction is reasonably accurate. For test flood event 2 with the adapted HEC-RAS grid, the average throughout the simulation is 0.72 (Figure 29). This indicates that the inundation extent is reasonably well predicted by the HAND.FLOW model. The drop in CSI visible at around $t = 90$ hours is caused by the slight difference in the moment of crossing the Oude IJssel dike: as still HEC-RAS floods across the dike before HAND.FLOW does, there is a larger mismatch between the areas of inundation of the models. The CSI then recovers as the HAND.FLOW model also crosses the Oude IJssel and starts flooding the area north of the dike. At around $t = 160$ hours, the HEC-RAS model experiences water volume losses through the downstream boundary that are greater than the volume of additional water coming through the dike breach. As explained before, the HAND.FLOW model does not have such an emptying mechanism, resulting in increasing inundated areas instead of decreasing. Therefore the CSI gradually drops after this point.

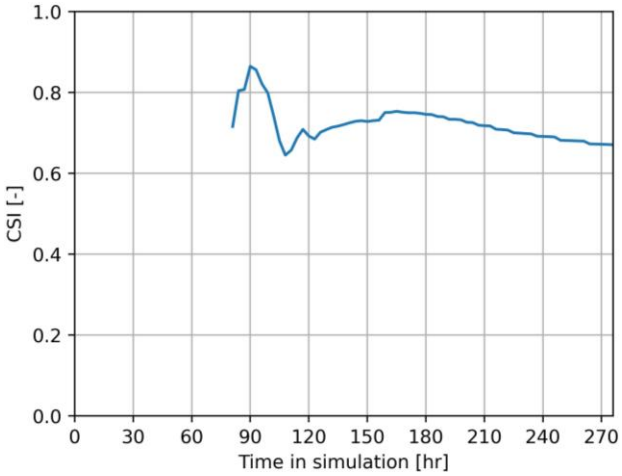


Figure 29 – CSI per time step for adapted test flood event 2 comparing HAND.FLOW to HEC-RAS

An important factor impacting the drop of the CSI at $t = 90$ is the distance limit that was developed for the HAND.FLOW model. It could be made more specific instead of the simple line that was used ($limit = 20000 + 5500t$), as will be described in chapter 7 on recommendations. However, since the HEC-RAS simulation used is not able to take into account small terrain features unless they are explicitly modelled, it remains the question which of the models is most accurate to reality. Perhaps there are more terrain features than the Oude IJssel and Stroomkanaal van Hackfort dikes that significantly influence the flow patterns. In any case, the NSE and CSI indicators show that the HAND.FLOW model is reasonable at recreating the HEC-RAS flood propagation patterns in this single corrected simulation.

However, even for this adapted simulation it is clear that the HAND.FLOW water depth predictions in a scatter plot are not matching with HEC-RAS (Figure 30). Two main reasons are responsible for this. First, it was described previously that the HAND.FLOW model does not have a downstream boundary to empty the study area, leading to ever-increasing water depths. This is visible in Figure 30 in the arches of points that curve towards the top left: these reflect grid cells where the HEC-RAS water depth decreases, but where the HAND.FLOW water depth is still increasing (see also Figure 32). The second reason is that the inundation extent of the HAND.FLOW flood along the edges of the flooded area is not very accurate. An example of this is the area of Montferland discussed on the previous page, as well as the edge regions of low NSE values. There, the HAND.FLOW model is often too late with predicting the arrival of the flood, which can be seen in Figure 31 as the large concentration of points above the red $y = x$ line.

The grid cells that exhibit this behaviour are largely in the edge region of the northern part of the flood. The reason for the late arrival times in this edge region is not directly linked to the distance limit relationship developed in section 3.3.2. Rather, it is caused by HAND.FLOW water levels rising slowly in the edge regions far away from the dike breach (see Figure 32 on the next page). Though in that example the HAND.FLOW flood arrives earlier than HEC-RAS, this is not the case in most of the cells. Mostly, the water levels rise quite slowly because of the assumption in the model that the water level is raised evenly across the study area. As a result, it takes a while before new cells are flooded in the edge regions far away from the breach. HEC-RAS, on the other hand, models water flow entering these areas quite rapidly, as seen in Figure 32 on the next page as well. This explains the large collections of late HAND.FLOW predictions above the red $y = x$ line in Figure 31.

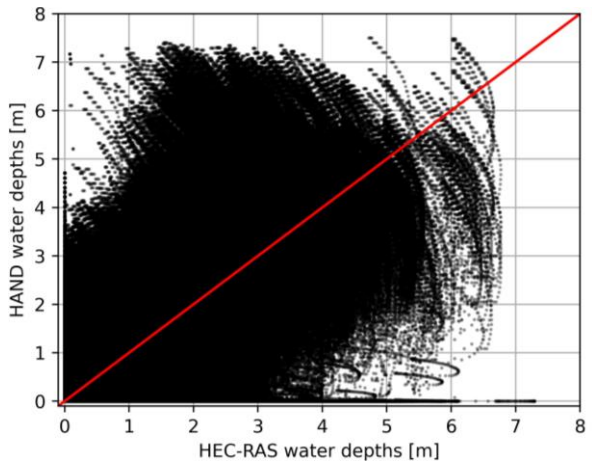


Figure 30 – Scatter plot of HAND.FLOW and HEC-RAS water depths for all grid cells and time steps in adapted test flood event 2

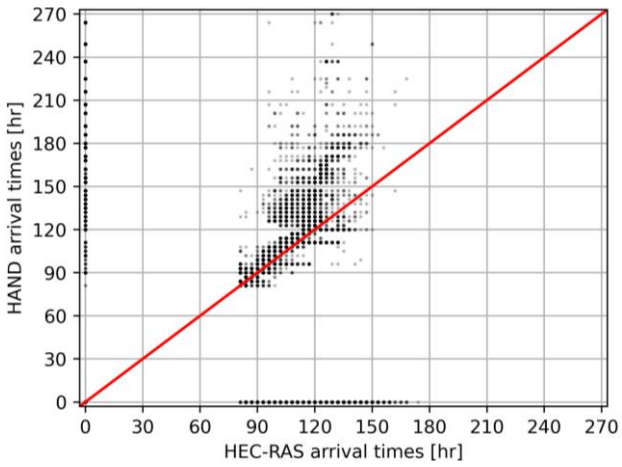
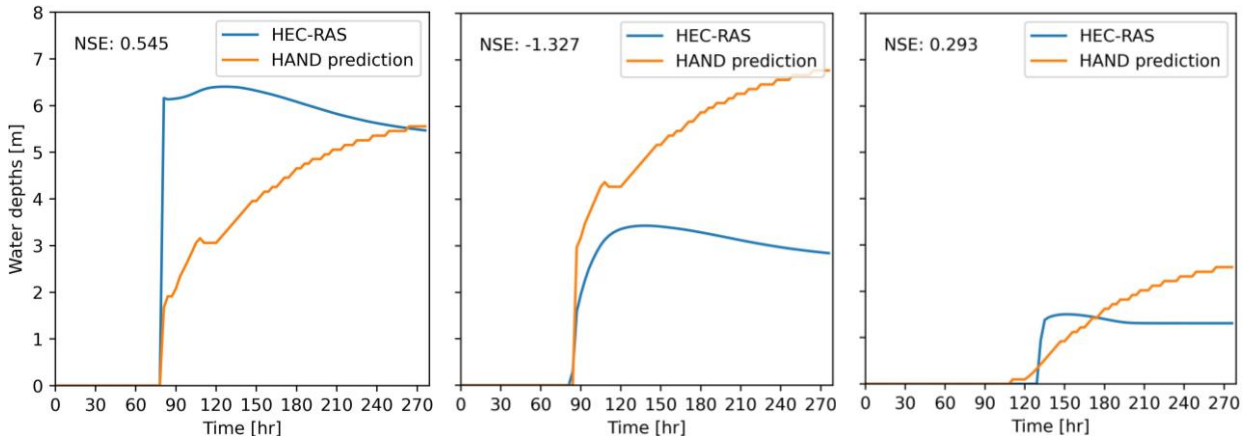


Figure 31 – Scatter plot of arrival times of HAND.FLOW compared to HEC-RAS for adapted flood test event 2

Zooming in from the scatter plots of all the grid cells to the same separate grid cells as for the LSTM (see Figure 28 for location), it can be seen that the water depths in the HAND.FLOW model do not match with the HEC-RAS model results, as expected (Figure 32). For the grid cell near the dike breach (Figure 32a), a limitation of the HAND.FLOW model is seen: it fills the entire accessible path of the flood limit with a single water level, based on the assumption of the original HAND model that water levels in rivers are more or less constant as they flow downstream. However, in a dike breach there is a strong gradient in the water depths; the closer to the breach, the more sudden and deep the water level increases. This steep increase is not accurately modelled in the HAND.FLOW prediction, which is why the prediction is only about 2 meters of water depth. For a grid cell in the centre of the study area (Figure 32b), it can clearly be seen that the water levels diverge as the HAND.FLOW model keeps filling up while HEC-RAS empties. Lastly, the HAND.FLOW model shows mediocre performance in the edge cell of Figure 32c, predicting that the water arrives about 20 hours earlier than in HEC-RAS. The difference in the steepness of the increase in water levels was explained on the previous page.



(a) Cell A near dike breach *(b) Cell B in centre of area* *(c) Cell C in edge region*

Figure 32 – HAND.FLOW and HEC-RAS water depths plotted for same three grid cells as LSTM for test flood event 2

4.3 HAND.FLOW AFTER DEM CHANGE

After changing the height of the A12 highway between the two service interchanges at Duiven and Westervoort in the Digital Elevation Model (DEM), the HAND.FLOW model and HEC-RAS model were both updated and used for a single simulation of flood test event 2. Unlike the retraining of an LSTM neural network, which would require all new training simulations, the HAND.FLOW model only requires a new DEM and Local Drainage Direction (LDD) map to be prepared. Updating these two maps took around 30 minutes. For comparison, preparing the training simulations that were used for the LSTM took around 800 hours in the study by Bomers (2021).

After the DEM had been changed, the flood propagation in the HEC-RAS simulation clearly changed, with water flowing further south-east until the water depth is great enough to flow over the A12 highway (dotted line in Figure 33). It can also be seen that the water depths have increased slightly in this area, as there is less space for the same volume of water. The HAND.FLOW model has also changed its behaviour with the new DEM (Figure 34), and comparing it to Figure 33 shows that it is quite accurate.

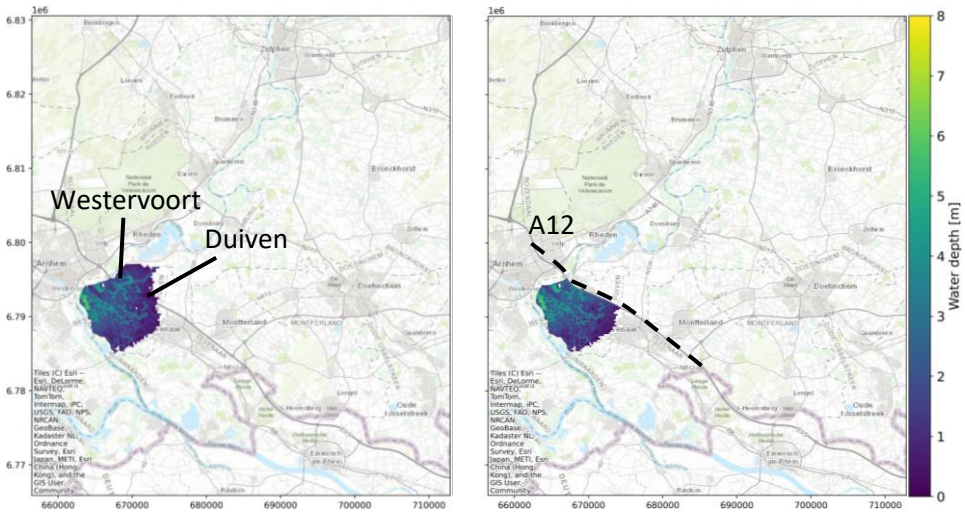


Figure 33 – Left: HEC-RAS before DEM change (first time step). Right: HEC-RAS after DEM change (first time step)

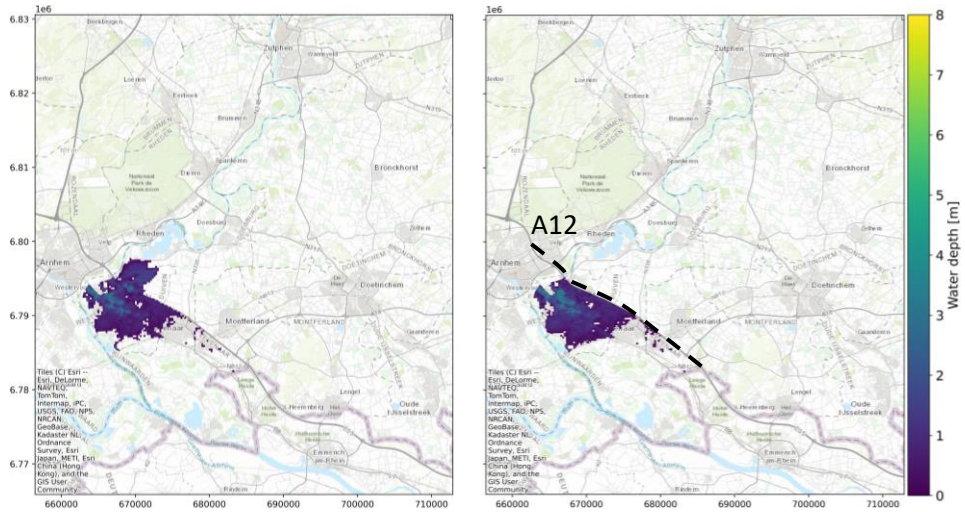


Figure 34 – Left: HAND.FLOW before DEM change (first time step). Right: HAND.FLOW after DEM change (first time step)

After the change in the DEM, both the HAND.FLOW and HEC-RAS models flow along the A12 highway instead of passing through the area between the two service interchanges. Additionally, the water depths of the HAND.FLOW model are also slightly higher in the situation after the DEM change, similar to HEC-RAS. However, in the second time step of the simulation, the HEC-RAS model has water depths large enough to flow over the highway, while the HAND.FLOW model flows further south for one more time step. It is thus one time step too late with flowing over the A12 highway compared to HEC-RAS. Due to the way the model is programmed, this extra time step delay in the beginning of the flood has consequences for the further flood propagation and the accuracy of the prediction of arrival times.

It takes the HAND.FLOW model one time step (3 hours) to find its way out of a pit, while HEC-RAS computes on a temporal resolution of 30 seconds. The effect of changing the temporal resolution of the HAND.FLOW model is covered in section 5.3.1 of the discussion. In this simulation, however, the effect of having an extra pit in the flow path close to the A12 slows down the flood propagation considerably compared to HEC-RAS. The reason is that the flow path no longer crosses between the two service interchanges at Duiven and Westervoort, but it flows over the highway at another point along the A12 further south-east. As a result, the flow path is longer than in the original simulation. Note that the distance limit function ($limit = 20000 + 5500t$) was not changed for the simulation with the new DEM, so the HAND.FLOW model needs more time to travel along the longer flow path. An example of the result of this is that model crosses the Oude IJssel river dike 5 time steps (15 hours) later than HEC-RAS. Therefore, the NSE values of the flood are slightly lower than before the DEM change (Figure 35 compared to Figure 28). Especially in the northern part of the flood, the NSE dropped from 0.8 – 0.9 before the DEM change to about 0.6 after.

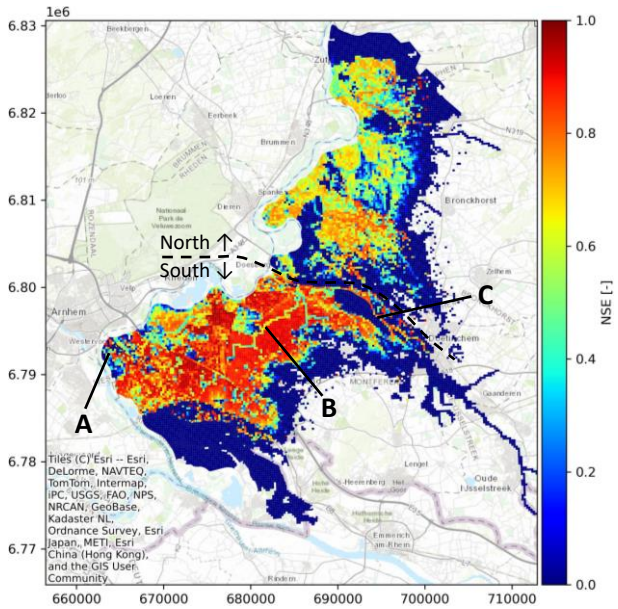


Figure 35 – NSE for grid cells in test flood event 2 with change in DEM (first half of simulation period)

Evaluating the scatter plot of arrival times in HAND.FLOW and HEC-RAS (Figure 36), it can be seen that the HAND.FLOW model predicts the arrival times often above the red $y = x$ line, indicating a delayed prediction compared to HEC-RAS. This was discussed above to be caused by the longer flow path, leading to a 5 time step delay for crossing the Oude IJssel river. A larger region of data points with even slower HAND.FLOW predictions is also seen. These points are mainly located in the edge regions, where HAND.FLOW is very late with predicting a flood compared to HEC-RAS. Evaluating the Critical Success Index

(CSI) of this event, two interesting drops can be seen. The CSI starts out at around 0.8 for the first time step, indicating large similarity between the flooded area of HAND.FLOW and HEC-RAS (mentioned also for Figure 33 and Figure 34). Then, the HAND.FLOW model crosses the A12 highway one time step later than HEC-RAS, leading to a mismatch in the inundated area and a lower CSI score (around $t = 90$). As HAND.FLOW crosses the A12, the inundated areas become more similar again and the CSI increases, before dropping around $t = 100$ due to the 5 time step delay in crossing the Oude IJssel dike. After this crossing, the score increases again as the HAND.FLOW model catches up. The average CSI of the complete simulation period is still 0.70, which is a reasonable score.

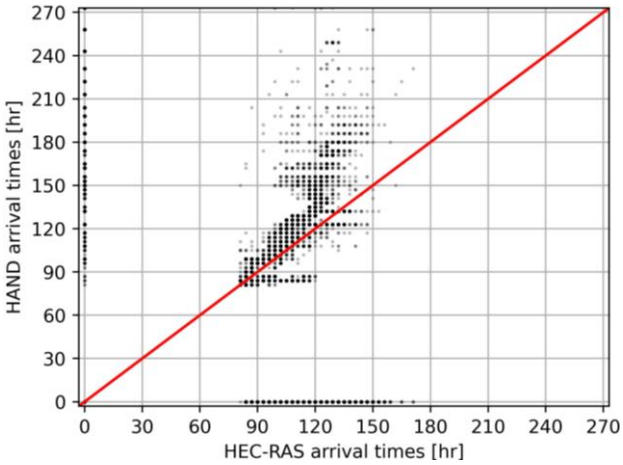


Figure 36 – Scatter plot of arrival times of HAND.FLOW compared to HEC-RAS for test flood event 2 with DEM change

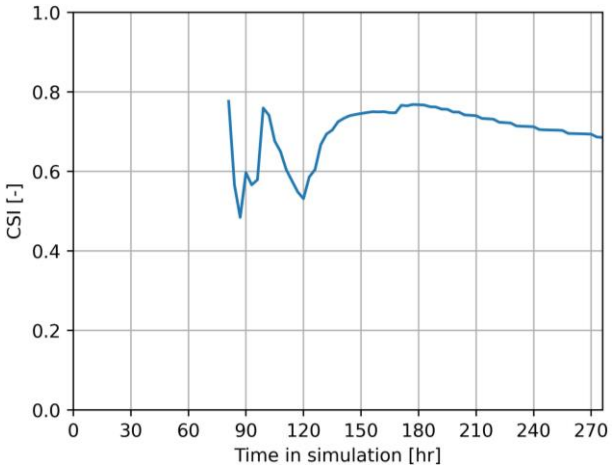
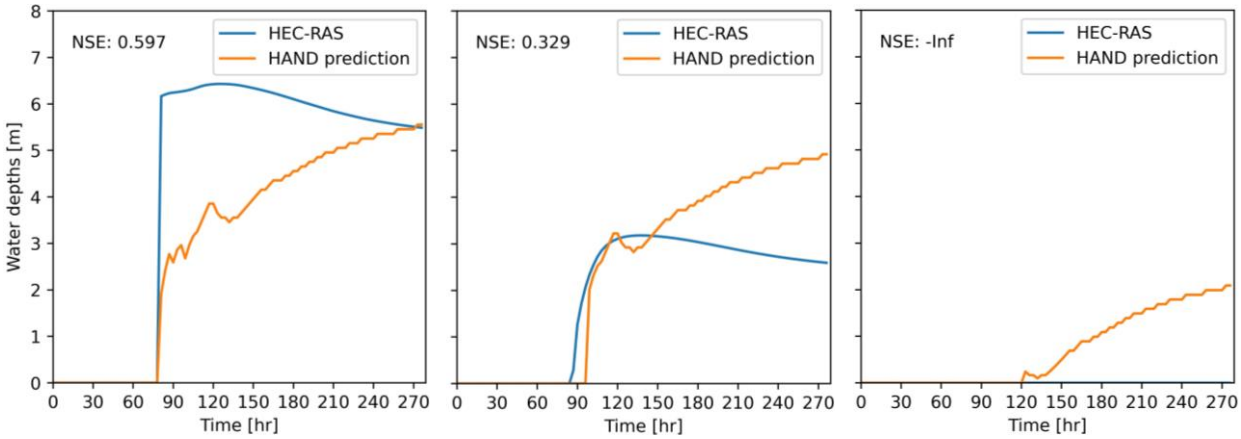


Figure 37 – CSI per time step for test flood event 2 with DEM change comparing HAND.FLOW to HEC-RAS

Looking at the water depths of separate grid cells in this event (see Figure 35 for locations), it is interesting to see that the performance in NSE is actually slightly better compared to the situation before the DEM change (Figure 38). However, as was mentioned before, the general performance throughout the study area is slightly worse. For two of the grid cells, it can be seen that the HAND.FLOW model does not predict the arrival time well compared to HEC-RAS. In the cell in the centre of the study area (Figure 38a), the HAND.FLOW model is too late with the prediction, which was explained to be the result of the longer flow path in the previous paragraph. In the cell in the edge region, the HEC-RAS model actually no longer predicts a flood, possibly due to additional water being held behind the raised A12 highway.



(a) Cell A near dike breach *(b) Cell B in centre of area* *(c) Cell C in edge region*

Figure 38 – HAND.FLOW and HEC-RAS water depths plotted for test flood event 2 with DEM change

5 DISCUSSION

This chapter first discusses several points that are valid for the two surrogate models in general, and afterwards the results and limitations of the LSTM and HAND.FLOW models are discussed separately.

5.1 GENERAL REMARKS

In this research, two surrogate models were developed for recreating the results of the 1D2D-hydrodynamic model HEC-RAS developed by Bomers (2021). This model was originally calibrated by Bomers, Schielen & Hulscher (2019) with the river water level data of the 1995 Rhine near-flood, and validated with the 1993 Rhine near-flood. These events did not lead to dike breaches, but it is assumed that the model is capable of simulating large overland flow. This is based on literature such as Moya Quirogaa et al. (2016), who used HEC-RAS to accurately reproduce the February 2014 Bolivian Amazonia flood. As such, it was assumed that the HEC-RAS model is representative of reality and suitable to be used as a base for the development of the LSTM and HAND.FLOW models. It does, however, mean that any errors present in the HEC-RAS model will be present in the LSTM and HAND.FLOW models too.

The most notable consequence of this was that the HAND.FLOW model noticed terrain features that were not taken into account in the HEC-RAS model. The 3-meter high dikes along the Oude IJssel and Stroomkanaal van Hackfort discussed in section 4.2.1 were invisible due to the relatively large HEC-RAS grid cell size of 150x150 meters. This grid cell size was required in the study by Bomers (2021) to make the computation times acceptable, as the study area was quite large. In this research, a single new simulation was conducted with a HEC-RAS model that did take these dikes into account, but it would have been more insightful if a larger set of scenarios could have been evaluated with this updated HEC-RAS model.

A second point is that the HEC-RAS output data available was generated once every 3 hours of the simulation. For the purposes of Bomers (2021), this output time interval was detailed enough. Generating the output every 3 hours of the simulation still resulted in a large amount of data, so to manage and store data from simulations with a shorter time interval, a flood event with less complexities could be considered for future research. This would especially be relevant for the area close to the dike breach. There, the velocities are high and the water depths rise fast, which now all happens in only one time step.

5.2 LSTM NEURAL NETWORK

The LSTM model accurately reproduced the results of the HEC-RAS simulations, as it is data-driven and can learn the patterns and dependencies present in the data. In the literature, this has been proven many times for time series prediction (such as Le et al. (2019), who demonstrated reliable flood discharge forecasting capabilities of an LSTM). Solvik et al. (2021) use an LSTM to calculate the probability of flooding for nearly 72,000 small lakes in the Great Plains of the US based on climate data and area-specific characteristics. Kilsdonk et al. (2022) successfully developed an LSTM to predict flood volume during a precipitation event for 230 manholes of a sewer system. As Bentivoglio et al. (2021) mention, deep learning had not yet been applied to dam and dike breach flood events, so this present study has proven that LSTM are accurate in reproducing water depths on grid cells in the hinterland after a dike breach as well. A number of interesting observations were made in this research that are worthy of discussion.

First, the LSTM network underpredicts the sharp increase in water depths in the first time step of the simulation. Such behaviour was also found by Bomers (2021) and Kilsdonk et al. (2022), and was hypothesized in section 4.1.3 to originate from the hidden state cells of the neural network, which are updated to reflect the flood's beginning during the first time step. The effect of their change is thus only visible starting from the second time step. Although this might not matter for the overall picture of the flood prediction, the impact can be decreased by modelling with a smaller time step than the 3 hours used in this research. Then, the neural network would adapt its hidden state cells while a smaller portion of the hinterland is flooded, leading to fewer cells with an incorrect prediction in the first time step. An additional insight gained by this would be to see how the LSTM behaves with the very high outflow hydrograph discharge inputs that correspond with this very first moment after the dike breach. With such a shorter time step, these will be more extreme than the average discharge of the first 3 hours used by this research.

Second, the best LSTM from the hyperparameter optimization was not the best in accurately predicting inundation extent. As discussed in section 4.1.2, the best model (model A) had a Mean Absolute Error (MAE) of 0.038 meters among all flooded cells and all time steps, but a poor predictive accuracy regarding the size of the flood. Model B had a slightly worse MAE of 0.045 meters, but predicted the size of the flood more accurately. The difference was attributed to the fact that model A did predict worse in the first time step, but better around the edges of the flood where grid cells flood in only some scenarios. Model B predicts better in the beginning, but worse in these edges. Apparently this difference leads to a slightly better MAE for model A, even though model B was considered more accurate and relevant for the flood propagation prediction. This is why model B was considered the best LSTM.

Other loss functions such as Mean Square Error and Root Mean Square Error were also tested, but did not seem to lead to better models, and often resulted in even worse models. While training of neural networks is a stochastic process, the loss function does affect the severity of changes to the weights that are propagated through the network, so a different loss function leads to different inner relations between neurons in the network. Perhaps another loss function is more appropriate than the MAE, such as a custom loss function that takes into account the distance of the grid cell to the dike breach, to be able to force the network to find a better balance between far away cells and cells close to the dike breach.

All in all, LSTM are great at learning patterns in data and forecasting after the training procedure. However, a lot of data is required for them to be trained to sufficient accuracy. Lu et al. (2021) refer to LSTM as having a "data-hungry nature", and use it to predict discharge in a data-scarce basin. They find reasonable prediction accuracy if the evaluated period is similar to the periods present in the training data, but poor performance if this is not the case. This shows that the LSTM requires a representative range of events in the training data, in order to not suffer from overfitting and out-of-distribution (dramatically different) predictions (Lu et al., 2021). Also in this research on dike breach inundation, a lot of training data and training time was needed to come to the accurate LSTM. Additionally, as mentioned in section 4.1.2, the training of a neural network is a stochastic process, which means that the performance can be different for a similar set of hyperparameters. Therefore, it can take quite some training iterations to find an accurate neural network, even in data-rich areas.

The implications of this for the use of an LSTM in a real-time flood forecasting system are significant. First, the LSTM that was trained in this research is only valid for this dike breach location. It cannot be applied for a dike breach that is a kilometre downstream, for example. As such, this LSTM set-up requires a lot of 1D-2D hydrodynamic simulations to be conducted for every potential breach location in a real-time flood

forecasting system, (800 hours for Bomers (2021) for one breach location).. It should then be trained for every potential breach location, which requires a lot of iterations and critical reflection of the modeller (see for example Figure 15 in section 4.1.2, in which the LSTM with the best MAE value was not the most accurate at flood inundation extent prediction). The training time for this study was also in the order of 24 hours (average 10-15 minutes per hyperparameter combination), making the total time investment required for an operational system using LSTM this way immense.

Although this version of the LSTM is impractical for use in a real-time flood forecasting system, the field of neural networks has been rapidly developing in recent years. With the interest for machine learning, artificial intelligence and algorithms in the tech sector, new neural networks are being developed and applied in more and more research fields (Chu et al., 2020). This study is the first to demonstrate that the often-used LSTM neural network is capable of learning the spatial and temporal characteristics of flooding due to a dike breach. In recent years, however, an entirely new type of neural network has been gaining momentum for time-series prediction, called the Convolutional Neural Network (CNN) (Bentivoglio et al., 2021). These are suitable for spatial analysis due to their ability to easily process raster files and images, a quality which the LSTM does not have of its own. Perhaps a better LSTM, another neural network such as CNN, or a combination of LSTM and another algorithm can be developed that is generalized for more dike breach locations in a study area, which is discussed in the recommendations in chapter 7.

5.3 HAND.FLOW MODEL

The HAND.FLOW model was less accurate in reproducing the HEC-RAS water depths than the LSTM. In section 4.2.1 on the results, a number of reasons were discussed. For example, unlike HEC-RAS, the HAND.FLOW model was not equipped with a downstream boundary, which resulted in water levels that kept increasing indefinitely. The HAND.FLOW model was still able to reasonably predict inundation extent throughout the flood, with CSI values of around 0.70. This is at the lower end of CSI values found by studies using the original HAND model, such as Chaudhuri et al. (2021), Li et al. (2022) and McGrath et al. (2018), who find CSI values ranging from 0.7 up to 0.9 depending on the case study.

The original HAND model works with setting a river water level on the grid cells that it determines to be the drainage cells of the area, as was described in section 2.3 and 3.3.2. This water level is then extrapolated to surrounding terrain, where all cells with a Height Above Nearest Drainage (HAND) value less than the water level are flooded. The model thus works on the assumption that the water level in a river is constant along the river reach. This is valid for the riverine floods spilling into floodplains usually modelled with the original HAND model. The HAND.FLOW model also uses this assumption when it floods the study area along the flow path. However, in dike breaches there is a relatively steep gradient in water depths: close to the breach the water depths are very high and increase very fast, while further away they increase slower and to a smaller depth. The importance of this effect not being included in the model would have to be investigated, since the area affected might be relatively small. If it is significant though, a future version of the model could apply a weight factor on grid cells based on their distance to the breach, to reflect this gradient. It would have to be seen if this weight factor can be generalized for dike breach events in other case studies too, or if it depends too strongly on location, discharge shape and duration.

A second major assumption made in the development of the HAND.FLOW model is the distance limit that is imposed on the pathfinding algorithm each time step ($limit = 20000 + 5500t$). The relationship was derived in section 3.3.2 using data from the 15 HEC-RAS test flood events (Figure 10). As a result, the

relationship is only valid for this study area. A new relationship has to be derived for a different dike breach location, which requires at least one HEC-RAS simulation to be conducted. The effect of the distance limit is also significant due to the way the model was programmed. For example, after the Digital Elevation Model (DEM) change at the A12 motorway (section 4.3), the flow path became significantly longer. The distance limit function was not altered for this situation, so the model took a longer time flowing over the path, leading to a 5 time step delay (15 hours) between HEC-RAS and HAND.FLOW crossing the Oude IJssel dike. Thus, the distance limit affects the propagation of the flood in the HAND.FLOW model, and it should be developed and used with care. Additionally, it means that the HAND.FLOW model in its current state is not independent; it needs input from another source to derive the distance limit relationship. For ways to solve this dependency, see chapter 7 on recommendations.

Third, the HAND.FLOW model has no notion of flow velocity of the water. During a dike breach however, and especially close to the breach, flow velocities are high and allow water to force its way across a higher terrain feature, such as a slightly elevated road. As the HAND.FLOW model cannot model this, situations can occur in which a sizeable volume of water will flow across a barrier in reality, but this area will not flood in the model. In the case study considered in this research, this situation does not occur. To solve it when it does occur, the introduction of a velocity component might be a solution. However, this could also further complicate the HAND.FLOW model beyond its intended goal as a fast surrogate model. As such, the applicability of the HAND.FLOW model in any location is not guaranteed, as some obstacles nearby the dike breach may divert the water in the HAND.FLOW model in an unrealistic manner.

Perhaps the solution to this problem is offered by another field of fast modelling of flood inundation: cellular automata (CA) models. These models consist of grid cells that have a set of transition rules determining the evolution of each cell, taking into account its neighbours (Teng et al., 2017). Guidolin et al (2016) developed a CA that was up to 8 times faster than the benchmark hydrodynamic Infoworks ICM model, and to reasonably good agreement. The model predicted water depth and used both rainfall and a point source of a sewer overflow as forcing. Jamali et al. (2019) compared a similar CA model to HEC-RAS, and it performed very well in areas with low-lying depressions filling up, but somewhat less in areas where floodwaters had higher velocities. Possibly, combining the velocity components of a cellular automata model with the principles of the HAND.FLOW model can help overcome the HAND.FLOW models aforementioned inability to flood across an obstacle with high velocity.

All in all, the HAND.FLOW model performed reasonably well in predicting the inundation extent, even though the water depth prediction was not equal to that of HEC-RAS. A number of use cases for the HAND.FLOW model are imaginable. First, the HAND.FLOW model can be used to quickly explore the severity of a flood after a dike breach in a particular location. Additionally, the model is relatively flexible in being used for a breach location that is, for example, a kilometre downstream. Second, the model showed a good understanding of changing flow patterns after a change in the Digital Elevation Model (DEM), meaning that it can be used for a rough estimate of changing flood inundation patterns after interventions such as compartmentalization in the hinterland. This cannot be said of the LSTM, which was mentioned to be inflexible in case of any such changes. Third, the HAND.FLOW model should be relatively quickly usable in other riverine areas, since the basic principles apply everywhere. Only the distance limit relationship requires an update to see how the flood propagates through time. Furthermore, since the model only requires the DEM as input, it should be applicable in data-scarce areas. The next section will discuss the performance in a data-scarce region with a coarser DEM. Considering all these use cases, there is certainly potential in using a the HAND.FLOW model in a real-time flood forecasting system.

Another important aspect of a model for a real-time flood forecasting system is that the set-up time and simulation time should be sufficiently short that multiple scenarios can be evaluated (Teng et al., 2017). As was discussed before, the LSTM neural network has a long preparation time for gathering all the training data, which has to be modelled for each situation and took 800 hours for Bomers (2021). The HAND.FLOW model, on the contrary, requires a much shorter preparation procedure for converting the Digital Elevation Model (DEM) into the Local Drainage Direction (LDD) of about 30 minutes. Regarding simulation time, the LSTM was near instant in predicting a flood event, while the HAND.FLOW model required about 30 minutes on a 10x10 meter resolution for a simulation period of 12 days with 3-hour time step. This is still quite long for the purposes of multiple scenario analysis. It is much faster than the HEC-RAS model, which takes about 6 to 10 hours for a simulation with a time step of 30 seconds on a 150x150 meter resolution. It should be noted that most of the flood propagation for this large flood event happened during the first 4 days of the flood, which the HAND.FLOW model simulated in 10 minutes. The effect of changing the resolution of the HAND.FLOW model on the simulation time and results is discussed in the next section.

5.3.1 HAND.FLOW resolution changes

The HAND.FLOW model that was used in this research utilized a spatial resolution of 10x10 meters and a temporal resolution of 3-hour time steps. This section discusses the changes that arise in the HAND.FLOW model and its results when the model is updated to work with other resolutions.

Spatial resolution

To investigate performance in a "data-scarce" region, the resolution is changed from 10x10 meters to the 150x150 meter resolution of the HEC-RAS model. The first step is to create the Digital Elevation Model (DEM) and Local Drainage Direction (LDD) maps. From the 10x10 meter DEM that is used as the base data set, a resampling algorithm is used in QGIS to calculate the average values of all the 10x10 meter data points in each 150x150 meter grid cell. With this new 150x150 meter DEM, the LDD can be derived in QGIS as well. The second step is to set the row and column coordinates of the dike breach. Third, the distance limit relationship requires an update, since in the updated DEM some of the terrain features are averaged away. The flow path from the dike breach downstream might thus take a slightly different route, leading to a longer distance covered especially in the first time step (Figure 39). In the figure, the raised railroad ascends towards the Westervoort bridge crossing the IJssel river, and descends into the hinterland. On the 150x150 meter resolution, the railroad is averaged away into the surrounding elevation sooner than on the 10x10 meter resolution, so the flow path crosses it earlier.

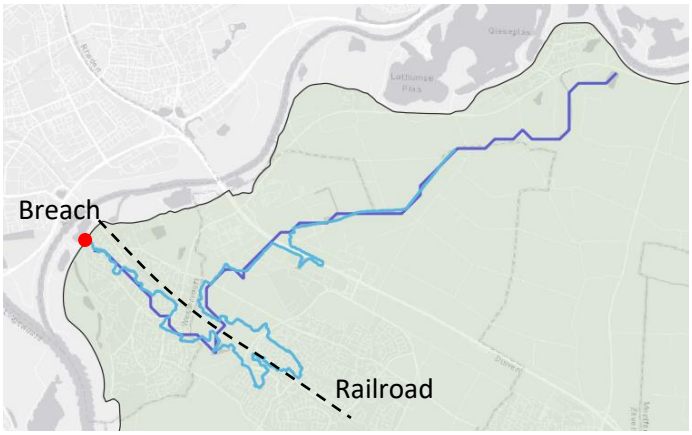


Figure 39 – Flow paths on 10x10m resolution (light blue) and 150x150m resolution (dark blue) in first time step

A benefit of a more coarse spatial resolution is that it leads to a much faster simulation time. The simulation of 12 days (93 time steps) on 10x10 meter resolution took 30 minutes, which is decreased to only 1.5 minutes on the 150x150 meter resolution. And as mentioned before, the flood propagation reaches its largest extent around the 4th day, which used to take 10 minutes on the 10x10 meter resolution, but takes only 3 seconds on the larger resolution. With such a short simulation time, scenario analysis for the purposes of real-time flood forecasting is more attainable.

However, the flood maps of the 150x150 meter flood show more of a fragmented inundation extent than what is expected (Figure 40), making the prediction less accurate. It is not entirely known why the flood in this first time step is disconnected in the HAND.FLOW model. The dry area between the two parts fills up in a later time step, but the remaining flood propagation in later time steps is also less accurate and more fragmented than on the more detailed spatial resolution. The problem of this fragmentation of the flood disappears at 75x75 meters resolution. In their research, Li et al. (2022) conclude that the DEM resolution is a substantial source of uncertainty in the original HAND model due to the resampling of features in a coarser DEM, possibly leading to overgeneralization. It thus seems like the HAND.FLOW model requires a more a detailed spatial resolution to predict flood propagation than HEC-RAS, and that the original HAND model also suffers from problems related to coarse resolutions.

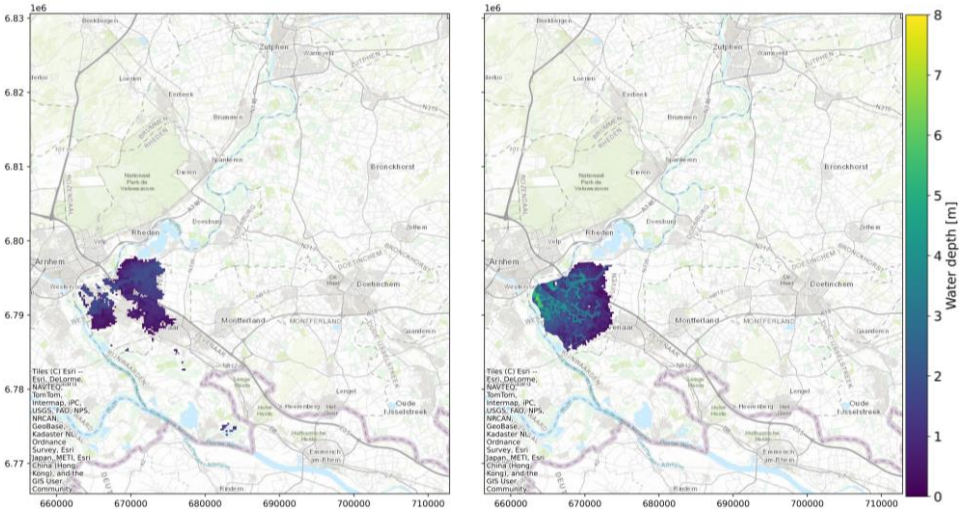


Figure 40 – Left: HAND.FLOW model on 150x150 meter resolution (first time step). Right: HEC-RAS model (first time step)

Temporal resolution

The temporal resolution of the HAND.FLOW model in this research was 3 hours per time step. For mapping purposes of flood propagation, this resolution was sufficient to show the flood gradually spilling into the hinterland. However, if more detail is desired, the temporal resolution can be made more detailed. In this example, a time step of 15 minutes will be used. Two steps are needed to change the temporal resolution of the HAND.FLOW model. The first step is to make sure that the input data of the outflow hydrograph is also in a 15 minute time step. The second step is to change the distance limit relationship; since in a 15 minute time step the distance travelled by the water is much less than in a 3 hour time step.

The first three days of one flood event were simulated using this higher temporal resolution in both the HAND.FLOW and HEC-RAS models. Figure 41 shows the flood results of the first time step of the simulation, and it can be seen that both the HEC-RAS and the HAND.FLOW model remain close to the dike breach. HEC-RAS can be seen to have slightly higher water depths, while the HAND.FLOW water depths are slightly more evenly spread. This has been explained before as the result of the assumption that the water level is constant along the drainage cells of the flow path.

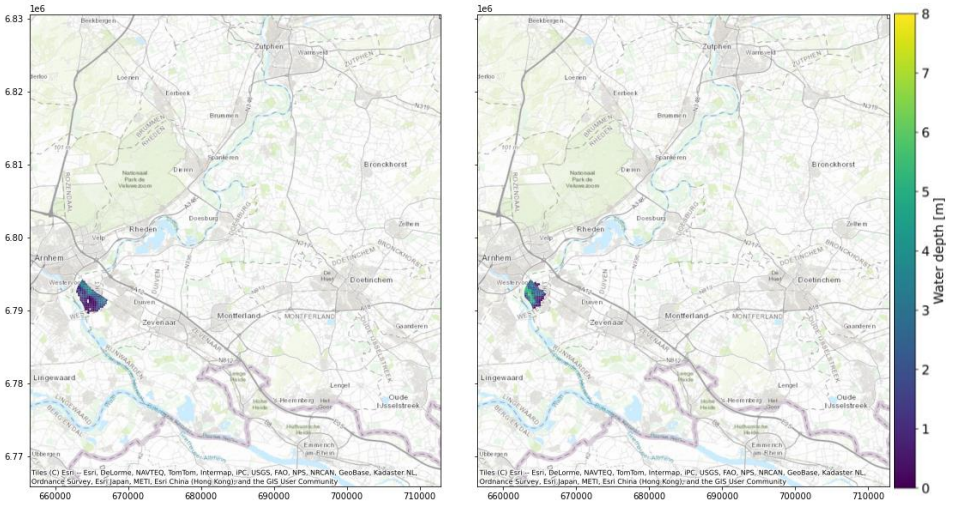


Figure 41 – Left: HAND.FLOW model on 15 minute resolution (first time step). Right: HEC-RAS model (first time step)

For the remaining part of the simulation, the HAND.FLOW model follows the same flow path as it did in the previous analysis of this study. This is because the DEM and LDD have not been changed, so the only difference is the allowed distance travelled per time step. As a result, the flood propagation patterns remain similar to HEC-RAS. One major drawback of increasing the temporal resolution is that the simulation takes more time. The complete simulation period of 12 days was not run using the 15-minute time step, but the first 4 days took the HAND.FLOW model 25 minutes on this temporal resolution. If the model will be used in another case study in later research, first a decision should be made on what part of the flood the focus will be; for a detailed look into the start of the flood a high temporal resolution is feasible, while for mapping the complete flood event a more coarse temporal resolution is needed if short simulation times are a requirement.

6 CONCLUSION

The aim of this study was to identify the most promising model for real-time flood forecasting after a dike breach, and investigate its capabilities for long-term use in case of changes in the hinterland. Three sub-questions were set up to guide the research and answer the main research questions. Each will be discussed below, and afterwards the main research question is answered.

1. *What is the performance of an LSTM network for an unknown set of dike breach flood events?*

The neural network that was developed in this study was a Long Short Term Memory (LSTM) neural network. Outflow hydrograph discharge data functioned as the input for the network per time step, and the water depth was predicted on every grid cell of the study area per time step. The LSTM architecture and hyperparameters were optimized for the lowest value of the error function on water depth (Mean Absolute Error (MAE), and it was checked if they predict the flood inundation extent accurately.

The performance of the LSTM on the 15 flood events used for testing was very accurate. The MAE of the network was 0.045 meters on an average water depth of 1.49 meters: an error of just 3%. Also, the NSE of the network was close to 0.99 for almost every grid cell in the study area. Close to the dike breach the NSE was slightly lower (0.95) and in the edge regions of the flooded area the NSE was worse. There, the LSTM had trouble with cells that flood in some scenarios and do not flood in others.

2. *How can the original HAND model be modified to model a dike breach flood?*

The HAND.FLOW model was created to be able to find the downstream path from the dike breach and travel a fixed distance along this path every time step. Using the Local Drainage Direction (LDD) map of the area, a pathfinding algorithm determines the steepest downstream path from the dike breach into the hinterland. To create a time component of simulation, a distance limit relationship was created using data from HEC-RAS simulations that limits the downstream distance covered by the pathfinding algorithm each time step to simulate flood propagation behaviour. Lastly, the volume of water flowing through the dike breach is divided over the hinterland that is accessible by this path at that time step, making use of the sub-catchment principle of the original HAND model. This results in the water depths per time step.

3. *What is the performance of the HAND.FLOW model for a set of dike breach flood events?*

The HAND.FLOW model was less accurate in predicting the water depths from HEC-RAS than the LSTM. The main reason for this is that the HAND.FLOW model was not equipped with a downstream boundary like HEC-RAS, so the HAND.FLOW model predicts water depths increasing indefinitely. Since HEC-RAS reaches its peak water levels about halfway during the simulation, the performance indicators were evaluated for this first half. The MAE was 0.37 m (error of 25%) and the majority of the grid cells in the area have reasonable NSE values of 0.7 to 0.8. Along the edges, the prediction is less accurate. However, it was found that the HEC-RAS model did not take into account two 3-meter high dikes compartmentalizing the study area, which the HAND model could detect due to its higher resolution. In a single corrected HEC-RAS simulation, the MAE became 0.21 meters (error of 15%) and the NSE was around 0.8 for large parts of the study area. Looking at the areas of the flood for each time step demonstrates that the HAND.FLOW model is a reasonable predictor of the flood propagation, with average CSI values of 0.72. This is important, as a good prediction of arrival time and flood propagation is arguably more important for policy makers than the precise water depth, as it determines when evacuation should be complete in a particular area.

4. *What is the performance of the HAND.FLOW model after a change in the hinterland topography?*

In this study, a section of the A12 highway between the service interchanges of Duiven and Westervoort was raised by 2 meters above ground level in the DEM, and the effect on the flood propagation was evaluated using both the HAND.FLOW and HEC-RAS models. Changing the DEM and re-calculating the LDD took around 30 minutes. This short time is advantageous for a flood forecasting system that will be used for longer periods of time, since the system will not be out of operation for a long time while it is updated.

The results show that both models flow along the highway instead of directly crossing it, only doing so only when the water depths are high enough to overcome the elevation difference. However, due to the way the HAND.FLOW model was programmed, the moment of flowing over the A12 and the subsequent pathfinding are slower than HEC-RAS, leading to a delayed prediction further downstream in the hinterland. Therefore, the NSE decreases downstream: close to the dike breach the NSE remains around 0.8, while further downstream the delay becomes larger and the NSE drops to around 0.6. All in all, the HAND.FLOW model is easily adapted to reflect a change in the hinterland and correctly predicts the new flood patterns. However, the accuracy of the flood arrival time for later time steps is subject to change.

For the answering of the main research question:
What are the drawbacks and benefits of neural networks and conceptual models in the context of real-time flood inundation forecasting after a dike breach, for current and future conditions of the hinterland?

The advantages of an LSTM network in a real-time flood forecasting system are that it is very fast and accurate after it has been trained, since “simulating” a flood event is near instant. The model is also easy to use, since it has no more settings to alter after training. The only input for operating the model is an outflow hydrograph. However, the main drawback of neural networks in a real-time flood forecasting system is that they require a lot of training data, and gathering this takes a lot of time (800 hours in the study by Bomers (2021)). Additionally, data gathering needs to be performed again after a change in the hinterland, or for a breach a kilometre downstream, since it may change the flood propagation patterns. All in all, a neural network can be trained to model any situation and a variety of output parameters (water depths, but also flow velocities), as long as enough relevant training data and time is available.

The benefits of using the HAND.FLOW model in a real-time flood forecasting are that it can be altered to reflect a change in the hinterland in just 30 minutes. Additionally, the model can be used for another dike breach location with relative ease. In its current state, the model is less accurate in reproducing the water depths from HEC-RAS than the LSTM counterpart. Still, it results in a reasonable representation of the propagation of the flood throughout the hinterland, which is arguably the most important metric for policy makers to plan an evacuation strategy. The simulation time is also important for real-time flood forecasting, and for the HAND.FLOW model the applicability will depend on the temporal and spatial resolution chosen. In this study, a large flood event was modelled on a resolution of 10x10 meters, which took the HAND.FLOW model 30 minutes. This is not very suitable for the large scale scenario analysis desired in real-time flood forecasting. On a more coarse spatial resolution of 150x150 meters used by HEC-RAS and the LSTM, the simulation took the HAND.FLOW model 1.5 minutes. This is much better, but the predictive accuracy decreased. Thus, the HAND.FLOW model in its current form should be used with care and with patience. In conclusion, this HAND.FLOW model offers in a relatively short time a reasonable first insight in how a flood will propagate into the hinterland, but it is insufficiently capable to be used for accurately predicting water depths. With further development, the model has the potential to be suitable and flexible enough to be successfully applied in a real-time flood forecasting system.

7 RECOMMENDATIONS

In this study, the LSTM neural network proved to be able to model dike breach flood events very accurately. Meanwhile, the developed HAND.FLOW model performs reasonably on predicting inundation extent, but lacks accuracy for water depth predictions. For practical use and further research, several recommendations are made:

- **Real-time flood forecasting system.** As was mentioned in the scope of this research in section 1.2, a real-time flood forecasting system consists of more components than a flood inundation model like the HAND.FLOW or LSTM. There should be a forecasting model for discharge in the river, and a 1D model of that river should determine water levels in order to determine when dike failure happens and what the outflow hydrograph of the breach will be. Many uncertainties are related to all these aspects, and research into the stacking of uncertainties when coupling these models is recommended. Still, a chain of these models is a goal to strive for, to enable policy makers to input an incoming upstream discharge wave and see the expected effects downstream.
- **Smaller flood.** The flood events used in this research consisted of events on a relatively large spatial and temporal scale. As such, the HEC-RAS simulations that were used to verify the LSTM and HAND.FLOW model were on a relatively coarse resolution. In future research, it is recommended to consider a smaller dike breach flood event to evaluate the LSTM and HAND.FLOW models on higher spatial and temporal resolutions.
- **LSTM in flood forecasting.** The LSTM constructed in this research showed that neural networks and deep learning are capable of simulating dike breach flood events accurately and fast. Realistically, however, their use in a flood forecasting system is limited due to the immense amount of time required for the training data of every potential dike breach location along a river system. Future research can focus on changing the setup of the neural network, to take into account characteristics of the area per grid cell. Possibly, the network can be taught the effect of the relative positions of each grid cell; such as that low lying cells experience higher water depths, and that cells behind a wall of higher elevations often flood later. This way, the LSTM would work for many different dike breach locations and hinterlands.
- **Amount of training data.** Since the data gathering for the neural network takes a long time for a flood of this size, it is recommended to research to what extent the LSTM neural network is "data-hungry". In this research, data from 58 dike breach events were used for the training procedure, while it could be that a well-selected group of 20 events also leads to a good neural network. Therefore, it is recommended to research the amount of training events required for a good LSTM.
- **Other neural networks.** Since the field of neural networks is developing so rapidly, other promising types of neural networks might be more suited to map flood extent spatially. As mentioned in the discussion, Convolutional Neural Networks (CNN) lend themselves well to the interpretation of raster files and images (Bentivoglio et al., 2021). This should make them applicable to flood inundation modelling. Perhaps a combination of LSTM and CNN architectures can even result in good predictions, as the LSTM has been proven in time-series prediction and the CNN is suitable for spatial analysis. Additionally, instead of a network predicting every cell, other approaches could be explored in which a network is made for only a selection of grid cells and then interpolated over the other cells (like in Kabir et al. (2020)). Future research is recommended to apply CNN networks to dike breach flood events.

- **HAND.FLOW in flood forecasting.** For policy makers, the current HAND.FLOW model is recommended to be used only as a first insight on where the flood waters will propagate. An advantage is that it can be quickly applied to another case study, requiring only a Digital Elevation Model (DEM) of the hinterland to be converted into a Local Drainage Direction (LDD) map, defining the location of the dike breach, and testing if the distance limit relationship is valid. For future research, it is recommended to apply the model to a case study in another area, to verify if the general approach behind the model is applicable there too.
- **Distance limit relationship.** In this research, the distance limit relationship for the HAND.FLOW model was dependent on simulation data from HEC-RAS. This makes its direct applicability to another case study questionable, without first simulating at least one dike breach there for reference. To solve this, it is strongly recommended to research if a more general relationship can be found based on characteristics of the study area. For example, the average slope and roughness along the flow path likely correlate with the distance water travels in a given time. Additionally, there is likely a relationship between the discharge through the breach and the distance travelled in the first time step, since larger discharges flow through the breach at high velocities and travel further. If such general relationships can be found between a number of dike breach case studies, the HAND.FLOW model could be made independent from HEC-RAS.
- **HAND.FLOW validation.** This research has evaluated the HAND.FLOW model for only a single dike breach location in a single hinterland. It is strongly recommended to research the performance of the model in other areas and breach locations too. Additionally, the flood events used for the creation of this model were simulated in HEC-RAS and did not happen in reality. Therefore, it is also recommended to apply the model to a historic flood event with real data. These steps would bring insight into the validity of the HAND.FLOW model beyond the IJssel river.

REFERENCES

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

Bermúdez, M., Cea, L., & Puertas, J. (2019). A rapid flood inundation model for hazard mapping based on least squares support vector machine regression. *Journal of Flood Risk Management*, *12*(S1), e12522. https://doi.org/10.1111/jfr3.12522

Bhola, P. K., Leandro, J., & Disse, M. (2018). Framework for Offline Flood Inundation Forecasts for Two-Dimensional Hydrodynamic Models. *Geosciences*, *8*(9), 346. https://doi.org/10.3390/geosciences8090346

Bomers, A. (2021). Predicting Outflow Hydrographs of Potential Dike Breaches in a Bifurcating River System Using NARX Neural Networks. *Hydrology*, *8*(2), 87. https://doi.org/10.3390/hydrology8020087

Bomers, A., Meulen, B., Schielen, R. M. J., & Hulscher, S. J. M. H. (2019). Historic Flood Reconstruction With the Use of an Artificial Neural Network. *Water Resources Research*, *55*(11), 9673–9688. https://doi.org/10.1029/2019WR025656

Bomers, A., Schielen, R. M. J., & Hulscher, S. J. M. H. (2019). Consequences of dike breaches and dike overflow in a bifurcating river system. *Natural Hazards*, *97*(1), 309–334. https://doi.org/10.1007/s11069-019-03643-y

Brownlee, J. (2020, August 16). Why Do I Get Different Results Each Time in Machine Learning? *Machine Learning Mastery*. https://machinelearningmastery.com/different-results-each-time-in-machine-learning/

Chaudhuri, C., Gray, A., & Robertson, C. (2021). InundatEd-v1.0: A height above nearest drainage (HAND)-based flood risk modeling system using a discrete global grid system. *Geoscientific Model Development*, *14*(6), 3295–3315. Scopus. https://doi.org/10.5194/gmd-14-3295-2021

Chu, H., Wu, W., Wang, Q. J., Nathan, R., & Wei, J. (2020). An ANN-based emulation modelling framework for flood inundation modelling: Application, challenges and future directions. *Environmental Modelling & Software*, *124*, 104587. https://doi.org/10.1016/j.envsoft.2019.104587

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

Hu, A., & Demir, I. (2021). Real-time flood mapping on client-side web systems using hand model. *Hydrology*, *8*(2). Scopus. https://doi.org/10.3390/hydrology8020065

Jamali, B., Bach, P. M., Cunningham, L., & Deletic, A. (2019). A Cellular Automata Fast Flood Evaluation (CA-ffé) Model. *Water Resources Research*, *55*(6), 4936–4953. https://doi.org/10.1029/2018WR023679

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

Karim, R. (2020, July 4). *Animated RNN, LSTM and GRU*. Medium. https://towardsdatascience.com/animated-rnn-lstm-and-gru-ef124d06cf45

Kilsdonk, R. A. H., Bomers, A., & Wijnberg, K. M. (2022). Predicting Urban Flooding Due to Extreme Precipitation Using a Long Short-Term Memory Neural Network. *Hydrology*, *9*(6), 105. https://doi.org/10.3390/hydrology9060105

Kingma, D. P., & Ba, J. (2017). Adam: A Method for Stochastic Optimization. *ArXiv:1412.6980 [Cs]*. http://arxiv.org/abs/1412.6980

Kumar, N. (2021, March 20). Optimizing Hyperparameters Using The Keras Tuner Framework. *MarkTechPost*. https://www.marktechpost.com/2021/03/20/optimizing-hyperparameters-using-the-keras-tuner-framework/

Le, X.-H., Ho, H. V., Lee, G., & Jung, S. (2019). Application of Long Short-Term Memory (LSTM) Neural Network for Flood Forecasting. *Water*, *11*(7), 1387. https://doi.org/10.3390/w11071387

Li, Z., Mount, J., & Demir, I. (2022). Accounting for uncertainty in real-time flood inundation mapping using HAND model: Iowa case study. *Natural Hazards*, *112*(1), 977–1004. https://doi.org/10.1007/s11069-022-05215-z

Lim, N. J., & Brandt, S. A. (2019). Are Feature Agreement Statistics Alone Sufficient to Validate Modelled Flood Extent Quality? A Study on Three Swedish Rivers Using Different Digital Elevation Model Resolutions. *Mathematical Problems in Engineering*, *2019*, e9816098. https://doi.org/10.1155/2019/9816098

Liu, M., Huang, Y., Li, Z., Tong, B., Liu, Z., Sun, M., Jiang, F., & Zhang, H. (2020). The Applicability of LSTM-KNN Model for Real-Time Flood Forecasting in Different Climate Zones in China. *Water*, *12*(2), 440. https://doi.org/10.3390/w12020440

Liu, Y. B., & Smedt, F. D. (2004). *WetSpa Extension, A GIS-based Hydrologic Model for Flood Prediction and Watershed Management*. 126.

Lu, D., Konapala, G., Painter, S. L., Kao, S.-C., & Gangrade, S. (2021). Streamflow Simulation in Data-Scarce Basins Using Bayesian and Physics-Informed Machine Learning Models. *Journal of Hydrometeorology*, *22*(6), 1421–1438. https://doi.org/10.1175/JHM-D-20-0082.1

McGrath, H., Bourgon, J.-F., Proulx-Bourque, J.-S., Nastev, M., & Abo El Ezz, A. (2018). A comparison of simplified conceptual models for rapid web-based flood inundation mapping. *Natural Hazards*, *93*(2), 905–920. https://doi.org/10.1007/s11069-018-3331-y

Mosavi, A., Ozturk, P., & Chau, K.-W. (2018). Flood prediction using machine learning models: Literature review. *Water (Switzerland)*, *10*(11). Scopus. https://doi.org/10.3390/w10111536

Nobre, A. D., Cuartas, L. A., Hodnett, M., Rennó, C. D., Rodrigues, G., Silveira, A., Waterloo, M., & Saleska, S. (2011). Height Above the Nearest Drainage – a hydrologically relevant new terrain model. *Journal of Hydrology*, *404*(1), 13–29. https://doi.org/10.1016/j.jhydrol.2011.03.051

Nobre, A. D., Cuartas, L. A., Momo, M. R., Severo, D. L., Pinheiro, A., & Nobre, C. A. (2016). HAND contour: A new proxy predictor of inundation extent. *Hydrological Processes*, *30*(2), 320–333. https://doi.org/10.1002/hyp.10581

Quirogaa, V. M., Kurea, S., Udoa, K., & Manoa, A. (2016). Application of 2D numerical simulation for the analysis of the February 2014 Bolivian Amazonia flood: Application of the new HEC-RAS version 5. *Ribagua*, *3*(1), 25–33. https://doi.org/10.1016/j.riba.2015.12.001

Rajaee, T., Ebrahimi, H., & Nourani, V. (2019). A review of the artificial intelligence methods in groundwater level modeling. *Journal of Hydrology*, *572*, 336–351. https://doi.org/10.1016/j.jhydrol.2018.12.037

Razavi, S., Tolson, B. A., & Burn, D. H. (2012). Review of surrogate modeling in water resources. *Water Resources Research*, *48*(7). https://doi.org/10.1029/2011WR011527

Scriven, B. W. G., McGrath, H., & Stefanakis, E. (2021). GIS derived synthetic rating curves and HAND model to support on-the-fly flood mapping. *Natural Hazards*, *109*(2), 1629–1653. Scopus. https://doi.org/10.1007/s11069-021-04892-6

Shao, J., Hu, K., Wang, C., Xue, X., & Raj, B. (2020). Is normalization indispensable for training deep neural network? *Advances in Neural Information Processing Systems*, *33*, 13434–13444. https://proceedings.neurips.cc/paper/2020/hash/9b8619251a19057cff70779273e95aa6-Abstract.html

Solvik, K., Bartuszevige, A. M., Bogaerts, M., & Joseph, M. B. (2021). Predicting Playa Inundation Using a Long Short-Term Memory Neural Network. *Water Resources Research*, *57*(12), e2020WR029009. https://doi.org/10.1029/2020WR029009

Speckhann, G. A., Borges Chaffe, P. L., Fabris Goerl, R., Abreu, J. J. D., & Altamirano Flores, J. A. (2018). Flood hazard mapping in Southern Brazil: A combination of flow frequency analysis and the HAND model. *Hydrological Sciences Journal*, *63*(1), 87–100. Scopus. https://doi.org/10.1080/02626667.2017.1409896

Staudemeyer, R. C., & Morris, E. R. (2019). Understanding LSTM -- a tutorial into Long Short-Term Memory Recurrent Neural Networks. *ArXiv:1909.09586 [Cs]*. http://arxiv.org/abs/1909.09586

Teng, J., Jakeman, A. J., Vaze, J., Croke, B. F. W., Dutta, D., & Kim, S. (2017). Flood inundation modelling: A review of methods, recent advances and uncertainty analysis. *Environmental Modelling & Software*, *90*, 201–216. https://doi.org/10.1016/j.envsoft.2017.01.006

Verwey, A., Kerblat, Y., & Chia, B. (2017). *Flood Risk Management at River Basin Scale: The Need to Adopt a Proactive Approach* [Working Paper]. World Bank. https://doi.org/10.1596/27472

Xie, S., Wu, W., Mooser, S., Wang, Q. J., Nathan, R., & Huang, Y. (2021). Artificial neural network based hybrid modeling approach for flood inundation modeling. *Journal of Hydrology*, *592*, 125605. https://doi.org/10.1016/j.jhydrol.2020.125605

Zhang, D., Lindholm, G., & Ratnaweera, H. (2018). Use long short-term memory to enhance Internet of Things for combined sewer overflow monitoring. *Journal of Hydrology*, *556*, 409–418. https://doi.org/10.1016/j.jhydrol.2017.11.018