Predicting flooding due to extreme precipitation in an urban environment using machine learning algorithms

Raphaël Kilsdonk

A thesis presented for the degree of
MSc Civil Engineering & Management

HydroLogic

UNIVERSITY OF TWENTE.

Faculty of Engineering Technology
University of Twente
The Netherlands
April 2, 2021

Preface

I have always had an interest in the development of AI and machine learning. The first time I applied machine learning techniques, was during my bachelor thesis, here I used machine learning for the optimisation of structural design. I realise now how little I knew then about machine learning and the wide range of applications in engineering. After following a course on machine learning in engineering during my master program, I was delighted to be presented with the opportunity to research the application of machine learning in hydrology for my master thesis. Together with Hydrologic we constructed a research objective for my master thesis: predicting flooding due to extreme precipitation in an urban environment using machine learning algorithms. During my master thesis research I have learned a lot about machine learning and the challenges one faces when applying machine learning techniques.

I would like to thank my supervisors Matthijs van den Brink, Anouk Bomers and Kathelijne Wijnberg for their support and guidance throughout my master thesis research. I would also like to thank Sam de Roover for his day to day support and discussions on the use of machine learning in hydrology.

I hope you enjoy your reading.

Raphaël Kilsdonk

Utrecht, April 2, 2021

Abstract

Pluvial flooding in an urban environment can occur quite sudden. Therefore, flood early warning systems with a short run time are desired. A method to reduce computational load is surrogate modelling. Response surface surrogate models (Machine learning (ML) algorithms) are a second level abstraction from reality. These algorithms do not emulate any internal component of the original simulation, but try to find relations between the input variables and output. They are, once trained, extremely fast in predicting the output from a given input. Therefore, the use of such ML algorithms as a flood early warning system is studied.

Precipitation time series are used for the prediction of flooding by the ML algorithms. This is the only input data used by the ML algorithms. Precipitation statistics are used for the construction of a synthetic precipitation event data set, as there is not enough recorded data from historic flood events available that can be used to train, test and validate the ML algorithms. Short term precipitation events with durations of 4, 8 and 12 hours, 7 different patterns and 6 precipitation intensities are constructed. This provides 126 different precipitation events.

Maximum flood volume and flood volume time series for all manholes in the area are obtained using a sewer model. This is a validated 1D sewer model, built with Infoworks Integrated Catchment Modelling (ICM). Topographic gradients of the surface are not included in the model. Runoff into the sewer system is determined by the shortest flow path to the nearest inlet. To limit the complexity and size of the data set, a case study area is chosen that is smaller than the whole municipality of Amersfoort. Therefore, the area of Hooglanderveen is chosen. This area has a combined sewer system, which historically has experienced frequent flooding.

The architecture of an Artificial neural network (ANN) makes it very suitable for the simulation of a sewer system. The hidden layer and weights that connect nodes, can simulate the non-linear interactions between manholes in a sewer network. Two ANN types are constructed and tested in the context of this research. First, the Multi-layer perceptron (MLP) is constructed for both classification and regression. With classification, flooding of manholes is classified for each precipitation event, predicting only if flooding occurs at each manhole. With regression, the maximum flood volumes that occur at each manhole is predicted for each precipitation event. Second, a Long short-term memory (LSTM) network is constructed for the regression of flood volume time series. Here, flood volume time series are predicted for all manholes in the studied area. The Python packages scikit-learn and Keras have been used for the construction and training of the MLP and LSTM respectively.

For the MLP in classification and regression the characteristics used for construction of the synthetic precipitation data set are used as input features (precipitation duration, intensity and pattern). The LSTM makes use of the full precipitation time series. After construction, the MLP and LSTM hyper-parameter configurations have been optimised using random search and Bayesian optimisation hyper-parameter optimisation respectively. Using hyper-parameter optimisation, important hyper-parameters, such as the size and number of hidden layers and learning rate are determined. For the training and testing of the algorithms 80% of the synthetic data set is used.

To test the LSTM algorithm on a realistic precipitation data set, 5 historic flood events from the case study area of Hooglanderveen are obtained. These are historic precipitation events that induced flooding, as reported by inhabitants of the area. The historic data is also fed into the sewer model to obtain flood volume time series for all manholes in the study area.

Validation of the ML algorithms is done using the validation data set, which is 20% of the synthetic data set. The MLP classifier obtained an accuracy of 99.29%, classifying if flooding occurs at a manhole or

not. The MLP regressor obtained a Mean absolute error (MAE) of 0.20 m^3 and a R^2 of 0.997, accurately predicting maximum flood volumes for each manhole. The LSTM algorithm obtained a MAE of 0.06 m^3 and R^2 of 0.99. Furthermore, the predictive capability of the LSTM was evaluated with the Nash-Sutcliffe efficiency (NSE), providing a mean NSE of 0.87 on the validation data set. Only the LSTM is evaluated by the NSE, as it is the only ML algorithm that predicts flood volume time series. The LSTM, therefore, has proven to have a high predictive capability of flood volume time series of a sewer system. The results also show, that the LSTM is able to time the peak of the flood wave and the duration of inflow and outflow in the sewer system.

Testing of the LSTM on the historic data provided a MAE of 0.19 m^3, NSE of 0.61 and R^2 of 0.99. The evaluation metrics are generally lower, with the NSE, being substantially lower. Still, high performance can be observed for the manholes that experienced large flood volumes.

From the validation on the synthetic validation data and testing on the historic data, two main observations of the LSTM were made. First, a tendency of underestimation of the peaks is observed. However, this underestimation is only observed at high peak flood volumes. Therefore, this underestimation does not reduce the predictive capability of the LSTM as a flood early warning system. Second, the LSTM has a high sensitivity to small perturbations in the precipitation input time series, this is especially noticeable with the historic data. However, when a manhole experiences frequent flooding due to extreme precipitation events, the LSTM is less sensitive to small perturbations in the input precipitation data.

Contents

List of Figures

List of Tables

Abbreviations

ANN	Artificial neural network
BP	Backpropagation
CE	Coefficient of efficiency
CPU	Central processing unit
FNN	Feedforward neural network
GPU	Graphics processing unit
GRU	Gated recurrent unit
ICM	Integrated Catchment Modelling
LSTM	Long short-term memory
MAE	Mean absolute error
ML	Machine learning
MLP	Multi-layer perceptron
MSE	Mean squared error
MSRE	Mean squared relative error
NSE	Nash-Sutcliffe efficiency
RNN	Recurrent neural network
STOWA	Stichting Toegepast Onderzoek Waterbeheer
SWE	Shallow water equations

1 Introduction

1.1 Background

Flood management is a vital part of combating climate change. Innovation in flood management is key in adapting to a changing environment and climate. The consequences of inadequate flood management are significant. Annual economic losses are up to tens of billions of US dollars with thousands of people killed due to flooding globally (Hirabayashi et al., 2013). Flooding can have several different causes. First, storm events can cause high water levels at sea and river dikes, causing flooding due to overtopping or structural failure. Second, extreme precipitation events, of both short and long duration, can cause flooding of areas. These extreme precipitation events can cause flooding locally or downstream of a catchment due to raising of the water levels in a river (Szöllösi-Nagy and Zevenbergen, 2018).

The present research will focus on local flooding due to extreme precipitation. Specifically flooding in an urban environment due to exceedance of the sewer capacity. Flooding in urban environments differs from other areas as there is a large amount of impervious surface area. This negates infiltration and increases the load on the sewer system. In e.g. a field, flooding due to extreme precipitation mostly occurs due to prolonged precipitation. The precipitation will first infiltrate the soil, once the soil is saturated and deep infiltration capacity is exceeded flooding will occur. Flooding in an urban environment is caused by short extreme precipitation events where infiltration is negligible.

Human induced climate change has increased the frequency and intensity of extreme precipitation events in the northern hemisphere (Min et al., 2011). Frequency of flooding due to these extreme precipitation events will subsequently also increase if sewer systems are not improved. Flood early warning systems can be used to predict flooding due to extreme precipitation if sewer systems are not capable to handle these extreme precipitation event and provide lead-times for evacuation and preparation.

Flood early warning systems are widely used around the world to provide a prediction of the time of flooding. The lead-time of the flood early warning systems varies greatly and is dependent on the system that is observed. For a large river system like the Rhine, flooding from a dike breaching due to precipitation in the Alps, can be predicted with a large lead-time. Generally the larger the lead-time the larger the uncertainty in time of flooding, due to e.g. weather forecasts (Verkade and Werner, 2011).

Flooding due to extreme precipitation events in an urban area can occur within a few hours. Therefore, a flood early warning system needs to be able to predict flooding almost instantaneously. The faster the flooding can be predicted the larger the lead-time. Current physics based modelling approaches are computationally expensive. These physics based models are highly detailed and can emulate a whole sewer system in a city. However, this level of detail comes with a computational price. Therefore, other approaches for flood prediction are studied (Ayazpour et al., 2019; Mounce et al., 2014; Li et al., 2011).

A method to reduce computational load is surrogate modelling. Surrogate modelling reduces the computational cost while approximating the original simulation model. Surrogate models are a second level abstraction from the original system. Response surface surrogate models, also called Machine learning (ML) algorithms, are a type of surrogate model (Razavi et al., 2012). They do not emulate any internal component of the original simulation model, but try to find relations between the input variables and output. They are, once trained, extremely fast in predicting the output from a given input (Razavi et al., 2012) and can subsequently do so on a continuous basis. This makes the use of ML algorithms advantageous in flood early warning systems.

The present research objective is to construct a ML algorithm that can predict urban flooding due to extreme precipitation. The algorithm should be able to predict, using precipitation data, flooding of all manholes in a specified area. The algorithm should therefore be able to model the non-linear interactions between manholes. The amount of interactions depends on the complexity of the system. One street with three manholes that are connected with piping will have a rather linear interaction, while a whole sewer system will be more complicated and non-linear.

1.2 Research Questions

A main research question is proposed that encompasses the research objective. Sub-questions are composed that aim to answer the main research question.

1.2.1 Main question

To what extent can machine learning algorithms be used to construct a location based flood early warning system for pluvial flooding in an urban environment?

1.2.2 Sub-questions

1. How can synthetic data be used for the construction of machine learning algorithms?
    - Extreme precipitation events that induce flooding are rare occurrences. Therefore, there are not enough recorded historic events for the training, testing and validation of ML algorithms. Synthetic data can be used to bridge this gap and provide enough data for training, testing and validation.
2. What hyper-parameter configurations are best performing?
    - With ML algorithms there are many 'higher order' parameters that need to be determined. These parameters are not trained by the ML algorithms and are called hyper-parameters. These hyper-parameters have no physical attributes and cannot be empirically determined. Therefore, many combinations of hyper-parameters need to be tested to determine the optimal configuration for the specific problem.
3. What is the performance of the algorithms and is this persistent when validated on historic data?
    - Final performance of the algorithms will determine applicability for the prediction of pluvial flooding in an urban environment. The ML algorithms are also tested on available historic data. This can give an indication if the ML algorithms trained on synthetic data, could be applied in a real world environment.

1.3 General research approach

A numerical sewer model is used to produce flood volume time series for each manhole in the studied area. This sewer model uses synthetically constructed precipitation time series as input to produce the flood volume time series output. This is done due to two main reasons. First, there are not enough recorded historic precipitation events that induced flooding for the training, testing and validation of ML algorithms. Rajaee et al. (2019) indicates that more than 100 samples are needed for this purpose. Second, there is no sensor data available from manholes in the study area, that can provide flood volume data.

It is important to note the levels of abstraction from reality present in this research. Fig. 1.1 shows a schematised overview of these levels of abstraction from reality. First, we have the real world sewer system, this is the basis for the numerical sewer model. This sewer model is a first level abstraction from reality. The ML algorithm is then trained on data produced by the sewer model, making it a second level abstraction from reality. Furthermore, synthetic precipitation data is used as input for the sewer model, introducing another level of abstraction from reality. To test the ML algorithms in a real world environment, historic data is used as input for the sewer model and trained ML algorithm. This shows if the ML algorithm can accurately predict flood volume time series in a real world environment. However,

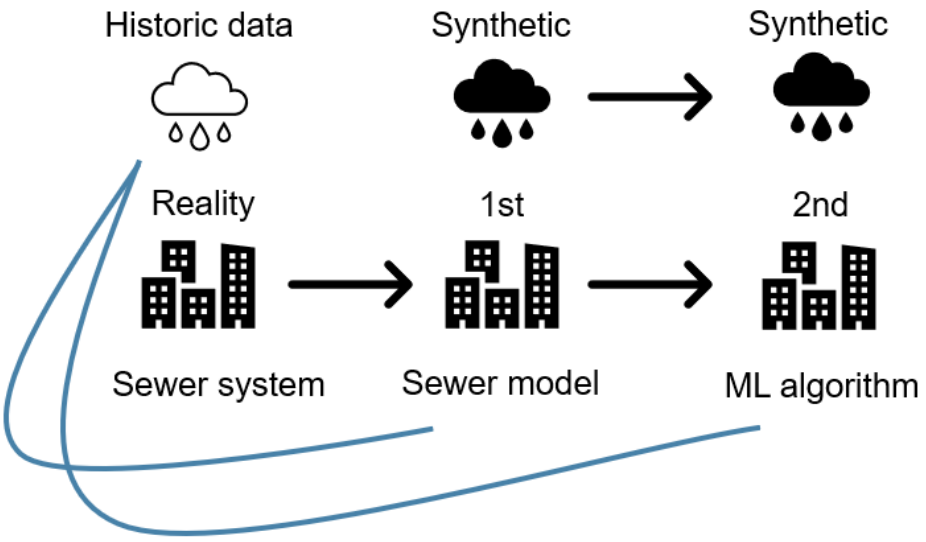


Figure 1.1: Levels of abstraction from reality

as the ML algorithm is still trained, tested and validated on the results produced by the sewer model, the predictive ability can only be as good or less as the sewer model it is trained on.

This thesis will first explore the sewer system used as a case study and its characteristics (chapter 2). In this chapter we determine how flooding occurs in the sewer system due to extreme precipitation. Second, the ML algorithms used are described and their mathematical formulations explained (chapter 3). Third, the methodology for the construction of the synthetic data set, use of the sewer model, data pre-processing, development of the ML algorithms, hyper-parameter optimisation and acquisition of historic data is explained (chapter 4). Fourth, the results of the algorithms and their performance is detailed (chapter 5). Last, the research is discussed, concluded and several recommendations for further research are made (chapter 6).

1.4 Case study area

Sewer model results from a specified area are used to train and validate the ML algorithms. For this purpose, the residential area of Hooglanderveen in Amersfoort, the Netherlands is chosen. Hooglanderveen is located to the northeast of Amersfoort, see Fig. 1.2. Hooglanderveen has a combined sewer system. This region has been chosen instead of using the whole area of Amersfoort for two main reasons. First, historically Hooglanderveen experiences frequent pluvial flooding. This makes it an interesting region for research into flood early warning systems. Second, relatively large amount of model runs and subsequently model results are needed to train a ML algorithm. Therefore, a smaller area has been chosen to accommodate construction time of this data set by the sewer model and limit the size of the data set. The whole area of Amersfoort has more than 2.3×10^4 manholes. If an output is taken from each manhole for each timestep the data set will become extremely large (30-60 GB), as indicated by Arcadis. Although the region of Hooglanderveen is chosen as a case study, the methods researched should be applicable to any residential area with a similar sewer system and topographical features.

1.5 Research affiliation

This research is part of the European SCOREWater project. The main goal of the project is to enhance the resilience of cities against climate change and urbanisation. The project is a cooperation between the municipalities of Barcelona, Amersfoort and Göteborg. In Amersfoort the project focus lies on pluvial flood detection and prevention while reducing environmental impact. The research is in cooperation with Hydrologic, the faculty Engineering Technology of the University of Twente, the municipality of Amersfoort and Arcadis.

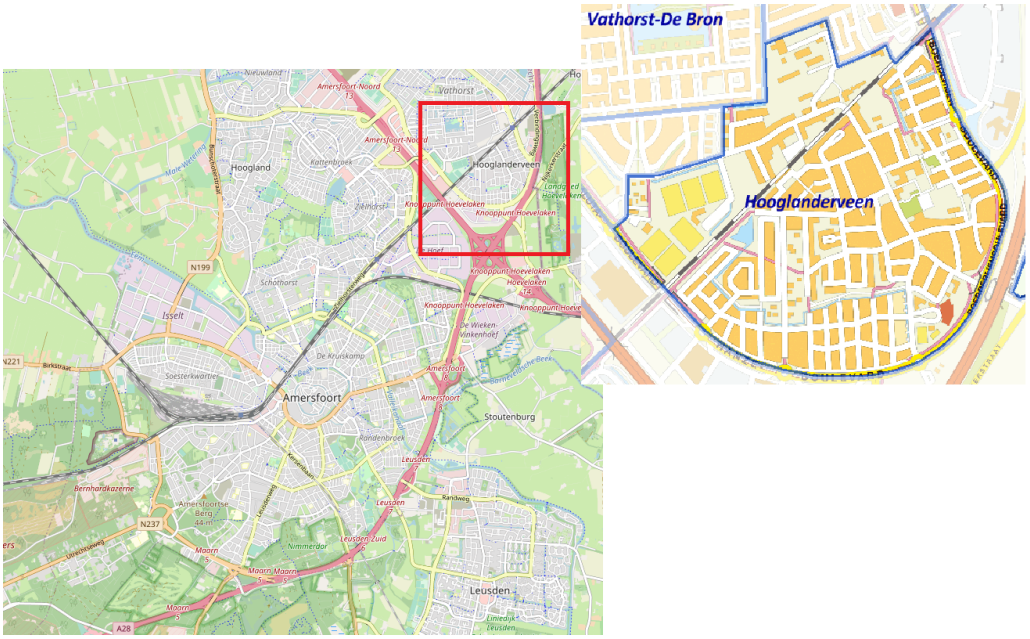


Figure 1.2: Location of the case study area Hooglanderveen.

2 System

2.1 Sewer system

When we look at the sewer system as a whole, we can define two components, these are defined by Szöllösi-Nagy and Zevenbergen (2018) as the major and minor system. In this chapter both will be described. The major sewer system is composed of streets, inlets, ditches and surface water channels, the system can be characterised as the surface system. The minor system is the subsurface system composed of interconnected piping, manholes, overflows and pumps. An overview of the storm-water flow through the major and minor systems of a combined sewer system is shown in Fig. 2.1.

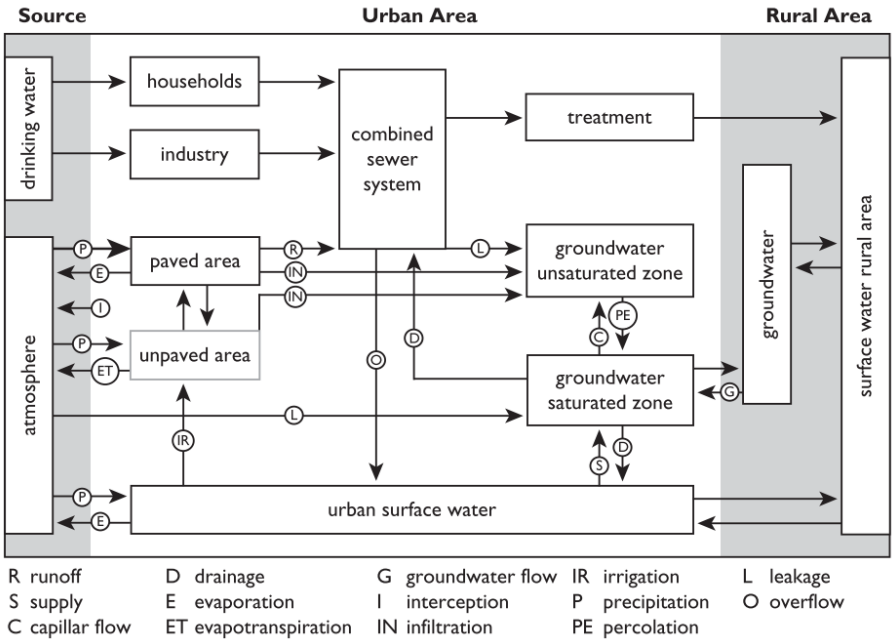


Figure 2.1: An overview of the urban water system. This study will focus on the flooding of the combined sewer system by precipitation from the atmosphere (Figure 5-1, (Szöllösi-Nagy and Zevenbergen, 2018)).

2.1.1 Major system

The major sewer system is composed of streets, inlets, ditches and surface water channels. The system can be characterised as the surface system. Precipitation will fall onto components of the major system after which it will flow into the minor system.

Precipitation will be transformed into water vapour, groundwater and storm-water runoff (Szöllösi-Nagy and Zevenbergen, 2018). The storm-water runoff will enter the minor system. The percentage of precipitation that will be transformed into storm-water runoff and enters the minor system is dependent on the environment. Fig. 2.2 shows the distribution of precipitation transformation for different environments. With pluvial flooding in an urban environment a runoff percentage of $30\% - 55\%$ is expected. For the

major system all impervious surfaces (roofing and streets) are included in the model to calculate flow into the inlets. The precipitation that falls on these impervious surfaces flows into the nearest inlet. The precipitation events studied in the present research are all of very short duration with high intensity, therefore, evaporation will be minimal.

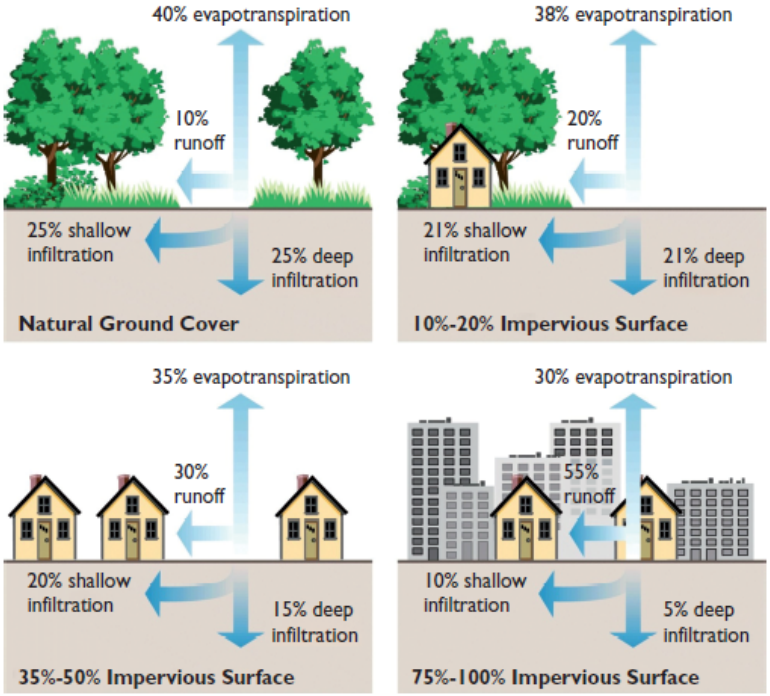


Figure 2.2: Transformation of precipitation to water-vapour (evapotranspiration), groundwater (shallow infiltration and deep infiltration) and storm-water runoff (Figure 5-2, (Szöllösi-Nagy and Zevenbergen, 2018)).

2.1.2 Minor system

The minor system is the subsurface system composed of interconnected piping, manholes, overflows and pumps (Szöllösi-Nagy and Zevenbergen, 2018). The minor system transports the domestic sewage, industry wastewater and storm-water runoff to a treatment facility. In the sewer network manholes are located wherever there is a change in gradient and/or alignment. Here the manholes can also branch the sewer network.

Precipitation enters the minor system via inlets. A schematised connection of these inlets to the subsurface sewer piping is shown in Fig. 2.3. A water lock is situated between the inlet and the sewer. This prevents unwanted odours from reaching the street level. This water lock also causes the system to be closed.

After entering the minor system precipitation will flow through the sewer piping to a treatment facility. Note that this only occurs if the system is a combined system where storm-water, domestic sewage and industry wastewater are transported together. Pumps are situated at different locations in the sewer network to limit the depth of the sewer network. They pump sewage from a lower lying sewer pipe to a higher sewer pipe or surface water (Rioned, 2020). This is required for a gravity sewer system, where transport of sewage is realised by gravity. Transport of sewage can also be done using pressure pumps, which is commonly used in areas with largely varying topographical gradients.

To prevent flooding, another structure is added to the sewer network. This structure is called an overflow. When the discharge in the sewer network exceeds a certain threshold the overflow will discharge sewage onto surface water. This can prevent flooding from the manholes. However overflows have a maximum capacity. If this capacity is exceeded the water level could keep rising, causing flooding at the service level.

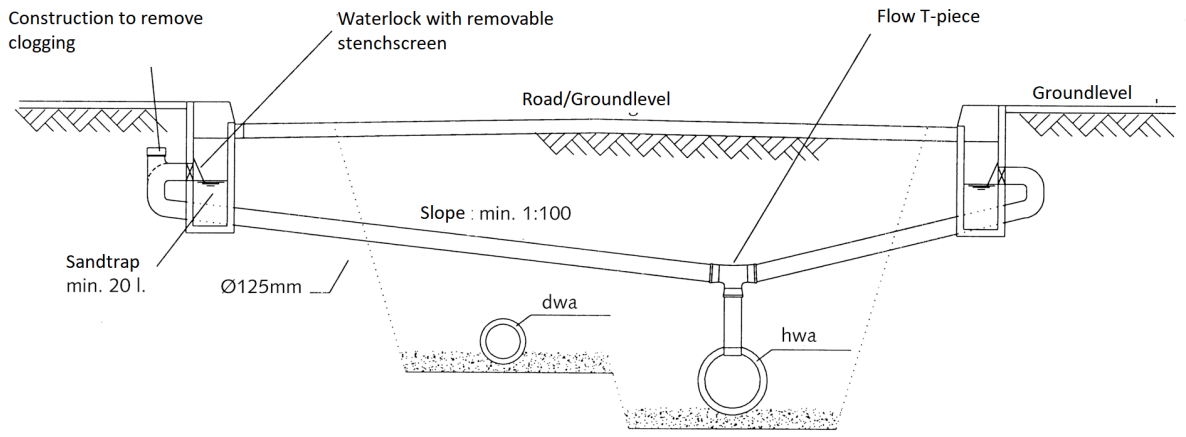


Figure 2.3: Schematised connection of inlets to the sewer piping (Rioned, 2020). Note that the dwa (domestic sewage) and hwa are seperated in this schematisation. The study area in the present reaseach has a combined system, here the dwa and hwa are combined in the sewer system/pipe.

2.1.3 Pluvial flooding

Flooding due to extreme precipitation occurs when the hydraulic head in the piping of the minor system exceeds the ground level of the manhole(s). This commonly occurs when the sewer system capacity is exceeded due to an extreme precipitation volume. With exceedance of the minor system capacity, capacity of one or several following components are exceeded:

- Combined pump discharge capacity
- Combined overflow discharge capacity
- Sewer storage capacity
- Sewer discharge capacity

Flooding occurs whenever and wherever the discharge capacity of the inlet into the minor system is exceeded. This can have several causes. First, flooding can occur when precipitation intensity exceeds the discharge capacity of the inlet. Water cannot enter the minor system and remains at the service level. Second, discharge capacity may be lower between sewer piping due to e.g. clogging or smaller pipe diameters, this can cause water to flow back onto the streets through the inlets or manholes. Third, a combined gravity driven sewer system has a larger discharge capacity than the pump at the end of the system. Therefore, a storage is designed in the minor system to accommodate this difference in capacity. This storage in the Netherlands is equivalent to approximately 7 mm - 9 mm precipitation (Rioned, 2020). When the storage capacity is exceeded and there is more water that enters the system, storm water will exit via the overflows present in the system. If the capacity of the overflows is exceeded storm water will flood the streets.

3 Machine learning algorithms

There are many machine learning algorithms that can be used for flood early warning systems. The choice depends mainly on the input data and preferred output. The present research used Artificial neural network (ANN) for the flood early warning system. The architecture of ANNs makes them very suited for the simulation of networks such as a sewer network. As the ANN is a network of connected neurons with weights, the ANN will, after training, model the non-linear interactions within a sewer network. The ANN can therefore model spatial relations between manhole nodes that other ML algorithms cannot.

Two ANN types will be used. First, a Multi-layer perceptron (MLP) will be implemented which is the most basic form of an ANN. Second, a Long short-term memory (LSTM) will be used to model temporal relations. Both variants of the ANN will be further detailed in this chapter.

3.1 Artificial neural network (ANN)

The ANN is the most commonly used ML algorithm. There are many variants that all work on the same basis. Dawson and Wilby (2001) provides an explanation of how such a network works. An artificial neural network is composed of three or more layers. An input layer with all the input variables, a hidden layer with neurons, and a transfer function for each neuron and an output layer with a neuron for each output and an activation function. An overview of such a network is shown in Fig. 3.1. This is the basis for each ANN type and is also called a MLP. Within the hidden layer a number of neurons are modelled. All neurons are connected to each input and output. If there is more than one hidden layer the neurons are also connected to each neuron in the other hidden layer. The input data passes from the left (input layer), through the hidden layers to the output layer. This is also called a Feedforward neural network (FNN).

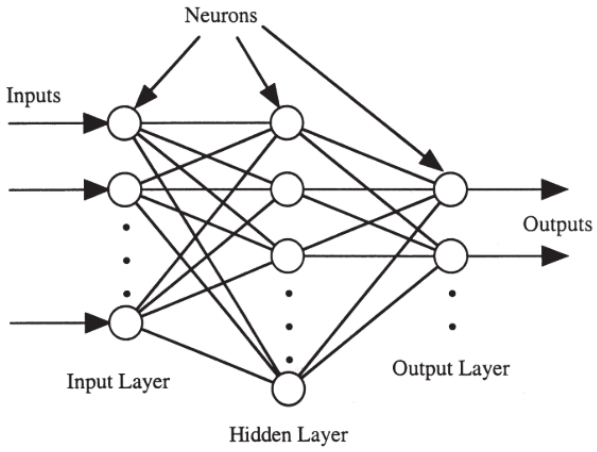


Figure 3.1: An overview of an artificial neural network (Figure 2, (Dawson and Wilby, 2001)).

The activation value of a neuron comes from the weighted sum of the input variables. This value is then used in the transfer function to determine the output of the neuron. Fig. 3.2 shows the activation of

a single neuron. Here u_i is the real-valued input from the input layer or a previous hidden layer, w_{ij} the weights from each input connected to neuron (j) and $f(S_j)$ the output of the neuron. The function for the output is then $f(S_j) = f(\sum_{i=0}^{n} w_{ij} u_i)$ (Dawson and Wilby, 2001), with f being the transfer function[1]. Several transfer functions can be used. An example, the sigmoidal function can be seen in Fig. 3.2 and is detailed in Eq. 3.1

$$f(x) = \frac{1}{1 + \exp -x} \tag{3.1}$$

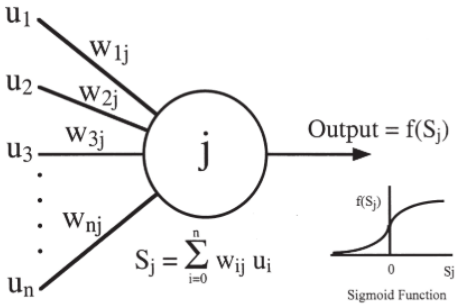


Figure 3.2: Activation of a single neuron (Figure 1, (Dawson and Wilby, 2001)).

The activation function(s) in the last layer (output layer) gives the output from the neurons in the previous layer. As mentioned before, all neurons in the layer before the output layer are connected to the output activation function(s). The output activation function can give a real-valued or classification output. In the case of a real-valued output of a continuous function the output activation function is a linear function (Moreno et al., 2011; Dawson and Wilby, 2001). For a classification output, the activation is a softmax function detailed in Eq. 3.2 (Goodfellow et al., 2016). The soft-max function is used to normalise the output of an ANN to a probability distribution between classes. In this equation, $\mathbf{x}$ is the vector of the summed input for each output neuron in the output layer.

$$\text{softmax}(\mathbf{x})_i = \frac{\exp x_i}{\sum_{j=1}^{n} \exp x_j} \tag{3.2}$$

Training of the ANN is done using a loss function and adjusting the weights in the neural network using e.g. back-propagation. The loss function quantifies the difference between the actual values and predictions made by the ANN. There are, as with the activation function, several loss functions to choose from. The choice is somewhat subjective, with a few loss functions that are designed for specific cases and algorithms. It is recommended to analyse performance of several loss functions.

A commonly used loss function for regression is the Mean absolute error (MAE) (Dawson and Wilby, 2001). There are more loss functions for regression that can be used such as the Mean squared relative error (MSRE), the Mean squared error (MSE), Coefficient of efficiency (CE) and Coefficient of determination (R^2). Dawson and Wilby (2001) mentions that the MSE provides a good measure for high river flows and the MSRE provides a more balanced estimate of the fit at moderate river flows. The CE and R^2 are useful for comparisons between studies as they are not dependent on the scale of the data.

In the present research, the MAE is used as the loss function for the regression algorithms as it has shown better performance, in training of the ML algorithms, than the MSE and MSRE in initial testing. The equation for the MAE is detailed in Eq. 3.3.

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{3.3}$$

where;

[1] In some literature, this is also called the activation function, to avoid confusion with the output activation function it is here called the transfer function.

MAE = the observed loss

y_i = the predicted value

$\hat{y}_i$ = the actual value

n = the number of predictions

For the classification algorithm, the cross-entropy loss function is used, see Eq. 3.4. Another commonly used loss function is the accuracy. The cross-entropy loss function estimates the loss in predicted probabilities instead of the discrete outputs.

$$L = -\log \Pr(y|p) = -(y \log(p) + (1-y) \log(1-p)) \tag{3.4}$$

where;

L = the cross-entropy loss or log loss

y = the true label $y \in \{0, 1\}$

p = the probability estimate $\Pr(y = 1)$

Backpropagation (BP) is the most commonly used technique to train an ANN. BP uses gradient-descent to adjust the weights of the network. The weights of the ANN are adjusted proportional to the partial derivative of the loss function with respect to the weights. This is done in each training iteration. Figure 3.3 shows how the gradient-descent is used to get the local minimum of a function (Goodfellow et al., 2016). BP propagates this change back throughout the network using the chain rule. This derivative is multiplied by a so-called learning rate, it defined the step size taken in gradient descent. The learning rate prevents overshooting and slow convergence in finding the local minimum (Goodfellow et al., 2016). When using large learning rates it is possible a positive feedback loop occurs, here large weights induce large gradient and so on. This causes overshooting and weights moving towards infinity. Therefore, the learning rate needs to be optimised to find one that provides fast convergence, but does not cause overshooting.

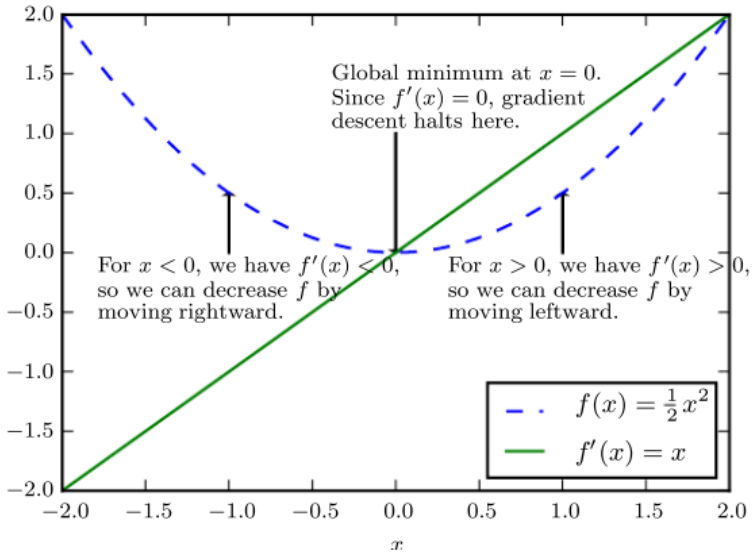


Figure 3.3: Illustration of how gradient-descent finds the local minimum of a function (Figure 4-2, (Goodfellow et al., 2016)).

3.2 Recurrent neural network (RNN)

In a recurrent neural network, the outputs of the hidden layer are stored for use in the next pass of data through the network. This gives the network a 'short term memory'. RNNs are especially useful in representing time relationships in a time series (Moreno et al., 2011). After each forward pass, the

outputs of the neurons are stored in a so-called 'context layer'. In the next run through the network, values stored in the context layer are fed back into the network. Figure 3.4 provides a conceptual overview of a Recurrent neural network (RNN). The time delay and frequency that the context layer is fed back into the model can be changed. The main limitation of this simple recurrent neural network is the limited long term memory. Information is not stored for more than one timestep. Therefore, other RNN architectures have been researched, with the main two architectures used being the LSTM (Hochreiter and Schmidhuber, 1997) and Gated recurrent unit (GRU) (Cho et al., 2014).

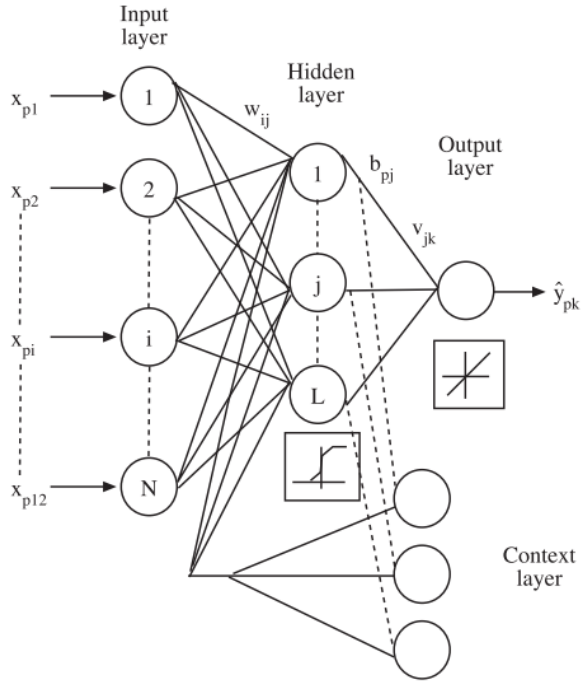


Figure 3.4: Layout of a recurrent neural network (Figure 3D, (Moreno et al., 2011)).

3.2.1 Long short-term memory (LSTM) and Gated recurrent unit (GRU)

An excellent explanation of the LSTM and GRU networks is given by Christopher Olah (2015), which will be summarised here. For further detail see Hochreiter and Schmidhuber (1997) and Cho et al. (2014). The LSTM cell tackles the issue that a simple RNN network cannot model long term dependencies. It cannot store information for later use. The LSTM cell uses the cell state to store information. This cell state is updated with the input (x_t) and previous output (h_{t-1}). Fig. 3.5 provides an overview of the data flow through an LSTM cell. Updates to the cell state are controlled by gates in the cell. The gates are a combination of a transfer function (yellow block) and pointwise operation (red circle). There are three gates present in the LSTM. First, the cell state is updated by the 'forget gate' (f_t), multiplying the cell state by the sigmoid transfer function of the input. As this sigmoid transfer function outputs a range of [0,1], this is called the 'forget gate'. Second, information is added to the cell state, this is called the 'input gate'. Here a sigmoid and tanh transfer functions of the input are multiplied and the result is added to the cell state. Last, the output is determined with the 'output gate'. Here a sigmoid transfer function of the input is multiplied by the cell state processed with a tanh function, determining the output (h_t). Note that there are weights between the input x_t, hidden state h_{t-1} and transfer functions which are updated during the training process.

The GRU cell proposed by Cho et al. (2014) is essentially a simplified LSTM cell. Combining the forget and input gate into a single 'update gate'. The cell state and hidden state are also combined, into just the hidden state. Due to its reduced complexity and the algorithm having less gates to train, training time is reduced compared to the LSTM cell. Fig. 3.6 provides an overview of the data flow through a GRU cell.

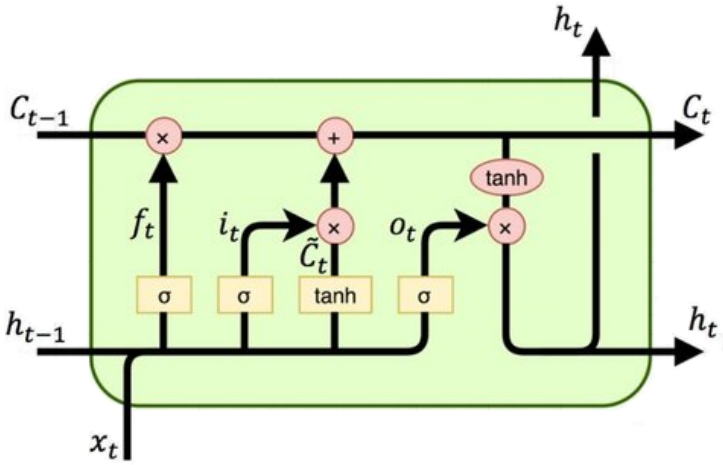


Figure 3.5: Illustration of a LSTM cell (Christopher Olah, 2015). The yellow blocks indicate transfer functions. The red circles indicate pointwise operations. x_t is the input, h_t and h_{t-1} the output of the current and previous timestep respectively and C_t and C_{t-1} the cell state for the current and previous timestep respectively. f_t, i_t, $\tilde{C}_t$, and o_t are the mathematical formulations of the transfer function using the input, weights, transfer function and bias.

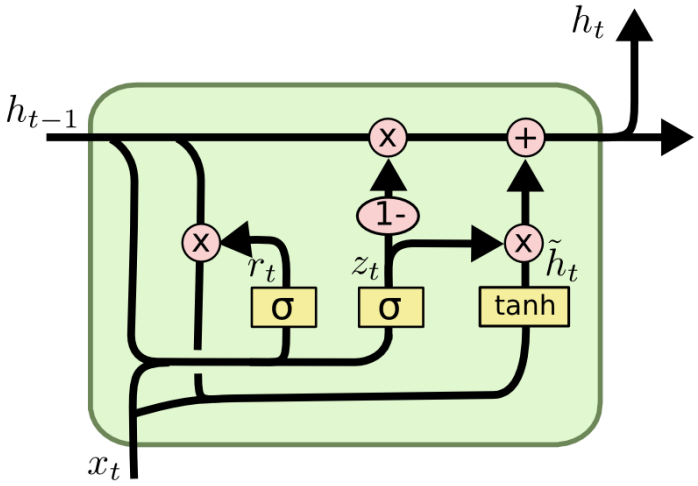


Figure 3.6: Illustration of a GRU cell (Christopher Olah, 2015). The yellow blocks indicate transfer functions. The red circles indicate pointwise operations. x_t is the input, h_t and h_{t-1} the output of the current and previous timestep respectively which is also the hidden state of the cell. r_t, z_t and $\tilde{h}_t$ are the mathematical formulations of the transfer function using the input, weights, transfer function and bias.

4 Methodology

The methodology used in each major step shown in Fig. 4.1 will be detailed in this chapter. First a synthetic precipitation data set is constructed using precipitation statistics. Then, this data set is used as input for a numerical sewer model and the model results are analysed. After the input and output data sets are obtained, three distinct ML algorithms are constructed. First, a MLP is constructed that classifies if flooding occurs at each manhole given a precipitation event. Note that only the precipitation features used for construction of the precipitation time series are used here. Second, a MLP is constructed that predicts maximum flood volumes at each manhole, using the same precipitation event features as the classifier. Last, an LSTM algorithm is constructed which is able to predict flood volume time series for all manholes in the area, given a precipitation time series. The hyper-parameter configurations of all three ML is optimised using random search and Bayesian optimisation. After construction, hyper-parameter optimisation and training the ML algorithms are validated using the validation data set to determine final performance of the algorithms. Furthermore, the LSTM is tested on historic extreme precipitation events that caused flooding in the study area. Only the LSTM is tested on the historic data as it was would be very cumbersome to extract the precipitation features used by the MLP algorithms from the historic data. Furthermore, extraction of precipitation features from flood volume time series introduces another layer of abstraction, which is undesired.

4.1 Extreme precipitation time series

4.1.1 Precipitation statistics

The Stichting Toegepast Onderzoek Waterbeheer (STOWA) is the central knowledge centre for water related research in the Netherlands. The STOWA has a yearly publication that is used as reference for precipitation events in the Netherlands. The latest publication from STOWA by Beersma et al. (2019), details the precipitation statistics and patterns for long and short term events. Due to the inherent early warning system that is proposed in the present research, the short term events are studied. Beersma et al. (2019) recommend precipitation durations of 4, 8 and 12 hours for short term events. The precipitation intensity curves are detailed by Beersma et al. (2019). Fig. 4.2 shows the curves for a return period of 2 to 1×10^3 years. The minimum and maximum precipitation intensity for these return periods and durations is 28 mm and 139 mm respectively.

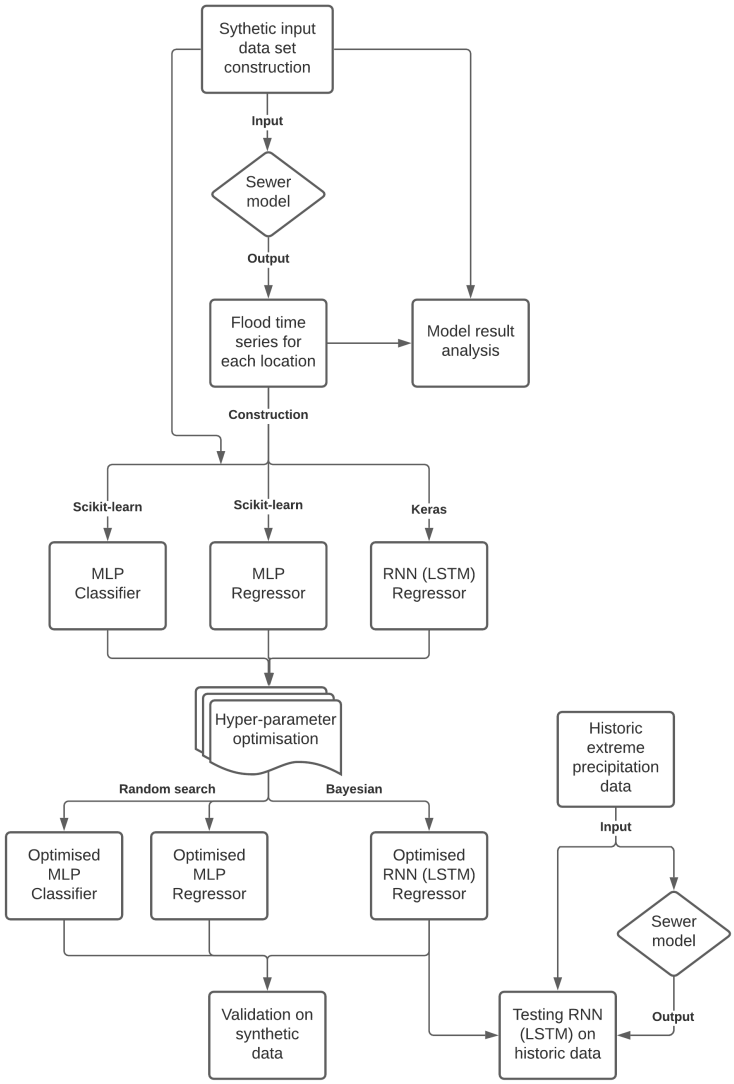


Figure 4.1: Flow chart showing the steps taken in the present research. Where the first step, is the construction of the synthetic data set and the last steps, ML algorithm validation on the synthetic data and testing of the LSTM on historic data.

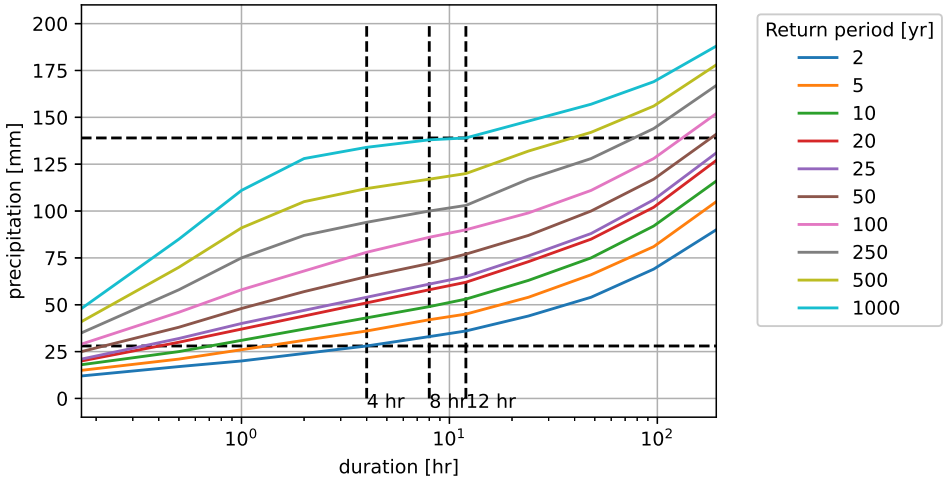


Figure 4.2: Precipitation intensity curves, the dashed black lines indicate maximum and minimum for the 4, 8 and 12 hour durations.

Precipitation patterns

Beersma et al. (2019) provide precipitation patterns for short term events. They indicate that these patterns are sufficient for the testing of quick-reacting systems such as sewer systems. A slow-reacting system is e.g. a farm field, which has a high infiltration and storage capacity, making it only susceptible to flooding from long term precipitation events. Beersma et al. (2019) provide seven distinct precipitation patterns:

1. Uniform: General uniform shape with minor changes between timesteps.
2. 1 peak - 12.5%: Pattern with one peak that has 12.5% of the total discharge in the peak.
3. 1 peak - 37.5%: Pattern with one peak that has 37.5% of the total discharge in the peak.
4. 1 peak - 62.5%: Pattern with one peak that has 62.5% of the total discharge in the peak.
5. 1 peak - 87.5%: Pattern with one peak that has 87.5% of the total discharge in the peak.
6. 2 peaks - short distance: Pattern with two peaks that has a small temporal distance between the two peaks.
7. 2 peaks - large distance: Pattern with two peaks that has a large temporal distance between the two peaks.

The patterns all provide a fraction of the total precipitation per hour. Tab. B.1, B.2 and B.3 (appendix. B) provide the used patterns for the different durations. All 7 precipitation patterns, for a duration of 8 hours, are shown in Fig. 4.3.

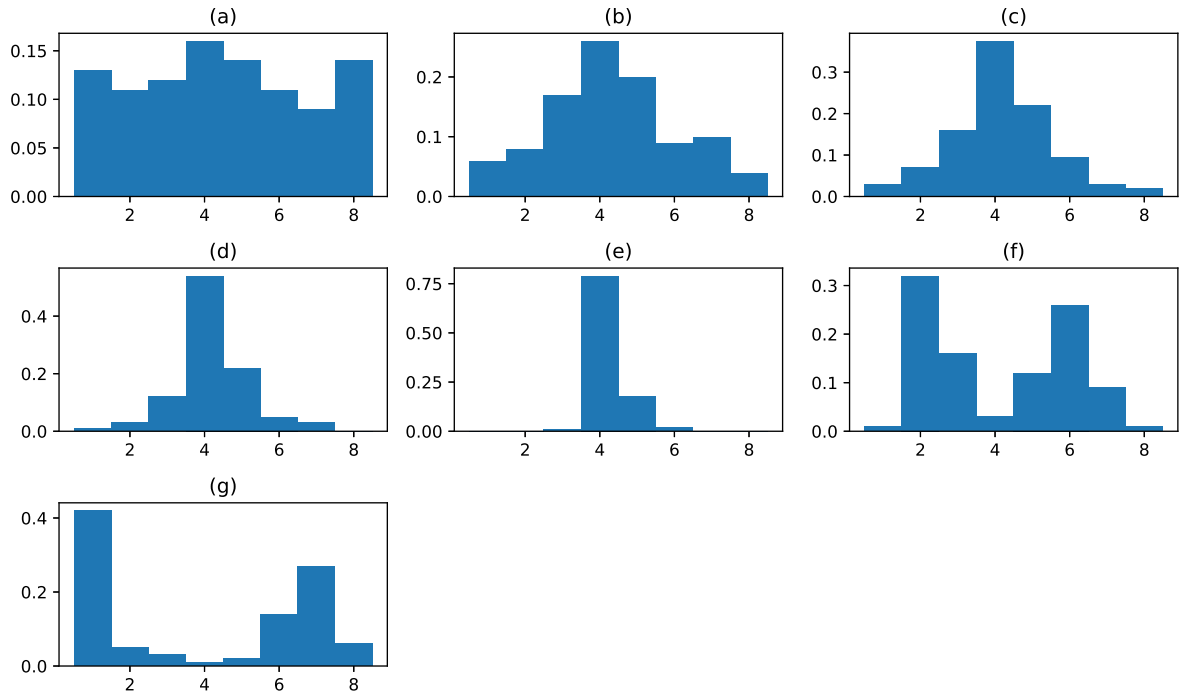


Figure 4.3: All 7 precipitation patterns for a duration of 8 hours, with (a) Uniform, (b) 1 peak - 12.5%, (c) 1 peak - 37.5%, (d) 1 peak - 62.5%, 1 peak - 87.5%, (e) 2 peaks - short and (f) 2 peaks - long. The x-axis represents the time in hours and the y-axis the fraction of total mm precipitation.

4.1.2 Generation of time series and combinations

For the construction of the synthetic precipitation data set, the three features provided by the precipitation statistics are used (precipitation intensity, precipitation pattern and precipitation duration). For this purpose, combinations are made between the features to generate unique precipitation events. The range of precipitation intensity is divided into 6 values with a minimum and maximum of 30 and 105 mm respectively. The minimum value is taken as the rounded minimum value given by the precipitation

intensity curves. To reduce the distance between the 6 values for the precipitation intensity, a lower maximum value is chosen, of 105 mm. In initial sewer model runs we found that the difference in output between 100 mm and 139 mm is minimal in complexity, only the volume that floods the street increases. Therefore, a lower maximum is chosen to create more values in the lower end of the range.

Constraints in provided model runs by external parties limits the amount of precipitation events that can be used as input. With 6 precipitation intensity values, the total amount of combinations is:

$$\text{combinations} = P_{intensities} * P_{patterns} * P_{duration} = 126$$

where;

$P_{events} = 6$, the amount of precipitation intensities

$P_{patterns} = 7$, the amount of precipitation patterns

$P_{duration} = 3$, the amount of durations studied

All possible values of each precipitation feature are shown in Tab. 4.1.

$P_{intensities}$	$P_{patterns}$	$P_{duration}$
30 mm	Uniform	4 hr
45 mm	1 peak - 12.5%	8 hr
60 mm	1 peak - 37.5%	12 hr
75 mm	1 peak - 62.5%	
90 mm	1 peak - 87.5%	
105 mm	2 peaks - short	
	2 peaks - long	

Table 4.1: All possible values for each precipitation event feature.

The majority of papers reviewed by Rajaee et al. (2019) use a data set size of 100 to 200 samples to train the ANNs. The 126 precipitation events should therefore be sufficient for training, testing and validation of the ANNs.

These precipitation events are used as input for the sewer model. The subsequent model results in combination with the input are used to train the ML algorithms. For the MLP, the three precipitation event features (precipitation intensity, precipitation pattern and precipitation duration) are used as features for the prediction of maximum flood volumes and flood classification (Tab. 4.3 shows how these features are used as input).

Interpolation of precipitation patterns

The precipitation patterns provided by Beersma et al. (2019) have a timestep of one hour. These patterns are interpolated to make precipitation time series with a timestep of one minute. This is done to accommodate the model which uses a timestep of one minute. A linear interpolation is chosen for a more realistic precipitation event. Furthermore, to facilitate the operationality of such a flood early warning system, the input time series is made to mimic a conventional precipitation forecast. For short term precipitation forecasts, a timestep of 5 minutes is used, as indicated by expert opinion from a meteorologist at Weather Impact. Therefore, the input time series will be a cascading precipitation pattern with a timestep of 1 minute which changes its value after every 5 minutes. This cascading pattern aggregates the linearly interpolated data to 5 minutes and produces a mean for each window.

Interpolation of an example precipitation pattern given in Fig. 4.3 is shown in Fig. 4.4.

Due to interpolation, the total precipitation is between $0-2\%$ lower than the given value. This is caused by losses in the interpolation process.

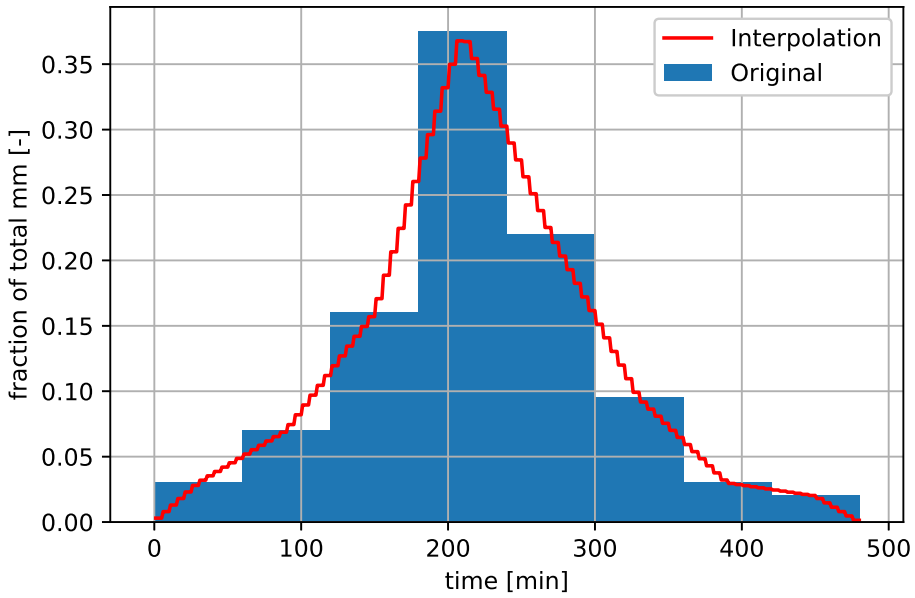


Figure 4.4: Example interpolation of a 8 hour precipitation pattern with a peak of 37.5% of the total precipitation.

4.2 Numerical sewer model

The numerical sewer model used to obtain inundation results is a validated model built with Infoworks Integrated Catchment Modelling (ICM). The sewer model is capable of physics based 1D and 2D hydraulic simulations, with all relevant sewer properties and structures included in the sewer model. The sewer model used is a 1D model of the minor system. The sewer model only looks at the area of the major system and the shortest flow path to the nearest inlet. No topographic gradients are included in the sewer model. Henonin et al. (2013) further details the modelling of such a 1D model. The model uses the Shallow water equations (SWE), also known as the depth integrated Navier-Stokes equations, to solve 1D flow in the minor system.

Fig. 4.5 provides an overview of the sewer piping levels and important structures included in the model, see appendix A for further detail. It can be seen that the sewer piping has a slope from the south-east to north-west. The general direction of the sewer flow will follow the same direction as this is a gravity based sewer system.

There are several clusters of structures that can be determined from the structures overview. In the northwest of the area the largest cluster of overflows and pumps is located. This is the downstream area of the Hooglanderveen sewer system. The pump with the highest capacity is also located here. The pump has a capacity of 111 m^3/hr. In the south-west of the area two overflows and an upstream pump is located. The overflows here transport sewer water to the surface water located nearby. The remaining pumps in the area are used to accommodate flow towards the north-west and south-west of the area.

The output flood volume time series are obtained from the sewer model, using the synthetic precipitation events as input. The input extreme precipitation events are synthetically constructed and thus are independent of each other. Therefore, the sewer model is reset to its initial conditions for each synthetic precipitation event. The model results are provided as flood volume time series per manhole. To obtain further insight into the sewer model, the sewer model results are analysed. Here we look at the spatial correlation between manholes and conduct a sensitivity analysis of the precipitation features.

4.2.1 Modelling of flood volumes

Flow through a pipe in the sewer network can be approximated by the shallow water equations for 1D flow. These are implemented in the Infoworks ICM sewer model. To determine flood volumes at each manhole location Eq. 4.1 is used. If the hydraulic head (H) exceeds the manhole cover level (h) sewage

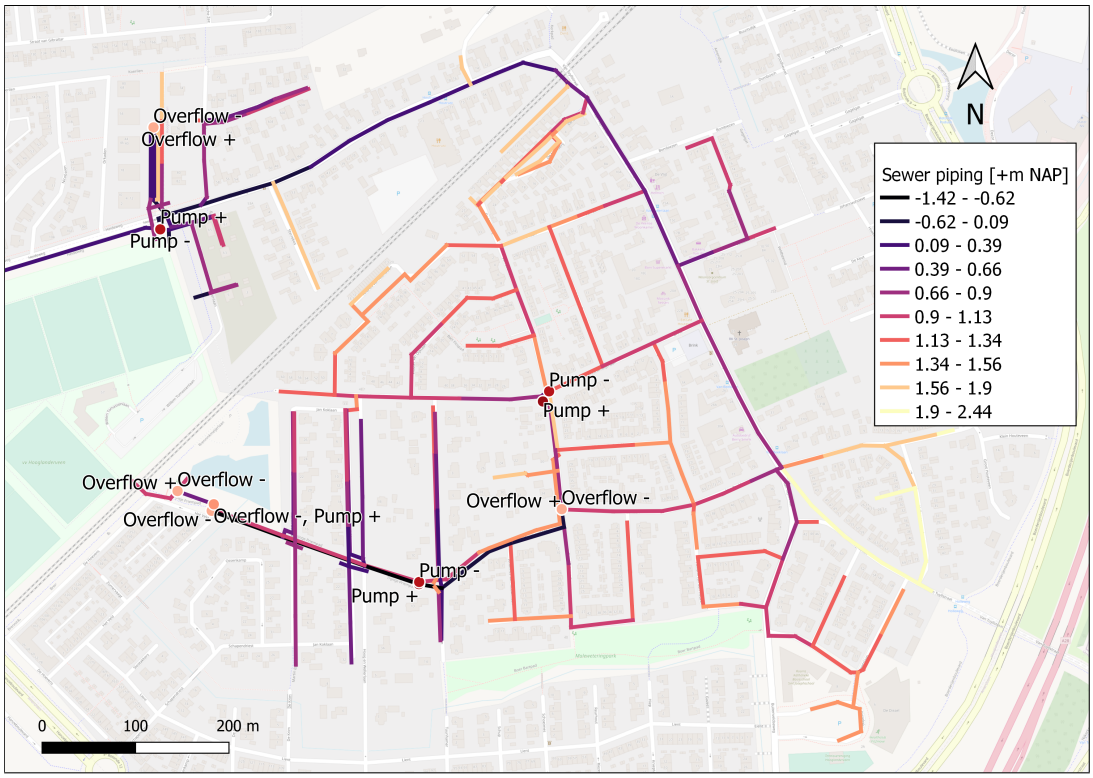


Figure 4.5: Important structures in the area and the level of sewer piping. The plus and minus signs indicate downstream and upstream nodes respectively.

will flood the street. The volume is calculated with the surface area (A) of the bounding box. Fig. 4.6 shows a schematised figure of this bounding box. In a 1D-hydrodynamic flow model, water that floods from the manhole is contained in this box and cannot flow along topographic gradients. Recent developments add a 2D terrain to the sewer model, allowing for surface water runoff and interaction between the sewer system and surface level flow paths. This 2D approach is not used in the context of this research.

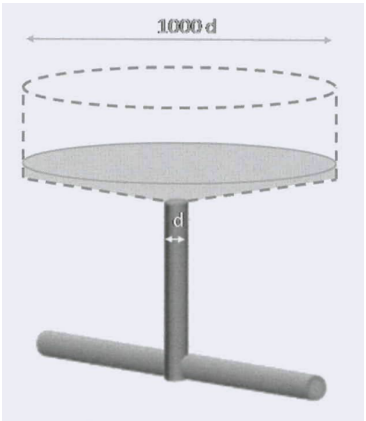


Figure 4.6: Visualisation of the bounding box on top of a manhole that holds the storm water surcharge (Fig. 2 (Henonin et al., 2013)).

The hydraulic head is computed using Eq. 4.2. Fig. 4.7 shows a schematised figure with parameters used in Eq. 4.1. Note that the reference level is set at the manhole cover level to provide negative flood volumes if the hydraulic head is below the level of the manhole cover. Therefore, in this case the ground level of the manhole cover is 0 m. The negative flood volume indicates the available storage at each manhole.

$$V_{flood} = (H - h) \cdot A \tag{4.1}$$

where;

V_{flood} = flood volume $[m^3]$

H = hydraulic head $[m]$

h = ground level of the manhole cover $[m]$

A = surface area of the bounding box $[m^2]$

$$H = \frac{p}{\rho g} + \frac{u^2}{2g} + z \tag{4.2}$$

where;

H = hydraulic head $[m]$

p = fluid pressure $[Pa]$

ρ = density of the fluid $[kg/m^3]$

g = gravity acceleration $[m/s^2]$

u = fluid velocity $[m/s]$

z = fluid elevation above a reference level $[m]$

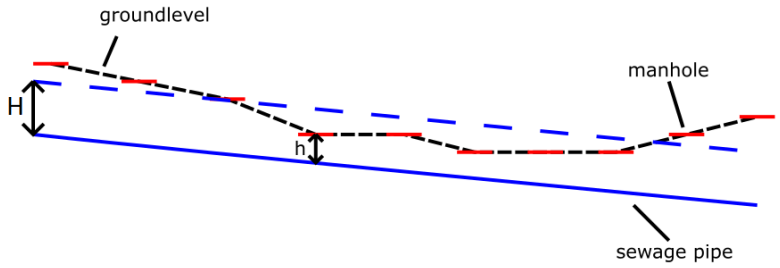


Figure 4.7: Schematised alignment of a street with manholes and an underground sewer pipe. The red lines represent manholes on the street alignment. The blue line is the sewage pipe, with the blue dashed line the equilibrium water level. The black dashed line is the ground level and schematised street alignment.

4.2.2 Spatial correlogram

The flood volumes at each manhole are expected to have a large spatial correlation, because manholes in a street close to each other will have equivalent flood volume time series. To inspect this property, flood volume model results of a 105 mm precipitation event with 87% of the total volume in the peak are taken. Distances between all manholes are calculated. This gives 52.9×10^3 total combinations of manholes with subsequent distances. The correlation between the flood volume time-series of each combination is determined and plotted against the distance in Fig. 4.8. The values are binned and shown as box plots.

We can see a high mean spatial correlation at 0 m to 50 m inter-location distance of approximately 0.97. The spatial correlation decreases as the inter-location distance increases. From 600 m inter-location distance, the mean correlation increases significantly. The distance is becoming large here with few samples. Therefore, no conclusions can be made from this increase in correlation. The spatial correlogram supports the hypothesis that there is a strong spatial correlation in flood time-series. Note that the inter-location distance is measured as the crow flies, while the true distance in the sewer network might be larger. This could influence the spatial correlation results, decreasing correlation for low inter-location distances.

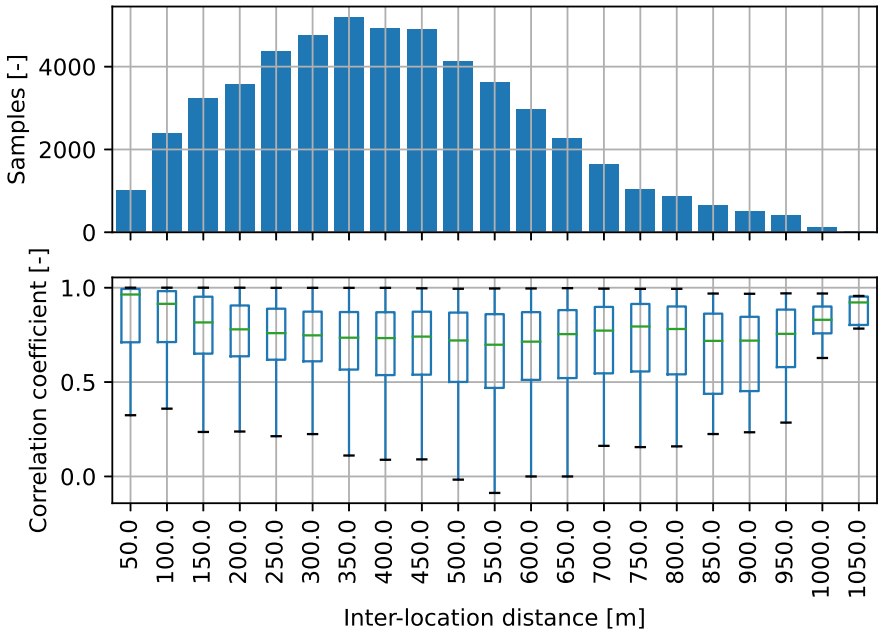


Figure 4.8: Spatial correlogram of flood volumes at manholes for a precipitation event of 105 mm *with* 87.5% *of the total volume in the peak. Values have been binned per* 50 m *inter-location distance. A box plot has been used to show the spread of data for each bin. The green line represents the mean correlation, the edges of the box represent the 25th and 75th percentile and the black lines the maximum and minimum.*

To further support this conclusion the spatial correlogram is plotted for four different precipitation events, see Fig. 4.9. It can be seen that the correlation in all four precipitation events decreases as the distance between manholes increases, with a near identical pattern as shown in Fig. 4.8.

4.2.3 Sensitivity analysis

For the MLP algorithm, the precipitation events are characterised by three features: precipitation duration, pattern and intensity. Rajaee et al. (2019) advises further analysis of input features before use, this is needed to determine if each feature is important for the predicted output. This can be done using a sensitivity analysis. If a feature has a low effect on the output, this feature can be eliminated from the input features used for the MLPs.

Fig. 4.10a, 4.10b and 4.10c show the spread of data for the precipitation duration, pattern and intensity respectively. The spread of the mean maximum flood volume for all locations is plotted with a boxplot. This is done for each value the features can take, as they are not continuous.

The sensitivity analysis shows that all three features influence the output flood volumes substantially. The precipitation intensity and duration provide a linear increase and decrease in the spread of data respectively as the value of the feature increases. Note that the total precipitation remains constant as the precipitation duration increases. The same total precipitation in *mm* will fall during a larger time span. We can see that the spread and mean (green line) of the data increase with the precipitation intensity. The sensitivity analysis for the precipitation pattern shows similar results for the change in peak precipitation intensity for precipitation patterns with 1 peak, with a linear increase in the spread and mean maximum flood volume. The 2 peak and uniform precipitation patterns both have a low spread of data. Concluding all three features have proven important for use in the MLP algorithms.

4.3 Data pre-processing

Before the synthetic precipitation data can be used in the ML algorithms, the input features need to be processed by normalisation and one-hot encoding. Data pre-processing of the output data is not required.

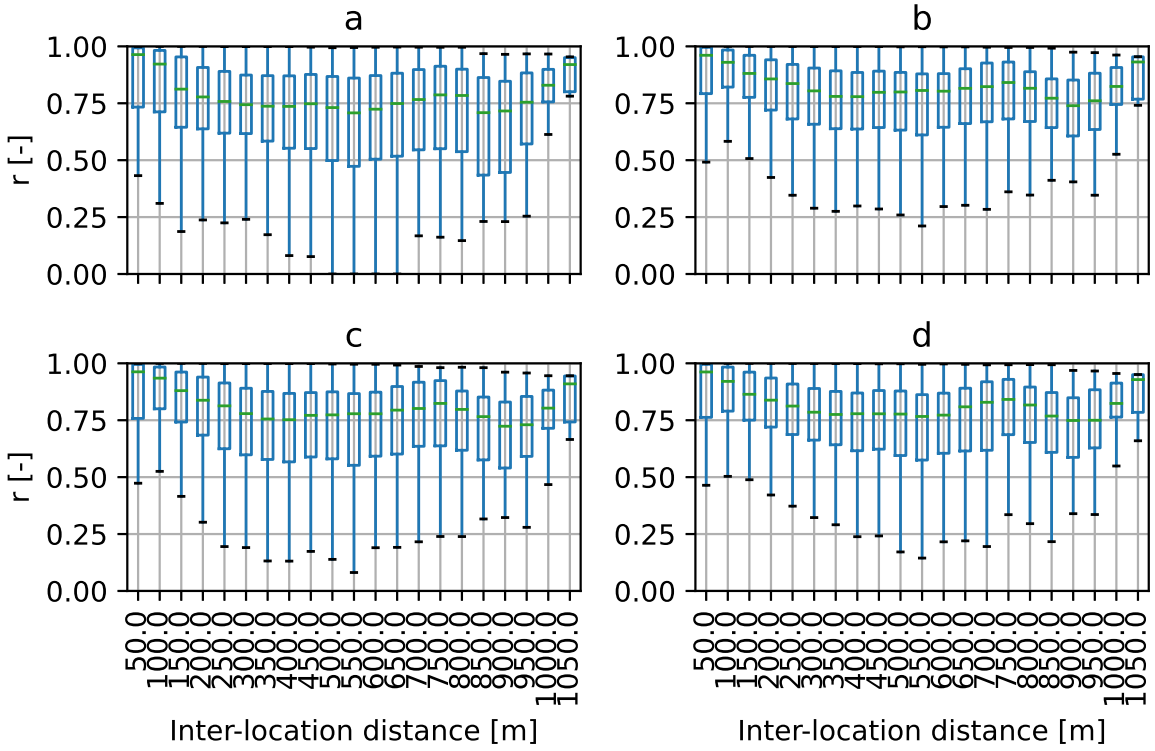


Figure 4.9: Spatial correlogram for 4 different precipitation events, with a = 4 hours with 1 peak of 87.5% *and* 100 mm *total precipitation, b = 8 hours with 1 peak of* 37.5% *and* 60 mm *total precipitation, c = 12 hours with 2 peaks with a long intermission and* 90 mm *total precipitation, d = 12 hours with 1 peak of* 62% *and* 75 mm. *Note values have been binned per* 50 m *inter-location distance. The amount of samples used in each bin is equivalent to Fig. 4.8.*

4.3.1 Normalisation

It has been proven that ANNs work best on data that is normalised (Dawson and Wilby, 2001). Therefore, it is important to process the data before training the ANN on it. The main component of data processing is normalisation of the data. With different ranges of values for each feature, the ANN tends to favour features that can have larger values (Dawson and Wilby, 2001). It is recommended to normalise the data to a $[0, 1]$ interval. Normalisation is done using Eq. 4.3. Note that normalisation can only be applied to ordinal variables, which have a natural numerical relationship. For example the precipitation intensity feature has a natural numerical relationship.

$$x'_i = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{4.3}$$

where;

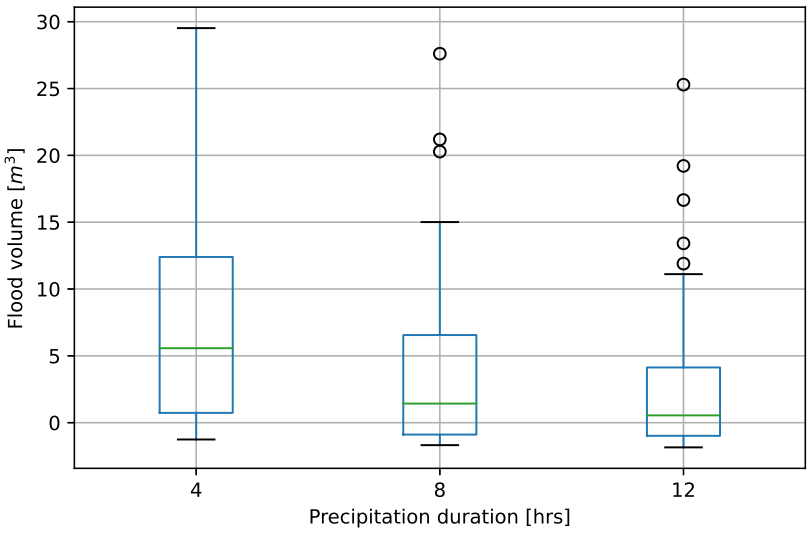
x'_i is the normalised value $(i = 1...n)$

x_{max} the maximum value of the feature

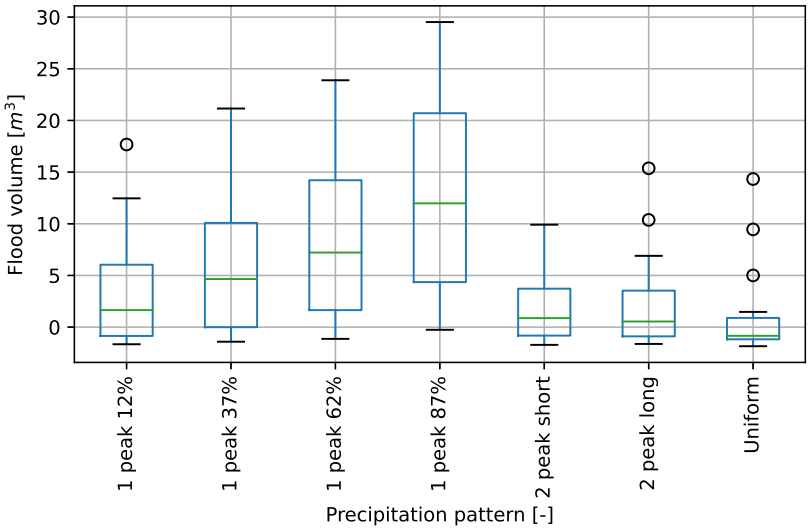
x_{min} the minimum value of the feature

4.3.2 One-hot encoding

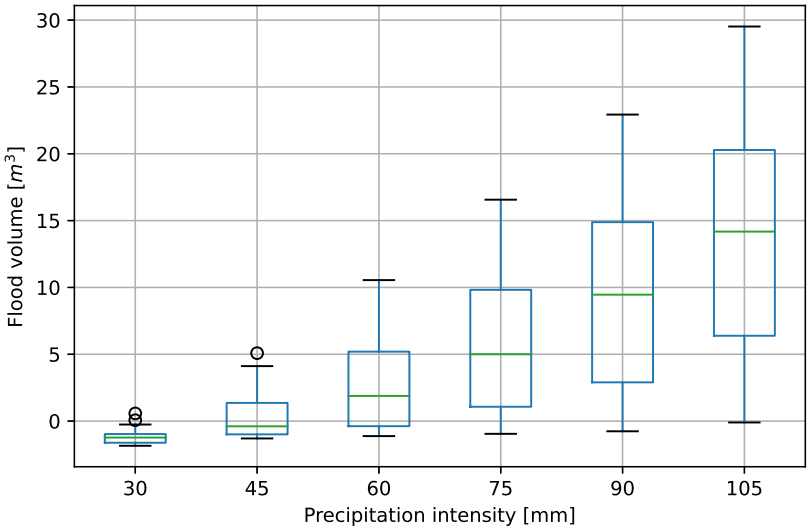
Compared to numerically discrete features, categorical features should be prepared differently. With categorical features, the values do not have a relation to each other. In the present research, the categorical feature is the precipitation pattern detailed in paragraph 4.1.1. With this feature, pattern seven is not seven times larger than pattern one, because they are not numerically related. Therefore, we cannot use a single feature to describe these patterns. In ML, a commonly used technique to tackle this is called



(a)



(b)



(c)

Figure 4.10: Box plots showing spread of data for (a) the precipitation duration, (b) pattern and (c) intensity. The mean of maximum flood volumes is taken over all manhole locations for each precipitation event. Negative values indicate storage left in the sewer system at a specific manhole. The samples used for each boxplot are in figures a, b and c are 42, 18 and 21 respectively.

one-hot encoding (Jason Brownlee, 2020). With one-hot encoding a feature is split into n features for all values the feature can take, where each feature can take a value of 0 or 1. For precipitation patterns, this one feature is split into seven features. The value is 1 if the specific pattern is present and 0 if not present.

Tab. 4.2 shows several examples of events before they are normalised and one-hot encoded. In Tab. 4.3, the normalisation is done for the duration and precipitation feature. Furthermore, one-hot encoding is applied to the precipitation pattern feature. From initial trial training and validation runs with the MLP algorithm, one-hot encoding improved results for classification and regression.

Event	Duration [hr]	Precipitation [mm]	Pattern [-]
1	4	105	1
2	8	30	3
3	4	75	5
4	12	105	2
5	4	45	7

Table 4.2: Example of precipitation event features before normalisation and one-hot encoding.

Event	Duration [hr]	Precipitation [mm]	Pattern [-]	1	2	3	4	5	6	7
1	0	1		1	0	0	0	0	0	0
2	0.5	0		0	0	1	0	0	0	0
3	0	0.6		0	0	0	0	1	0	0
4	1	1		0	1	0	0	0	0	0
5	0	0.2		0	0	0	0	0	0	1

Table 4.3: Example of precipitation event features after normalisation and one-hot encoding.

4.4 Flow and transformation of data

The full flow and transformation of data for the MLP and RNN (LSTM) can be seen in Fig. 4.11 and 4.12 respectively. The data is split into training, test and validation data sets according to the average of studies studied by Rajaee et al. (2019). They found that 60%, 18% and 22% of the total data was used for training, testing and validation respectively. In the present study, a split of 60%, 20% and 20% is used for simplification. The training set will be used to train the ML algorithms. The test set is used to test the ML algorithms and calibrate hyper-parameters. The validation set is only used to evaluate the ML algorithm final performance.

For the prediction of maximum flood volumes and flood classification with the MLP, the three features used for construction of the flood volume time series are used (precipitation intensity, precipitation duration and precipitation pattern) after normalisation and one-hot encoding. For the RNN (LSTM), the input precipitation time series are used after normalisation (see chapter 4.3.1). The input and output of the MLP algorithm has a 2D dimension. For the RNN (LSTM) algorithm the input and output are shaped into 3D matrices. The first, second and third dimension are the samples, time steps and features respectively.

4.5 Construction of the machine learning algorithms

The construction of the MLP and RNN algorithms is done using Python with the packages scikit-learn and Keras. Scikit-learn is used for the construction of the MLP regressor and classifier. Keras is used for the construction of the LSTM network. Values for the hyper-parameters are determined using hyper-parameter optimisation.

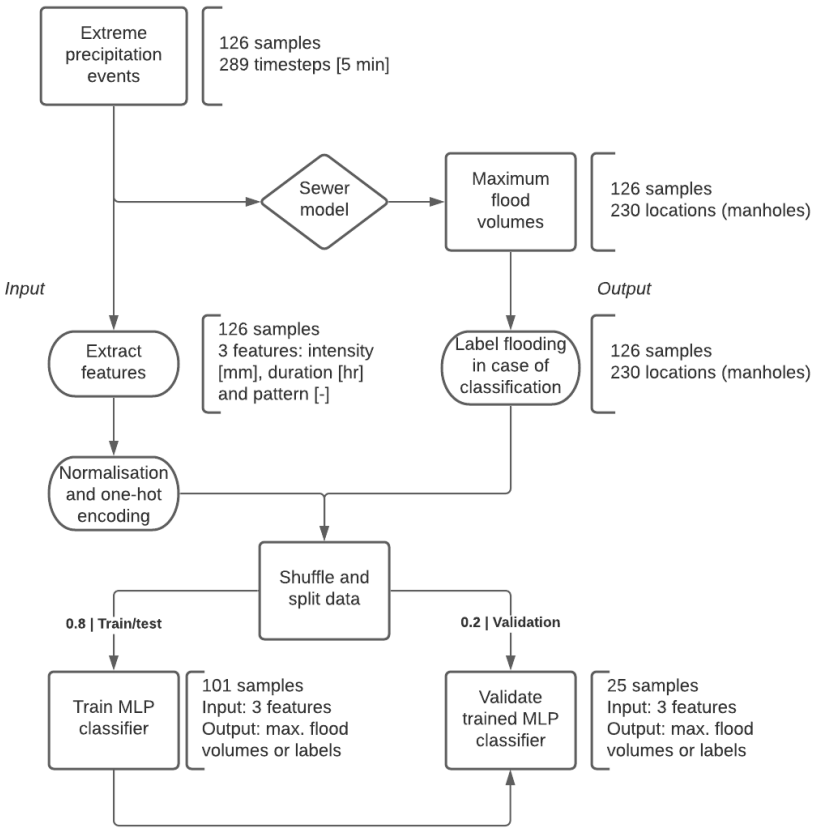


Figure 4.11: Flow and transformation of data for training and validation of the MLP classifier and regressor. Maximum flood volumes are labelled 'no flood' or 'flood' (0 or 1) only for the classification MLP.

4.5.1 Scikit-learn (MLP)

For the classifier and regressor, the functions MLPClassifier[1] and MLPRegressor[2] from scikit-learn are used. Both functions have the same parameters. However, as the former outputs a class and the latter a real-valued flood volume, the algorithms use different activation and loss functions. The activation functions used are detailed in chapter 3, these are the softmax and linear function for the MLPclassifier and MLPregressor respectively. The loss function for the MLPClassifier and MLPRegressor are the cross-entropy and MAE respectively. The script in Listing C.1 shows a basic setup of an MLPRegressor. The hidden layer sizes defines the number of hidden layers and neurons in each hidden layer.

4.5.2 Keras RNN (LSTM)

Keras uses a different framework than scikit-learn and does not use a single Python function for the construction of a model. With Keras we create a 'sequential' model. This means that we create a sequence of layers that the data passes through. This sequential model works like any ANN where weights connect the input feature to transfer functions which then passes the data to the next layer via weights and so on. The script in Listing C.2 shows an example of an LSTM sequential model built in Keras.

The LSTM layer is added as the first layer in the network. Note that the input shape specifies the time steps and features of the input. No transfer function is specified for the LSTM layer. This is done due to time constraints in the hyper-parameter optimisation. To use the Graphics processing unit (GPU) for training of the LSTM, a specified set of hyper-parameters need to be used, else the GPU will not be used and only the Central processing unit (CPU) is utilised. In initial testing of the training process, the training time with the CPU on a large LSTM layer (> 500 units) could be up to 80 min. With hyper-

[1] see: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html
[2] see: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html

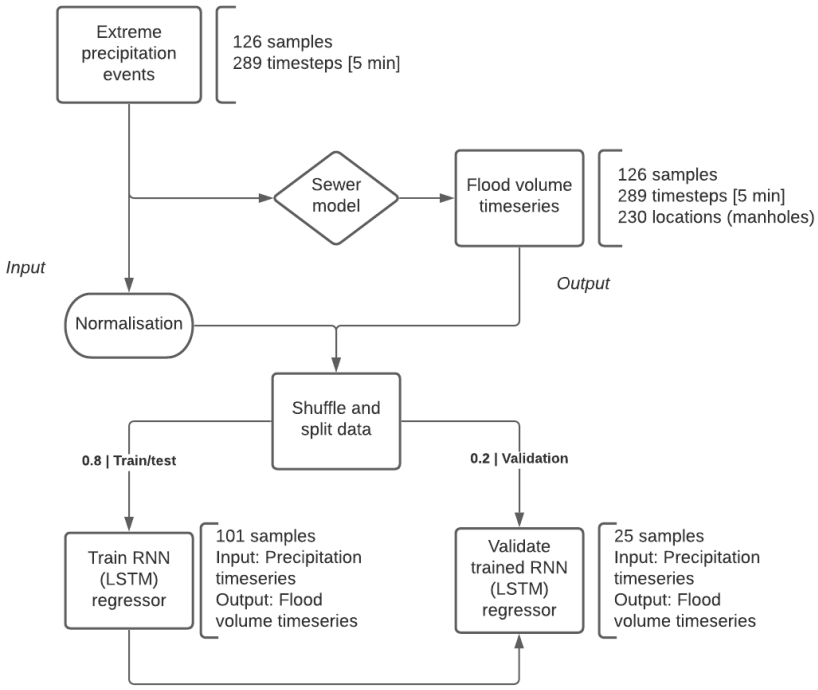


Figure 4.12: Flow and transformation of data for training and validation of the LSTM regressor. Note that data is reshaped into a three dimensional matrix as required by Keras.

parameter optimisation, a model needs to be ran many times to find the best set of hyper-parameters. Therefore, this GPU optimised LSTM layer[3] setup is used. The transfer functions used in the GPU optimised LSTM layer are the tanh and sigmoid functions. This is equivalent to the layout of the LSTM with transfer functions shown in Fig. 3.5.

The third layer is a standard hidden layer with a transfer function, a so called 'Dense' layer in Keras. As this is the last layer in the sequential model the transfer function is the output activation function. The units in the last layer need to be equal to the output dimension of targets. There are 230 manholes in the study area, therefore, the last layer in the sequential model needs 230 units. The activation function of the last layer is a linear function.

The sequential model is compiled with the MAE loss function. The model is fit to the train/test data set. The Keras model can define a test split (called the validation split in keras), therefore the train and test data sets are combined. A test split of 0.25 is set to obtain a division of 0.6 and 0.2 for the train and test data sets respectively. The number of epochs is set to 5×10^2, which defines the number of training iterations. The batch size is set to 10, this defines the amount of samples that are processed simultaneously.

4.6 Hyper-parameter optimisation

Optimisation of hyper-parameters can be done in several ways. The main three used approaches are grid search, random search and Bayesian optimisation. A study by Bergstra and Bengio (2012) has shown that random search is more efficient and better at finding the optimum hyper-parameter configuration than manual search and grid search. Therefore, in the present research only random search and Bayesian optimisation is applied. Random search is applied for the optimisation of the MLP classifier and regressor and Bayesian optimisation is applied for the RNN (LSTM) regressor. Bayesian optimisation was not applied for the MLP due to limitations in scikit-learn with Bayesian optimisation. Here, it was not possible to input a range of possibilities for the number of hidden layers and neurons in each hidden layer. Therefore, random search has been chosen for the MLP algorithms.

[3] see: https://keras.io/api/layers/recurrent_layers/lstm/

4.6.1 Random search

Grid search and random search differ only in the sampling technique used. With grid search, each sample on a pre-defined hyper-parameter space is evaluated. With random search, a pseudo-random sample of hyper-parameter values is evaluated for a defined amount of iterations. Therefore, the grid spacing in the hyper-parameter space can be set significantly smaller than for grid search. The dimensionality and size of the hyper-parameter space can become quite large, making it impossible to evaluate each combination of hyper-parameters with grid search due to time constraints. Fig. 4.13 shows how, using random search, performance for the important hyper-parameter is better determined. It is easier to find high performance of the algorithm on the important hyper-parameter value using random search. To find the highest performance with grid search in Fig. 4.13, the grid should be refined to a smaller interval. However, with the same number of samples, the random search did approximate the hyper-parameter value with the highest performance better.

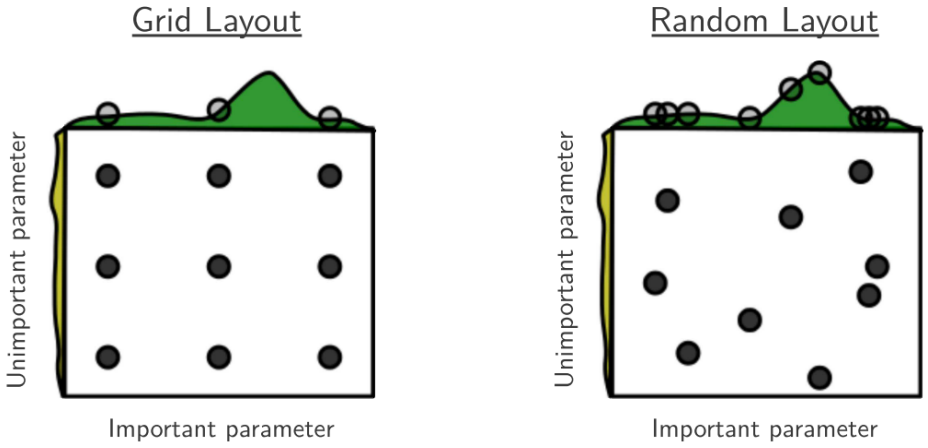


Figure 4.13: Comparison between grid and random search in finding the optimal hyper-parameter configuration. Note that random search is better at sampling the important hyper-parameter. This is caused by the removal of grid spacing limitations.

The full setup for the optimisation of hyper-parameters using random search for the MLP classifier and regressor is detailed in Listing D.1. The scoring is changed to a negative loss function for the regressor. For optimisation the objective measure needs to be maximised, therefore the MAE is made negative.

For the hidden layer and number neurons hyper-parameters a list of possibilities is created. Each value in the list defines the number of hidden layers and neurons in each hidden layer. A maximum of 10 hidden layers is defined, with a maximum of 500 neurons for each layer. Note that there is no variation in the number of neurons between the hidden layers. This is done to limit the size of the hyper-parameter space.

All possible activation functions and learning rate types are included in the hyper-parameter space. The learning rate types define if the learning rate remains constant or changes over the training iterations. The initial learning rate has a minimum of 1×10^{-4} and a maximum of 0.1 with a step size of 1×10^{-4}. The total amount of possible combinations is approximately 6×10^7 and the number of iterations ran is 1×10^5. A cross validation split of 5 is chosen to test each hyper-parameter configuration five times, increasing the total amount of iterations to 5×10^5.

4.6.2 Bayesian optimisation

Bayesian optimisation is a more extensive technique, where Bayesian techniques are used to determine probabilities of a set of hyper parameters being better than previously tested sets. In other words, it finds the probability of a score given a set of hyper-parameters. The probability function built is also called the surrogate function or model (Dewancker et al., 2015). This surrogate is easier to optimise than the underlying objective function. The Bayesian methods thus find the next set of hyper-parameters to test on the objective function from the surrogate. The surrogate is then updated from the results and a new sample set is chosen. This continues until the maximum number of iterations is reached.

Bayesian optimisation is applied for the optimisation of the LSTM. The Python package 'keras-tuner' is used. This package works different from the optimisation in scikit-learn. With the keras-tuner the sequential model is built with Keras and a range of hyper-parameters is given where needed. The full setup for the Bayesian optimisation with the keras-tuner can be seen in Listing D.2.

The range of LSTM neurons has a minimum of 30 and a maximum of 1200, with a step size of 1. The dropout rate has a minimum value of 0.0 and a maximum of 0.2, with a step size of 0.05. The neurons in the last layer are set to a fixed 230 as this is the output layer of the network. The learning rate has a minimum value of 1×10^{-4} and a maximum of 0.01, with a step size of 1×10^{-4}. The learning rate maximum is set significantly lower than the MLP network of 0.1. When the range was set higher than 0.01 the optimisation crashed, as values were going to infinity. Therefore, the maximum of the learning rate range is kept at 0.01. The total amount of combinations is 4.68×10^5. Due to time constraints the maximum iterations for the Bayesian optimisation is set to only 100.

4.7 Validation

Final algorithm performance is determined using the validation data set (20% of the data). Several performance indicators are used for the different ML algorithms. The performance of the MLP classifier is evaluated by the accuracy, calculating the percentage of all classes that are correctly predicted. The performance of the MLP regressor is evaluated with the MAE and R^2. The performance of the LSTM regressor is also determined by the MAE and R^2. However, since the LSTM regressor outputs flood volume time series, the Nash-Sutcliffe efficiency (NSE) is also used to determine the predictive ability of the algorithm. The NSE proposed by (Nash and Sutcliffe, 1970), is a commonly used measure of the predictive ability of hydrological models. The NSE is determined using the 'hydroeval' Python package by Hallouin (2019). For some precipitation events, nodes in the north of the area had NSE approaching negative infinity. Therefore, the bounded version of the NSE, proposed by Mathevet et al. (2006), is applied. NSE values are now bounded to the interval $[-1, 1]$.

4.8 Historic data

For further testing of the LSTM, radar precipitation data from historic extreme precipitation events is obtained. A list of 6 flood events in Hooglanderveen has been provided by the municipality of Amersfoort. Note that the flooding is reported by inhabitants. Flooding can also occur at other locations in the area that has not been reported. As the LSTM is tested on sewer model results the location of the reported flooding does not matter. It is important to note that using the historic data removes one abstraction from reality; the synthetically constructed precipitation data.

The historic precipitation time series are shown in Fig. 4.14. The precipitation time series has been obtained from precipitation radar data provided by Hydrologic. The time series start one day prior to the date reported as there can be a delay between flooding and reporting. All events except event 1 and 4 show large peaks in precipitation up to 106 mm/hr. This is higher than the peak precipitation from the synthetic precipitation time series, which have a maximum precipitation of 88 mm/hr. Events 1 and 4 have relatively lower precipitation peaks of 16 mm/hr and 5 mm/hr respectively. Flooding at these dates could be caused by e.g. blockage of the inlets by leaves.

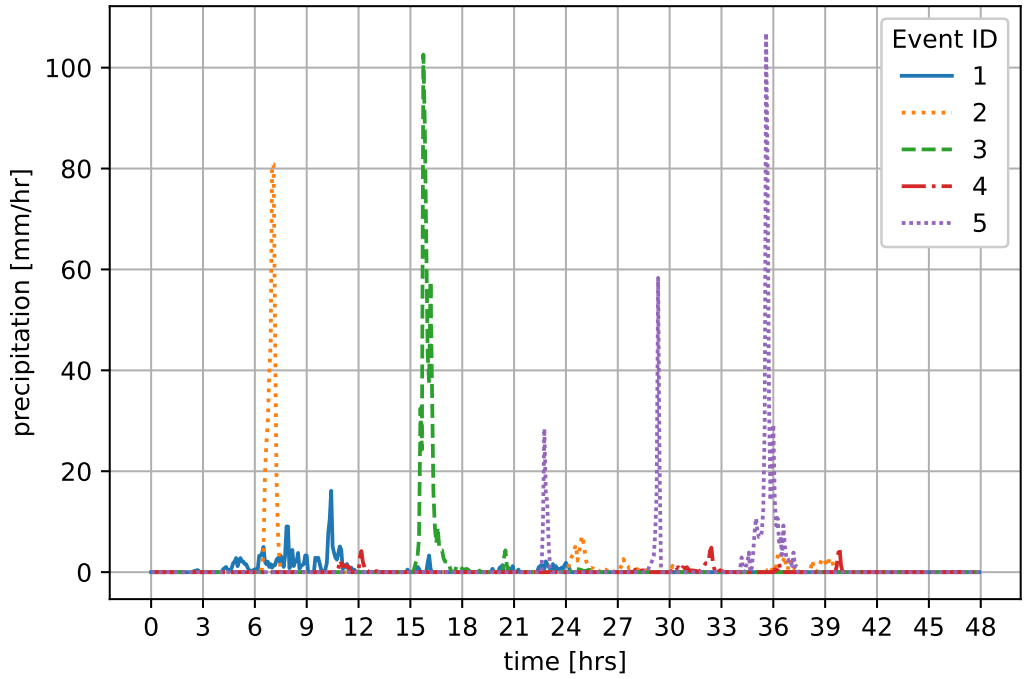


Figure 4.14: Precipitation time series for historic flood events in Hooglanderveen. All time series start one day prior to the reported flooding as there can be a delay in reporting. This can be seen with event 2 and 3.

5 Results

5.1 Machine learning algorithm validation

5.1.1 Multi-layer perceptron classification

After optimisation with random search, an accuracy of 99.29% is achieved on the validation data set. The confusion matrix can be seen in Tab. 5.1. The algorithm had a run time of 1 ms on the validation data set. The hyper-parameters determined using random search, are detailed in Tab. 5.2.

	predicted no flood	predicted flood
actual no flood	4592	16
actual flood	25	1117

Table 5.1: Confusion matrix of the validation data set for the classifier, with hyper-parameters optimised using random search.

Hyper-parameter	Value
Hidden layers	5
Neurons in each hidden layer	152
Initial learning rate	0.0027
Learning rate type	adaptive
Activation function	relu
Metric	Value
Accuracy	99.29%

Table 5.2: Hyper-parameter and evaluation values of the MLP classifier after random search optimisation.

5.1.2 Multi-layer perceptron regression

The random search-optimised MLP regressor reached a MAE of 0.20 m^3 on the validation data set. The hyper-parameters determined using random search, are detailed in Tab. 5.3. The run time of the MLP regressor for evaluation of the validation data set is 0.79 ms. A scatter plot of the predicted and actual results can be seen in Fig. 5.1. Note that negative values are not plotted. Negative values indicate storage left in the sewer system at a specific node. These values are not important for a flood early warning system. Predicted flood volumes follow the actual flood volumes well. The $R^2 = 0.997$ which is very high, showing that the MLP regressor can accurately predict maximum flood volumes for all nodes.

Hyper-parameter	Value
Hidden layers	3
Neurons in each hidden layer	366
Initial learning rate	0.0038
Learning rate type	constant
Activation function	relu
Metric	**Value**
MAE	0.20 m^3
R^2	0.997

Table 5.3: Hyper-parameter and evaluation values of the MLP regressor after random search optimisation.

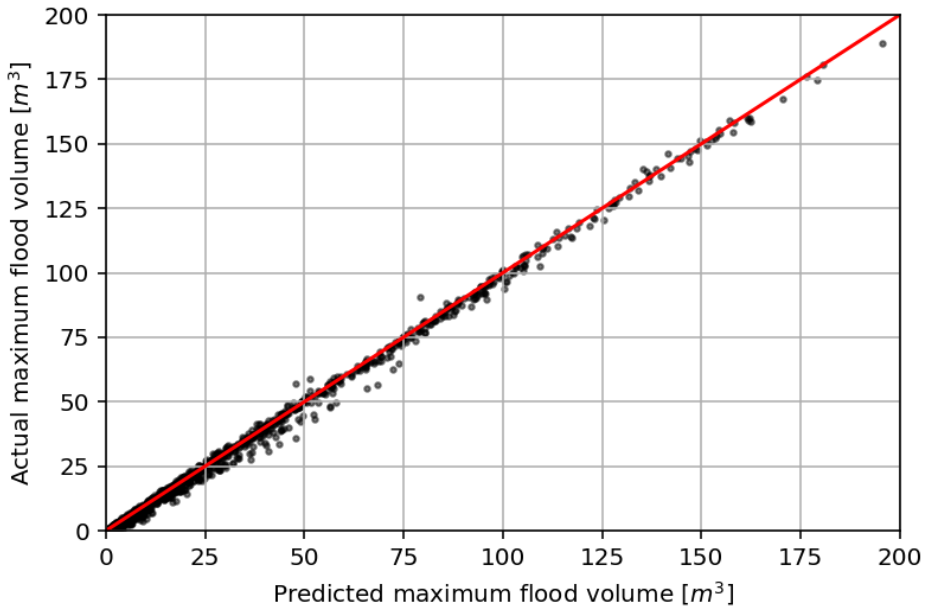


Figure 5.1: Scatter plot of random search optimised MLP regressor evaluated on the validation data set (R^2 = 0.997). Negative flood volumes are not plotted, as these are not of importance for a flood early warning system.

5.1.3 Recurrent neural network (LSTM) regression

The LSTM after optimisation using Bayesian optimisation, has a MAE on the validation data set of 0.0621 m^3. The run time of the LSTM algorithm on the validation data set (25 samples) was 1.89 s. The hyper-parameters determined using Bayesian optimisation, are shown in Tab. 5.4.

Hyper-parameter	Value
LSTM units	636
Learning rate	0.01
Dropout	0.00
Metric	**Value**
MAE	0.06 m^3
NSE	0.87
R^2	0.99

Table 5.4: Hyper-parameter and evaluation values of the LSTM sequential model after Bayesian optimisation.

The NSE of the Bayesian optimised LSTM can be seen in Tab. 5.4. This is the mean NSE of all nodes and time series. Fig. 5.2 shows the fraction of nodes that have a higher NSE than the value on the x-axis. NSE values are very high for the majority of nodes in the studied area. With few nodes providing low NSE values.

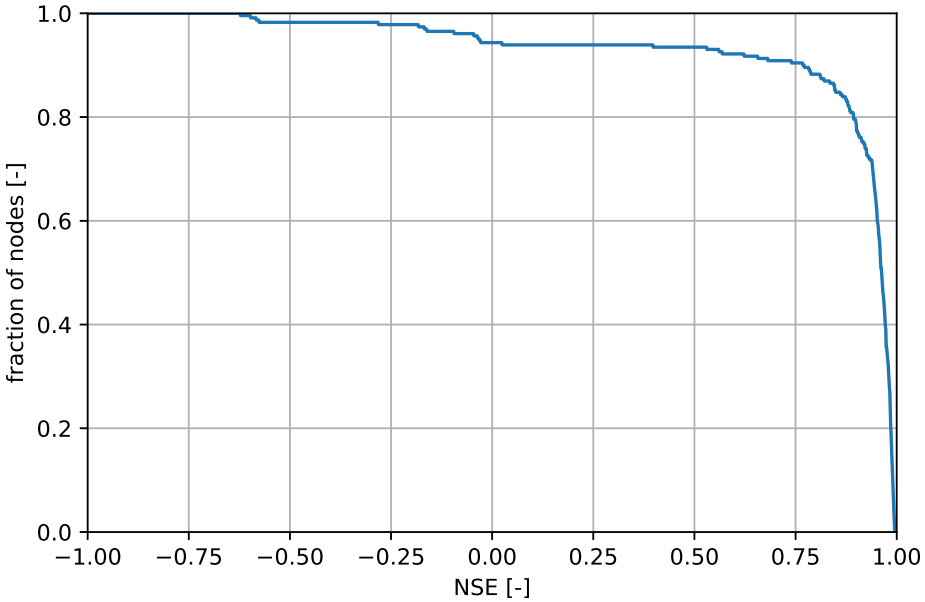


Figure 5.2: Fraction of nodes, for the LSTM regressor prediction on the validation input precipitation data set, with a greater NSE than the value on the x-axis.

Looking at a node with a high NSE value, a good fit can be seen, see Fig. 5.3. This node is located in the centre of the area. In this area flooding occurs at most nodes. The LSTM algorithm underestimates the peak flood volume slightly. This can be seen in many nodes that have a large peak flood volume. A scatter plot of the predicted values set against the actual values, shows this tendency of underestimating, see Fig. 5.4.

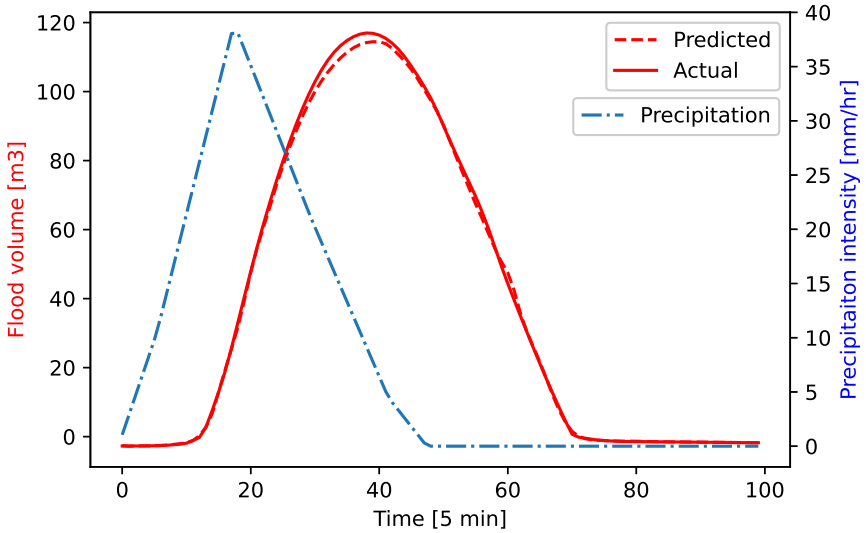


Figure 5.3: Flood volume time series, for the LSTM network validated on synthetic data, at a node in the centre of the area (node 110072*, NSE* = 0.94*).*

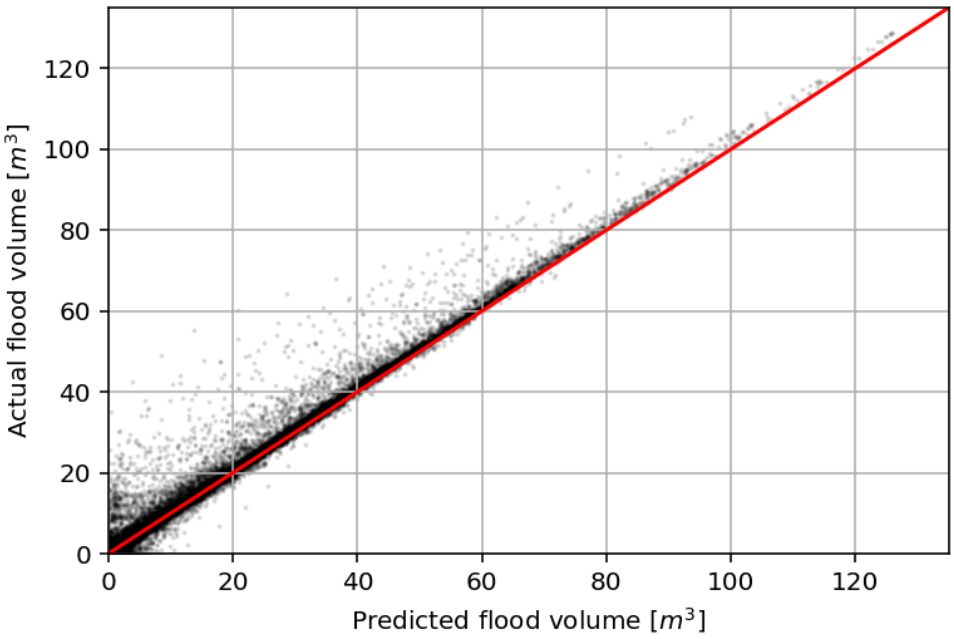


Figure 5.4: Scatter plot of the predicted and actual flood volumes for the LSTM regressor ($R^2 = 0.99$). Negative flood volume are not plotted, as they are not of importance for a flood early warning system.

To further investigate the location of the NSE values, a map with the values for each node in the studied area is made, see Fig. 5.5. It can be seen that the nodes with a very low negative NSE (dark blue nodes) value are clustered together. It is unclear why a low goodness-of-fit is observed in this street. Fig. 5.6 shows a flood volume time series of a node in this cluster. For these nodes constant negative flood volumes are observed in the sewer model results. The predicted flood volumes from the sewer model stay negative and constant, no flooding occurs here in the validation data set. This behaviour is observed for all nodes in the street. Therefore, this area is not of importance for the flood early warning system and the negative NSE values can be neglected.

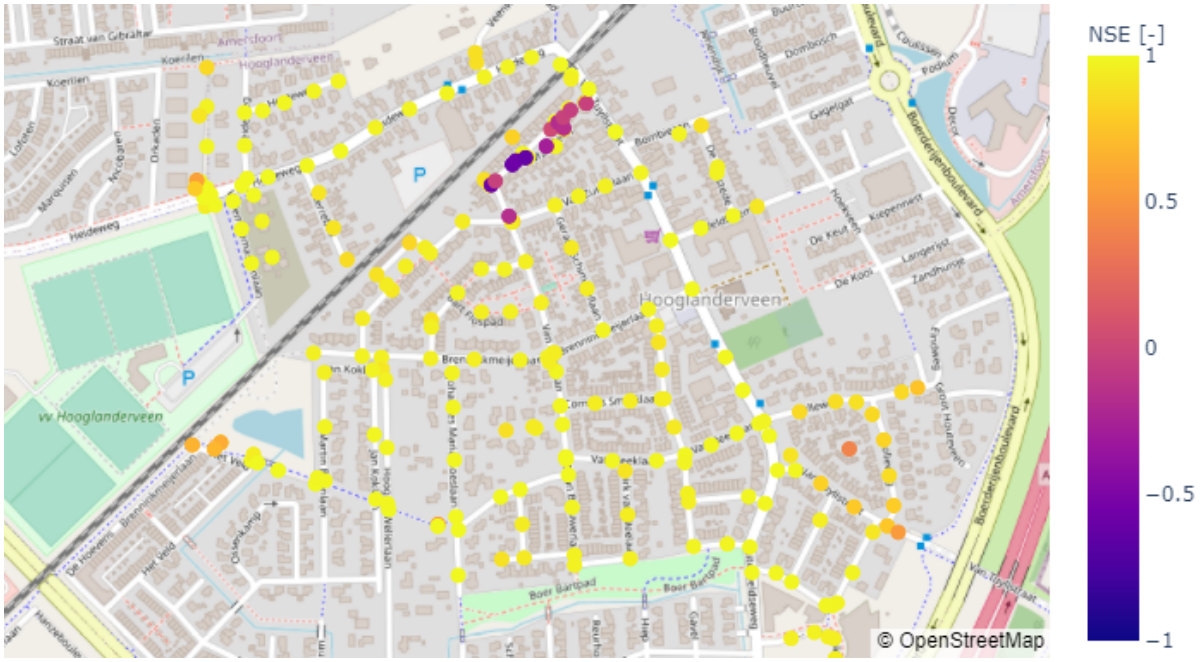


Figure 5.5: NSE values for each node in the case study area (mean NSE = 0.87). NSE values have been calculated with the predicted flood volume time series by the LSTM and sewer model. The NSE is calculated for each time series and a mean is taken for the nodes.

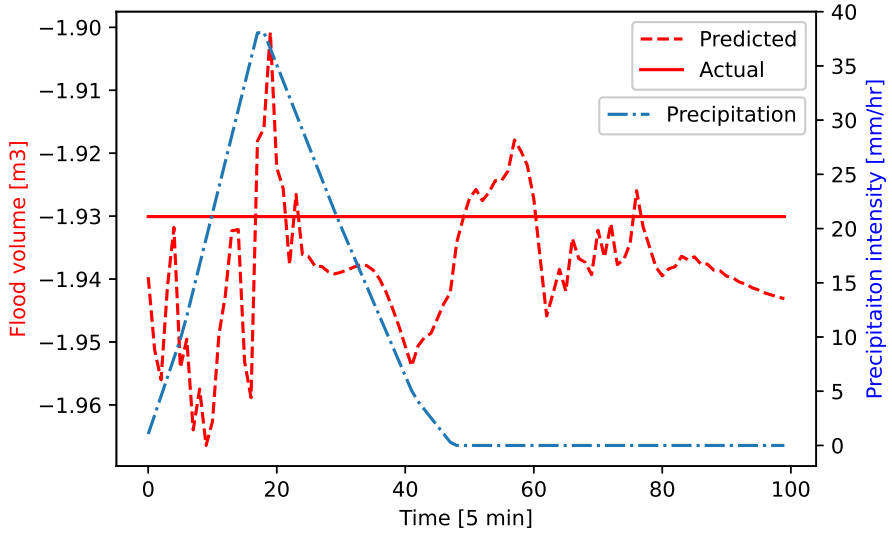


Figure 5.6: Flood volume time series for the LSTM network validated on synthetic data, at node 110197 *(NSE = −0.58). This is one of the dark blue dots, in the north of the area, in Fig. 5.5.*

Lower NSE values can also be seen in the north-west and south-west of the area in Fig. 5.5. These are both locations where a pump is located. Therefore, flooding rarely occurs here and flood volumes are in the negative. It could be that the LSTM algorithm has a hard time modelling the activation of the pump and therefore the flood volumes are harder to predict close to pumps.

Furthermore, low NSE values are observed in the south-east of the area. Nodes in this area do exhibit flooding, but less frequent and with less flood volumes than other nodes. Therefore, it seems that the LSTM algorithm cannot predict peak flood volumes well. Fig. 5.7 shows two flood volume time series of one of the nodes in this area. Flooding is observed, but much lower than the nodes in the centre of

the area, also the form of the flood volume time series is more complex. This difference in complexity can clearly be seen compared with Fig. 5.3 and could be the cause of the decrease in NSE values.

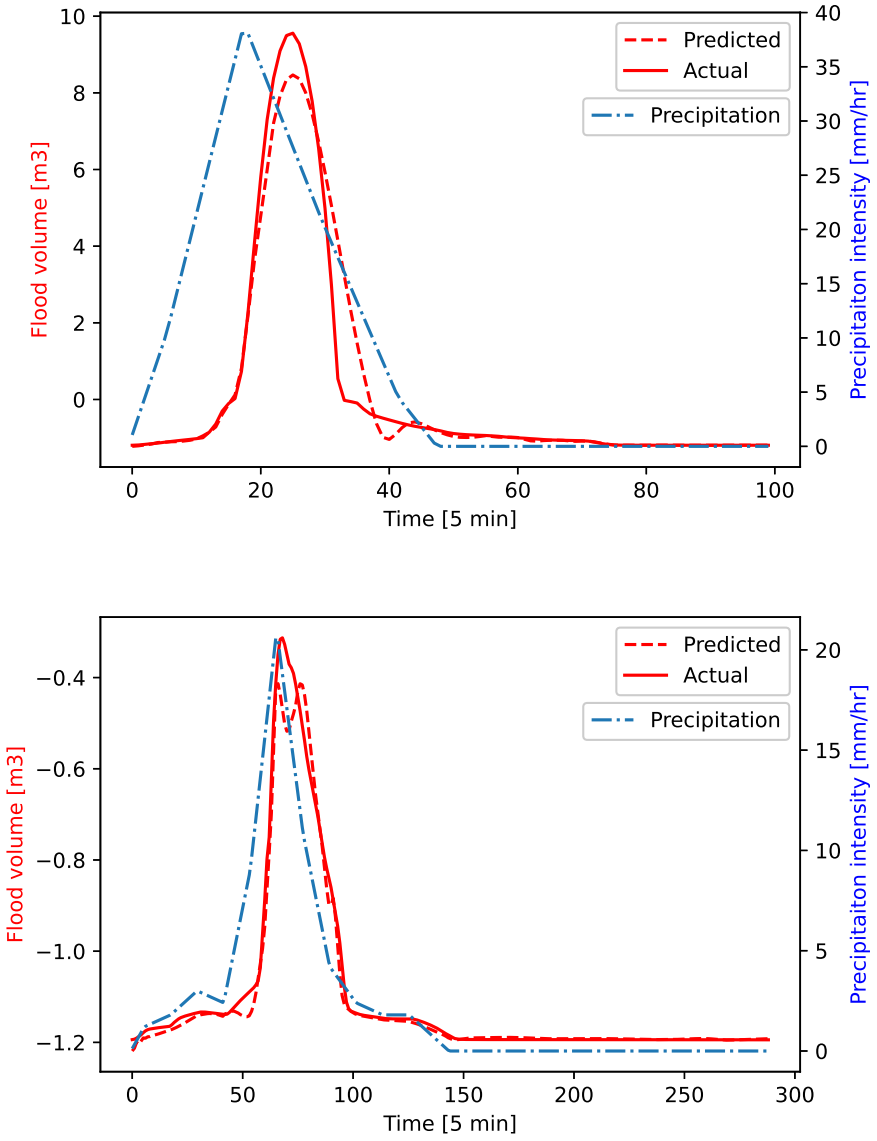


Figure 5.7: Two flood volume time series, for the LSTM network validated on synthetic data, at a node in the south-east of the area (node 110104*, NSE* = 0.82*).*

As mentioned before, it is important to note that nodes that show a low goodness-of-fit do not flood in the the validation data set. This behaviour is confirmed with a map of the NSE values, only for nodes that experience flooding (nodes that have a maximum flood volume larger than 0), see Fig. 5.8. The mean NSE for the nodes that experience flooding is also higher, with a NSE of 0.92. Only 38% of the nodes in the studied area experienced flooding on the validation data set.

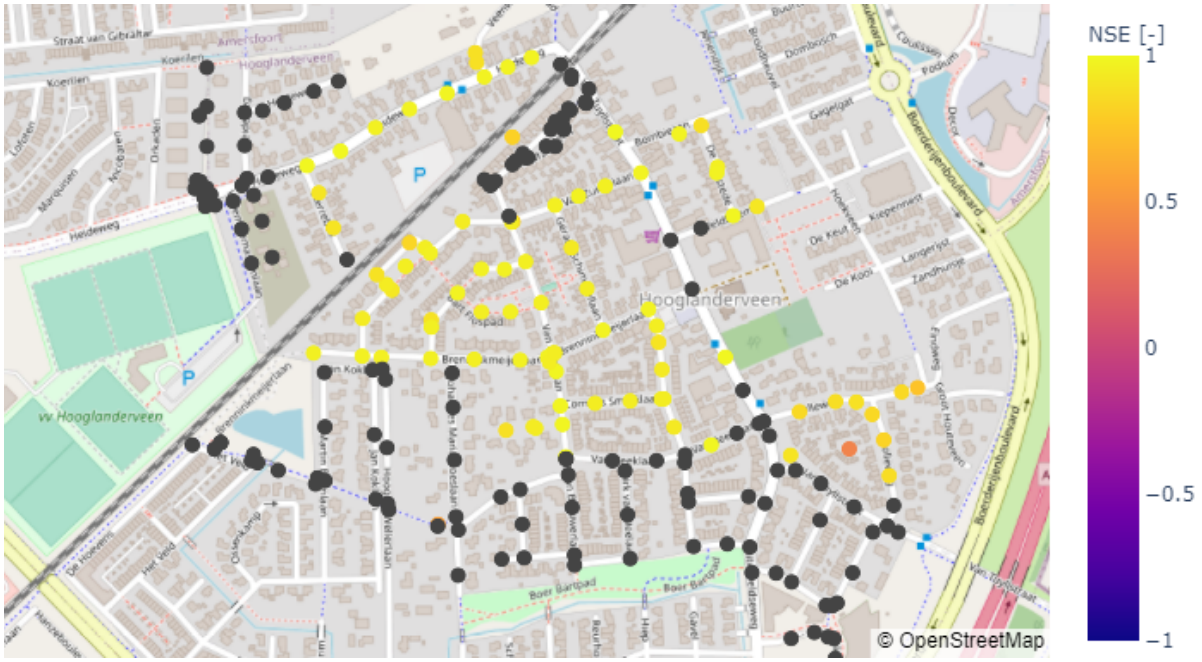


Figure 5.8: NSE values for each node in the case study area that experiences flooding from the validation data set (mean NSE = 0.92). NSE values have been calculated with the predicted flood volume time series by the LSTM algorithm and sewer model. The NSE is calculated for each time series and a mean is taken for the nodes. Dark grey nodes indicate locations where no flooding occurs.

5.2 Testing of the LSTM on historic data

To further test the LSTM algorithm, historic precipitation events that caused flooding in the area have been identified (see chapter 4.8 for more detail on the historic data). The precipitation events have been fed into the sewer model and LSTM algorithm and subsequent results are obtained. Historic events 1 and 4 (see Fig. 4.14), do not cause any flooding in the area. This is persistent in both the predicted values from the LSTM and the sewer model results. The sewer model is therefore unable to model the cause of flooding during these events. Both events have very low precipitation values and nodes remain mostly at their (negative) flood volume storage values. Therefore, further NSE values are all calculated without these two events. However, for the calculation of the MAE all events are included.

The MAE and NSE values can be seen in Tab. 5.5. Fig. 5.2 shows the fraction of nodes that have a higher NSE than the value on the x-axis. It can be seen that in general the LSTM has a lower goodness-of-fit when predicting flooding from historic precipitation data. This is expected, as the historic data has very different precipitation patterns than the LSTM is trained on.

Metric	Value
MAE	0.19 m^3
NSE	0.61
R^2	0.99

Table 5.5: Goodness-of-fit evaluation for the LSTM algorithm tested on historic data. For the calculation of the mean NSE value, historic events 1 and 4 are excluded. These do not cause any flooding in the sewer model results and are therefore, not important for the evaluation of a flood early warning system.

The NSE values for each node is shown in Fig. 5.10. A larger range of values can be seen, compared to the validation of the LSTM. Nodes that show frequent flooding and have high flood volume peaks, have the highest NSE values. These nodes are located in the centre of the area. Low NSE values can be observed in the same areas where the LSTM algorithm also had relatively low performance on the synthetic validation data set.

The LSTM algorithm has generally worse NSE values when tested on the historic data set. There are

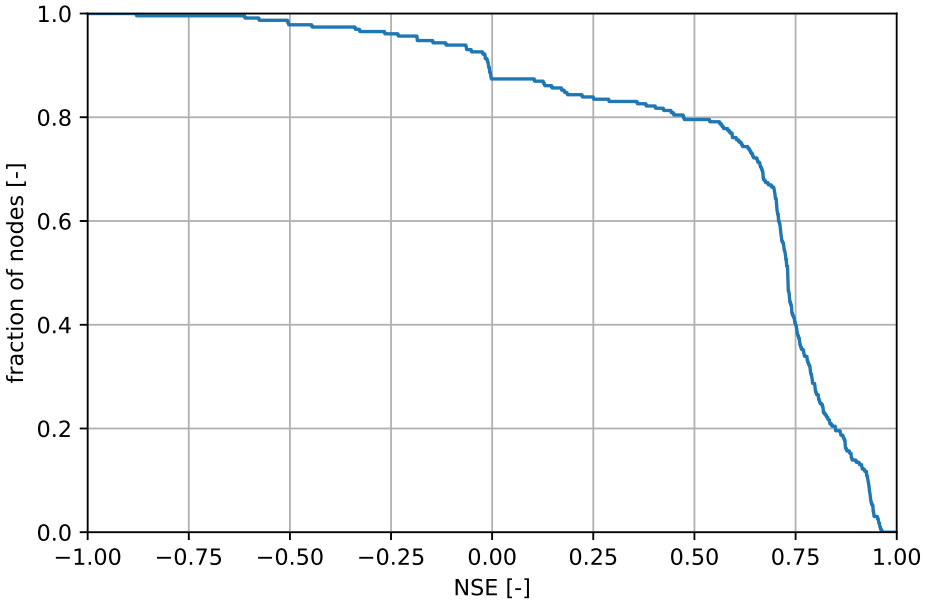


Figure 5.9: Fraction of nodes, for the LSTM regressor prediction on the historic input precipitation data set, with a greater NSE than the value on the x-axis.

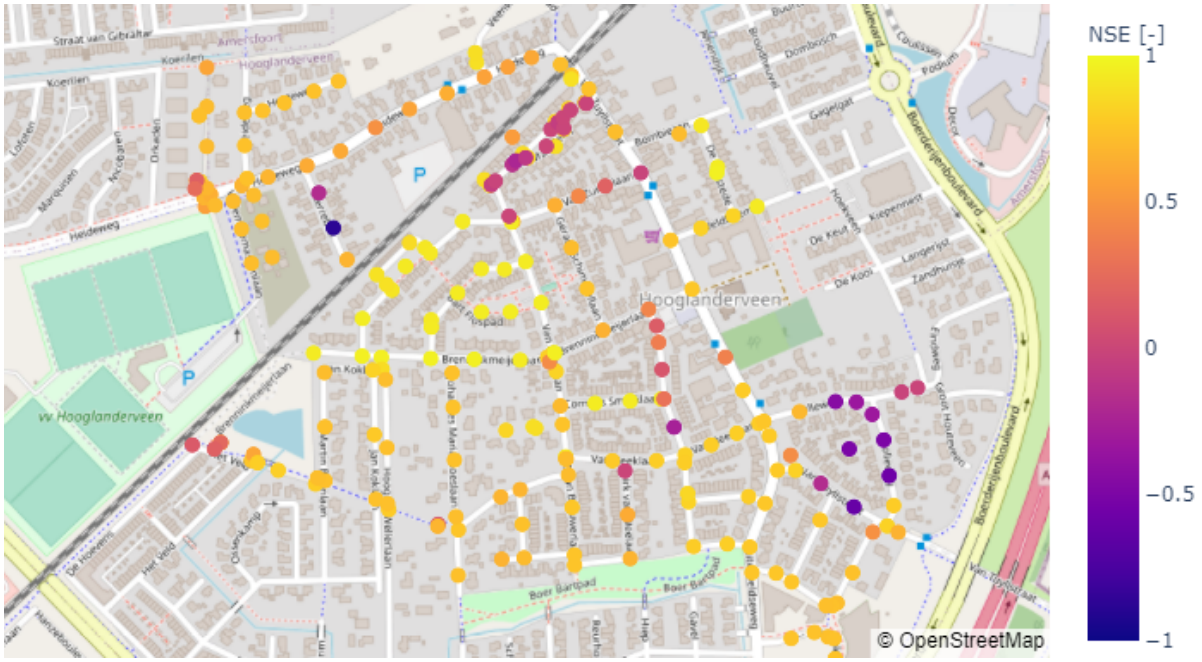


Figure 5.10: NSE values for each node in the case study area of the historic precipitation event prediction (mean NSE = 0.61). Note that the dark grey nodes represent values that are large negative values values.

two main differences between the historic precipitation and synthetic precipitation time series. First, the historic data precipitation peaks are larger and confined in a smaller time span, compared to the synthetic training data set. Second, the historic data has many small fluctuations in the precipitation data.

The in general lower performance of the LSTM on the historic data shows, that the trained algorithm performs best when large and smooth precipitation intensities are given as input and the output is also a large flood volume time series, matching the precipitation patterns from the training data set. Small perturbations in the input causes large fluctuations in the predictions, which can be seen in Fig. 5.11. Looking at a node with large flood volumes, it can be seen that the LSTM algorithm, even on historic

data, can predict the size and duration of the flood volume well, see Fig. 5.12. When looking at the scatter plot of the predicted and actual values a larger spread can be seen, see Fig. 5.13. The tendency to underestimate high flood volumes is still present and has increased. Fig. 5.14 shows a map of the NSE values for the nodes that experience flooding. The mean NSE increases for the nodes that experience flooding to 0.66, compared to all the nodes. Only 24% of the nodes experience flooding from the historic precipitation events. For the nodes that experience flooding the NSE values are still high, only the nodes in the south east of the area show a large decrease in goodness-of-fit. In this area the peak of the flood waves is underestimated greatly. This underestimation can be seen in Fig. 5.15. Aside from the underestimation, the timing of the peak with the inflow and outflow is predicted well.

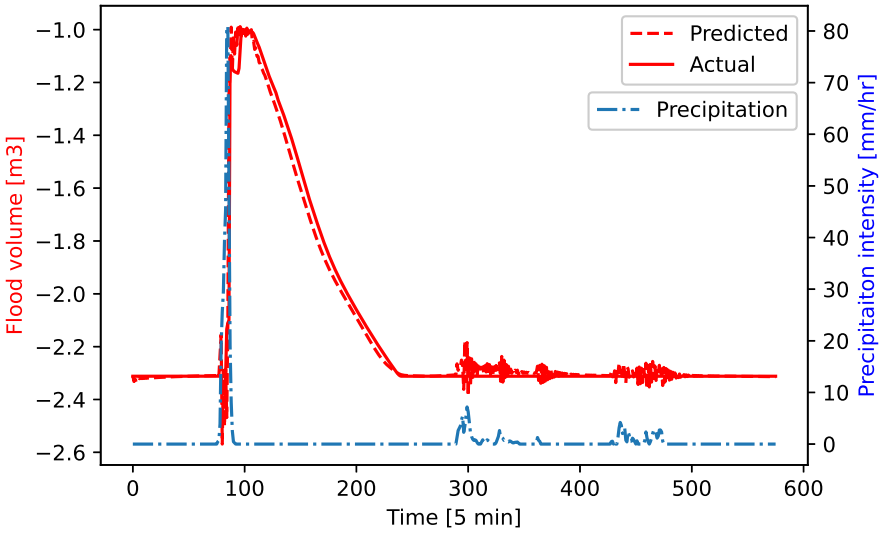


Figure 5.11: Flood volume time series, for the LSTM network validated on historic data, at a node in the south of the area (node D1128V, NSE = 0.73).

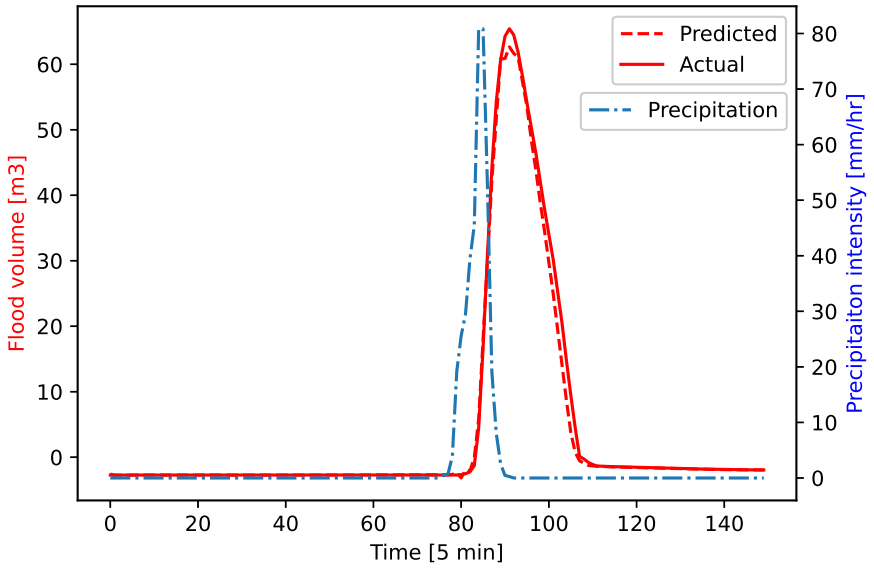


Figure 5.12: Flood volume time series, for the LSTM network validated on historic data, at a node in the centre of the area (node 110050, NSE = 0.96).

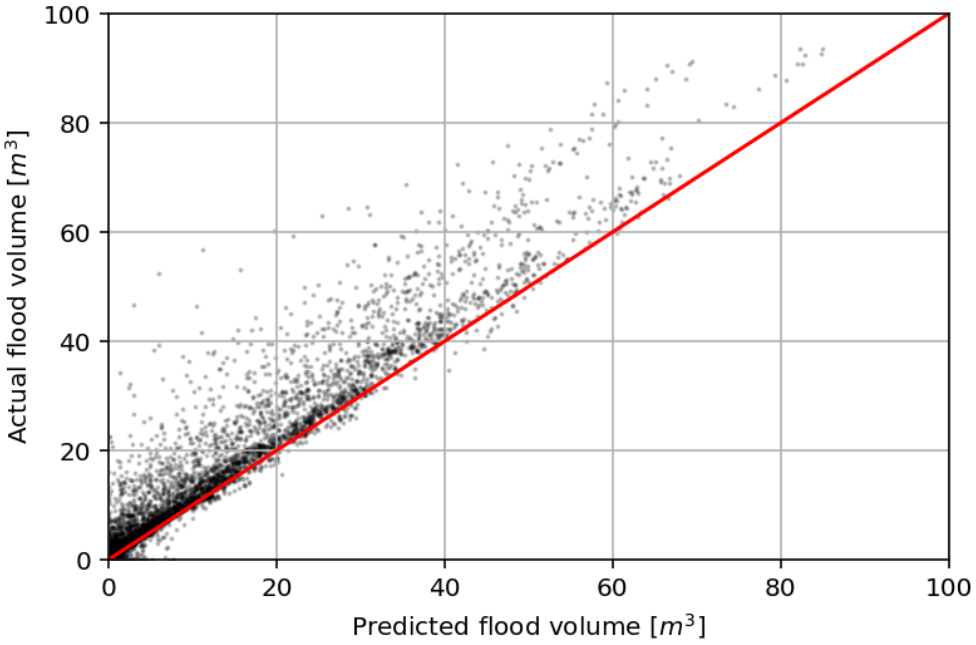


Figure 5.13: Scatter plot of the predicted and actual flood volumes taken from the historic data evaluation ($R^2 = 0.99$). Negative flood volumes are not plotted, as they are not of importance for a flood early warning system.

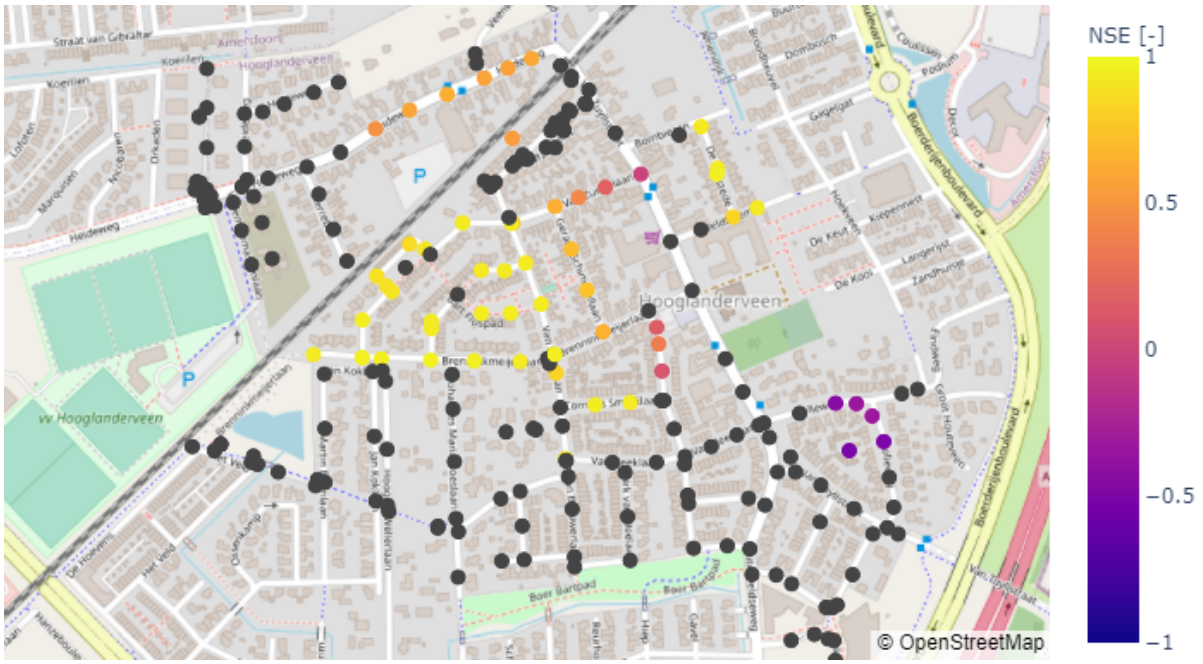


Figure 5.14: NSE values for each node in the case study area that experiences flooding from the historic data set (mean NSE = 0.66). NSE values have been calculated with the predicted flood volume time series by the LSTM algorithm and sewer model. The NSE is calculated for each time series and a mean is taken for the nodes. Dark grey nodes indicate locations where no flooding occurs.

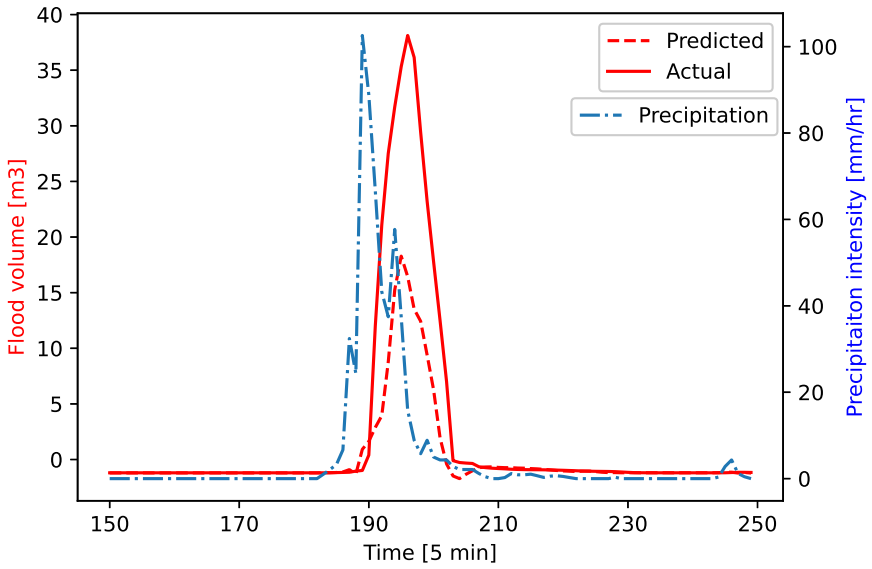


Figure 5.15: Flood volume time series, for the LSTM network validated on historic data, at a node in the south east of the area (node 110104*, NSE* $= -0.5$*).*

6 Discussion, conclusion & recommendations

6.1 Discussion

All three ML algorithms achieved high evaluation scores. However, the real world applicability of the MLP algorithms has shown to be low. As the MLP algorithms are trained on the precipitation features defined using precipitation statistics from Beersma et al. (2019), they also require these features to be present in the historic data. Due to the inherent more fluctuating and irregular nature of real precipitation data, these features would be hard to obtain. This makes it cumbersome to use the MLP algorithm, as configured in this study, as a flood early warning system. Furthermore, extraction of the precipitation features would introduce another layer of abstraction, which is undesired. Adjusting the precipitation statistics and subsequent precipitation features to more realistic extreme precipitation events observed, could increase the viability of the MLP algorithms as a flood early warning system.

The use of synthetic precipitation data for the training, testing and validation of the ML algorithms proved useful and easy to use. A large drawback of using ML algorithms is the amount of data that is required to train, test and validate the ML algorithm. In this study synthetic extreme precipitation events were constructed using precipitation statistics. This data was used to obtain flood volume time series from a sewer model for all manholes in a specified area. After training the LSTM on the synthetic data, the algorithm performed well in predicting flooding from historic precipitation data. Interestingly we can see that the LSTM has learned the behaviour of the manholes reacting to a range of synthetic precipitation events with different intensities, patterns and durations. Then when tested on historic data, that had much larger precipitation peaks which occurred in a smaller time span, the LSTM could predict flood volumes accurately for most manholes in the area.

The LSTM algorithm, capable of predicting flood volume time series for each manhole in the studied area has shown high performance, even when tested on historic data. The LSTM is therefore, applicable for use as a flood early warning system. However, two drawbacks were observed. Firstly, peak flood volumes of large peaks were underestimated slightly for the validation on synthetic data. This underestimation increased when tested on historic data. It is unknown what the cause is of this underestimation. One probable cause is the loss function used when training the LSTM network. In the present study the MAE loss function was used. This loss function takes the mean absolute error over the whole time series to determine the loss. No special attention is given to the peak flood volumes in this loss function. This could be the cause of the underestimation. By using a loss function that weighs the loss at high flood volume values more, underestimation of peaks could be reduced. Secondly, from the results it can be seen that the LSTM algorithm is more sensitive to small fluctuations in the input precipitation than the sewer model. When the LSTM was tested on historic data, this sensitivity was more apparent. The sensitivity to small fluctuations in the input can have several causes. Firstly, the inherent architecture of a ANN and LSTM, causes the algorithm to always respond to the input. As the input precipitation is translated to an output flood volume via weights, the flood volume fluctuates heavily with small fluctuations in the input precipitation. This behaviour is not present in the sewer model data the LSTM is trained on. Secondly, the synthetic data the LSTM is trained on has very smooth precipitation patterns, with no noise is present in the input. The noise that is present in the historic data is therefore unfamiliar for the trained LSTM. Smoothing the historic input precipitation data eliminating noise or training the LSTM

on precipitation data with noise present, could eliminate this drawback.

The use of the LSTM as a flood early warning system looks promising. In this study a relatively small area was chosen to test the applicability of ML algorithms as a flood early warning system. This could be expanded to a whole city like Amersfoort, where the algorithm predicts flood volume time series for each manhole. However, there are several complications that come with expansion of the area observed by the algorithm. Firstly, training times will increase significantly as weights between each manhole need to be trained. Training of the LSTM for the case study area, already had large training times ($\approx 30\,\text{min}$). Training a network for the whole city could take years, making it unfeasible. Secondly, the network uses a uniform precipitation input, where no spatial variation is present in the input. Therefore, non-uniform precipitation cannot be used as input for the ML algorithms. This assumption is acceptable for the size of the area chosen, which is approximately $1\,\text{km}^2$, but would yield unrealistic results when upscaled to larger areas. Therefore, an approach should be chosen that takes into account the spatial variation in precipitation. Two possible options are the constructing of multiple ML algorithms for areas of a city or constructing ML algorithms for each manhole.

6.2 Conclusion

The goal of this research was to construct ML algorithms that can predict location-based flooding due to extreme precipitation in an urban environment. We can conclude that the behaviour of the existing numerical sewer model and its characteristics has been successfully reproduced by the ML algorithms. Both in classification and regression, the MLP and LSTM algorithms are able to predict flooding and flood volumes with high accuracy, with an accuracy of 98.29% for the MLP classifier and MAE values of $0.20\,\text{m}^3$ and $0.06\,\text{m}^3$ for the MLP and LSTM regressor respectively. In addition to the MLP, the LSTM predicts the temporal aspects of the flood wave; the duration of the flood wave and subsequent emptying accurately. Especially locations with frequent flooding show high NSE values. However, note that the ML algorithms are only as good in predicting flooding, as the sewer model it is trained on.

Hyper-parameter configurations for the MLP classifier and regressor and LSTM regressor have been determined using random search and Bayesian optimisation. Interestingly the number of neurons in the hidden layers increased with algorithm complexity. Here, 152, 366 and 636 neurons were determined best performing for the MLP classifier, MLP regressor and LSTM respectively. The optimised MLP classifier has less neurons per layer than there are manholes in the study area. This indicates a strong correlation between manholes, i.e. their flooding behaviour is more or less identical. The learning rate was determined significantly lower for the MLP classifier and regressor than the LSTM. For the LSTM the learning rate was even set at the maximum of the given range. Increasing the range maximum further, caused the optimisation to crash. Therefore, the maximum was maintained at 0.01. Once again an increasing value for the hyper-parameter can be seen with increasing complexity. Here, a learning rate of 0.0027, 0.0038 and 0.01 were determined best performing for the MLP classifier, MLP regressor and LSTM respectively. We can conclude that the hyper-parameter configurations determined are sufficient and have shown high evaluation scores.

The precipitation feature used as input, provided enough information for the ML algorithms to predict the output flood volumes. Extraction of features (precipitation intensity, duration and pattern), provided enough variation and information for the MLP algorithms to train and predict flooding and maximum flood volumes. However, the features used to create the synthetic data set are directly present in the historic data, and would have to be derived. Therefore, the MLP classifier and regressor, as configured in this study, are harder to use with actual precipitation forecasts than the LSTM regressor.

Testing of the LSTM on historic data, shows that the LSTM can accurately predict flooding for historic precipitation events. These precipitation patterns, found in historic events, are different from the synthetic data. Here, the precipitation is confined to a very short interval, where precipitation with a high intensity falls. In the synthetic data, the pattern has a more gradual increase in precipitation intensity before the peak. Interestingly, the LSTM still was able to predict peak flood volumes and durations accurately. When testing the LSTM on the historic data, we essentially removed one level of abstraction from reality; the synthetically constructed precipitation data. Still, the LSTM that was trained on the synthetic data was able to predict flood volume time series from the historic precipitation data. The LSTM is therefore, able to learn the inherent behaviour of the sewer model from the synthetic precipitation data and sewer model results. Note however, that the LSTM is till a second level abstraction from

reality as it is trained on sewer model data.

6.3 Recommendations

From the three ML algorithms the LSTM has proven to be the best performing and robust algorithm. For further use and research of the LSTM as a flood early warning system, several recommendations are made:

- **One vs Many** In initial testing of the LSTM network, a network was evaluated with only a single location. This one manhole LSTM network showed good performance. Therefore, we recommend further research in the use of many algorithms all trained for individual locations, as opposed to one algorithm for multiple locations. The use of one algorithm for each manhole also makes it possible to use a non-uniform precipitation forecast, as opposed to the uniform forecast used in the present research. The non-linear interactions in the sewer network could be present in the output sewer model data of each individual manhole. These non-linear interactions are already simulated in the sewer model. Creation of a ML algorithm for the whole area could, therefore, be unnecessary. Furthermore, one could use different ML regression algorithms, that are not ANNs, for this purpose.
- **Underestimation** The LSTM underestimates flood volumes slightly for many precipitation events. The cause of this underestimation is unknown. It could be caused by the MAE loss function used. Further research into the use of different loss functions and its effect on underestimation or overestimation is recommended. Recommended is the use of a loss function that weights losses at peak flood volumes higher than at low flood volumes.
- **Sequential model setup** The amount of hidden layers, of the LSTM, was kept relatively small compared to the MLP, this was done to reduce the hyper-parameter space for hyper-parameter optimisation. Further research is recommended in the setup of e.g. multiple LSTM layers and added hidden layers in the sequential model. These could also be interchanged where a hidden layer is set between two LSTM layers. Algorithm performance might be improved using a more complex sequential model setup.
- **Flood early warning system** The LSTM has shown good performance on the historic data set, confirming that the LSTM can be used as a flood early warning system. It is recommended, however, to further research the implementation and operationalisation of such an LSTM as a flood early warning system. Here, several aspects should be considered. Firstly, due to the fast run time, a stochastic input can be used to determine a range and chance of flood volumes. Secondly, the LSTM needs to be further evaluated with precipitation forecasts that are less extreme than the historic events but still induce flooding. The inherent properties of the trained LSTM to react heavily to small fluctuations, could hinder its performance when such precipitation events are considered.
- **Extreme precipitation events** The historic extreme precipitation events are defined by precipitation that falls during very short time periods. The statistics provided by Beersma et al. (2019) are essentially longer precipitation events scaled down to 4, 8 and 12 hour periods. Therefore, we recommend the inclusion of shorter ($< 4\,\mathrm{hr}$) extreme precipitation events in the precipitation statistics used by Beersma et al. (2019) and in the training data for the ML algorithms.
- **Sewer systems** The present research case study area sewer system is a combined system, with very slight topographic gradients. Further research into the application of the LSTM on other systems, e.g. a separated sewer system, or e.g. a region with different topography, will provide further insight into the general applicability of the LSTM for flood prediction in an urban environment.

6 Bibliography

Ayazpour, Z., Bakhshipour, A., and Dittmer, U. (2019). Combined sewer flow prediction using hybrid wavelet artificial neural network model. *Green Energy and Technology*, pages 693–698.

Beersma, J., Hakvoort, H., Jilderda, R., Overeem, A., and Versteeg, R. (2019). Neerslagstatistiek en reeksen voor het waterbeheer 2019. Technical report, Stichting Toegepast Onderzoek Waterbeheer.

Bergstra, J. and Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13:281–305.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation.

Christopher Olah (2015). Understanding lstm networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/. Last checked on 08/02/2021.

Dawson, C. and Wilby, R. (2001). Hydrological modelling using artificial neural networks. *Progress in Physical Geography*, 25(1):80–108.

Dewancker, I., McCourt, M., and Clark, S. (2015). Bayesian optimization primer.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

Hallouin, T. (2019). Thibhlln/hydroeval: General enhancements.

Henonin, J., Russo, B., Mark, O., and Gourbesville, P. (2013). Real-time urban flood forecasting and modelling - a state of the art. *Journal of Hydroinformatics*, 15(3):717–736.

Hirabayashi, Y., Mahendran, R., Koirala, S., Konoshima, L., Yamazaki, D., Watanabe, S., Kim, H., and Kanae, S. (2013). Global flood risk under climate change. *Nature Climate Change*, 3(9):816–821.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Jason Brownlee (2020). Why one-hot encode data in machine learning? https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/. Last checked on 28/12/2020.

Li, X., Zhou, F., and Lodewyk, S. (2011). Applications of Artificial Neural Networks in Urban Water System. *Watershed Management*, pages 508–519.

Mathevet, T., Michel, C., Andréassian, V., and Perrin, C. (2006). A bounded version of the nash-sutcliffe criterion for better model assessment on large sets of basins. *IAHS-AISH Publication*, 307:211–219.

Min, S.-K., Zhang, X., Zwiers, F., and Hegerl, G. (2011). Human contribution to more-intense precipitation extremes. *Nature*, 470(7334):378–381.

Moreno, J. J. M., Pol, A. A. P., and Gracia, P. M. (2011). Artificial neural networks applied to forecasting time series. *Psicothema*, 23(2):322–9.

Mounce, S., Shepherd, W., Sailor, G., Shucksmith, J., and Saul, A. (2014). Predicting combined sewer overflows chamber depth using artificial neural networks with rainfall radar data. *Water Science and Technology*, 69(6):1326–1333.

Nash, J. and Sutcliffe, J. (1970). River flow forecasting through conceptual models part i — a discussion of principles. *Journal of Hydrology*, 10(3):282–290.

Rajaee, T., Ebrahimi, H., and Nourani, V. (2019). A review of the artificial intelligence methods in groundwater level modeling. *Journal of Hydrology*, 572:336–351.

Razavi, S., Tolson, B., and Burn, D. (2012). Review of surrogate modeling in water resources. *Water Resources Research*, 48(7).

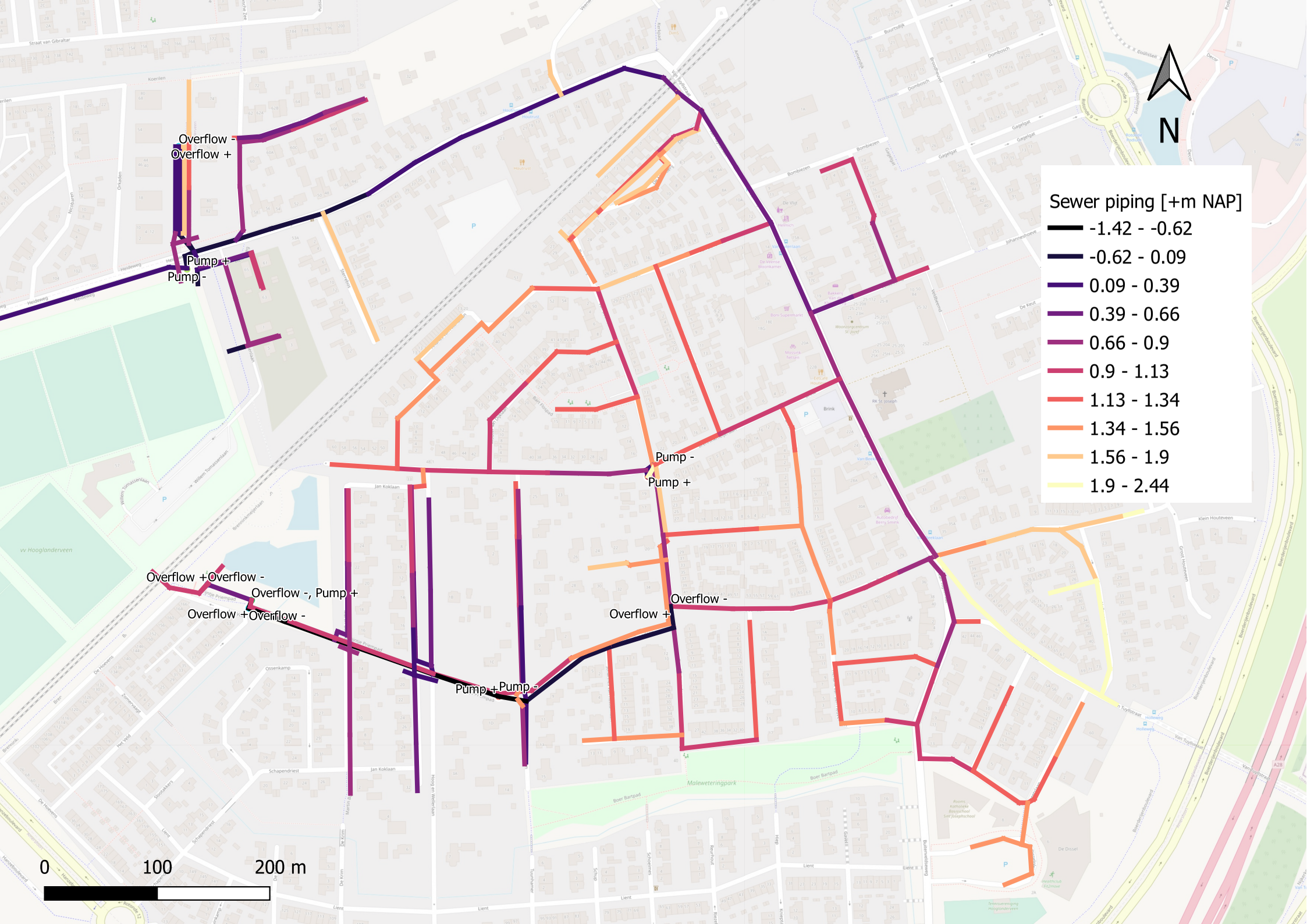
Rioned, S. (2020). Kennisbank.

Szöllösi-Nagy, A. and Zevenbergen, C. (2018). *Urban flood management*.

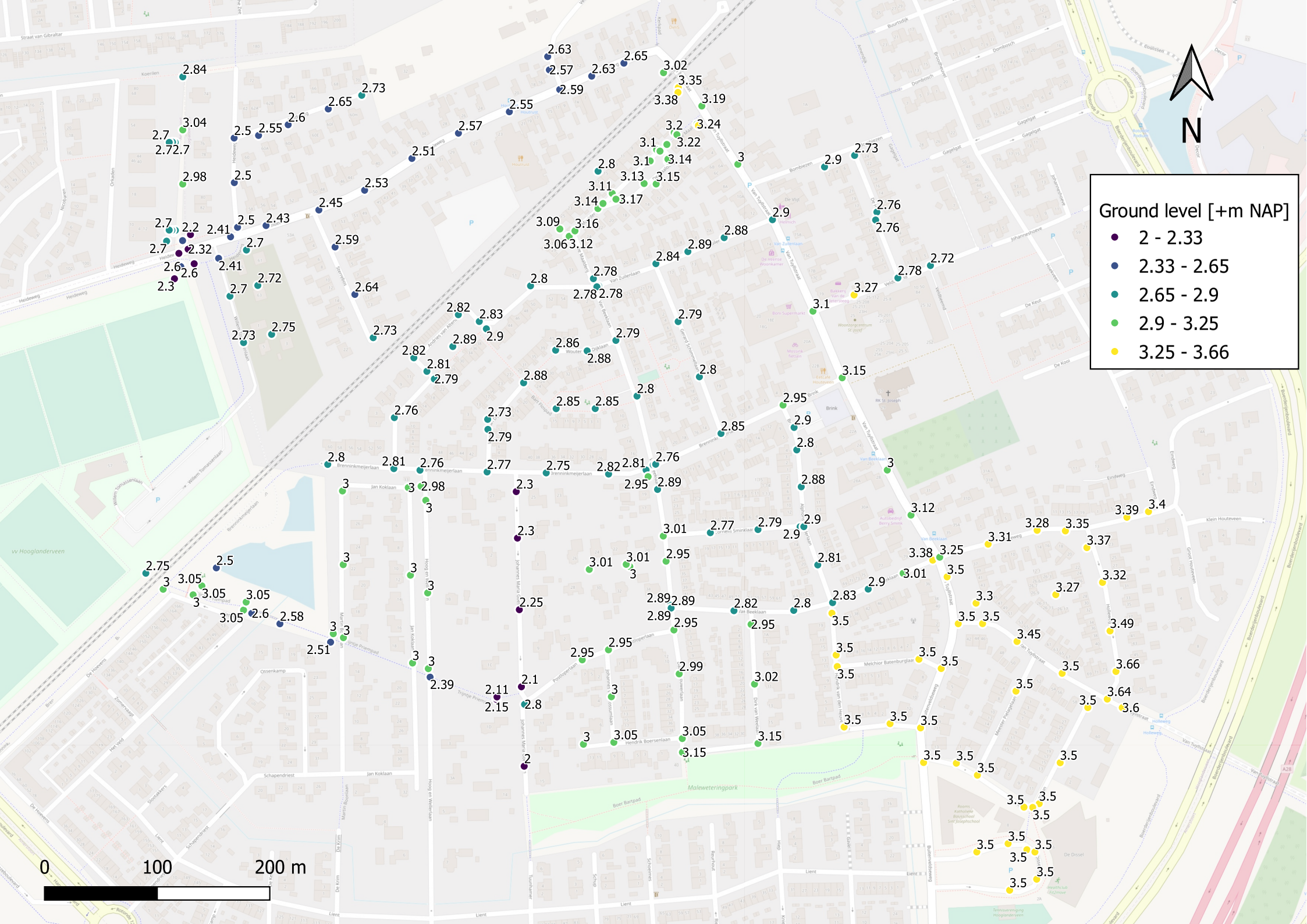
Verkade, J. and Werner, M. (2011). Estimating the benefits of single value and probability forecasting for flood warning. *Hydrology and Earth System Sciences*, 15(12):3751–3765.

A Sewer model overview

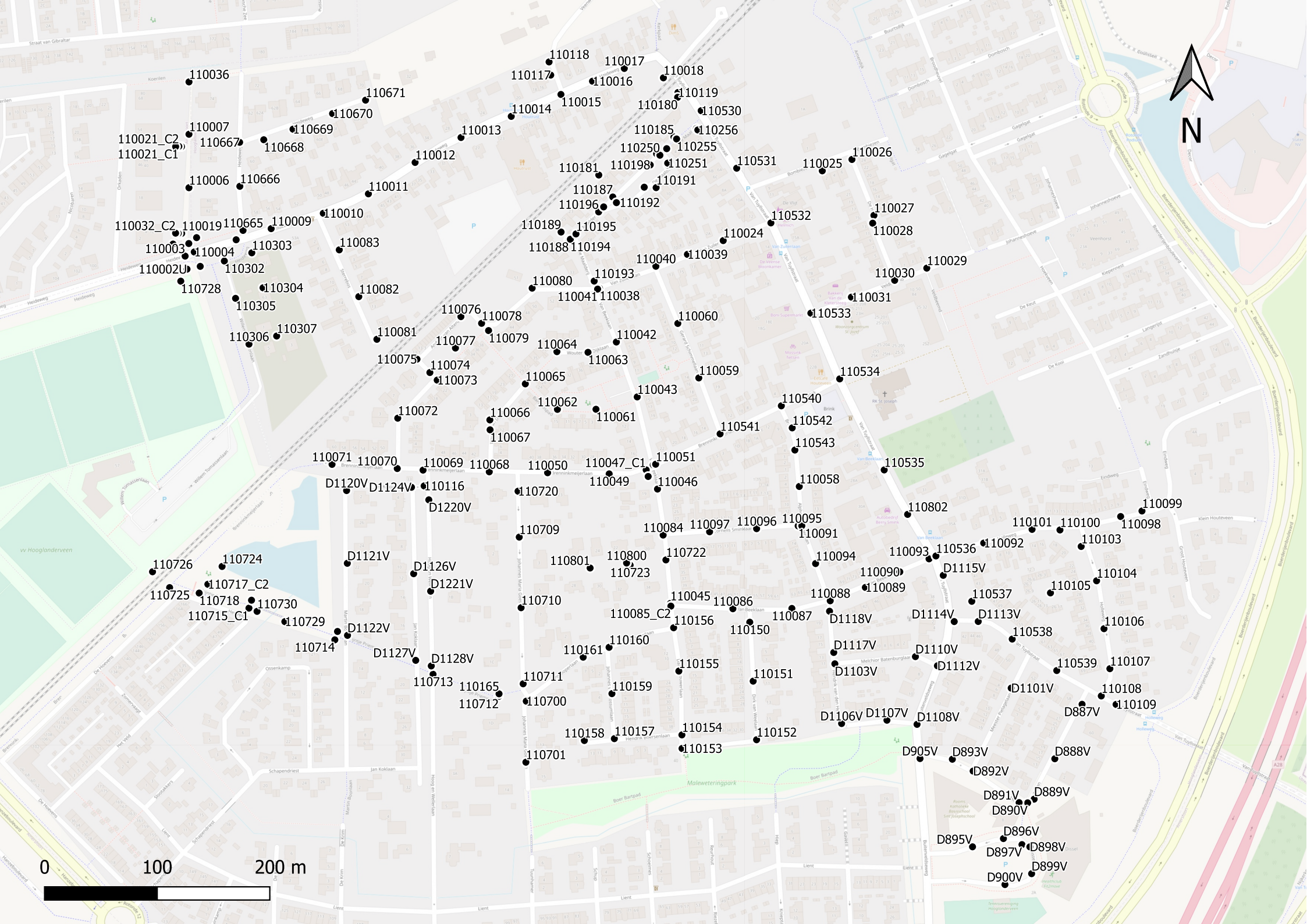
A.1 Sewer piping & structures


Overflow -
Overflow +
Pump +
Pump -
Pump -
Pump +
Overflow +Overflow -
Overflow -, Pump +
Overflow +Overflow -
Pump +Pump -
Overflow +
Overflow -
N
Sewer piping [+m NAP]
-1.42 - -0.62
-0.62 - 0.09
0.09 - 0.39
0.39 - 0.66
0.66 - 0.9
0.9 - 1.13
1.13 - 1.34
1.34 - 1.56
1.56 - 1.9
1.9 - 2.44
0
100
200 m

A.2 Ground level


Ground level [+m NAP]
2 - 2.33
2.33 - 2.65
2.65 - 2.9
2.9 - 3.25
3.25 - 3.66
N
0
100
200 m

A.3 Node ID


110118
110017
110036
110117
110016
110018
110671
110015
110119
110670
110014
110180
110530
110007
110669
110013
110185
110256
110021_C2
110667
110668
110250
110255
110021_C1
110012
110181
110198
110251
110531
110025
110026
110006
110666
110191
110011
110187
110192
110196
110532
110027
110032_C2
110019
110665
110009
110189
110195
110028
110010
110024
110003
110083
110188
110194
110004
110303
110040
110039
110029
110002U
110302
110030
110728
110304
110082
110080
110193
110031
110041
110038
110305
110533
110076
110078
110060
110306
110307
110081
110042
110077
110079
110064
110075
110063
110534
110074
110059
110073
110065
110043
110540
110072
110066
110062
110061
110542
110541
110067
110543
110071
110070
110069
110068
110050
110047_C1
110051
110535
D1120V
D1124V
110116
110049
110046
110058
110720
D1220V
110802
110099
110709
110084
110097
110096
110095
110101
110100
110098
110091
110103
D1121V
110801
110800
110722
110092
110726
110724
D1126V
110723
110094
110093
110536
D1115V
110090
110104
110725
110717_C2
D1221V
110089
110105
110718
110730
110710
110045
110086
110088
110537
110715_C1
110085_C2
110156
110087
D1118V
D1114V
D1113V
110729
110150
110106
D1122V
110538
110714
D1117V
110160
110161
D1110V
D1127V
D1128V
D1112V
110155
110539
110107
110713
110151
D1103V
110165
110711
110159
D1101V
110108
110712
110700
D1106V
D1107V
D1108V
D887V
110109
110158
110157
110154
110152
D905V
D893V
D888V
110153
110701
D892V
D891V
D889V
D890V
D896V
D895V
D898V
D897V
D899V
D900V
N
0
100
200 m

B Precipitation Patterns

Type/Hour	1	2	3	4
Uniform	0.21	0.29	0.24	0.26
1 peak - 12.5%	0.16	0.41	0.28	0.15
1 peak - 37.5%	0.14	0.52	0.28	0.06
1 peak - 62.5%	0.09	0.62	0.26	0.03
1 peak - 87.5%	0.01	0.87	0.12	0.00
2 peaks - short distance	0.42	0.12	0.40	0.06
2 peaks - large distance	0.43	0.05	0.03	0.49

Table B.1: 4 hour precipitation patterns as a fraction of total precipitation [-] (Beersma et al., 2019)

Type/Hour	1	2	3	4	5	6	7	8
Uniform	0.13	0.11	0.12	0.16	0.14	0.11	0.09	0.14
1 peak - 12.5%	0.06	0.08	0.17	0.26	0.20	0.09	0.10	0.04
1 peak - 37.5%	0.03	0.07	0.16	0.38	0.22	0.10	0.03	0.02
1 peak - 62.5%	0.01	0.03	0.12	0.54	0.22	0.05	0.03	0.00
1 peak - 87.5%	0.00	0.00	0.01	0.79	0.18	0.02	0.00	0.00
2 peaks - short distance	0.01	0.32	0.16	0.03	0.12	0.26	0.09	0.01
2 peaks - large distance	0.42	0.05	0.03	0.01	0.02	0.14	0.27	0.06

Table B.2: 8 hour precipitation patterns as a fraction of total precipitation [-] (Beersma et al., 2019)

Type/Hour	1	2	3	4	5	6	7	8	9	10	11	12
Uniform	0.08	0.07	0.09	0.05	0.09	0.14	0.12	0.08	0.07	0.06	0.07	0.08
1 peak - 12.5%	0.03	0.04	0.02	0.06	0.13	0.27	0.16	0.08	0.07	0.05	0.05	0.04
1 peak - 37.5%	0.02	0.03	0.05	0.04	0.15	0.35	0.18	0.07	0.04	0.03	0.03	0.01
1 peak - 62.5%	0.00	0.00	0.02	0.05	0.11	0.47	0.21	0.07	0.04	0.02	0.01	0.00
1 peak - 87.5%	0.00	0.00	0.00	0.02	0.05	0.73	0.15	0.04	0.01	0.00	0.00	0.00
2 peaks - short distance	0.01	0.01	0.06	0.28	0.12	0.06	0.04	0.12	0.20	0.05	0.03	0.02
2 peaks - large distance	0.03	0.27	0.16	0.02	0.02	0.01	0.01	0.01	0.02	0.15	0.25	0.05

Table B.3: 12 hour precipitation patterns as a fraction of total precipitation [-] (Beersma et al., 2019)

C Machine learning algorithm setup

Listing C.1: Scikit-learn MLP regressor algorithm setup

```
reg = MLPRegressor(
                    hidden_layer_sizes=[100,100,100,100],
                    learning_rate='constant',
                    learning_rate_init=0.01,
                    activation='logistic'
                    )
```

Listing C.2: Keras LSTM sequential algorithm setup.

```
model = Sequential()

model.add(LSTM(300,input_shape=(289,1),return_sequences=True))
model.add(Dropout(hp.Float('dropout',
                           min_value=0,
                           max_value=0.2,
                           step=0.05)))
model.add(Dense(230))
model.compile(optimizer=keras.optimizers.Adam(lr=0.01),
              loss='mean_absolute_error')

modelhist = model.fit(
                     x_traintest,
                     y_traintest,
                     validation_split=0.25,
                     epochs=5e2,
                     batch_size=10
                     )
```

D Hyper-parameter optimisation

Listing D.1: Random search setup for the MLP classifier and regressor. For the regressor the scoring is changed to 'neg_mean_absolute_error'.

```
hidden_layers = 10
neurons = list(range(10,500,1))

m = [0]*(hidden_layers*len(neurons))

for i in range(1,hidden_layers+1):
    for idx,i2 in enumerate(neurons):
        m[((i-1)*len(neurons)) + (idx)] = [neurons[idx]]*i

param_space = {
    'hidden_layer_sizes':   m,
    'activation':           ['identity', 'logistic', 'tanh', 'relu'],
    'learning_rate':        ['constant','invscaling','adaptive'],
    'learning_rate_init':   np.arange(1e-4,0.1+1e-4,1e-4)
    }

clf_rand_search = RandomizedSearchCV(clf,
                                     param_space,
                                     n_iter=10000,
                                     scoring='neg_log_loss',
                                     verbose=True,
                                     cv=5,
                                     n_jobs=-1)

search = clf_rand_search.fit(x_traintest, y_traintest)
```

Listing D.2: Bayesian optimisation setup for the LSTM regressor.

```
def build_model(hp):
    model = Sequential()

    model.add(LSTM(hp.Int('lstm_units',
                          min_value=30,
                          max_value=1200,
                          step=1),
                   input_shape=(289,1),return_sequences=True))
    model.add(Dense(230))
    model.compile(optimizer=keras.optimizers.Adam(lr=hp.Float(
        'lr',
        min_value=1e-4,
        max_value=0.01,
        step=1e-4)),
        loss='mean_absolute_error',metrics='mean_absolute_error')
    return model

tuner = BayesianOptimization(
    build_model,
    objective='mean_absolute_error',
    max_trials=100,
    executions_per_trial=1,
    directory='my_dir',
    project_name='LSTM_optimisation')

tuner.search(x_traintest,
             y_traintest,
              epochs=500,
              batch_size=10,
              validation_split=0.25,
              verbose=3)
```

E Recurrent neural network architecture evaluation

To accommodate choosing an RNN architecture for further use, the simple RNN, LSTM and GRU have been briefly evaluated. Fig. E.1 shows the training results. 25% of the train/test data set is used for testing. The validation data set is not used here, it will only be used for final algorithm evaluation. The loss function used is the MAE. The data shuffle has been done once to ensure good comparisons. The setup of the Keras RNN algorithms is equivalent to Listing C.2. It can be seen that the simple RNN underperforms the LSTM and GRU significantly. The minimum MAE validation loss for the simple RNN, LSTM and GRU is $1.29\,\mathrm{m}^3$, $0.07\,\mathrm{m}^3$ and $0.10\,\mathrm{m}^3$ respectively. The LSTM outperforms the GRU slightly. The GRU, however, does have a shorter training time. Therefore, due to the better performance, the LSTM is chosen as RNN architecture for time series regression.

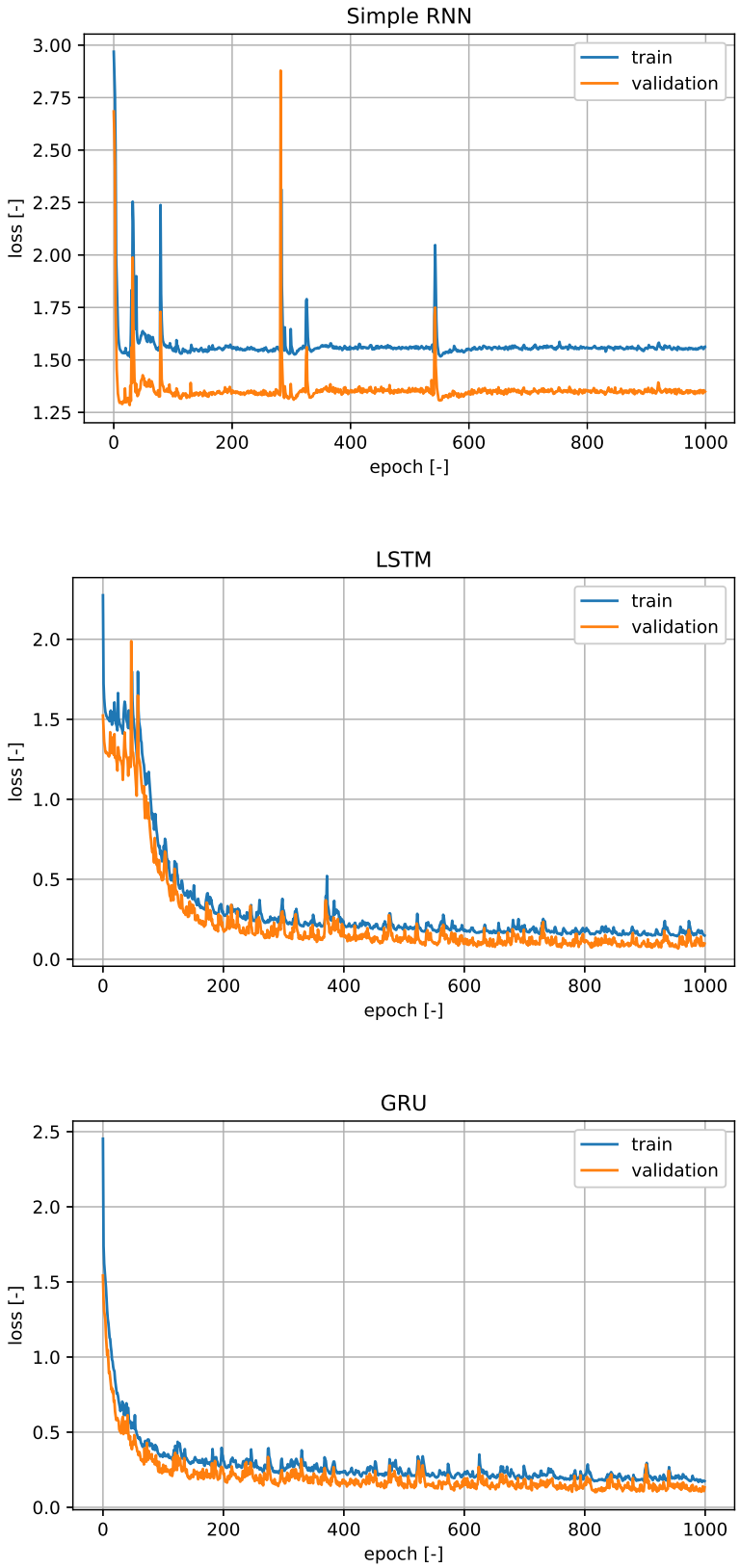


Figure E.1: Evaluation of three RNN architectures using mean absolute error loss. For all three the same hyper-parameters have been used. The RNN layer has 230 *units or neurons with the default activation function provided by Keras. A learning rate of* 0.01 *has been used. The batch size is 10 with 1000 epochs. It can be seen that the Simple RNN significantly under performs compared to the LSTM and GRU networks. The LSTM outperforms the GRU slightly, with an MAE of* $0.07\,\mathrm{m}^3$*. Note that the validation loss is lower due to a dropout layer of 0.2 that has been added after the RNN layer.*