River Flood Forecasting using Long-Short Term Memory Neural Networks

Lars-Eric Ertlmeier

Intelligent and Reliable Systems, IRS10-1,
Spring Semester 2022

Master's Project

Energy Technology
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
River Flood Forecasting using Long-Short Term Memory Neural Networks

Theme:
Master's Thesis

Project Period:
Spring Semester 2022

Project Group:
IRS10-1

Participant(s):
Lars-Eric Ertlmeier

Supervisor(s):
Zhenyu Yang
Benjamin Refsgaard

Copies: 1

Page Numbers: 77

Date of Completion:
May 28, 2022

Abstract:

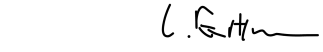
This project attempts to solve the problem of river flood forecasting by providing a data-driven methodology that can be generalised to any catchment using Long-Short Term Memory (LSTM) neural networks. A feature selection method to automatically select moving average blocks for a tapped delay line of rainfall measurements using the multiple Spearman rank correlation coefficient was developed. We applied our method on Abelones Plads, Vejle. A Deep Residual LSTM with 8 layers was found to obtain the best validation performance of all trained networks. We show that it explains up to 67% of the variance in the water level change with a prediction time of 3 hours. We then show that all events where the LSTM fails to outperform the zero-order forecast benchmark model can be attributed to highly spatially variable rainfall which we fail to accurately capture given the location of the rain gauge available to us. We conclude that LSTM models can be used to forecast floods.

Contents

Preface

This project was conducted in cooperation with Vejle Spildevand A/S with the intention of developing a flood prediction method that can be applied at Abelones Plads, Vejle. It is based on separate prior work by Benjamin Refsgaard and the author who worked on a related problem in a previous semester.

Aalborg University, May 28, 2022



Lars-Eric Ertlmeier
<lertlm17@student.aau.dk>

Chapter 1

Introduction

In this chapter, we will introduce the problem of water level prediction that we are attempting to solve in this Master's project. We will review the relevant literature and then discuss the use of neural networks in this field. We will then introduce the subject of the case study on which we apply the developed methodology. The purpose of this project is to find a solution to the problem of water level prediction as a flood mitigation strategy with a methodology that can be generalised to different catchments.

1.1 Problem Statement

In July 2021, Germany and Belgium experienced severe floods along the rivers Rhine and Meuse caused by record-breaking precipitation[8], an event that took more than 200 lives. In the same month, China's Henan province experienced flooding caused by 60 centimetres of precipitation in a single day; as much as expected over the course of an entire year. The official death toll for this event was given as 398 by the Chinese government[10].

The Intergovernmental Panel on Climate Change's (IPCC) report 2021 concluded with "high confidence" that the frequency of extreme rainfall has increased since the 1950s and with "medium confidence" that the increasingly common extreme rainfall events will translate into an increase in flood risk, intensity and damage[22].

The European Union's European Environmental Agency (EEA) has assessed that climate change is driving an increase in the frequency of extreme precipitation events[1]. The World Meteorological Organization (WMO) reports that 44% of disasters reported worldwide between 1970 and 2019 have been associated with floods which caused 16% of reported deaths and 31% of reported economic losses (relative to the total of disaster-related deaths and economic losses). For river floods specifically, the numbers are 24% of disasters, causing 6% of deaths and 21% of

reported economic losses. The WMO also reports that disaster-related deaths have declined, which the report relates to improvements to warning systems. However, the number of floods has seen a marked increase, for example, 66% of reported disasters in Africa between 2000 and 2019 were floods.[49]

River floods are one of the major type of natural disasters and are generally caused by high-intensity rainfall. While measurements and forecasts of precipitation are often readily available, forecasts of the stream levels are usually done with detailed hydrological models that are specific to their respective locations. The purpose of this project is to develop a methodology for short-term forecasting of critical points in streams that can be generally applied to any catchment in which flooding is a known problem. The longest forecasting time that is achievable is highly dependant on the specific system and on the availability of precipitation forecasts.

1.2 Literature Review

In this section, we will discuss different methods used in the field of hydrological modelling, the most recent advancements regarding the use of neural networks in this field and how those approaches compare to physics-based models.

1.2.1 Types of Hydrological Models

Generally speaking most hydrological models can be distinguished according to several properties, how they were obtained, how they divide the area of interest and whether their output is deterministic or stochastic.

A conceptual model is obtained by finding physical relationships between different elements (for example sub-catchments). These relationship require knowledge of a number of system parameters, some of which can be measured directly, like elevation maps, others require calibration against measurements of the system. On the other hand, black box models are based on time series data that measure the relevant system states, inputs and outputs (for example water levels and precipitation measurements). Functional relationships between inputs and outputs are then obtained from data.

Hydrological models can be further classified into lumped, distributed and semi-distributed. Lumped models lump the catchment together and only provide the output for one location (generally the discharge at the exit of the catchment), whereas distributed and semi-distributed models separate the catchment into small areas (for example cells on a grid map). The model states then correspond to local water levels or discharges. This allows the user to gain a fine-grained insight into the hydrological system.

Finally, a hydrological model may be formulated in a deterministic or stochastic environment. A deterministic model may also provide a stochastic prediction when given stochastic inputs. If a neural network model is used, stochasticity may be introduced through the use of dropout layers. The advantage of a stochastic model lies in the fact that it can produce the distribution of the prediction which allows the user to compute statistics of that distribution. This can be used to provide a measure of prediction uncertainty.

1.2.2 Comparison of Neural Network and Conceptual Approaches

The main advantage in conceptual approaches lies in the fact that they can make intermediate states available (such as water flow estimates) that are often times unavailable. This allows the usage of the model for planning. For example a conceptual model may allow the user to estimate the effect on flooding behaviour given the placement of a hypothetical dyke. Furthermore, they are based on sim-

plifications on physical interaction between nodes or grid points and are thus explainable.

On the other hand, neural networks are based purely on measurement data, such as water level measurements and data from rainfall gauges. The advantage of this is that neural networks can be trained with just data. Knowledge of the underlying processes is not required, unlike conventional models which require, for example, detailed elevation maps and a number of other parameters, some of which need to be estimated empirically. Furthermore, neural networks can be much lighter in terms of computational cost compared to conceptual models which often require discretisation of the St. Venant shallow water equations. Since neural networks can approximate these dynamics, it allows real-time predictions of one or more relevant water levels at little computational cost.

However, we can only predict water levels at locations where enough data is available, which requires that water level sensors have been placed at the critical locations a sufficiently long time ago (a few years is assumed to be sufficient). The drawback of data-driven approaches is that unmonitored locations cannot be predicted at all, whereas with conventional models you can make a prediction (albeit the prediction quality would suffer in the absence of data to calibrate against).

We summarise, that neural networks are a convenient and powerful tool for forecasting the water level at specific locations of interest as a function of rainfall for the purpose of flood forecasting. However, they do not allow easily accessible insight into the underlying dynamics and they cannot be used to obtain intermediate states, or model the system behaviour given a user-specified change in some physical parameter. Neural networks are easier to obtain given sufficient measurements and have the capacity to outperform conceptual models which makes them well suited to address the problem of flood forecasting.

1.2.3 State-of-the-art hydrological modelling

Machine learning has become a focus of research in many environments in recent years. Hydrology in particular has seen many attempts at modelling discharge or water levels, in part due to the lack of a complete understanding of the underlying physical process. Since it is the objective of this work to develop a generalisable methodology of short-term water level forecasting, we will review the recent research regarding the application of neural networks in the field of hydrology.

The Long-Short Term Memory (LSTM) neural network introduced by Hochreiter and Schmidhuber[19] is considered among the state-of-the-art of neural network architectures for time series prediction and are preferred over standard Recurrent Neural Networks, primarily due to its capability to learn long-term time dependencies present in the input-output data. This is relevant for purposes of stream water level forecasting as catchments can maintain water for a long time.

Liu et al. trained LSTM models for short-term real-time forecasting in Fuzhou, China, and showed that the forecasts generated by the LSTMs were accurate three to six hours into the future (depending on the location)[32]. Le et al. trained an LSTM model for discharge prediction of the Da river basin, Vietnam, which showed good performance on a 24-hour prediction. Lee et al. coupled an LSTM model using radar precipitation measurements with a Storm Water Management Model (SWMM) to achieve accurate water level predictions in Seoul for prediction times ranging from 30 to 90 minutes[29].

Hu et al. trained both an LSTM and a Feed-Forward Neural Network (FFNN) for runoff prediction with a 1 to 6 hour prediction time on the Fen river basin, China, and showed that both LSTM and FFNN outperformed conceptual models. Furthermore, the LSTM model outperformed the FFNN model for all prediction times[21]. Shalev et al. trained a single LSTM model for daily runoff prediction for several hundred basins by encoding basin-specific static information in the LSTM input and showed that this model outperformed conceptual models calibrated to the individual basins[43]. Rahimzad et al. found that LSTM outperforms linear regression, multilayer perceptron (MLP) models and Support Vector Machines (SVM)[37].

The prediction quality depends heavily on the catchment itself. LSTM models have also been applied on fast-flowing rivers. Luppichini et al. trained an LSTM model for water level prediction on the Arno river basin, Italy, where parts of the basin have runoff times of just four hours[34]. Li et al. investigated two small and medium sized basins in China and found that LSTM models outpeformed comparative models such as Convolutional Neural Networks (CNN) and SVM[30].

A major challenge that Machine Learning faces, especially so in the field of hydrology, is the lack of understanding in how exactly the model works internally. This has become a focus of research in recent years. For example, Kratzert et al. showed that the LSTM cell states correlate to known states of the modelled hydrological system that the LSTM is unaware of[28].

1.2.4 Challenges of Data-driven Approaches

While hydrology has become a hotspot for machine learning in recent years, it is also a field characterised by its resistance against data-driven approaches that lack a direct connection to the underlying physical processes.

The lack of explainability is commonly held against neural networks and other data-driven approaches, for example, hydrologist and former President of the International Association of Hydrological Sciences (IAHS) Vit Klemeš provided a rather scathing rebuke of data-driven approaches in his paper "Dilettantism in Hydrology: Transition or Destiny". An excerpt is given below:

> "For a good mathematical model it is not enough to work well. It must

> work well for the right reasons. It must reflect, even if only in a simplified form, the essential features of the physical prototype."[26]

We believe it is necessary to acknowledge his criticisms. Klemeš primarily criticises data-driven approaches for extrapolation far into the future (he humorously remarks that he was once asked to compute a 1-million-year flood). He further comments on the lack of validation of data-driven approaches. We believe that our methodology survives his test for two reasons: we compute the change in water level from now at most a few hours hours into the future (this is a short period given the slow change of the system) and we evaluate our model on a withheld set of data that the network is not trained on. Thereby, we can reasonably well extrapolate how well the model would perform on the data it may be given in a real-life implementation. Secondly, we note that the model must be retrained periodically as there is no guarantee that the underlying physical processes remain the same, especially in the face of climate change; this is the kind of extrapolation that Klemeš warned against.

By only using data from events in a multi-year-long window, we can reasonably assume that the model performance on unseen events in the past is an approximate indicator of the model performance on events in the near future. We recommend a moving window approach; after one year of runtime, we drop the oldest year of events in the overall set, add all events in the past year and then retrain the model on the new set of events. This ensures that we only extrapolate the system dynamics one year into the future.

We point out that in prior work by the author[11], an LSTM network was trained to model the water level and then used to extract local transfer function estimates. It was shown that the step responses exhibited roughly first-order behaviour which is consistent with our knowledge of the system. This suggests that the internal states of the LSTM correspond in some way to the physical states of the system; thus the LSTM implicitly contains a sort-of conceptual model that could theoretically be extracted. We assume that this property of the trained LSTM is what allows acceptable performance on unseen data.

1.3 Vejle as Case Study

In order to develop a generalisable methodology for local water level forecasting which can then be used to perform flood forecasting, we require a test system to be used as a case study. In this work, we will discuss the stream system of Vejle which has been facing floods at Abelones Plads, a location for which measurement data has been available since December 2015, which makes the city well suited to serve as our case study. This work is done in cooperation with Vejle Spilevand A/S to provide our partner with a methodology to aid them in mitigating the flood problem in Vejle specifically but the methodology itself can be applied to other catchment.

Vejle is Denmark's ninth largest city with 58 777 inhabitants (2021,[36]). It is located at the north-western edge of the administrative subdivision *Region Syddanmark* (Region of Southern Denmark) of whose regional government Vejle is the administrative seat. Vejle is also located at the confluence of the rivers Vejle Å and Grejs Å (which enter the city from west and north, respectively) as well as at the estuary of Vejle Å, the starting point of Vejle Fjord. Most of the city is located in the low-lying areas of the Vejle Å valley, the larger of the city's two rivers. Both rivers have special significance; the Vejle Ådal is Denmark's longest tunnel valley and the Grejsdal is Denmark's longest canyon.

The catchment of the Vejle Å is around 245 square kilometres, whereas the catchment of the smaller Grejs Å is about 78 square kilometres. The latter is almost entirely within Vejle Kommune, with the exception of a small area around the village of Ulkær, which belongs to the Region Midtjylland. Figure 1.3 shows the full extent of the Grejs Å catchment.

Shortly after entering Vejle, the Grejs Å is divided into the two streams of Mølleåen and Omløbsåen by a distribution structure (called the *fordelerbygværk*) located next to the intersection of Vedelsgade and Vesterbrogade. This structure employs three pipes of 140 centimetre diameter with valves that control the flow through the structure. This structure separates Grejs Å and Omløbsåen; the connection between Grejs Å and Mølleåen is uncontrolled. The purpose of the fordelerbygværk is to control the water levels within Omløbsåen. A control algorithm that maintains a constant water level at Aagade, a location approximately central along Omløbsåen, was developed in the previous semester[11]. The controller is intended to automate a currently manual operation; a human operator adjusts the opening percentage of the distribution structure to feed as much flow into Omløbsåen without causing a flood; the excess flow is then shunted into Mølleåen.

In order to understand the frequent occurrence of flooding events in Vejle, it is necessary to understand the geography of Vejle and Vejle Kommune as a whole. According to the cooperative partner, the majority of floods occur in the area of Abelones Plads, where Mølleåen tends to exceed its bank given suffient flow. The

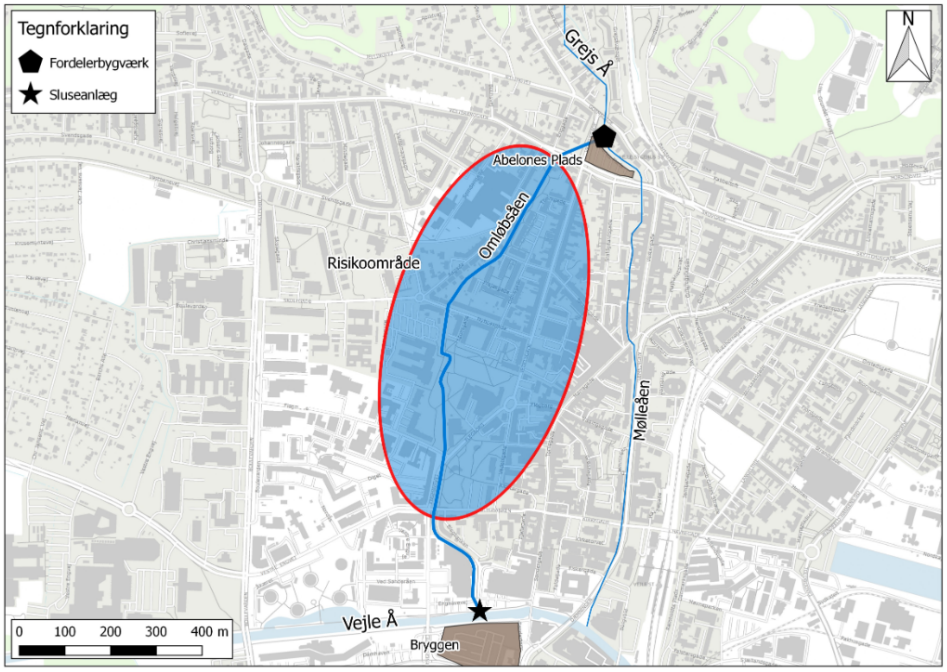


Figure 1.1: Map of Vejle with the locations of the water level measurements stations marked. *Risikoområde* means risk environment. Figure taken from [39, Figure 2.2]

excess water drains into Abelones Plads and then flows down the streets of Vedelsgade and Staldgårdsgade to finally drain into Omløbsåen near Frejasgade.

However, floods at Abelones Plads cannot be prevented entirely by automating the distribution structure; this only allows minimising the flow into Mølleåen by maximising the capacity of Omløbsåen. If the flow through Grejs Å exceeds the combined capacity of Omløbsåen and Mølleåen, a flood must occur and it will, generally, occur near Abelones Plads. Figure 1.1 provides a map of the city Vejle and 1.2 provides a map of Vejle Kommune with the relevant locations marked.

Because of the rapid response of the Grejs Å catchment and the availability of a 2-hour precipitation rainfall, we will limit the water level forecast to a maximum of 4 hours.

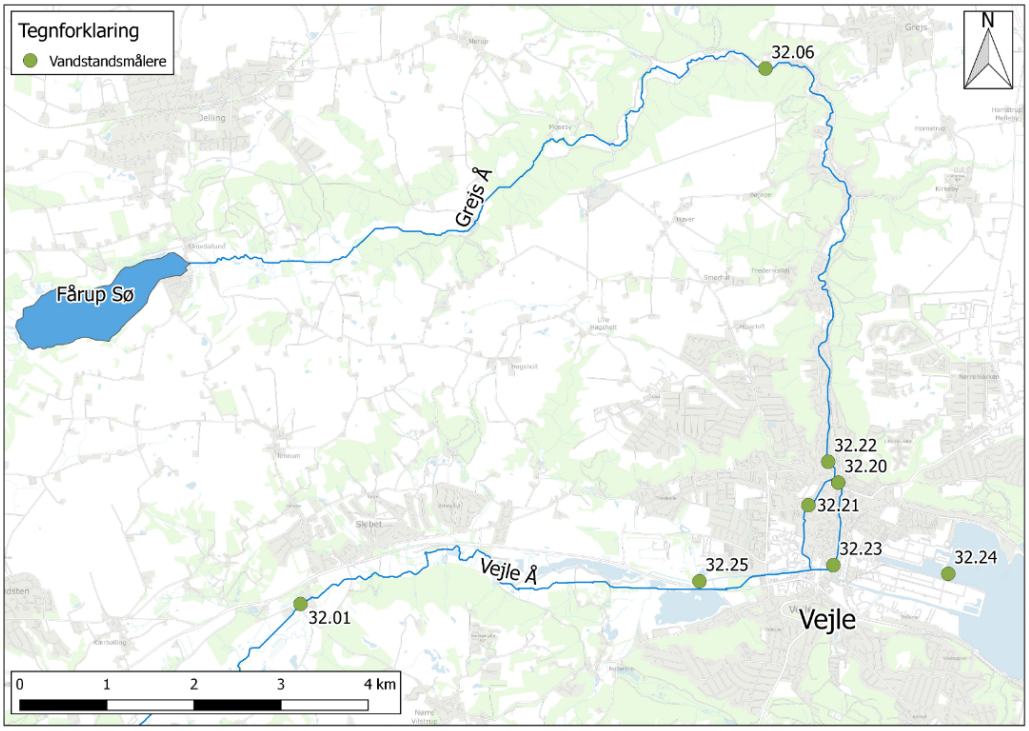


Figure 1.2: Map of Vejle Kommune with the locations of the water level measurement stations marked. Figure taken from [39, Figure 2.1]

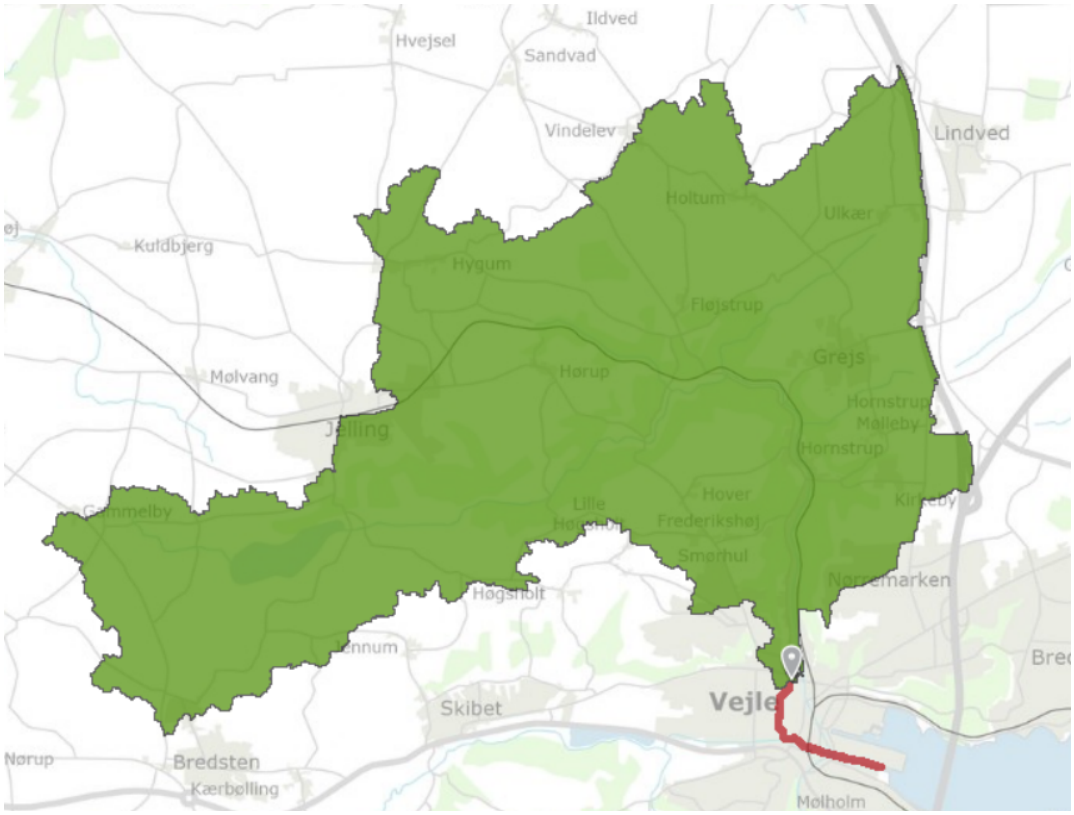


Figure 1.3: Catchment of the Grejs Å until the fordelerbygværk. Figure was obtained from Scalgo Live.[40]

1.4 Structure of the Report

The following report is divided into five chapters. We will begin chapter 2 by discussing the data obtained for this project with a focus on the normalisation procedure for each different type. In chapter 3, We will then introduce a method to automatically select an optimal set of moving averages applied on the rainfall measurements and provide the theoretical basis for the developed heuristic algorithm. We then introduce the LSTM network used in this project in 4 which was applied successfully in prior work[11] and show that the LSTM is a universal approximator for dynamical systems, as well as introducing the Residual LSTM we will apply. In 5, we will discuss how to identify events from data, classify them according to their characteristics and how the neural network training procedure is implemented with a focus on the validation loss. In chapter 6, we will evaluate the effect of a number of hyperparameters onto the validation loss, obtain the best performing model and analyse its performance over the validation events and over the entire dataset. We will further investigate the reasons for any observed network failure in individual events.

Finally, we will summarise and evaluate the model performance in chapter 7 and discuss possible future work in chapter 8.

Chapter 2

Data Processing

In this chapter, we will first discuss the different measurements that have been made available to the author by Vejle Spildevand A/S and discuss their relevance to flood prediction with respect to different prediction times. We will also discuss the peculiarities involved with processing of the different measurements.

2.1 Water Level Measurements

Along the streams of Grejs Å, Omløbsåen and Mølleåen, there are six water level measurements available. The records start in December 2015 with an initial sampling time of 15 minutes, later on 5 minutes. In order to obtain a uniform sampling time of the data, we upsample the initial period to a sampling time of 5 minutes using linear interpolation. We further interpolate missing periods of data when those periods are at most 5 hours long. This can be justified by the slow dynamics of the system. Furthermore, all normalised measurements below zero were replaced with linear interpolations. It should be noted that the water level measurements are provided in local time (i.e. CET/CEST) and are converted to UTC for ease of usage.

Table 2.1 provides the names, locations as well as the ground and bank levels used for normalisation following the procedure 2.1.

$$x_n = \frac{x - x_{ground}}{x_{bank} - x_{ground}} \tag{2.1}$$

Name	Location	Ground Level [m]	Bank Level [m]
ST.32.06	Grejsdalens Planteskole	27.35	29.4
ST.32.20	Abelones Plads	1.47	2.8
ST.32.21	Aagade	0.28	1.4
ST.32.22	Ny Grejsdalsvej	1.67	3.6
ST.32.23	Mølleåen Dæmningen	-0.46	2

Table 2.1: Measurements stations in Vejle, excluding those along Vejle Å. Ground and bank levels were provided by Benjamin Refsgaard.

2.2 Rainfall Measurements

In order to accomplish long-term predictions, rainfall measurements are needed. Within Vejle Kommune, there are several gauges located. Those gauges utilise a tipping bucket mechanism with 0.2 millimetre buckets. The number of tips is counted each minute to generate the rainfall measurements. In order to obtain the same sampling time as the water level measurements, we keep the rainfall measurements at the same time indices and to those add the four preceding rainfall measurements respectively. The rainfall measurements from the gauges are available since 2000 and are given in UTC. It is noted here that the measurements of the rainfall sampled over 5 minute intervals exhibit significant left skew, which may pose issues in the neural network training process. We normalise using a log transform as given in 2.2, where k is computed based on the rounded maximum measurement. The transform maps a measurement in the range $x \in [0, \max(x)]$ to the range $x_n \in [0, 1]$. Table 2.2 provides the rounded maximum values for a selected set of moving average window lengths.

$$\begin{aligned} x_n &= \ln(1 + kx) \\ k &= \frac{1}{\text{round}(\max(x))}(e - 1) \end{aligned} \tag{2.2}$$

Window Length [min]	Rounded maximum [mm]
10	618
20	420
25	344
30	290
50	174
60	145
120	77
180	53
205	49
240	45
300	38

Table 2.2: Rainfall moving average windows and associated rounded maximum values used for log normalisation for selected window lengths (including the window lengths generated by the feature selection algorithm, see section 3.4)

2.3 Radar Measurements

While the rain gauges can accurately capture the temporal variability in the rainfall, they do not capture the spatial variability of the rainfall to the same extent as radar data as the radar generates a 2d image of rainfall data over a large area. The cooperative partner stated that there are around 1200 radar measurements available over the catchment of the Grejs Å. This means that a neural network based on rain gauge data will perform well in scenarios that exhibit uniform rainfall but its predictive capability will degrade significantly in scenarios described by a high spatial variability in the rainfall. For example, the system may be driven by rainfall that is not measured by the rain gauges. In that case, the network will under-predict the water level. This spatial variability can be captured using radar measurements of the Grejs Å catchment which are originally measured with a spatial-temporal resolution of 1 kilometre and 10 minutes which were then up-sampled to a resolution of 200 metres and 1 minute. While the radar measures reflectivity, this is related to the rainfall by the semi-empirical Z-R relationship 2.3, where a, b are semi-empirical coefficients that depend on the type of rainfall as well as the raindrop size distribution but are generally estimated as constants $a = 200$ and $b = 1.6$.[24]

$$Z = aR^b \tag{2.3}$$

Radar data is not available for this project and we are limited to the rain gauge located in the parking lot of Vejle Spildevand A/S at Toldbodvej 20, 7100 Vejle. We point out that this rain gauge is not located in the catchment of the Grejs Å

but rather south of Vejle Å. The effect of this onto the prediction quality will be investigated in this report.

2.4 Yearly Variation

It is known that rainfall in the winter causes a higher increase in water level compared to rainfall in the summer. It was explained by the cooperative partner that in the winter, the soil can store little water and any rainfall will proceed to flow into Vejle, whereas in the summer, the storage capacity of the soil will capture a portion of the incoming rainfall.

Beyond temperature alone, there are a number of correlated variables such as air humidity, evaporation, sunlight intensity etc. that express seasonal behaviour with a yearly period. As these inputs were available, we instead opted to accommodate the seasonality by providing the neural network model with an artificial measurement called *yearly variation*, denoted as t_A as given in 2.4, where t_{year} denotes the time that has passed since the beginning of the current year (YYYY.01.01 00:00:00), and τ_{year} denotes the duration of the year. Note that the argument of the cosine must be dimensionless, i.e. the units of t_{year}, τ_{year} must agree.

$$t_A = \frac{1}{2}\cos\left(\frac{t_{year}}{\tau_{year}}\right) + \frac{1}{2} \quad \in [0,1] \tag{2.4}$$

The yearly variation t_A is high in the winter and low in the summer. A visual representation of this artificial input is provided in 2.1. In the absence of inputs with a seasonal dependency and a direct physical interaction with the system, the yearly variation will be given to the LSTM as an additional input to account for the effect of seasonality.

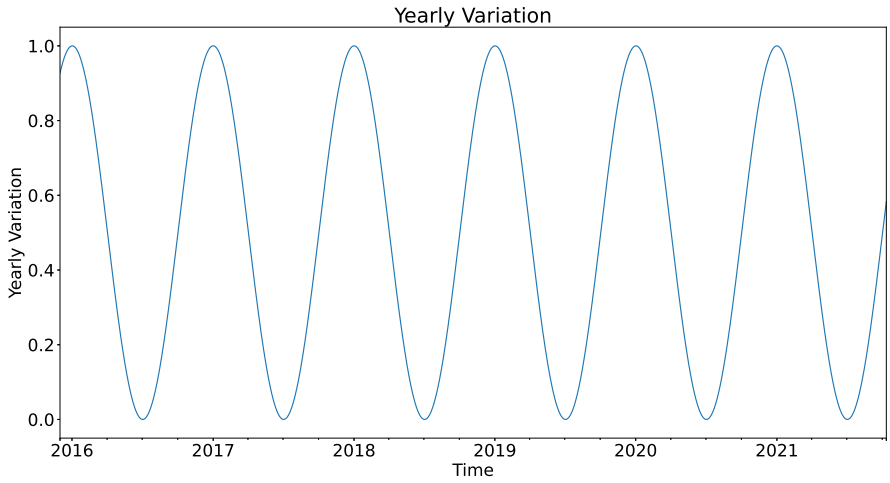


Figure 2.1: Yearly variation over the entire dataset

Chapter 3

Feature Selection using Multiple Spearman Correlation

Correlation Analysis shows that the instantaneous rainfall measurement reported by the rain gauge located in the parking lot of Vejle Spildevand A/S is nearly uncorrelated with the water level. The correlation coefficient was found to be $r \approx 0.06$ over the entire the entire dataset and $r \approx 0.13$ when only considering the events. The small degree of correlation should be understood in the context that the water level at Abelones Plads is driven by the water stored in the catchment and the instantaneous rainfall measurement is a part of time derivative of the water storage level. This complicates training as from the neural network perspective, the instantaneous rainfall measurement is effectively noise. Furthermore, the rainfall measurements exhibit significant left skewness.

Since the driver of the water levels within near Abelones Plads is the stored water within the catchment of the Grejs Å, it is reasonable to compute the accumulation of rainfall over different intervals in the past. This technique was used successfully by Benjamin Refsgaard in prior work[39], who utilised contiguous, non-overlapping moving average blocks. With this approach, it is necessary to determine which moving average blocks to use and in what temporal resolution. A method for feature selection using multiple correlation analysis has been developed and will be presented below. We will begin with some theoretical background that allows us to use the multiple correlation coefficient as a measure of information carried by the rainfall with respect to the change in water level, if the two are related by a system governed by mostly monotonous dynamics. We will show that this condition generally holds for the system of the change of water level caused by rainfall. Then, we will show how the optimal method may be implemented and provide a suboptimal but computationally more efficient heuristic method.

3.1 Approximation of the Information Coefficient

Suppose we define the *information coefficient* $IC(y|x)$, which is a generalised version of the coefficient of determination R^2. It describes the ratio of variance in y that can be explained given x. This specifies the upper bound of the model performance in terms of variance. However, the IC is only a theoretical concept and there is no way to obtain it for a general system. The complement of the IC might be easier to understand; the *fraction of unexplained variance* which is given as $FVU(y|x) = 1 - IC(y|x)$. This denotes the ratio of variance of y that cannot be causally explained given x.

As an example: consider $y = x_1 + x_2$, where x_1, x_2 are independent of each other and have the same distribution, i.e. $Var[x_1] = Var[x_2]$ and $Var[y] = 2Var[x_1]$. Suppose, you attempt to make a prediction of y given only x_1. In that case, you can compute the IC through the Pearson correlation coefficient as $IC(y|x_1) = 0.5$. This allows you to claim that your predictor $\widehat{y} = f(x_1)$ can at most explain 50% of the variance of y, the remaining 50% cannot be causally explained given only x_1. A model with a better performance necessarily requires additional information to make the prediction. It is trivial to see that $IC(y|x_1, x_2) = 1$ and thus the model $\widehat{y} = f(x_1, x_2)$ can explain 100% of the variance of y given x_1, x_2.

It can be seen that $IC(y|x)$ specifies the amount of causal information carried by x with respect to y. To our knowledge, no formula exists that captures the entire nonlinear, asymmetric dependency of $y = f(x)$. However, we can approximate the information coefficient from correlation coefficients. For example, suppose all dependencies are linear. In that case the square of the Pearson correlation coefficient can be used to estimate $IC(y|x) \approx R^2(y|x)$.

It is known that the system is nonlinear (for example, the flow velocity increases as the water level increases), therefore the Pearson R-squared value will underestimate the information coefficient. An alternative is a rank correlation coefficient like Spearman's ρ given in 3.1, which is effectively the Pearson correlation coefficient computed on the rankings of x, y. The benefit of the Spearman correlation is that it captures nonlinear, monotonous dependency. For example, given $y = x^3$, with $x \sim \mathcal{U}(-0.5, 0.5)$, the Spearman Rho-squared is $\rho^2 = 1$, while the Pearson R-squared is $r^2 = 0.84$. Since, given x, one can compute y, the information coefficient is $IC(y|x) = 1$. For this nonlinear, monotonous function, the Spearman Rho-squared is more accurate approximation compared to the Pearson R-squared which misses 16% of the dependency that we know to exist.

$$\rho_{y,xi} = \frac{\mathrm{Cov}(R(y), R(x_i))}{\sigma(R(y))\sigma(R(x_i))} \tag{3.1}$$

For a multivariate input $x = \begin{bmatrix} x_0 & \dots & x_{M-1} \end{bmatrix}^T$, the *Multiple Spearman Rho-squared* can be computed using the pairwise Spearman correlation coefficients as given in

3.2. If the input variables in x are independent, then $P_{xx} = I_M$ and $\rho^2 = \sum_{i=0}^{M-1} \rho_{xi,y}^2$.

$$\begin{aligned} IC(y|x) \approx \rho^2 &= \rho_{xy}^T P_{xx}^{-1} \rho_{xy} \\ \rho_{xy}(i) &= \left[\rho_{xi,y}\right] \\ P_{xx}(i,j) &= \left[\rho_{xi,xj}\right] \end{aligned} \tag{3.2}$$

3.2 Proof of Monotonicity

It has been shown that the square of the Spearman rank correlation coefficient ρ is an approximation of the information coefficient if the function f that maps x to y is monotonous. In order to base a method of feature selection on the multiple Spearman R-squared, we need to show that the water level is a monotonous function of the rainfall.

This can be approached intuitively through a thought experiment (see theorem 1). Let $r_1(t)$ be the rainfall curve and let $w_1(t)$ be the water level curve caused by the rainfall through an unknown physical system $r(t) \to w(t)$. Suppose a different rainfall curve $r_2(t)$ which dominates $r_1(t)$, i.e. $r_2(t) \geq r_1(t) \quad \forall t$ and $r_2(t) > r_1(t)$ for some value(s) of t. Then, monotonicity is given if and only if the water level $w_2(t)$ (caused by $r_2(t)$) dominates $w_1(t)$. We assume based on our intuition that this should generally hold. A strict proof is not given and not required either; simply showing that a majority of the system dynamics are monotonous with respect to the rainfall is sufficient.

Theorem 1 *The system $r(t) \to w(t)$ is monotonous, if and only if the following condition is satisfied:*

$$\begin{aligned} w_2(t) &\begin{cases} \geq w_1(t) \text{ if } r_2(t) \geq r_1(t) \\ \leq w_1(t) \text{ if } r_2(t) \leq r_1(t) \end{cases} \\ r_i(t) &\to w_i(t) \quad i = 1,2 \end{aligned} \tag{3.3}$$

We can also approach this from the perspective of mass flow and mass storage during a steady state situation. The water level at Abelones Plads is a function of the local mass flow. Due to conservation of mass, the sum of mass flows through Mølleåen and Omløbsåen, and the mass input through rainfall must be equal, assuming the system is in its steady state. The total mass storage in the system is the result of the integration of the difference between mass input and mass output. If the water level at Abelones Plads is constant, this implies that the mass storage is constant. If the rainfall now increases, the steady state of the water level has to increase due to conservation of mass. Note, water level is a nonlinear, monotonous function of mass flow.

These considerations imply that the system has a global equilibrium point which is a monotonous function of the rainfall (as well as the water level in the harbour, as outflow depends on it). This leads us to assume, in general, an increase in rainfall somewhere causes an increase in water level somewhere, which aligns with theorem 1.

3.3 Optimal Dimensionality Reduction based on the Multiple Rho-squared

In this section, we will derive the optimal dimensionality reduction method (optimal in the sense of the least reduction of the ρ^2 for a reduction of a given number of dimensions; i.e. given 10 rainfall features, what are the set of 10 features that maximise the ρ^2).

Let $m_k^{(i)}$ be the value of the rainfall feature i at time k and Let L_i be the length of the moving average block (in samples) that generates feature i; we might initialise this as $L_i = 1$.

We then compute the Multiple Rho-squared for the whole set of features $IC(y|m)$, where $m = \{m^{(i)}|i = 0, .., M-1\}$ is the set of moving averages. For $|m|$ moving average blocks, there are $|m| - 1$ pairs of neighbours that can be combined. For every pair $(i, i+1)$, we compute the average weighted by L, replace element i in m, L with the new average and length and drop element $i+1$ in m, L. Let the new set of features be $m^{(1,i)}$, where i denotes the first element of the pair that is combined. We compute the set of information coefficients $IC(y|m^{(1,j)})$ and then select the largest value. We iteratively choose the combination that yields the highest IC, i.e. the combination with the least loss of information carried by the set of features with respect to y until some user-specified termination criterion is met.

This algorithm has two main problems, the computational complexity and the numerical precision issues both associated with inverting the correlation matrix R_{xy}. Consider that the optimal algorithm starts with a high number of features and then iteratively combines them to reach a compact representation that captures the majority of information carried by the rainfall with respect to the water level prediction to be made.

3.4 Heuristic Feature Selection Algorithm

Due to the computational issues faced by the optimal algorithm, a less optimal, but significantly more efficient heuristic algorithm was developed and will be introduced in this section. We have reason to assume that monotonocity holds generally. From prior experience of working with water level prediction in Vejle[11], we know that the system exhibits local first-order-like behaviour. We note here

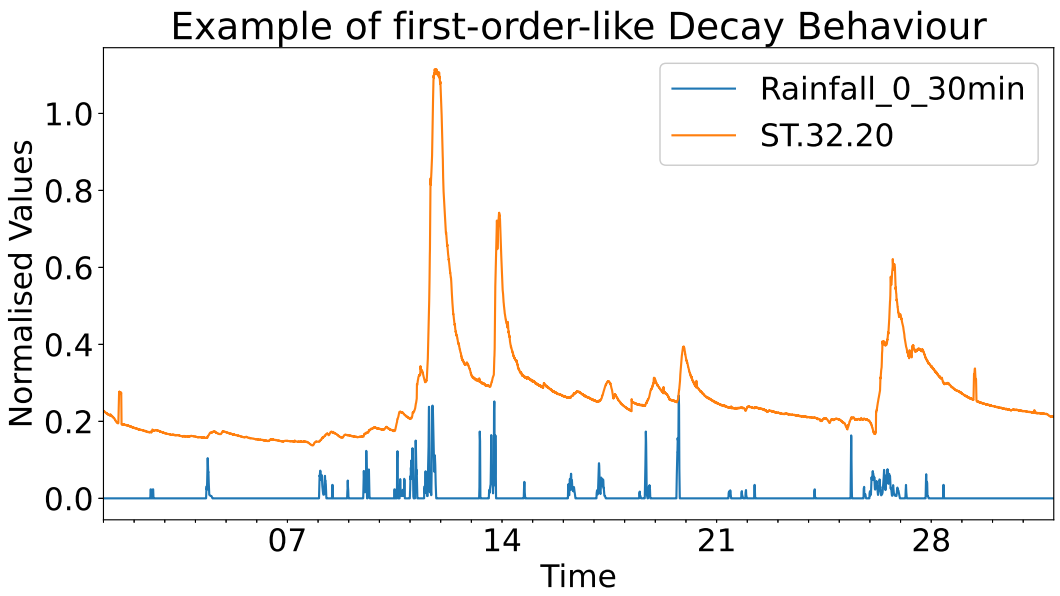


Figure 3.1: Example of first-order-like decay behaviour. Figure shows the water level at Abelones Plads and the log-normalised rainfall accumulated over the last 30 minutes.

that the system may potentially be better understood as a damped higher-order system, but even then, monotonicity still holds. Figure 3.1 shows a first-order-like decay to the base water level after the rainfall has ended.

Since the decay behaviour appears like that of a first-order system, we expect a roughly exponentially decaying impulse response, i.e. the concentration of information carried by a measurement with respect to the water level reduces as we progress backwards in time. Therefore, it intuitively makes sense that the features start short and become increasingly long, since merging neighbouring samples will lose information and therefore merging low-information samples loses less information.

Since the most relevant information is found in the recent past, we can iterate along the tapped delay line and create the moving average blocks in sequence instead of creating all possible blocks at once and then merging them. This greatly reduces computational cost. It furthermore handles the question of how long the tapped delay line should be.

Suppose we use a set of moving average blocks as well as the yearly variation t_A to compute the change in water level. In the winter, rainfall causes a higher increase than the same rainfall would cause in the summer, so we assume that a significant portion of the dependency between t_A and the water level is also monotonous, therefore the Spearman rank correlation coefficient is a meaningful approximation of the information coefficient 3.4.

$$IC(\Delta w|t_A, m) \approx \rho^2(\Delta w|t_A, m) \tag{3.4}$$

We define the shorthand $\rho^2(i)$ which is the Spearman Rho-squared of Δw with respect to the input set $\{t_A, m_0, ..., m_i\}$. We then define the *added information* $\Delta\rho(i) = \Delta\rho(i) - \Delta\rho(i-1)$, which denotes the information of Δw we gained by adding m_i to the set of inputs. Since the impulse response exhibits exponential decay, we expect $\Delta\rho^2(i) > \Delta\rho^2(i+1)$ (the difference between the two will decrease as i increases). In order to achieve a dimensionality reduction, we merge blocks that add little in terms of non-redundant information. Suppose we define the *minimum added information* $\Delta\rho^2_{min} = 0.02$ which enforces that all inputs add at least 2% monotonous explanatory power. If a block does not satisfy this requirement, it is extended into the past.

If rainfall falling within either the block or the sample(s) to be included in the block, has a similar effect onto the water level that we are attempting to predict, then extending the block will increase the added information. Note: $\Delta\rho^2(0) = \rho^2(0) - \rho^2(\Delta w|t_A)$.

The heuristic algorithm is given below. A simplified flowchart is given in 3.2.

1. Specify $\Delta\rho^2_{min}, L_{min}, \epsilon, i = 0$ and set $L = \{\}$.
2. Create set of inputs $\{t_A, m_j | j \in [0, i]\}$
3. Set $l = L_{min}$ and add l to L
4. compute $\Delta\rho^2(i, l)$
5. if $\Delta\rho^2(i, l) \geq \Delta\rho^2_{min}$, set $i = i + 1$, add m_i to m, add $L_i = L_{i-1}$ to L, else set $l = l + 1$ and $L_i = l$
6. if $l \geq 10$ and $\Delta\rho^2(i, l) < \max_{j=0}^{l-1} \Delta\rho^2(i, j)$, remove m_i, L_i from m, L and terminate algorithm.
7. if $l \geq L_{max}$:
   (a) if also $\Delta\rho^2(i, l) \geq \epsilon$, terminate algorithm
   (b) else, remove m_i, L_i from m, L and terminate algorithm.
8. return to 2.

3.5 Results

Consider the set of auxiliary inputs $\widetilde{X}$. We know that the impact of the rainfall onto the water level change is dependent on the season, thus we include the yearly

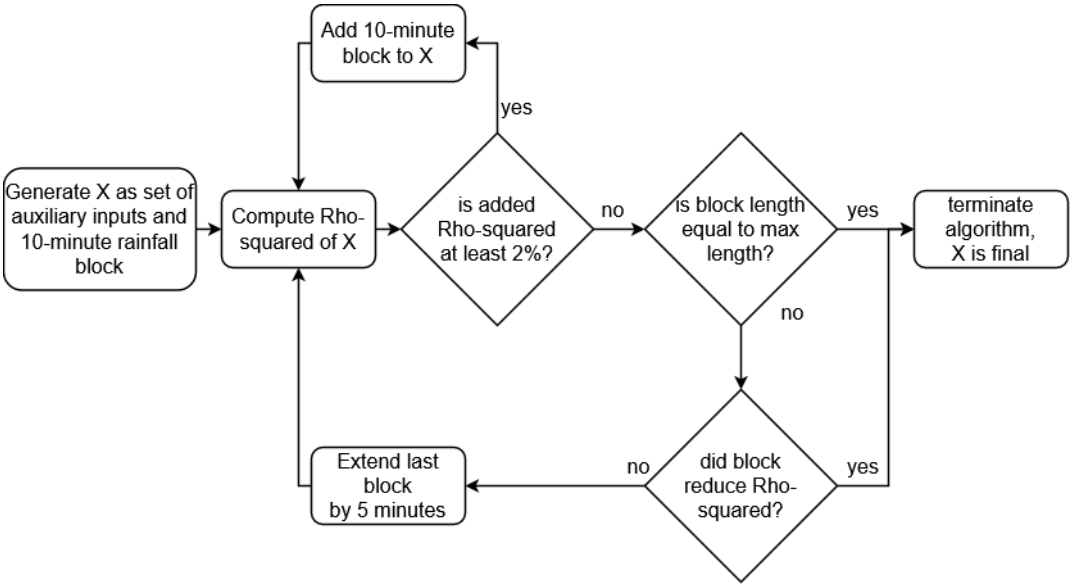


Figure 3.2: Flowchart of the feature selection algorithm given in simplified form for reasons of readability.

variation t_A in the set of inputs $\widetilde{X}$. Since t_A changes very slowly, it is effectively a different constant for each event, therefore no processing is necessary.

It is known from prior work[11], that the water level at one point can be predicted with a short prediction time using only one water level measurement from an upstream location and one from a downstream location. Since in this work the neural network predicts the change in the water level, it is useful to provide the water level inputs as differences as well. In order to find the best interval ΔT over which to compute the change in water level $\Delta w[k] = w[k] - w[k - \Delta T]$ for the measurements from Planteskole and Mølleåen Dæmningen, we compute the Spearman Rho-squared value of the water level change for prediction times ranging from 30 minutes to 4 hours with respect to the change of the two upstream/downstream water levels over intervals of length 5 minutes to 12 hours. Note, that the upstream water level change carries significantly more information than the downstream water level change. This is because the change at Abelones Plads lags behind the change at Planteskole but leads the change at Mølleåen Dæmningen. In prior work[11], it was established that for prediction at Aagade, the downstream water level was in fact necessary because Aagade was low enough to be affected by tides, which are not present at Abelones Plads (which has a ground level of 1.47 m). Figures 3.3,3.4 show the Rho-squared curves for both locations. It can be seen that the Rho-squared values increases briefly (likely due to the fact that differences for small interval length are affected by noise) and then decreases as the interval grows in length.

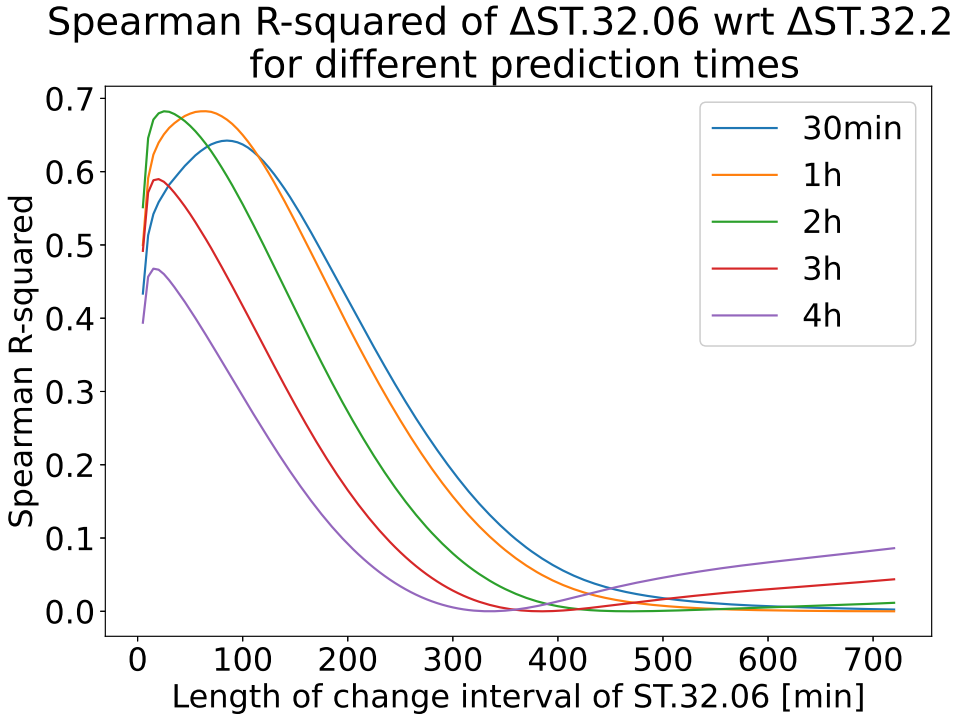


Figure 3.3: Spearman R-squared of the change of the water level at Abelones Plads over different prediction times (30 minutes to 4 hours) with respect to the change of the water level at Grejsdalens Planteskole over intervals of 5 minutes to 12 hours

For reasons of simplicity, we propose using one 30-minute interval for both measurements, i.e. the change in water level at Planteskole and Mølleåen Dæmningen over the past 30 minutes. Since the water level behaviour changes very slowly, it is considered unnecessary to add blocks further into the past. Let $\widetilde{X}$ be a set containing the yearly variation t_A as well as the change in water level at Planteskole and Dæmningnen over the last 30 minutes.

Given the established set of auxiliary inputs $\widetilde{X}$, we seek an acceptably optimal set of moving averages to generate the complete set of inputs $X = \{\widetilde{X}, M\}$ in the sense of $\rho^2(Y|X)$. Consider that the information carried by the water level difference blocks with respect to the target Y decreases as the prediction time increases (the further we try to predict into the future, the more we need to rely on rainfall measurements). We therefore apply the heuristic algorithm developed in 3.4 onto the change of the water level at Abelones Plads for a prediction time of 4 hours into the future given the auxiliary inputs $\widetilde{X}$. We further set the minimum information added to 2%.

The output of the heuristic algorithm is given in 3.1, which gives the block lengths and types (mean or difference). Note that the blocks are always sequential with no space between blocks and no overlaps. The block number indicates the

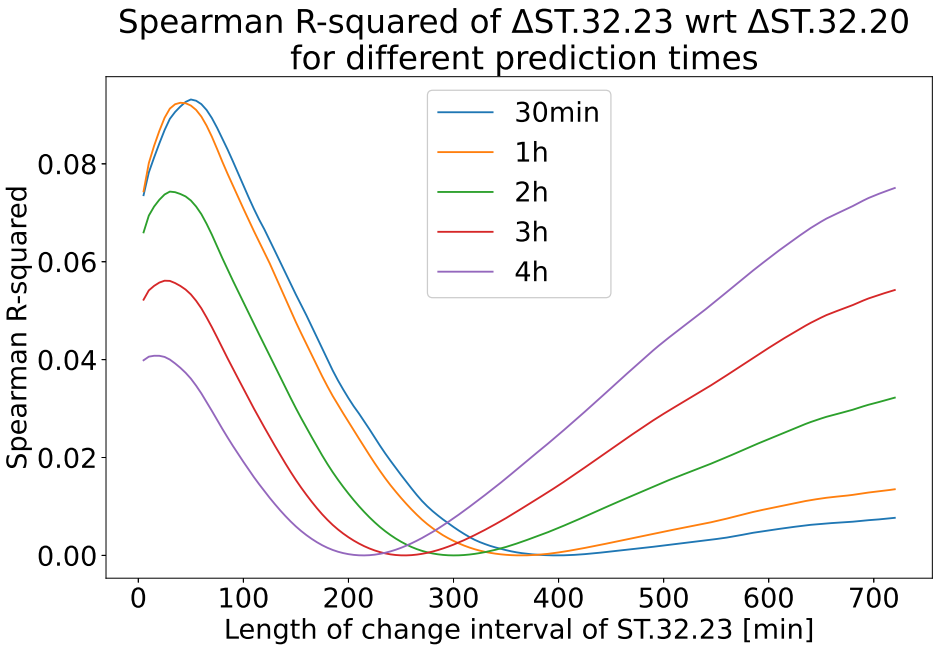


Figure 3.4: Spearman R-squared of the change of the water level at Abelones Plads over different prediction times (30 minutes to 4 hours) with respect to the change of the water level at Mølleåen Dæmningen over intervals of 5 minutes to 12 hours

position of the block along the tapped delay line; the block most recent in the past is denoted as 0 and always takes the current measurement as its first element of the block. For example, the input denoted by 'Delta_ST.32.06_0_30min' is the current water level at Planteskole minus the water level at Planteskole 30 minutes ago. Similarly, 'Rainfall+2h_0_10min' is the mean of the rainfall forecast at the gauge 2 hours into the future in the 10-minute interval $[t + 120min, t + 111min]$. Note that the rainfall tapped delay line effectively consists of a 2-hour window of forecasted data followed by a tapped delay line containing the rainfall measurements.

The table shows that the generated set of input features allow us to explain 69.73% of the variance of the water level change at Abelones Plads 4 hours into the future. The Spearman R-squared $\rho^2(Y|X)$ only accounts for monotonous, non-recurrent information. The LSTM should be capable of extracting the unmeasured dependency (non-monotonous and recurrent) and therefore should be able to outperform this measure of dependency.

As a measure of confidence, we took the total rainfall window considered by our set of inputs X (10 hours and 10 minutes) and split it up into 61 10-minute blocks (the smallest block size generated by the algorithm) and compute the multiple Rho-squared as 71.85%. The heuristic algorithm captures 2.5% monotonic information less (equivalent to 3.45% loss of relative information), while using 6

Input	Block	Block Length	Block Type	ρ^2	$\Delta\rho^2$
t_A	-	-	-	0.017	0.017
ST.32.23	0	30 min	Difference	0.0394	0.0411
ST.32.06	0	30 min	Difference	0.4108	0.4519
Rainfall+2h	0	10 min	Mean	0.1325	0.5844
Rainfall+2h	1	20 min	Mean	0.0217	0.6061
Rainfall+2h	2	25 min	Mean	0.0229	0.6290
Rainfall+2h	3	50 min	Mean	0.0208	0.6498
Rainfall+2h	4	300 min	Mean	0.0205	0.6703
Rainfall+2h	5	205 min	Mean	0.0200	0.6973

Table 3.1: Total Spearman R-squared and added Spearman R-squared for the set of inputs with respect to the water level change at Abelones Plads given a prediction time of 4 hours.

rainfall measurements instead of 61. We note here, that the loss of information is likely irrelevant due to the LSTM's ability to maintain memory and thus extract relationships between past values of the rainfall blocks and the prediction is trained to make. We conclude that the suboptimal heuristic algorithm developed in 3.4 is sufficient for our purposes.

Chapter 4

Neural Network Architectures

In this chapter, we will discuss different neural network architectures that may be applied to the problem of flood forecasting and discuss their internal processes in detail. We will also discuss the Universal Approximation Theorem in detail and show that the LSTM is a universal approximator for dynamical systems. This chapter is meant to give the reader a detailed understanding about the implementation of the neural networks that were trained during the course of this project.

4.1 Feed-Forward Neural Networks

The simplest neural network structure is the *Feed-Forward Neural Network* which gains its name based on the fact that it only feeds through the input and maintains no memory. The output at time k is only a function of the input at time k. This type of network architecture is generally implemented as a set of densely connected layers with a nonlinear activation function (such as ReLU). An implementation of this structure with one hidden layer is given in 4.1, where x_k, h_k, y_k are the input, hidden and output signals, W_h, b_h, W_y, b_y the network parameters and f a nonlinear activation function (such as ReLU) applied element-wise.

$$\begin{aligned} h_k &= f(W_h x_k + b_h) \\ y_k &= W_y h_k + b_y \end{aligned} \tag{4.1}$$

It is important to note, that f must be a nonlinear function. If f is omitted, then 4.1 simplifies to 4.2, which shows that a linear Feedforward Neural Network with any number of hidden layers is equivalent to a linear transformation $y_k = Ax_k + b$.

$$y_{2,k} = A_2 A_1 x_k + A_2 b_1 + b_2 = Ax_k + b \tag{4.2}$$

It is known that the single hidden layer Feed-Forward Neural Network is a universal approximator for continuous functions in a bounded region. Impor-

tantly, Kolmogorov's Superposition theorem[27] shows that any multivariate function $f(x) = f(x_1, ..., x_n)$ can be represented as a superposition of univariate functions as given in 4.3.

$$f(x) = f(x_1, ..., x_n) = \sum_{q=0}^{2n} \Phi_q \left(\sum_{p=1}^{n} \phi q, p(x_p) \right) \tag{4.3}$$

Lorentz, and Sprecher then showed that the functions $\Phi_q, \phi_{q,p}$ do not need to be different functions and proved that $f(x)$ can be represented as given in 4.4.[33][45]

$$f(x) = f(x_1, ..., x_n) = \sum_{q=0}^{2n} \Phi \left(\sum_{p=1}^{n} \lambda_p \phi(x_p + \mu q) + q \right) \tag{4.4}$$

It is well established that Feedforward Neural Networks with at least one hidden layer are capable of approximating all continuous functions to any degree of accuracy given a sufficient amount of neurons in the hidden layer.[20][5][12][9]. In the context of dynamical systems, this means that if the system can be discretised as $x_{k+1} = f(x_k, u_k)$, a Feed-Forward Neural Network is sufficient to predict x_{k+1}. Furthermore, if there is a mapping of current and past states and inputs $z = g(x_k, x_{k-1}, ..., u_k, u_{k-1}, ...)$ such that $x_{k+1} = f(z_k)$, and the mapping g is available (for example, those can be moving averages), then the Universal Approximator Theorem still applies and the dynamical system can be modelled accurately. This means that in practice, such a network is highly dependent on what inputs are provided, as no memory of past inputs are maintained. Benjamin Refsgaard showed that prediction of water levels in Vejle can be done with a Feed-Forward Neural Network when rainfall inputs are provided as block-wise moving averages of 30 minute length.[39]

4.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are extensions of the aforementioned Feed-Forward Neural Network. It uses the same feed-forward structure with added terms that relate the current activations to the previous activations. A simple version (known as the Elman Network) is given in 4.5, where x_k, h_k, y_k are the input, hidden state and output signals, W_h, b_h, W_y, b_y are the model parameters and f_h, f_y are activation functions.

$$\begin{aligned} h_k &= f_h(W_h x_k + U_h h_{k-1} + b_h) \\ y_k &= f_y(W_y h_k + b_y) \end{aligned} \tag{4.5}$$

It has been shown that a single hidden layer RNN is a universal approximator for a dynamical system with continuous derivatives[42]. In practice, however, the

RNN suffers from the problem of exploding and vanishing gradients which is caused by the state signal circling through the static gain U_h (see 4.5. Consider the sample case of such an RNN with a single state. In that case, U_h is scalar. If $U_h < 1$, then the gradient decays exponentially; if $U_h > 1$, then the gradient grows exponentially. This causes a significant problem for the training of an RNN. If the gradient explodes, the training process may become unstable. If the gradient vanishes, the RNN is unable to learn the dynamics associated with those gradients. In practice, this hampers the RNNs ability to learn dynamics that require long-term memory.

4.3 Long-Short Term Memory

Due to the aforementioned problems faced by RNNs, Schmidhuber and Hochreiter developed the *Long-Short Term Memory* (LSTM)[19], which evades the problem of vanishing and exploding gradients by gating their internal state. The internal working process of an LSTM layer is given in 4.6, where x_k, h_k, c_k denote the input, hidden states and cell states, i_k, f_k, g_k, o_k denote the input, forget, cell and output gates, $\sigma, \tanh$ denote the sigmoid and hyperbolic tangent functions and $\odot$ denotes the Hadamard (element-wise) product.

$$\begin{aligned}
i_k &= \sigma(W_{ii}x_k + b_{ii} + W_{hi}h_{k-1} + b_{hi}) \\
f_k &= \sigma(W_{if}x_k + b_{if} + W_{hf}h_{k-1} + b_{hf}) \\
g_k &= \tanh(W_{ig}x_k + b_{ig} + W_{hg}h_{k-1} + b_{hg}) \\
o_k &= \sigma(W_{io}x_k + b_{io} + W_{ho}h_{k-1} + b_{ho}) \\
c_k &= f_k \odot c_{k-1} + i_k \odot g_k \\
h_k &= o_k \odot \tanh(c_k)
\end{aligned} \tag{4.6}$$

Various different modifications of the LSTM structure have been developed, a notable one being the Gated Recurrent Unit (GRU)[6] which merges the hidden and cell states as well as the input and forget gates of the LSTM into an update gate. The GRU is governed by 4.7, where r_k, z_k, n_k are the reset, update and new gates, x_k, h_k are the input and hidden state, W, b denote weight matrices and biases, respectively, $\sigma, \tanh$ denote the sigmoid and hyperbolic tangent functions respectively and $\odot$ denotes the Hadamard (element-wise) product.

$$\begin{aligned}
r_k &= \sigma(W_{ir}x_k + b_{ir} + W_{hr}h_{k-1} + b_{hr}) \\
z_k &= \sigma(W_{iz}x_k + b_{iz} + W_{hz}h_{k-1} + b_{hz}) \\
n_k &= \tanh(W_{in}x_k + b_{in} + r_k \odot (W_{hn}h_{k-1} + b_{hn})) \\
h_k &= (1 - z_k) \odot n_k + z_k \odot h_{k-1}
\end{aligned} \tag{4.7}$$

4.3.1 Universal Approximation Theorem For LSTM

Consider an LSTM with a single hidden layer as given in 4.6 and an output layer governed by $y_k = W_y + h_k$. We can now recover the basic, single hidden layer Recurrent Neural Network by setting $f_k = 0$ and $i_k \odot g_k = c$, which yields a constant cell state $c_k = c$. This reduces 4.6 to 4.8.

$$\begin{aligned} h_k &= \tanh(c) \odot \sigma(W_{io} x_k + b_{io} + W_{ho} h_{k-1} + b_{ho}) \\ y_k &= W_y h_k \end{aligned} \tag{4.8}$$

Since $\tanh(c)$ is a constant term in 4.8, we define $h_k = \tanh(c) \odot h'_k$. This yields 4.9.

$$\begin{aligned} h'_k &= \sigma(W_{io} x_k + b_{io} + W_{ho} \tanh(c) \odot h'_{k-1}) \\ y_k &= W_y \tanh(c) \odot h'_k \end{aligned} \tag{4.9}$$

Since, h'_k is scaled element-wise by $\tanh(c)$, this can also be written as the multiplication with the diagonal matrix $\text{diag}(\tanh(c))h'_k = \tanh(c) \odot h'_k$. Inserting this equality into 4.9 yields 4.10, where $V = \text{diag}(\tanh(c))$ and $W_h = W_{ho}\,\text{diag}(\tanh(c))$. For the sake of brevity, we rename $W_i = W_{io}, b = b_{io} + b_{ho}$.

$$\begin{aligned} h'_k &= \sigma(W_i x_k + W_h h'_{k-1} + b) \\ y_k &= V_y h'_k \end{aligned} \tag{4.10}$$

It can be seen that 4.10 is the simple, single-hidden-layer Recurrent Neural Network with a sigmoid activation function. It is well known that such a network is a universal approximator for dynamical systems[42]. As we have shown, all single-hidden-layer RNNs of the structure 4.10 can be written as LSTM networks. Therefore, the single-hidden-layer RNN is a subset of the single-hidden-layer LSTM, i.e. if all dynamical systems can be represented by a single-hidden-layer RNN, the same must apply to the single-hidden-layer LSTM, i.e. the Universal Approximation Theorem holds for the single hidden layer LSTM to the exact same extent as to which it applies to the single hidden layer RNN.

4.3.2 Universal Approximation Theorem For GRU

Consider a single hidden layer GRU governed by 4.7 and an output layer $y_k = W_y h_k$. Similar to the preceding proof, we fix the reset update gates in place by setting $r_k = 1, z_k = 0$. This yields 4.11.

$$\begin{aligned} h_k &= \tanh(W_{in} x_k + W_{hn} h_{k-1} + b_{hn}) \\ y_k &= W_y h_k \end{aligned} \tag{4.11}$$

As we have shown the GRU is the superset of the class of RNNs with a single hidden layer and a hyperbolic tangent activation function. As proven by Zimmermann and Schäfer[42], this class of RNN is a universal approximator for dynamics systems. Note, that while Zimmermann and Schäfer gave proof for the Universal Approximator Theorem for RNNs with sigmoid functions, this refers not the function commonly denoted as *sigmoid* (more specifically known as the *logistic function*), but to the class of sigmoid functions (i.e. functions that are monotonic, differentiable and bounded) which includes the hyperbolic tangent function.

4.4 Residual Networks

He et al. introduced the *Residual Network*[16], an architecture which has been found to outperform and generalise better than deep stacked neural networks. He et al. hypothesised that it is easier for a neural network to learn residual mappings as opposed to unreferenced mappings (i.e. $y = f(x) + x$ as opposed to $y = f(x)$. The commonly understood working principle of Residual Networks is to learn pertubations from the identity functions. Hauser[15] showed that the norm of the pertubation decreases with block number (i.e. the early blocks create higher changes). Jastrzębski et al.[23] reported that the residual layers iteratively refine the estimate. Veit et al. showed that residual networks in practice behave like an ensemble of shallow networks. Siu showed that Residual Networks behave like a boosting algorithm, meaning when a shared linear classifier is applied on the output of the residual blocks, each will be an estimate of the target with decreasing error as we proceed along the feed-forward path. Wang et al. proposed that the effectiveness of residual connections can be explained by studying the loss landscape; they showed that the loss landscape of a deep residual network remains well-behaved when the number of layers is increased whereas the loss landscape becomes chaotic without the residual connections in place. Su et al[50] hypothesises that Residual Networks generalise better because they are less sensitive to noise. Arora et al.[3] showed that noise sensitivity is correlated to generalisability. This allows deep Residual Networks to generalise even though the number of parameters may be larger than the number of samples.

In addition to the Universal Approximation Theorems that were shown to hold for single-hidden-layer NNs with infinite width, Lin et al.[31] showed that a single-neuron Residual Network with infinite depth is also a universal approximator. It follows that a Residual Network with a fixed width can approximate any given function arbitrarily well given enough layers.

We discussed the problem of vanishing gradients in section 4.3. We note that the LSTM handles the problem of gradients vanishing along the recursive path. Recall from 4.6 that the signal has to pass through at least one sigmoid function to pass through one LSTM layer. The maximum gradient of the sigmoid function

is 0.25, which implies that for a deep stacked LSTM, we lose 75+% of the gradient for each layer, which introduces the problem of gradients vanishing along the feed-forward path. It has been shown that residual connections allow gradients to bypass nonlinear activation functions and thus the gradients do not degrade[47]. Zaeemzadeh et al.[51] showed that Residual Networks preserve the norm of the gradients and thus lead to a stable back-propagation process. This property enables Residual Networks to become very deep, compared to Deep Stacked Networks which become increasingly harder to train as depth increases. The original authors (He et al.[17]) showed that Residual Networks with non-identity mappings along the residual connections either perform worse than models with identity mappings or fail outright. This result underscores the importance of using as few non-identity mappings as possible.

In this work, we employ a Deep Residual LSTM network architecture which we refer to as *ResLSTM* based on an architecture proposed by Wang et al.[48]. It features residual connections connecting the inputs of each block to its output. Each block consists of one LSTM layer followed by a Dropout layer (see section 4.12). The residual connection employs a linear layer (with bias) if the block's input and output dimensions differ; if they are identical, the mapping is simply the identity function. After the final block, a linear output layer is employed to generate the predictions. All mappings along the residual connections except the first are identity mappings.

The application of Deep Residual LSTMs such as the ResLSTM used in this work appears to be sparse in the field of hydrology. A comprehensive 2020 review of machine learning in the field by Sit et al.[44] mentions Residual Networks only in the context of Residual CNNs being used for precipitation nowcasts.

Figure 4.1 gives a block diagram of a 2-layer ResLSTM network. Note that the only non-identity functions along the residual path are the initial mapping (required to change dimensions from the input dimension to the hidden dimension) and the output layer which is required to change the dimension to the output dimension. For an abitrary hidden dimension, this the minimum number of non-identity mappings along the residual path. Importantly, both are linear mappings which do not squash the gradient like the sigmoid functions inside the LSTM layers do.

4.5 Dropout

In this section, we will briefly explain the concept of Dropout since it is our method of choice of dealing with the problem of overfitting and thus obtaining a model that can generalise to unseen data. In prior work[11], it was observed that when using Dropout, the validation loss at one point levels out instead of increasing again.

Introduced by Hinton and Srivastava et al.[18][46], Dropout randomly drops

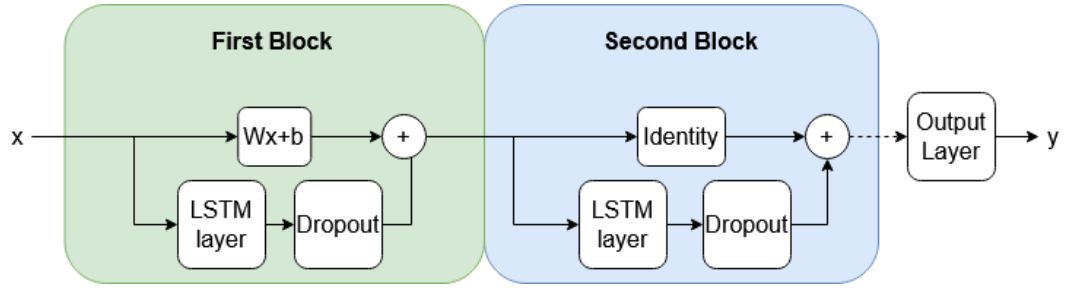


Figure 4.1: Example of a 2-layer ResLSTM network. Any potential further blocks will have the same structure as the second block.

out neurons during the forward passes in the training process as shown in 4.12, where y denotes the output of a layer (i.e. the input to the dropout layer), δ is a vector of Bernoulli random variables that is generated at the beginning of each iteration and $\tilde{y}$ is the output of the dropout layer.

$$\begin{aligned} \delta^{(l)} &\sim \text{Bernoulli}(p) \\ \tilde{y} &= \delta^{(l)} \odot y \end{aligned} \tag{4.12}$$

In each iteration, Dropout creates a different subnetwork by dropping out different neurons each iteration. At the end of the training process, the Dropout is deactivated and the weights are scaled down to account for the increase in the number of active neurons. This is similar to training an ensemble of neural networks and then averaging their outputs.

The exact reason for why Dropout works is not very well understood yet and several different explanations have been proposed. The original paper hypothesised that Dropout is effective because it reduces coadaption between neurons. Coadaption means a neuron is only helpful in the context of other neurons. Dropout then reduces coadaption by dropping those neurons out. The authors showed that dropout in effect leads to sparsity in the hidden activations. In contrast, Hahn et al.[14] proposed that the effectiveness of Dropout is due to Dropout leading to the hidden activations entering the saturation regions of the nonlinear activations functions thus achieving better generalisation. They empirically confirmed that Dropout lead to an increase in activation variance. Nalisnick et al.[35] proposed that Dropout adds multiplicative noise which induces shrinkage priors on the network parameters. Gal and Ghahramani[13] hypothesised that Dropout leads to an approximation of the probabilistic deep Gaussian process. Baldi and Sadowski[4] argue that Dropout averages out the ensemble of possible subnetworks. Ahmad and Scheinkman[2] argued that Dropout induces sparse representations in the hidden layers which leads to a better generalisation. They showed that models trained with Dropout are less sensitive to additive input noise com-

pared to model trained without Dropout which the authors relate to difference in sparsity in the representation as models without Dropout learn dense representations.

The original paper[18] used a dropout rate of 50%, which was also used in prior work by the author[11] and was found to be sufficient for the purpose of avoiding the problem of overfitting. We therefore choose $p = 0.5$ for the neural networks to be trained.

Chapter 5

Flood Forecasting Methodology

The methodology developed in this report is intended to provide flood warnings for a single point (or at most a few points). Recall that a vast majority of floods in Vejle originate at Abelones Plads, along Mølleåen. This location has been equipped with a measurement station (ST.32.20) of which data has been available starting in December 2015. This data can be used to train a neural network to obtain long-term predictions which are related to flooding events. If the water level at Abelones Plads is predicted to exceed the bank level, a flooding of Abelones Plads is to be expected. The longer this continues and the higher the water level in excess of the bank, the larger the flood extent. If some information regarding flood extent for historic floods is available for periods in which water level data is also available, then some correlation can potentially be made.

In the following sections, we will discuss how to identify and classify events that exhibit dynamics of interest, we will discuss the set-up of the neural network model to evade the known problem of the network converging to the zero-order-forecast and we will introduce additional metrics for training and evaluation using the zero-order-forecast model as a benchmark model. Furthermore, we will discuss the training process with a special focus on preventing an undue prioritisation of a prediction by the network. The following sections about data generation will refer to Vejle specifically as they are based on data of Vejle; when applied to other catchments, the process works in principle, but the user may need to adjust the process based on the dynamics exhibited by that catchment.

5.1 Event Identification

The provided water level data starts in 2015 and the provided rainfall data starts in 2000. Most of it is irrelevant data as it reflects baseflow behaviour with little rainfall and a normalised water level of approximately 0.2 at Abelones Plads. We will therefore need to identify events of interest to be used as training and validation

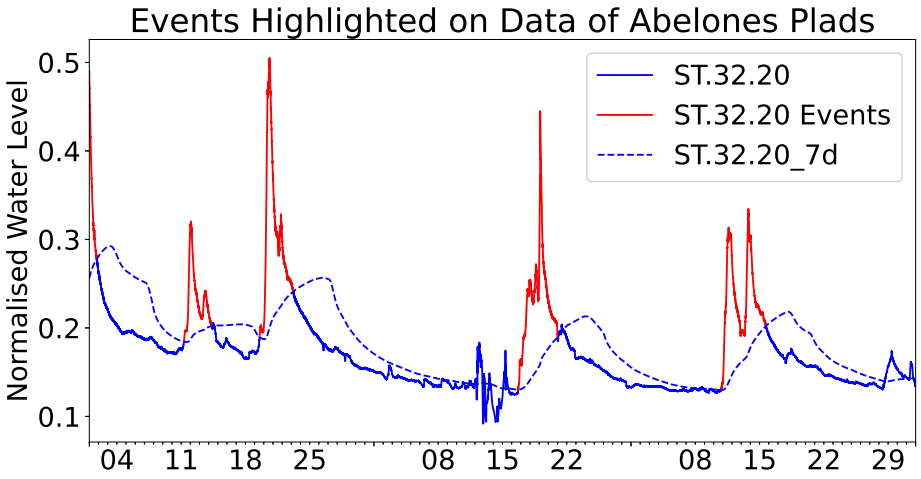


Figure 5.1: Sample Events in January-March 2021 highlighted on water level data from Abelones Plads

data. As we do not possess knowledge of the base level from which we deviate during events, we employ a moving average with a window of seven days to be used as an estimate. Let $x(t)$ be the water level and $\overline{x} = \mu(x(t); T)$ the moving average of the water level with a window length of T. In the discrete-time domain $\overline{x}$ is defined as given in 5.1, where T_s is the sampling time of the data.

$$\begin{aligned} \overline{x}[k] = \mu(x[k], N) &= \frac{1}{N} \sum_{l=k-N-1}^{k} x[l] \\ N &= \frac{T}{T_s} + 1 \end{aligned} \tag{5.1}$$

We then define events as continuous intervals of exceedance of the base line estimate. Let $\Delta x[k] = x[k] - \overline{x}[k]$. Then the set of start and end indices k_{start}, k_{end} are the values of k that satisfy 5.2.

$$\begin{aligned} k_{start} &: \Delta x[k_{start}] > 0 \text{ and } \Delta x[k_{start} - 1] \leq 0 \\ k_{end} &: \Delta x[k_{end}] > 0 \text{ and } \Delta x[k_{end} + 1] \leq 0 \end{aligned} \tag{5.2}$$

Event E_i then encompasses the samples $E_i : k^i_{start} \leq k \leq k^i_{end}$, including both the start and end indices. A visualisation of the events during the first three months of 2021 is given in 5.1 We then compute the maximum exceedance $\max(\Delta x)$ within the event E_i as given in 5.3.

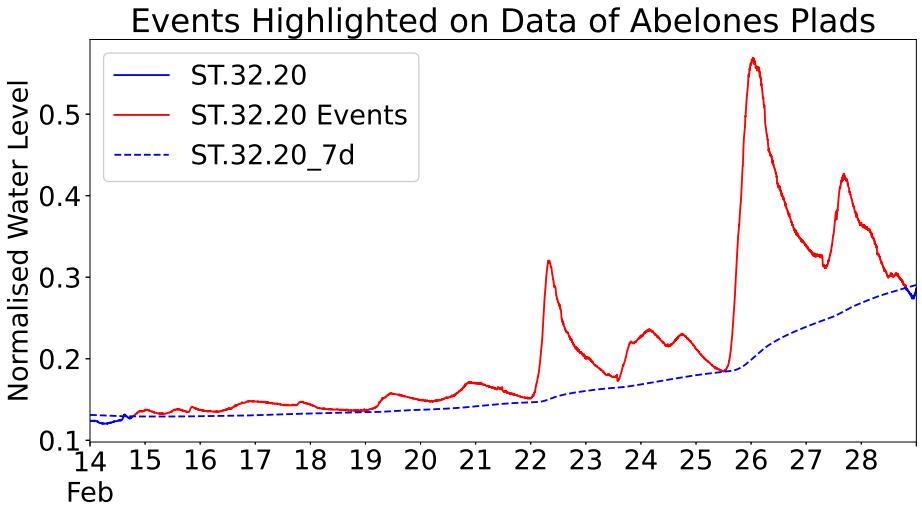


Figure 5.2: Longest Event in E

$$\max_{k \in E_i}(\Delta x[k]) = \max_{k^i_{start} \leq k \leq k^i_{end}}(\Delta x[k]) \tag{5.3}$$

Since we are comparing a signal against its own moving average, there will be many spurious events caused by some natural variation. For example the 2410 events of set E have a combined length of 697 days, but the 90% percentile of maximum exceedance is 0.015 and the 95% percentile is 0.06, indicating that most events do not exhibit a sufficiently significant response that can be used for training. In order to isolate events with dynamics of interest, we compute the maximum exceedance values over each event to eliminate spurious events. Suppose, we impose the criterion 5.4.

$$\max_{k \in E_i}(\Delta x[k]) \geq \Delta x_{min} \tag{5.4}$$

For example, imposing $\Delta x_{min} = 0.05$ will reduce E to 138 events of a combined length of 393 days which corresponds to 6% of events making up 56% of its length in time. Imposing $\Delta x_{min} = 0.1$, reduces E to 73 events of a combined length of 235 days and finally imposing $\Delta x_{min} = 0.2$, reduces E to 35 events worth 120 days.

As it turned out during preliminary testing, these events contained too much data, which lead to a sluggish training process. We therefore created a new set of modified events M. The longest event in E contains mostly spurious data before the actual event occurs. In order to cut off spurious tails as the one seen in 5.2, we enforce a minimum exceedance criterion ϵ on during the entirety of the events.

Suppose we generate the set of start and end indices as given in 5.5 which is based on the exceedance criterion $\Delta x[k] > \epsilon$.

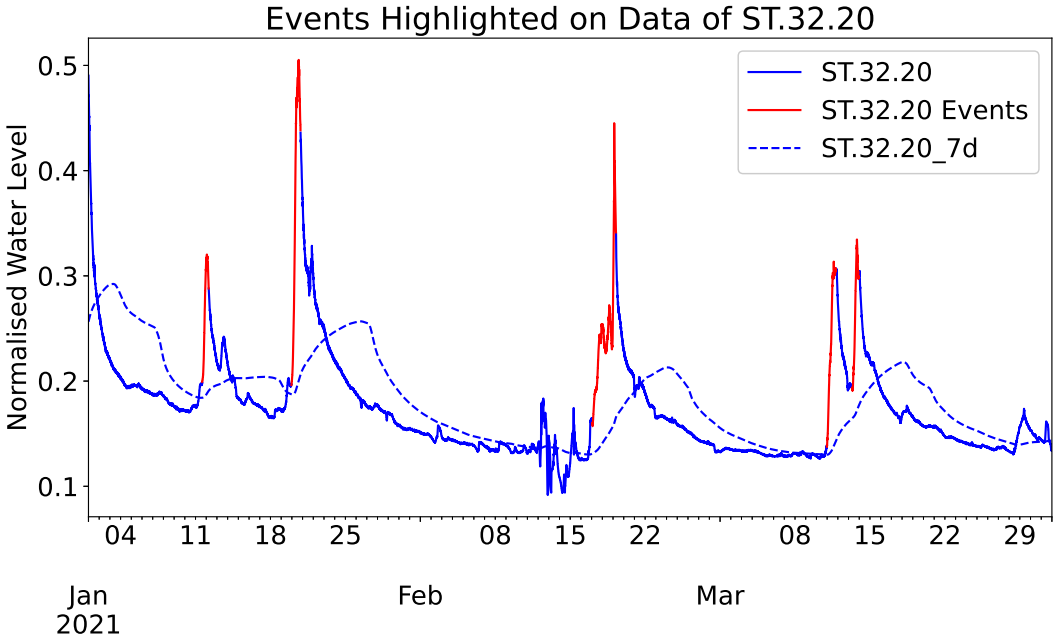


Figure 5.3: Modified Events Highlighted on Data of Abelones Plads from 2021, see figure 5.1 for a comparison

$$\begin{aligned} k_{start} &: \Delta x[k_{start}] > \epsilon \text{ and } \Delta x[k_{start} - 1] \leq \epsilon \\ k_{end} &: \Delta x[k_{end}] > \epsilon \text{ and } \Delta x[k_{end} + 1] \leq \epsilon \end{aligned} \tag{5.5}$$

Recall that event ends are identified by the indices at which the water level returns to meet the moving average. Generally, the water level behaviour is marked by spikes followed by first-order-like decay in the absence of rainfall where the catchment drains. From communication with the cooperative partner, it is known that it takes about five to seven days for the decay to be completed. This means that perhaps half of each event is simple decay. While we do need the trained network to understand when the decay will occur, it does not need to estimate the decay precisely, since, if the water level is going down, no flood warning will be necessary. We therefore opt to cut the events after the peak within the event. Let k_{peak} be the index of the highest peaks within each event. We then cut off the events shortly after the peak by setting $k_{end} = k_{peak} + k_{afterpeak}$, where $k_{afterpeak}$ corresponds to four hours. Since the starting indices do not reflect the true beginning of the deviation, all starting indices are left-shifted by four hours.

Figure 5.3 visualises the modified events using $\Delta x_{min} = 0.1$ over the first three months of 2021. The set M now contains 82 events with a combined length of 63 days. Table 5.1 contains the hyper-parameters of the event identification process.

Minimum normalised exceedance	0.05
Left-shift start indices	4h
Right-shift end indices	4h
Truncate after peak	Yes
Minimum time between events	0
Minimum normalised peak exceedance	0.1

Table 5.1: Specification of hyper-parameters of event identification process

Class	Training	Validation	All	Definition
4	19	10	29	Long, High-Intensity
3	8	4	12	Long, Low-Intensity
2	19	10	29	Short, High-Intensity
1	8	4	12	Short, Low-Intensity

Table 5.2: Class definitions and number of events by class and set. Medians are given as $L = 13.5\,\mathrm{h}$ and $\sum r = 695.6874\,\mathrm{mm}$. The ratio of events used for training is 0.66.

5.2 Event Classification

In order to ensure that our network is trained on a variety of different types of events and generalises to different types of events, it is necessary to split the set of events into different classes and then draw events from those classes to generate the training and validation sets at random such that each class is represented in the training and validation sets at the same ratio as they are represented in the entire set. This supports our ability to claim that the model generalises to unseen data based on the model performance on the validation set because the validation set is representative of the entire set of events.

Consider that increases in the water level at Abelones Plads are driven by rainfall. It is reasonable to split events by rainfall intensity (the total rainfall measured during the event). The evolution of the water level also depends on the length of the event; the longer the event, the more time for the catchment to drain which leads to smaller increases in water level compared to events with short, high-intensity rainfall.

We split each axis (length and total rainfall) by their median values. This leads to similar class sizes for the two classes along the regression line (short, low-intensity and long, high-intensity) and similar class sizes for the two off-line classes (long, low-intensity and short, high-intensity). This discrepancy is due to the correlation between the two axes; the longer the event, the more rainfall we tend to accumulate.

5.3 Delay Phenomenon

This section serves to discuss an important phenomenon that was already observed in prior work[11] which greatly affects the considerations we made in regards to what inputs to provide to the neural network. For lack of a better term, we call this the *delay phenomenon*. Recall that the water level at Abelones Plads is a the result of a relatively slow-moving dynamical system (relative to our sampling time). Therefore, the delay (or zero-order-forecast) model $\widehat{y}[k+T] = y[k]$ generates an accurate prediction in purely numerical terms. For example, on the set of events, the Mean Absolute Error (MAE) of the zero-order-forecast model for a 1h-ahead prediction is generally in the range of 2 to 3 centimetres. Compared to the ground-to-bank distance of 133 centimetres of Mølleåen at Abelones Plads, this seems to imply a 2% relative error.

However, the zero-order-forecast model does not capture any information about the change of the system, i.e. it makes no statement whether the water level will go up or down, which is precisely the information relevant to us. For that reason, the zero-order-forecast model is useful as a benchmark model because it is the non-parametric model with the least loss (Refsgaard et al. also used it as a benchmark model[39]). If any neural network model is observed to have a smaller loss, then it must necessarily have captured some information about the dynamics. Conversely, a model with a worse loss likely captures erroneous information irrelevant to the evolution of the system.

Since the zero-order-forecast model has a high degree of numerical accuracy, a Neural Network model that operates on a set of inputs which includes the state to be predicted, then that model tends to converge to that model. This delay phenomenon was observed in prior work[11], where it was observed that including the state as input leads to the model failing to extract the underlying dynamics.

In the prior work, the problem posed by the delay phenomenon was handled by omitting the state from the inputs and instead using only water level measurements upstream and downstream of the point of prediction. It was noted that this leads to a model that behaves in parallel with the actual water level. Since the models trained in that work did not interact with the state measurements at all, the prediction exhibited an approximately constant error. As the purpose of the prior work was the design of a control algorithm, this behaviour was deemed acceptable.

However, for the purpose of flood forecasting, we are interested in the absolute values of the water level at Abelones Plads some time into the future, and therefore such a constant error is deemed unacceptable. We therefore employ an architecture that has the benefit of using the state as an input (no offset) and the benefit of excluding the state (avoiding the delay phenomenon). Consider that the largest portion of the water level prediction $\widehat{y}_{k+T}$ is provided by the current water level y_k.

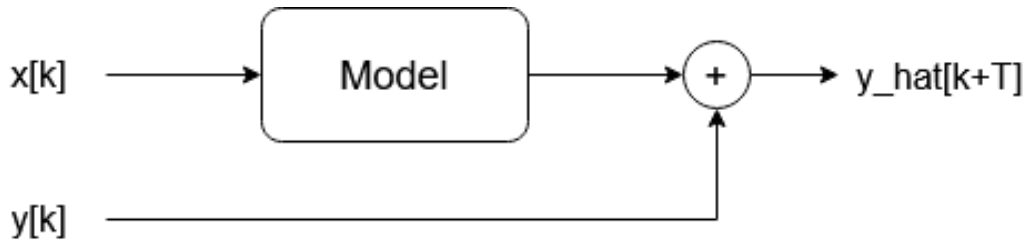


Figure 5.4: Block diagram of model trained to predict change in water level, $x[k]$ denotes the input to the neural network, $y[k]$ the current water level and $yhat[k+T]$ the prediction of the network.

Therefore, it makes sense to train the neural network to predict the change in value Δy_k as given in 5.6. We hypothesise that it is easier for the neural network to learn changes of a variable over time than to learn to predict the value of that variable. A similar logic has been applied to explain why residual networks outperform classic deep networks[16].

$$\widehat{y}_{k+T} = y_k + \Delta\widehat{y}_k \tag{5.6}$$

A block diagram of this model architecture is given in 5.4, where the block denoted as model can be any neural network model. Note, that this architecture can be understood as a sort of residual network as the current water level measurement bypasses the model and thus generally avoids the delay phenomenon.

5.4 Zero-Order-Forecast Benchmark Model

In order to evaluate the forecast accuracy of a neural network model, we compare the Mean Square Error (MSE) of the NN with the MSE of the zero-order-forecast model to compute the *Skill Score vs Delay Model* (SSD), as given in 5.7. An SSD of 1 indicates a perfect model, an SSD of 0 indicates a model with performance equal to the zero-order-forecast model.

$$SSD = 1 - \frac{MSE_{NN}}{MSE_{delay}} \tag{5.7}$$

We primarily use the SSD for a qualitative assessment; i.e. if a model shows a strictly positive median SSD for the validation set, then we say the model provides an acceptable forecast for the water level at Abelones Plads. It should be noted that any single-prediction model that minimises the MSE also maximises the SSD. However, the SSD faces the same issue as the MSE regarding the qualitative interpretation, since both have the dimension of water level squared.

For that reason, we introduce two related comparative metrics, the *Root Skill Score vs Delay* (RSSD) and the *Absolute Skill Score vs Delay* (ASS), which are based on the ratio of RMSE and MAE, respectively, as given in 5.8.

$$
\begin{aligned}
RSSD &= 1 - \frac{RMSE_{NN}}{RMSE_{delay}} = 1 - \sqrt{\frac{MSE_{NN}}{MSE_{delay}}} = 1 - \sqrt{1 - SSD} \\
ASSD &= 1 - \frac{MAE_{NN}}{MAE_{delay}}
\end{aligned}
\tag{5.8}
$$

The two new metrics can be easily interpreted; an ASSD or RSSD of x indicates that the model reduces the associated metric by $100x\%$ compared to the benchmark delay model. We point out that the SSD is computed similar to the Nash-Sutcliffe Efficiency (NSE) for the water level change assuming a mean water level change of zero (which is generally given considering the water level is bounded). This is importantly not the case for our event data as the events are truncated after the peaks and therefore tend to end at a higher level than the starting level. Simply computing the R-squared or NSE would then be identical to computing the SSD with $\Delta\widehat{x} = \mu_x$ as benchmark model which (considering our event data yields $\mu_x > 0$) is not meaningful. Instead, the SSD is computed using the zero-order-forecast benchmark model $\Delta\widehat{x} = 0$ and then interpreted in the same manner as the R-squared. The fraction of explained variance is equal to the SSD and the fraction of unexplained variance is equal to the complement of the SSD (we later denoted this quantity as the NMSE, see 5.9).

5.5 Network Training

In this section, we will briefly explain the training process that we use to obtain neural network models.

The training process is an interative procedure. Each step, we execute a hotstart using a 4-hour window preceding each event and extract the final hidden and cell states. We then compute the predictions for the training events using the extracted states with Dropout switched on. We then switch the Dropout off and compute the predictions for the validation events using the states extracted from the hotstart. We note that the only data kept from the hotstart are the internal states; we do not require the prediction made over the hotstart windows.

We note an important fact here: if one wishes to make a single prediction, the Mean Square Error (MSE) is sufficient for that purpose. However, for a network making multiple predictions, this will prioritise the longer prediction times as it contributes more to the overall error. In general, network outputs must be normalised. We have done so with the water level by scaling it such that the ground and bank level map to 0 and 1. We note that the network output is the change in water level. The more time passes, the larger the expected change will be. Therefore, the longer prediction times will provide a much larger part of the overall MSE and thus provide the most significant opportunity to decrease the overall MSE. The result is that the shorter prediction times may potentially be neglected by the model. However, we expect this issue to be somewhat remedied by the fact that the different water level changes to be predicted are correlated, therefore temporal features extracted by the LSTM that are helpful for one prediction are likely helpful to the other predictions as well.

Ordinarily, we would handle this by scaling the changes to have, for example, unit variance. However, the change itself is not computed in the data; instead we simply add the water levels at the time at which we make the prediction to the network output. The water level changes are not explicitly computed. This is primarily due to the fact that scaling all these variables independently and then scaling them back to the original scale adds obscurity which increases the risk of the network generating a wrong prediction through user error. Recall that we scaled the water levels according to bank and ground levels as this scale is easily understandable to the user. Therefore, instead of scaling the outputs, we instead opted to scale the errors.

Recall that the magnitude of the errors increases with time. We therefore introduced the SSD as a relative metric that compares the model prediction with a zero-order forecast benchmark model, so that the errors are equally scaled. Let NMSE denote the *normalised MSE* as given in 5.9. We point out that the NMSE is simply the complement of the SSD. This is beneficial as a model that minimises the NMSE for several predictions, also maximises the SSD. For a model trained

using the MSE, this only holds if the model makes a single prediction, due to the difference in scales, as the MSE is the average of the prediction MSEs, whereas the SSD is a weighted average of the prediction MSEs.

$$NMSE = \frac{MSE(NN)}{MSE(ZOF)} = 1 - SSD \tag{5.9}$$

Using the water level measurements at the prediction times, the water levels to be predicted and our predictions, we compute the NMSE for the training and validation events. The former is used for parameter updates, while the latter is done without maintaining gradients to withhold the validation events from the network. The parameter updates are generated using the Adam optimiser with the Amsgrad implementation[25, 38].

We begin the training procedure with an initial learning rate of 0.01. While the network is being trained, we monitor the validation NMSE. The learning rate is modified by a factor of 0.5 if the validation NMSE stops improving using a 'ReduceLROnPlateau' learning rate scheduler (see [7]). We terminate the training process when the learning rate decays below $1e-7$ as we consider a learning rate of that magnitude to generate insignificant parameters updates. During the procedure, we maintain the minimum validation NMSE and the associated parameter set. Once training concludes, we load the best performing parameter set into the model.

5.6 Sign Conformance Metric

We have explained the SSD and the complementary NMSE metrics before, as they are the most relevant metrics for both model evaluation (SSD) and training (NMSE). In addition, we introduce another evaluation metric that helps us quantify the model performance, which we call the *Sign Conformance*. We point out that this metric should only be used to complement the quantitative performance analysis made using the SSD scores.

The Sign Conformance SC is a categorical evaluation; given a change in water level to be predicted, the SC specifies the likelihood that the sign of the model prediction conforms with the sign of the change as given in 5.10 (i.e, the likelihood of the model correctly predicting the direction of the change in water level). This metric is perhaps easier to understand than the SSD due to its categorical nature. Recall that the zero-order-forecast is a numerically good prediction but captures no information about the expected evolution of the system. Since it always predicts zero change, it has a Sign Conformance of 0%. A neural network that incurs some prediction error but has a sign conformance of 100% is a very good predictor since it outperforms the ZOF benchmark model in a categorical sense. Thus, the SC can also be understood as a categorical relative metric. We note that some care has to

be taken here; a hypothetical model whose prediction is a small positive constant will have an SC of approximately 50%. Therefore, it is desirable to split up the SC into two partial metrics for the two cases of $\text{sgn}(\Delta w)$; note that we omit the case of $\text{sgn}(\Delta w) = 0$.

$$SC = p(\Delta w > 0 \wedge \Delta \widehat{w} > 0) + p(\Delta w < 0 \wedge \Delta \widehat{w} < 0) \tag{5.10}$$

The SC can be computed as follows. Let $\epsilon(x)$ be zero close to $x = 0$ and 1 everywhere else as given in 5.11. The epsilon function can be understood as a relevance function with the criterion ε which we use to drop out irrelevant values.

$$\epsilon(x) = \begin{cases} 0 \text{ if } & |x| \leq \varepsilon \\ 1 \text{ if } & |x| > \varepsilon \end{cases} \tag{5.11}$$

We primarily care about getting the sign correct for changes of high magnitude. For changes of very low magnitude, whether the sign conforms is irrelevant. We compute the pairwise Sign Conformance sc_k in 5.12 using ε as the relevance criterion. Let $y_k = \Delta w_k$ be the exact change and $\widehat{y}_k = \Delta \widehat{w}_k$ be the predicted change.

$$sc_k = \epsilon(y_k)\text{sgn}(y_k)\text{sgn}(\widehat{y}_k) \tag{5.12}$$

The pairwise sign conformance is 1 if y is relevant and the signs of $y, \widehat{y}$ conform, it is zero if y is irrelevant, and -1 is y is relevant but the signs of $y, \widehat{y}$ do not conform. The number of relevant samples in sc is given by $\sum_{k=1}^{N} |sc_k|$.

The Sign Conformance SC is then given in 5.13, where $u(x)$ denotes the unit step function, which replaces the -1 entries with 0 in the numerator to enforce a lower bound of 0. We note that $u(0) = 0$.

$$SC = \frac{\sum_{k=1}^{N} u(sc[k])}{\sum_{k=1}^{N} |sc[k]|} \in [0, 1] \tag{5.13}$$

Consider that $\epsilon(x)u(x)$ is the step function with a dead-zone around the origin. We can use this function to determine which samples have a relevant positive change y and which have a relevant negative change y (denoted by y^+, y^-, respectively, in 5.14).

$$\begin{aligned} y_k^+ &= \epsilon(y_k)u(y_k) = \begin{cases} 1 \text{ if } & y_k > \varepsilon \\ 0 \text{ if } & y_k \leq \varepsilon \end{cases} \\ y_k^- &= \epsilon(y_k)u(-y_k) = \begin{cases} 1 \text{ if } & y_k < -\varepsilon \\ 0 \text{ if } & y_k \geq -\varepsilon \end{cases} \end{aligned} \tag{5.14}$$

We can now compute the partial Sign Conformances $SC(y^+)$ and $SC(y^-)$ as given in 5.15.

$$\begin{aligned} SC(y^+) &= \frac{\sum_{k=1}^{N} y_k^+ u(sc_k)}{\sum_{k=1}^{N} y_k^+} \\ SC(y^-) &= \frac{\sum_{k=1}^{N} y_k^- u(sc_k)}{\sum_{k=1}^{N} y_k^-} \end{aligned} \tag{5.15}$$

The two partial Sign Conformances are related to the overall Sign Conformance depending on the occurrence of relevant positive and negative changes. Note that the set of relevant positive changes and the set of negative changes are mutually exclusive and exhaustive with respect to the set of relevant changes as given in 5.16. If a model has both partial Sign Conformances above 50%, then it is a better predictor than the zero-order-forecast benchmark model in a categorical sense.

$$SC = \frac{\sum_{k=1}^{N} y_k^+}{\sum_{k=1}^{N} \left(y_k^+ + y_k^-\right)} SC(y^+) + \frac{\sum_{k=1}^{N} y_k^-}{\sum_{k=1}^{N} \left(y_k^+ + y_k^-\right)} SC(y^-) \tag{5.16}$$

The Sign Conformance metrics will be used to evaluate the categorical performance of the neural network model. For the purposed of flood forecasting, we are primarily concerned with detecting an incoming rise in water levels which warrants action to be taken. While a quantitatively correct prediction allows that (this can be measured using the SSD), a network that does not predict the exact extent of the rise in water level but does correctly predict that the water level will rise is valuable. The Sign Conformance may thus be understood as a relaxed performance criterion compared to the SSD.

Chapter 6

Testing

In this chapter, we will train neural networks for different hyper-parameter choices and evaluate the effect onto the model performance. We will start by comparing a model trained with the rainfall moving average blocks obtained in 3.4 with a model trained with 30 minute blocks covering the same time interval. This will allow us to evaluate whether the feature selection algorithm managed to extract the relevant temporal features. Then, we will compare a model trained with rainfall inputs as well as water level inputs with a model trained with only rainfall inputs. This will allows us to evaluate how well the rainfall input can compensate when the water level inputs are dropped, i.e. how much information carried by the water level input is redundant. Since the water level inputs themselves are functions of the rainfall, ideally, no decrease in model performance should be observed. However, since our rainfall input does not fully capture the spatial variability of the rainfall, we expect to see a decrease in performance. We will then compare models with different numbers of layers to obtain the best performing model. We will also evaluate models with and without Dropout to evaluate the effect of Dropout with regards to overfitting.

Finally, we will evaluate the performance of the obtained model on the validation events and attempt to explain under what circumstances the model performs well and when it performs poorly. We hypothesise that the insufficient coverage of the rainfall is responsible for a significant part of the failure.

The general structure of the models (unless explicitly stated otherwise) is a ResLSTM with 20 neurons in each hidden layer, a Dropout layer (with $p = 0.5$) located after each LSTM layer. There is no Dropout along the residual connections and no Dropout along the recursive connections within the LSTM layers (see figure 4.1). This structure (20 neurons, 50% Dropout) was found to be effective in prior work by the author[11]. We train all models to predict the change in the water level at Abelones Plads with prediction times of 30 minutes, 1 hour, 2 hours, 3 hours and 4 hours. Table 6.1 provides a summary of hyper-parameters shared by all models

Hyper-parameters	Values
Width	20
Output	ΔST.32.20
Prediction Times	[30 min, 1 h, 2 h, 3 h, 4 h]

Table 6.1: Hyper-parameters shared by all models

Loss	Name	Unit
NMSE	Normalised Mean Square Error	-
MAE	Mean Absolute Error	[m]
RMSE	Root Mean Square Error	[m]
SSD	Skill Score vs Delay Benchmark Model	-
SC	Sign Conformance	[%]
SC^+	Sign Conformance for $\Delta w > 0$	[%]
SC^-	Sign Conformance for $\Delta w < 0$	[%]

Table 6.2: Loss functions used for model performance evaluation

trained in this chapter.

The purpose of this section is to be able to give a recommendation on how to obtain the best performing model. While there are numerous possible combinations of hyper-parameters to test, each combination requires a model to be trained. We therefore opt to test iteratively, i.e. we determine optimal choice of hyper-parameters sequentially and not in parallel. All evaluations are done over the validation events and do not involve the training events. This will allow us to evaluate the expected model performance in a real-life implementation during an event. Table 6.2 provides a list of loss functions we will evaluate.

6.1 Evaluation of the Feature Selection Algorithm

In this section, we will evaluate the model performance using the moving average blocks generated by the feature selection algorithm which is compared against the model performance of a model using 30-minute blocks covering approximately the same interval. In section 3.4, we only compared the two sets of inputs via the Spearman Multiple Rho-squared which only covers monotonous dependency. There might in fact be some relevant non-monotonous dependency which the feature selection algorithm cannot account for.

We train two ResLSTM models. The first is given the input set 3.1 and the other is given the same yearly variation and water level inputs along with 21 moving average blocks of length 30 minutes. Both models are trained as described in 5.5 and then compared via their median SSD scores over the validation events. We

MA type	30 min	1 h	2 h	3 h	4 h	Mean	Number of Parameters
custom	0.325	0.488	0.633	0.632	0.608	0.537	12865
fixed	0.293	0.408	0.639	0.629	0.636	0.521	14365

Table 6.3: Median SSD scores for 4-layer ResLSTM models using custom or fixed rainfall inputs for different prediction times.

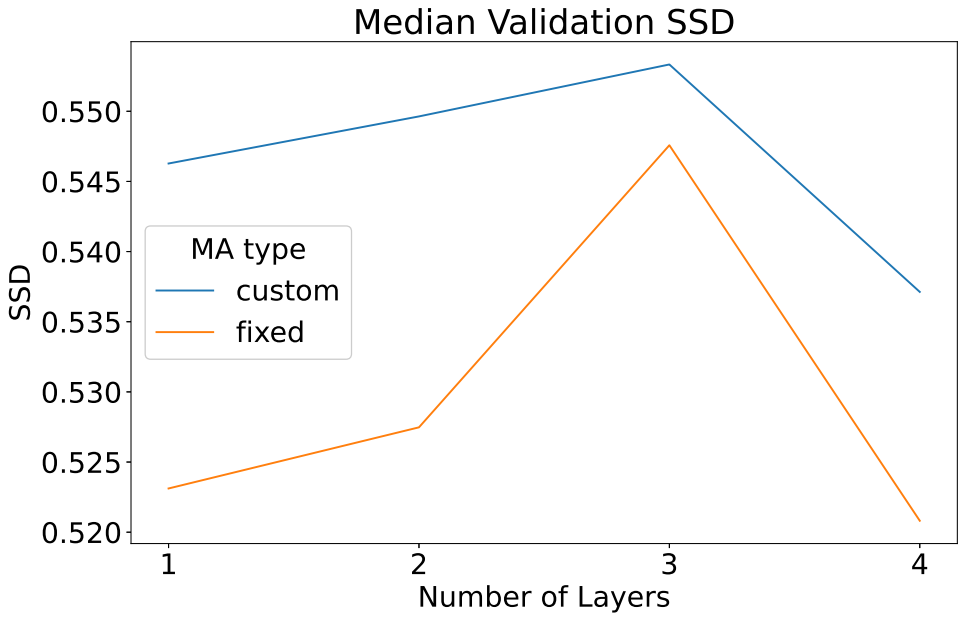


Figure 6.1: Comparison of 2 ResLSTM models using the custom and fixed moving average sets for the rainfall input for 1 to 4 layers. The custom set is generated by the heuristic feature selection algorithm.

note here that neural networks are initialised with random parameters and there is no convenient way to account for the effect.

Table 6.3 provides the median SSD scores for the two models for the different prediction times along with the overall mean. We also trained the two ResLSTM models with depths ranging from 1 to 4 layers. Figure 6.1 visualises the median SSD of the two ResLSTM models with one to four layers (the individual median SSD scores from the different predictions were average out to generate the figure).

Table 6.3 shows that the 4-layer ResLSTM model with the rainfall features generated by the algorithm introduced in 3.4 yields an increase of 0.016 in SSD (this is equivalent of explaining an additional 1.6% of the variance of the water level change) while using 1500 parameters less compared to the model using a set of 30-minute blocks. Since the models were initialised with random parameters, a hypothesis test would be appropriate to determine if the difference is indeed sig-

nificant; however doing so would require numerous trained models which is not feasible. We therefore conclude that the difference may not be significant. This means the feature selection algorithm accomplished the same model performance using less model parameters and less inputs.

6.2 Evaluation of Models using only Rainfall Inputs

In the previous section, we showed that the model using rainfall features generated by the feature selection algorithm yields similar or perhaps better median SSD scores over the validation events while also reducing the dimension of the rainfall input. We will therefore proceed with using features generated by the algorithm.

In this section, we will compare two 1-layer ResLSTM models, the first using inputs given in table 3.1, the second using the yearly variation t_A along with rainfall features generated by the feature selection algorithm given t_A as auxiliary input (i.e. the second model does not use any water level information). This test is meant to investigate the effect of water level information onto the model performance for the different prediction times. We will also train a ResLSTM model using only t_A and the water level inputs from table 3.1. We hypothesise that water level information is primarily useful for shorter prediction times (30 minutes and 1 hours) but not useful for longer prediction times (4 hours). We know from prior work[11] that the water level can be predicted a very short time into the future using only water level information from two points up- and downstream of the prediction point.

Table 6.4 provides the median of the Validation SSD scores with respect to prediction time and input set. Figure 6.2 visualises the data. It can be seen that up until a prediction time of two hours, the model using rainfall and water level information has similar performance as the model using only water level information. The latter's performance quickly degrades for prediction times in excess of two hours. In contrast to that, the performance of the model using only rainfall information increases as the prediction time increases and surpasses the performance of the model using only water level information for a prediction time of four hours. This underscores the relevance of using both water level and rainfall information to obtain best possible performance (we note that adequate rainfall inputs may make water level inputs redundant). The test was repeated using 4-layer ResLSTM models and yielded similar results.

6.3 Evaluation of Dropout

In this section, we will train two 1-layer ResLSTM models using the inputs given in table 3.1, the first with Dropout of 50% after each LSTM Layer (see figure 4.1), the second without Dropout. On average the model with Dropout achieved an im-

Inputs	30 min	1 h	2 h	3 h	4 h
Rainfall	0.150	0.244	0.368	0.447	0.576
Rainfall+Water	0.345	0.515	0.645	0.630	0.606
Water	0.351	0.502	0.637	0.511	0.366

Table 6.4: Median validation SSD of 1-layer ResLSTM models trained using three different input sets. The rainfall sets were both generated by the feature selection algorithm using the yearly variation, and the yearly variation and the water levels as auxiliary inputs.

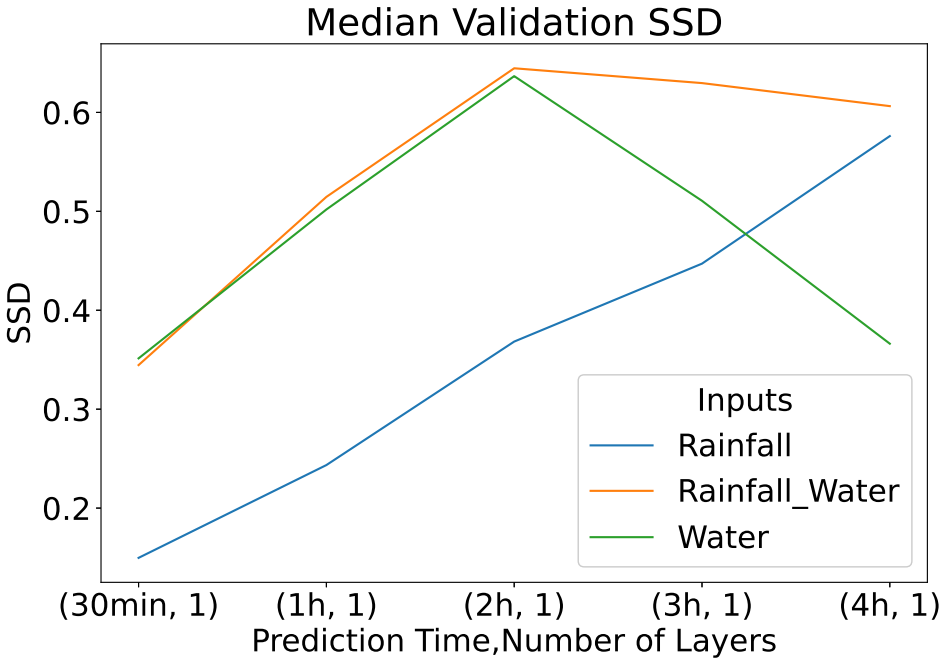


Figure 6.2: Comparison of median validation SSD as a function of prediction time for three models using different input sets.

Dropout	30 min	1 h	2 h	3 h	4 h
0%	0.301	0.458	0.597	0.571	0.563
50%	0.345	0.536	0.652	0.685	0.591

Table 6.5: Comparison of 1-layer ResLSTM models with Dropout of 0% and Dropout of 50%

Model	30 min	1 h	2 h	3 h	4 h	Mean
Multi Prediction	0.321	0.501	0.661	0.659	0.607	0.550
Single Prediction	0.346	0.458	0.639	0.602	0.550	0.519
Difference	-0.025	0.043	0.022	0.058	0.059	0.031

Table 6.6: Comparison of median validation SSD scores for multi-prediction model and ensemble of single-prediction models.

provement of 0.06 in median validation SSD when compared to the model without Dropout, i.e. the model with Dropout explains an additional 6% of the variance in the water level change which we attempt to predict.

The model without Dropout terminated after 791 iterations of training, whereas the model with Dropout terminated after 2215 iterations. This implies that the learning rate scheduler together with the termination criterion imposed onto the learning rate terminated the training of the first model after the validation loss stopped improving (i.e. the model failed to generalise). Without this safeguard in place, the model without Dropout would have likely exhibited a significantly worse validation performance. We conclude that a Dropout of 50% is effective at inducing a higher generalisability of the model and preventing overfitting of the model.

6.4 Evaluation of Single vs Multiple Prediction

In this section, we will briefly analyse whether the training process places an undue priority upon an individual prediction time at the cost of neglecting the other prediction times in the process by training one model with multiple prediction times and then training an ensemble of models for each prediction time. All models have the same hyper-parameters other than the prediction time(s). Since we train using the NMSE as loss function, we hypothesise that the single prediction models will not perform significantly better than the multiple prediction model.

Table 6.6 shows the obtained median validation SSD scores. The multiple-prediction model outperforms the ensemble of single-prediction models for all prediction times except 30 minutes. The average difference is 0.031, which means the multiple-prediction model can explain an average of 3.1% of the variance of the water level changes at Abelones Plads that the ensemble of single-prediction models cannot explain. We hypothesise that training with multiple objectives has a

Layers	1	2	3	4	5	6	7	8
Median SSD	0.535	0.538	0.501	0.549	0.545	0.548	0.547	0.559
Parameters	2785	6145	9505	12865	16225	19585	22945	26305

Table 6.7: Comparison of median validation SSD scores and number of trainable parameters for ResLSTM models with 1 to 8 layers.

Hyper-parameters	Values
Input	see table 3.1
Output	ΔST.32.20
Prediction Times	[30 min, 1 h, 2 h, 3 h, 4 h]
Dropout	50%
Width	20
Layers	8
Parameters	26305

Table 6.8: Optimal hyperparameter values with respect to the median validation SSD.

regularising effect onto the model which allows better generalisation. We assume that effect is the reason why the ensemble of five models with 2701 parameters each does not outperform the multi-prediction model with 2785 parameters.

6.5 Evaluation of Network Depth

In this section, we will train ResLSTM models with the inputs given in table 3.1 of depths ranging from 1 to 8 layers to evaluate the effect of the network depth onto the model performance. It can be seen in table 6.7 that the 8-layer model yields the best performance seen so far with a average median validation SSD of 0.559.

6.6 Summary of Hyperparameter Evaluation

In this section, we will summarise the results from the evaluated hyperparameter options to settle on a set of hyperparameters. Table 6.8 provides the optimal values obtained for the hyperparameters we tested for. The model that performs best on the validation set in terms of the highest median SSD averaged out over the 5 different prediction times is the 8-layer ResLSTM model using rainfall and water level information and Dropout of 50%. We retain the 8-layer model from the previous section for further analysis which we will refer to as ResLSTM-8.

Table 6.8 specifies the hyperparameters used to obtain the ResLSTM-8 model in a previous test (see 6.5). We will evaluate the model performance over the set of validation events.

Prediction Time	30min	1h	2h	3h	4h
MAE [cm]	0.84	1.60	2.83	3.53	4.48
RMSE [cm]	1.24	2.07	3.30	4.47	5.54
SC [%]	85.55	88.76	89.56	90.64	88.03
SC^{-} [%]	84.34	90.44	90.85	84.94	83.10
SC^{+} [%]	90.45	92.26	92.56	94.94	91.83
NMSE	0.615	0.449	0.333	0.330	0.398
SSD	0.385	0.551	0.667	0.670	0.602
ASSD	0.321	0.368	0.338	0.441	0.409
RSSD	0.216	0.330	0.423	0.426	0.369

Table 6.9: Obtained values for different loss functions and prediction times of the ResLSTM-8 model applied on validation events.

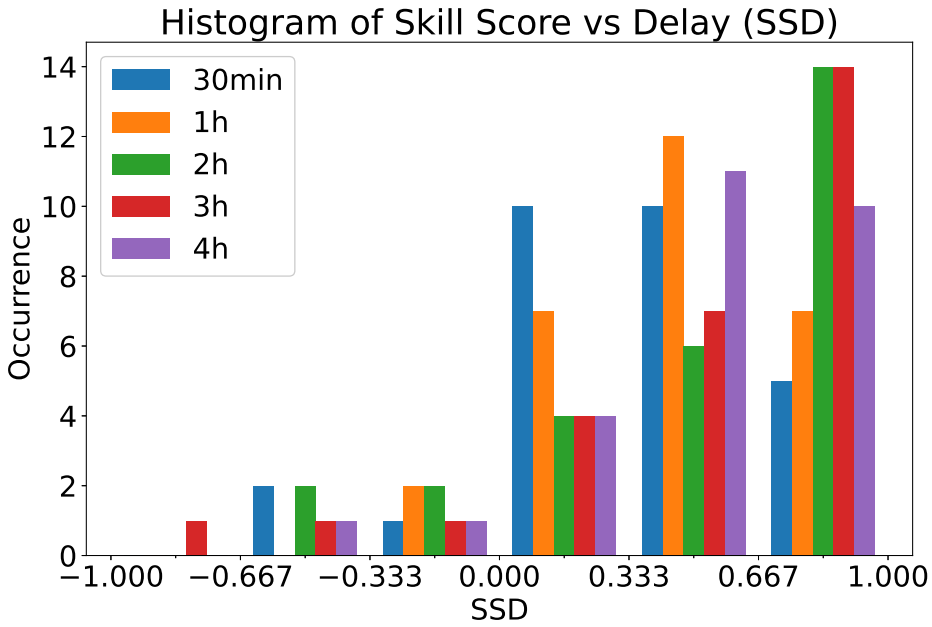


Figure 6.3: Histogram of event-specific SSD scores computed over the set of validation events (histogram cuts off one event with SSD=-1.59 for a prediction time of 4h).

Prediction Times	30min	1h	2h	3h	4h	Total
SSD < 0	3	2	4	3	3	15
SSD > 0	25	26	24	25	25	125

Table 6.10: Occurrence of positive and negative SSD scores computed over the validation events. In 15 out of 140 predictions, the SSD is negative which indicates a model performance worse than the zero-order forecast benchmark model.

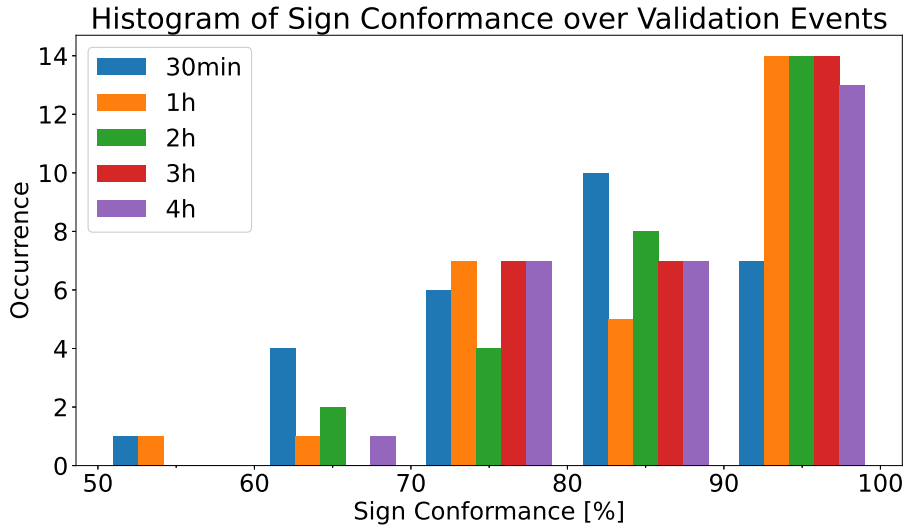


Figure 6.4: Histogram of the event-specific Sign Conformance values computed over the validation events.

Figure 6.3 provides a histogram of the validation SSDs which exhibit a significant skew behaviour. From the figure we see that the bin with the highest occurrence for prediction times of 2 and 3 hours is [0.667, 1]. Table 6.9 provides the median of the event losses for different loss functions and prediction times. The best individual median validation SSD is 0.67 for a prediction time of 3 hours, which indicates that in a real-life event, there is a 50% chance that the event SSD for that prediction time is above 0.67. The distribution of loss function values over the events is skewed significantly by a few events characterised by a poor model performance. Table 6.10 provides the occurrence of positive and negative validation SSD. The 15 cases belong to only 5 events, thus the per-event chance is 83% that the ResLSTM-8 model will outperform the zero-order benchmark model on all prediction times in a real-life implementation. On a per-event and per-prediction basis, the chance is 90%.

6.7 Event-Based Evaluation of Best-Performing Model

In this section, we will analyse the ResLSTM-8 model that was found to have the best performance of all trained models during testing (the model hyperparameters were given in table 6.8). We apply the model onto the validation events and evaluate the event-specific performance.

A preliminary investigation showed that the 5 events with partially or completely negative SSD scores, are all events with a below-median peak water level deviation measured at Abelones Plads, yet (with one exception) report above median total rainfall over the duration of the event. Figure 6.5 provides a scatter plot of the validation events with respect to total accumulated rainfall and peak water level deviation with the events colour-coded by SSD. Recall that the rain gauge used in this work is not located in the Grejs Å catchment but rather to the south of Abelones Plads. All 5 events in which the ResLSTM-8 model fails to make an acceptable prediction (i.e. SSD$\geq$ 0), are events where we measure significant rainfall, yet a less-than-expected increase in water level. We hypothesise based on this knowledge that in these 5 events, the rainfall in the catchment was less than the rainfall over the gauge. The ResLSTM-8 model is unaware of the spatial non-uniformity and attempts to extrapolate the rainfall in the catchment from the measured rainfall and in the process, it over-predicts the rainfall. One of the events is depicted in 6.6. This behaviour is exhibited by four of the 5 events where at least one prediction reports a negative SSD score. The remaining event (also the one with the least total rainfall) does not exhibit any significant over-prediction, but only the 30-minute-ahead prediction reports a negative SSD which is due to some oscillatory behaviour in the prediction. Based on this, we conclude that the inadequacy of the rainfall information used by the ResLSTM-8 model causes at least 4 of the 5 failures reported among the 28 validation events and 14 of the 15 failures reported among the 140 pairs of validation events and prediction times.

From conversations with our cooperative partner, we know that the rainfall distribution is dependent on the season. It is therefore valuable to analyse the network performance on validation events based on when the event occurs in the year. For simplicity, we will refer to the months of October to March as winter months and to the months of April to September as summer months. Table 6.11 provides the median validation SSD over the entire set of validation events and over the set of validation events in the winter and summer months. Figure 6.7 provides the normalised total and peak rainfall for the 12 months over a period of 5 years. It can be seen that while the total rainfall is roughly constant, the peak rainfall exhibits clear seasonal behaviour. This indicates that rainfall in the winter months tends to be more constant over time, whereas rainfall in the summer months is characterised by short, heavy bursts. This change in temporal distribution of the rainfall has a significant effect. For example, the 10-minute rainfall block for the

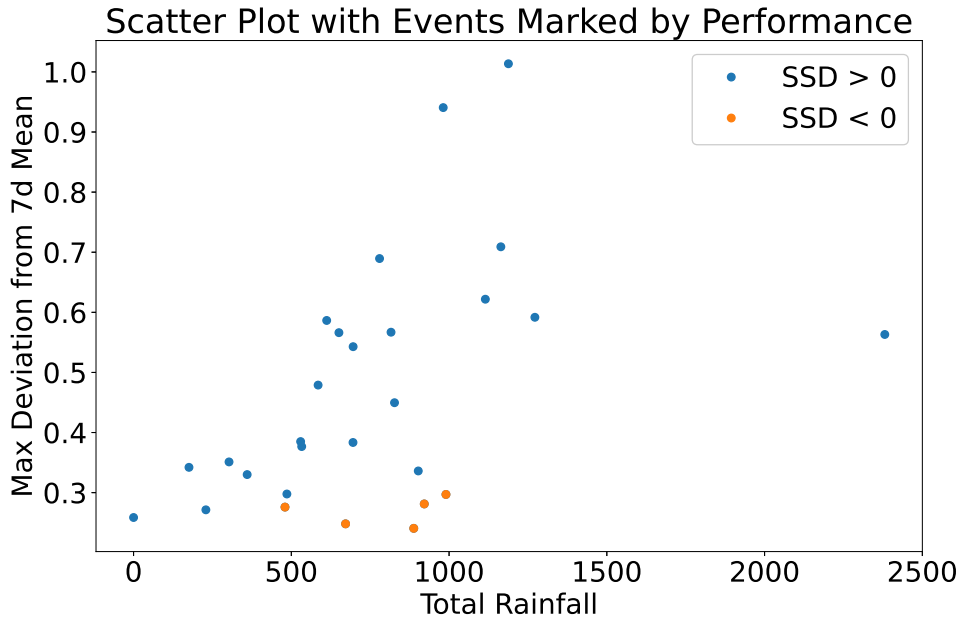


Figure 6.5: Scatter plot of validation events by total accumulated rainfall and peak water level. Note the placement of events which report a negative SSD.

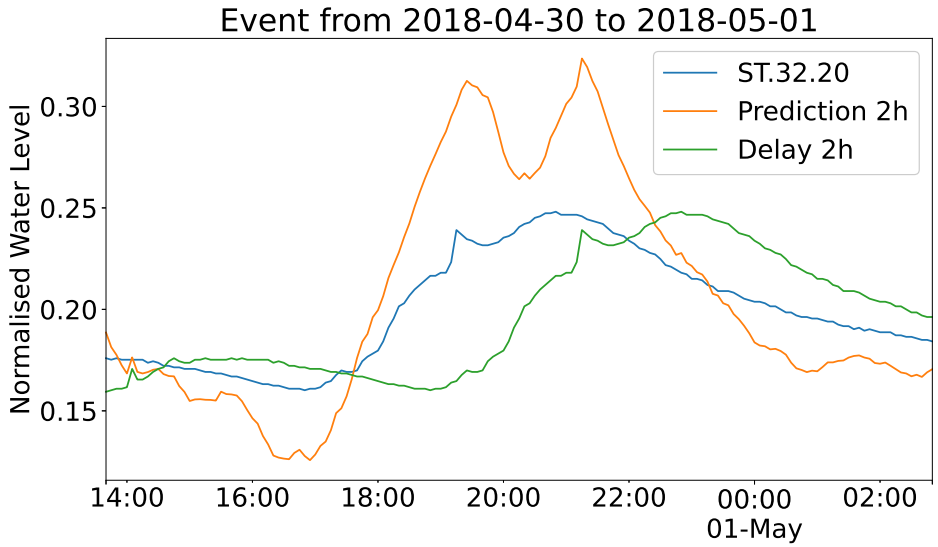


Figure 6.6: Example of network over-prediction due to rain falling over the gauge but not in catchment; SSD=-0.5, SC=94%.

Time	30min	1h	2h	3h	4h	Mean
Overall	0.385	0.551	0.667	0.670	0.602	0.575
Winter	0.470	0.582	0.682	0.694	0.655	0.616
Summer	0.203	0.265	0.255	0.257	0.175	0.231

Table 6.11: Median Validation SSD computed according to the part of the year in which the validation events occurred.

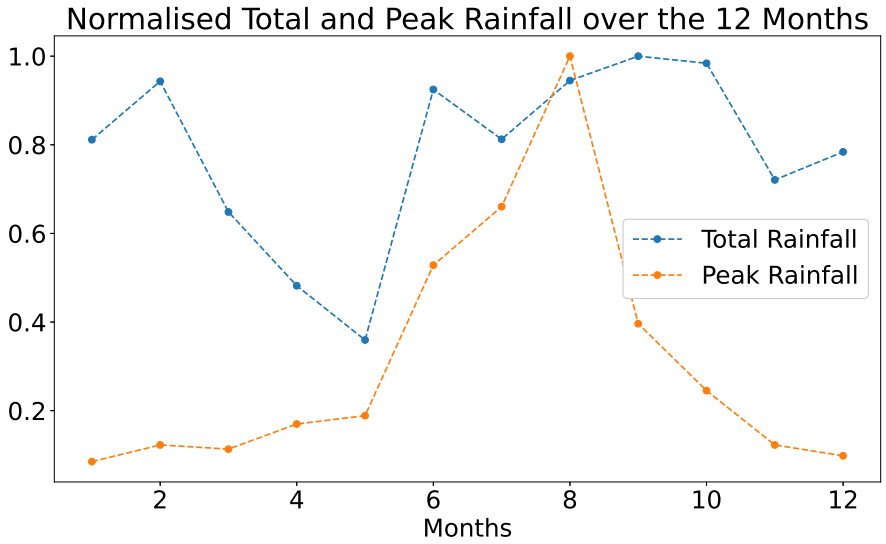


Figure 6.7: Normalised Total and Peak Rainfall for each of the 12 months, computed over the period 2016-01-01 to 2020-12-31.

winter validation events has a maximum value of 0.15, in the summer validation events it reaches a maximum of 0.57.

Furthermore, the set of events is dominated by winter events (65 vs 17). This coupled with the difference in temporal rainfall distribution causes the network to significantly over-predict the effect of summer rainfall since the network is primarily trained on winter rainfall.

Finally, we will look at two more validation events. The first (6.8) is one that exhibits acceptable performance. The second event (6.9) starts with acceptable performance and then is effected by an unmeasured input (likely the fordelerbygværk).

6.8 Evaluation of Overall Performance

In the previous sections, we analysed the model performance on the events in the validation set. However, the future data that we want to apply the model on does not only include events but also non-events (periods where the water level at Abe-

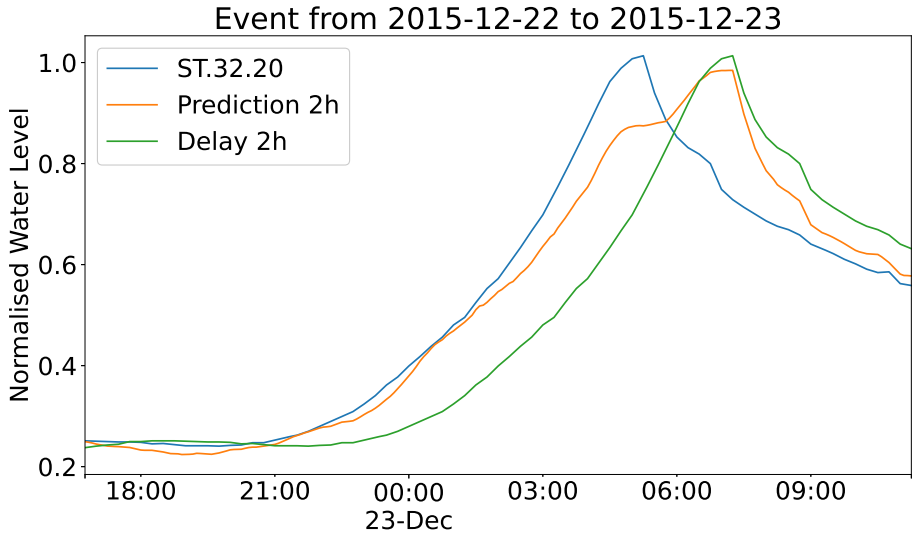


Figure 6.8: Example of acceptable prediction on a validation event; network predicts general direction, but under-predicts rise in water level; SSD=0.75, SC=94%.

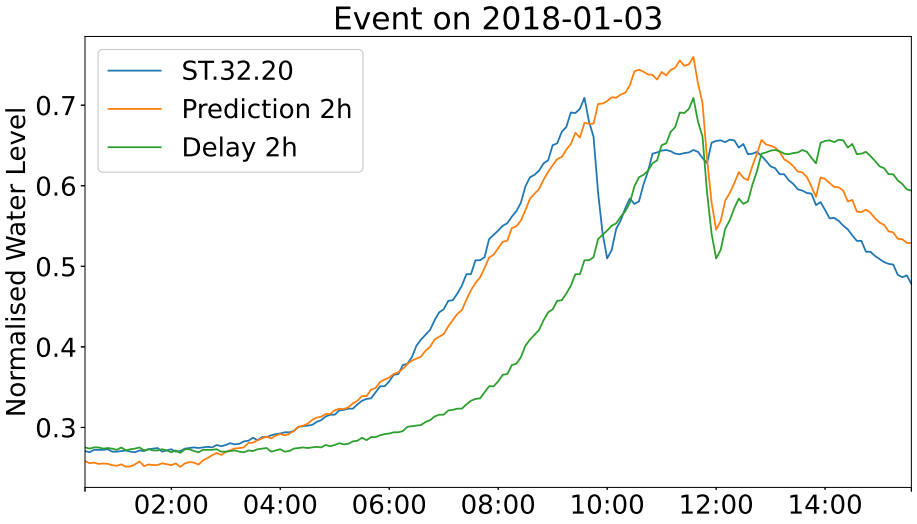


Figure 6.9: Example of prediction error when the water level is driven by an unaccounted input; SSD=0.68, SC=79%.

lones Plads does not diverge significantly from the 7-day moving average). We apply the ResLSTM-8 model on the entire dataset and compute the loss function values. Computing the five predictions over a period of almost 6 years took approximately 4 minutes. Missing values are handled by linear interpolation in the case of the water level at Abelones Plads and 0 for the rainfall and water level change inputs. We note that since the LSTM is a stable system (it was trained on bounded inputs and bounded outputs), the effect of the error introduced by replacing the missing values is expected to vanish with time.

The computed loss function values for the different prediction times are given in 6.12. It can be seen that while the absolute losses MAE and RMSE were reduced significantly, the relative losses (SSD and associated metrics) do not outperform the zero-order forecast benchmark model in any significant fashion. We caution against considering the SSD and similar metrics (also including the Sign Conformance) when the model is applied on time periods where the stream is in a baseflow scenario. In such cases, the water level is nearly static, therefore the zero-order-forecast benchmark model is very accurate. Similarly, the ResLSTM prediction will vary around 0, therefore the SC metrics are distorted by the effect of noise. However, the absolute loss metrics indicate that the model does not diverge significantly in baseflow situations compared to events of interest. The fact that the ResLSTM fails to outperform the zero-order-forecast benchmark model during baseflow situations is considered acceptable given that no flood warnings are required in such cases.

Figure 6.10 visualises the comparison of the 2h prediction and the measured water levels over the time period of 2021 (up until 2021-10-11, where the dataset ends) and figure 6.11 visualises the error of the 2h prediction over the same time period (note that we define the error as $w - \widehat{w}$; i.e. a positive error indicates the model under-predicts the water level). It can be seen that the network tracks the peaks of the winter events but under-predicts the water level in base-flow situations in the winter. Furthermore, the predictions in the summer exhibit intermittent behaviour in excess of the actual water level. Recall that we truncated the events after the time at which the water level peaked. This has the result that the network has learned that the water level has to drop when rainfall has ceased but it has not learned the full extent of the decay behaviour. We hypothesise that this is responsible for the under-prediction in the winter. The error spikes in the summer are likely due to the difference in temporal distribution in the rainfall data, i.e. the network over-predicts the effect of a short rainfall burst.

Table 6.12 suggests that the Sign Conformance metrics are lower than the results from the validation events would suggest. We hypothesise that this is due to the high occurrence of changes with small magnitudes in the dataset since it is dominated by baseflow situations. We therefore compute the *Sign Conformance Profile*, which is the sign conformance metric considering only changes of magnitude

Prediction Time	30min	1h	2h	3h	4h
MAE	0.34	0.52	1.22	1.23	1.27
RMSE	0.59	0.94	1.81	2.10	2.51
SC	63.89	67.21	68.63	70.33	70.94
SC^-	79.34	82.96	87.14	88.80	89.39
SC^+	39.15	41.9993	39.00	40.78	41.42
NMSE	1.047	0.866	1.113	0.831	0.802
SSD	-0.047	0.134	-0.113	0.169	0.198
ASSD	-1.279	-1.027	-1.631	-0.889	-0.559
RSSD	-0.023	0.069	-0.055	0.088	0.104

Table 6.12: Obtained values for different loss functions for the different prediction times computed over the entire dataset.

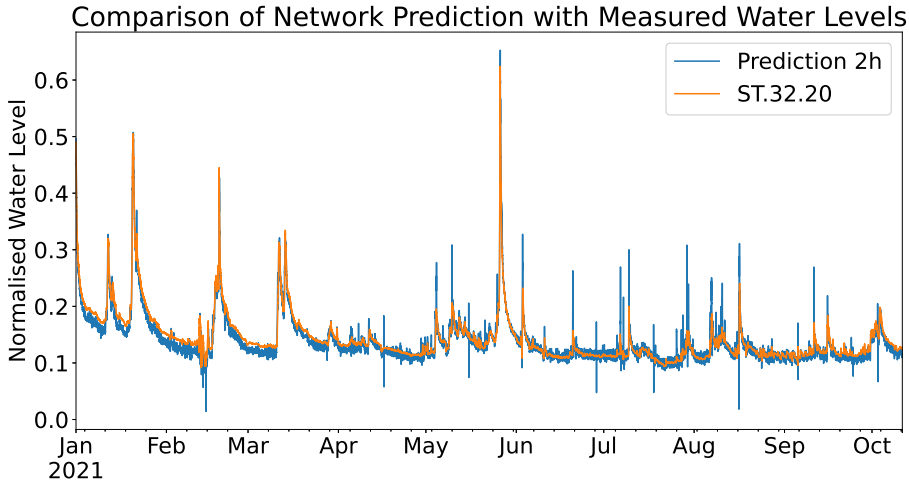


Figure 6.10: Comparison of the 2h prediction with the measured water levels computed over time period of 2021 as far as data was available.

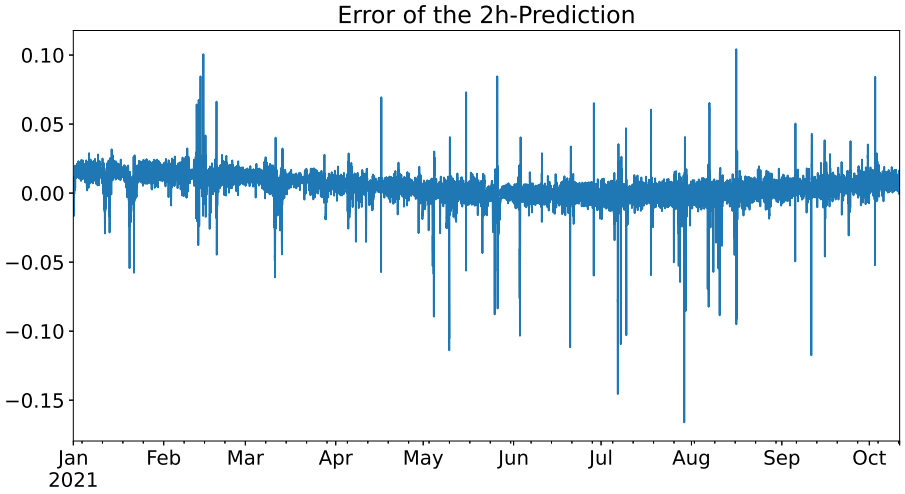


Figure 6.11: Error of the 2h prediction computed over the time period of 2021 as far as data was available.

ε where ε is a set of test values. Additionally, we compute the occurrence of those changes over the dataset as given in 6.1.

The Sign Conformance $SC(\varepsilon)$ and the associated Occurrence $o(\varepsilon)$ for test values $\varepsilon \in [0\,\text{cm}, 5\,\text{cm}]$ are shown in figure 6.12. Table 6.13 provides numerical values of Sign Conformance and Occurrence for selected values of ε. As can be seen, the SC metrics increase roughly monotonically (with the exception of the 30 minute prediction) as a function of the test value. The Occurrence rapidly decreases monotonically as a function of the test value which indicates that Δw is approximately normally distributed. We note that the Occurrence should be understood as an estimate of the complement to the cumulative distribution function. This would suggest that $o(0) = 1$. However, Δw is a discrete variable (with a numerical precision of 0.1 cm). The algorithm that computes the SC was set up to omit all samples where $|\Delta w| \leq \varepsilon$, i.e. $SC(0)$ considers only samples $|\Delta w| > 0$ where. Consequently, the occurrence of zero-change samples is simply $1 - o(0)$. Due to the slow nature of the system, the discretisation, the frequency of baseflow situations, the occurrence of zero-change samples is between 45% and 17% depending on the prediction time. We can conclude that in a real implementation, we expect an SC of more than 85% when predicting a change of at least 1 cm. This aligns with our findings regarding the median validation SC (see table 6.9)

$$o(\varepsilon) = \frac{1}{N} \sum_{k=1}^{N} u(|\Delta w_k| - \varepsilon), \quad u(0) = 0 \tag{6.1}$$

Recall that we emphasised the need to investigate the Sign Conformance based

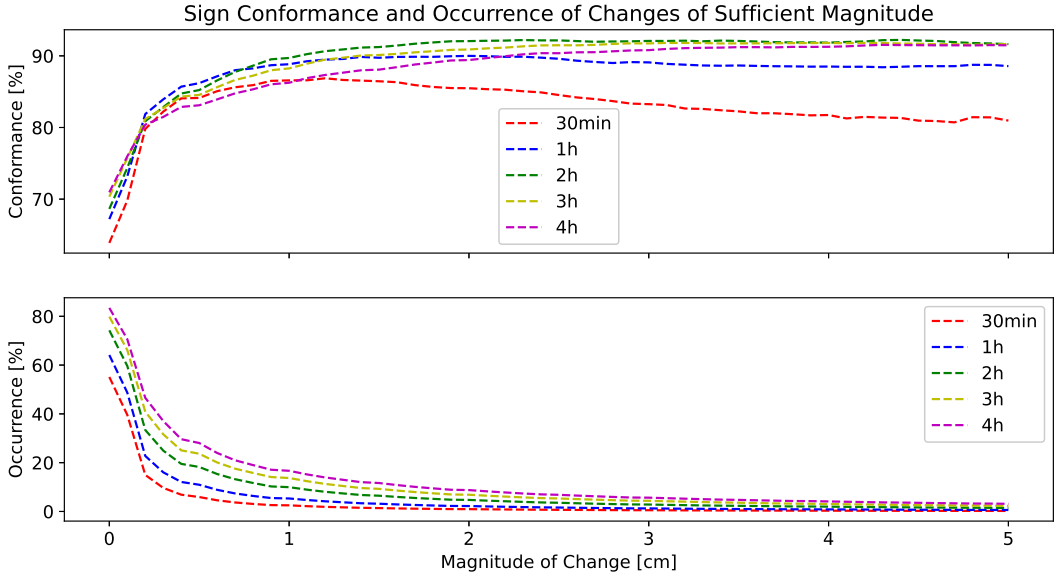


Figure 6.12: Sign Conformance Profile computed over the entire dataset using cut-off values for the magnitude of the change Δw from 0 to 5 centimetres. The figure also shows the occurrence $p(|\Delta w| > \varepsilon)$.

ε [cm]		0	1	2	3	4	5
$SC(\varepsilon)$ [%]	30min	63.89	86.57	85.48	83.26	81.74	80.96
	1h	67.21	88.83	90.00	89.09	88.51	88.57
	2h	68.63	89.71	92.06	92.08	91.84	91.63
	3h	70.33	88.25	90.89	91.74	91.80	91.66
	4h	70.94	86.24	89.41	90.80	91.27	91.47
$o(\varepsilon)$ [%]	30min	55.13	2.51	0.95	0.51	0.32	0.22
	1h	64.13	5.33	2.22	1.26	0.85	0.60
	2h	74.26	9.96	4.73	2.85	1.97	1.44
	3h	79.86	13.72	6.86	4.38	3.06	2.31
	4h	83.44	16.67	8.77	5.65	4.11	3.14

Table 6.13: Sign Conformance and Occurrence values for the different prediction times and selected values of ε.

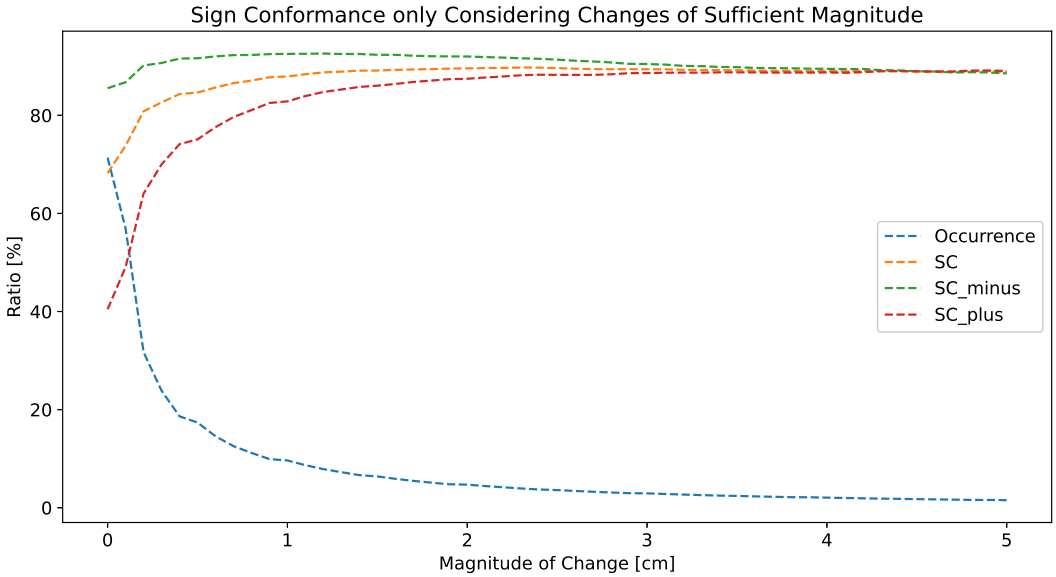


Figure 6.13: SC, SC^+ and SC^- and Occurrence averaged over the prediction times as a function of ε.

ε [cm]	0	1	2	3	4	5
$o(\varepsilon)$ [%]	71.36	9.64	4.71	2.93	2.06	1.54
$SC(\varepsilon)$ [%]	68.20	87.92	89.57	89.39	89.03	88.86
$SC^-(\varepsilon)$ [%]	85.53	92.53	91.99	90.46	89.48	88.54
$SC^+(\varepsilon)$ [%]	40.47	82.84	87.45	88.61	88.74	89.04

Table 6.14: Sign Conformance and Occurrence values for selected values of ε.

on the sign of the change. Figure 6.13 shows the three different SC metrics (averaged over the prediction times) along the Occurrence as a function of the test value. Table 6.14 provides numerical values for selected values of ε. The figure shows that the model performs better for negative changes $\Delta w < 0$ compared to positive changes $\Delta w > 0$ but the two metrics converge as ε increases. Recall that for the set of validation metrics, the opposite was found ($SC^+ > SC^-$). We hypothesise that this difference is due to the fact that the ResLSTM-8 model was found to under-predict the change during base-flow situations in the winter. There, the occurrence of small positive and small negative changes is roughly even but since the model tends to predict $\Delta\widehat{w} < 0$, the Sign Conformance metric is punished more harshly for positive changes. As we increase the cut-off value, we expect this effect to disappear which aligns with our findings.

Chapter 7

Conclusion

In this chapter, we will summarise the work done in this project and make conclusions regarding the performance of the specific application in Vejle as well as about our methodology as a generalisable method of flood forecasting applicable to other catchments.

A generalisable method of flood forecasting via the LSTM-based prediction of water levels at locations of interest has been developed. We detailed the event identification and event classification process to generate training and validation data useful for training. We further developed a heuristic feature selection algorithm to automatically generate a set of moving average blocks for the rainfall measurements and forecasts given some other inputs to maximise the monotonic explanatory power of the input set with respect to some prediction time. The overall methodology is purely data driven and requires no knowledge of the underlying physical processes; only water level measurements from a critical point along with neighbouring water levels and/or rainfall measurements are required. Thus, the method is generally applicable to any catchment in which flooding is a concern, assuming sufficient data is available.

We conducted a case study by applying the method on the city of Vejle where we trained a neural network model to predict the water levels at the critical point of Abelones Plads with prediction times ranging from 30 minutes to 4 hours into the future. We showed that the model performance on the withheld validation events is superior (median SSD = 0.575) to the zero-order forecast model which was chosen as benchmark model. The SSD can be interpreted similar to the R-squared, i.e. the model is expected to explain 57.5% in the variance of the water level change that the model attempts to predict.

An important limitation was discovered during testing. The model performed acceptably well during the winter (median SSD = 0.616) but poorly during the summer (median SSD = 0.231). We attributed this difference in performance to a lack of data for summer events as well as to a different spatio-temporal distribution

of rainfall in the summer and winter. We conclude that in the winter the network performs well due to the spatio-temporal uniformity even though the rainfall data is generated by a gauge located outside the catchment. We have shown that all events, where any prediction fails to outperform the zero-order forecast, exhibit disproportionally high amounts of rainfall over the event given the deviation of the water level at Abelones Plads from the baseline level. We conclude that the poor model performance for those events are due to inadequate measurements of the rainfall and hypothesise that the difference in performance would be mitigated if radar measurements were included in the model inputs.

Based on our experiences with the subject of the case study which is a catchment with a quick response time and thus difficult to model (as stated by the cooperative partner), we conclude that the methodology developed in detail in this work is generally capable of providing short-term flood forecast for river floods in a real-life implementation. We have shown that a stream can be modelled to an acceptable degree with limited rainfall information; even though the rain gauge we rely on was located outside the catchment itself, the neural network model was able to extrapolate the effect of the catchment rainfall from the rainfall outside the catchment. We have shown on a catchment with a quick response time that a prediction of up to 4 hours into the future can be made using a 2h rainfall nowcast. For catchments with slower response times, the maximum prediction time is expect to be larger.

Finally, we wish to make recommendations for the usage of the methodology in Vejle. We recommend usage primarily in the winter. Since the model may incur an error in predicting the numerical change in water level but has been found to generally predict the direction correctly (SC between 85% and 90%), we propose to modify the quantitative prediction to obtain a categorical prediction by keeping the sign and omitting all predictions of insufficient magnitude. The network output can then be represented as an arrow pointing either up or down. This will aid a human operator in gauging whether a flood is likely to occur or not and will remove unneeded and potentially misleading information from the eyes of the operator. Another possible approach is to keep Dropout active during inference, compute a number of stochastic predictions and then plot the prediction with confidence intervals. A more effective but also more time-consuming solution would be to train the model using radar data which we expect to significantly to lead to a significantly better performance.

Chapter 8

Discussion

In this chapter, we will discuss potential work that we think could to be done to improve upon the methodology. We will discuss the performance issues arising from the insufficient rainfall inputs and potential solutions. We will then discuss the problem of interpretability. This is especially important to data-driven flood forecasting approaches as it is necessary to convince the relevant government authorities to trust the model that was trained on the catchment. Showing that the LSTM cell states are a representation of the actual system states is helpful for building trust in neural network models.

We have shown during testing that all events in which the model failed to outperform the zero-order forecast exhibit a water level deviation from the baseline that is disproportionally low given the rainfall measured during the event. Based on this knowledge, we have concluded that in those events, the rainfall is spatially non-uniform and the rain gauge captures more rainfall that the catchment, and thus the model over-predicts the effect of the rainfall. Conversely, in events where the rainfall is primarily located in the catchment and not above the gauge, the network will under-predict the change in water level to some extent. In both cases, the model will get the direction correctly (as rainfall measurements from different locations are positively correlated) but it fails to extrapolate the rainfall in the catchment from the measurements. This cause of failure is likely solvable if the network is given more rainfall data, ideally radar data, to properly capture the spatial distribution of the rainfall and thus reduce the prediction error. For radar data, we propose feeding the radar images through a Convolutional Neural Network and the output of that as well as all other inputs into a ResLSTM. A comparable approach could be to conduct a dimensionality reduction on the radar image to obtain modes that describe the system (Principal Component Analysis or Dynamic Mode Decomposition[41]), reduce the high-dimensional radar image to a small set of modes or components and then feed them into the model. The LSTM is capable of extracting temporal features in the input data by learning how to gate the cell

state, however it is not capable of extracting spatial features since the LSTM input is a flat vector (reducing a 2d image to a 1d vector loses the spatial information).

With adequate rainfall inputs it may be possible to generate accurate predictions using only rainfall inputs. In that case, water level inputs may be omitted and only a single water level sensor located at the critical point is needed. If multiple water level measurements are available, all of them can potentially be predicted since none of them have to be withheld to be used as inputs. Furthermore, the change in water level is affected by noise (since it is a finite difference) and therefore introduces noise into the prediction. This can be seen in figure 6.10, where there is apparent noise in the prediction during a base-flow situation.

We have discussed the issue of interpretability before. Additional work should be done in this field. If a conceptual hydrological model is available, we can attempt to map the LSTM cell states to the conceptual model's internal states. Since the cell states can be interpreted as mass or energy storage, they ought to be correlated with the states of the conceptual model. This would allow us to claim that the LSTM managed to extract a representation of the system states without being given those states explicitly. If it can be shown that the LSTM extracts an implicit representation of those states, this supports our argument for using neural network models for flood forecasting. Recall how Klemeš stated that a hydrological model should reflect the underlying physical system; this criterion would then be fulfilled.

Chapter 9

Acknowledgements

I would like to express my gratitude to my professor Zhenyu Yang who provided academic guidance throughout my Master's education. I also wish to thank Benjamin Refsgaard from Vejle Spildevand A/S, the cooperative partner of this work who proposed this project, provided expert knowledge and the necessary data and also laid the groundwork to this project with his Master's thesis.

I would like to further thank my parents Sandra and Christian Kienert who supported my education in Denmark and even drove me to Esbjerg in 2017, when I started my education in a foreign country at the age of 17. In that regard, I would like to thank my former English teacher Anne Bittner who encouraged my ambitions to start an education conducted in English when I was still a high school student at Scharnhorst Gymnasium Hildesheim, as well as Aalborg University Esbjerg which has been my academic home for the past 5 years.

Finally, I would like to thank my fellow Master's students Jakob Damkjær, Spyridon Lazaris and Ubaid Bashir Wani with whom I have shared a table in our group room and who provided moral support throughout this project.

Bibliography

[1] European Environmental Agency. *Europe's changing climate hazards - Wet and dry - heavy precipitation and river floods*. Nov. 19, 2021. URL: https://www.eea.europa.eu/publications/europes-changing-climate-hazards-1/wet-and-dry-1/wet-and-dry-heavy (visited on 05/04/2022).

[2] Subutai Ahmad and Luiz Scheinkman. *How Can We Be So Dense? The Benefits of Using Highly Sparse Representations*. 2019. DOI: 10.48550/ARXIV.1903.11257. URL: https://arxiv.org/abs/1903.11257.

[3] Sanjeev Arora et al. *Stronger generalization bounds for deep nets via a compression approach*. 2018. DOI: 10.48550/ARXIV.1802.05296. URL: https://arxiv.org/abs/1802.05296 (visited on 05/13/2022).

[4] Pierre Baldi and Peter J Sadowski. "Understanding Dropout". In: *Advances in Neural Information Processing Systems*. Ed. by C.J. Burges et al. Vol. 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper/2013/file/71f6278d140af599e06ad9bf1ba03cb0-Paper.pdf.

[5] A.R Barron. "Universal approximation bounds for superpositions of a sigmoidal function". eng. In: *IEEE transactions on information theory* 39.3 (1993), pp. 930–945. ISSN: 0018-9448.

[6] KyungHyun Cho et al. "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches". In: *CoRR* abs/1409.1259 (2014). arXiv: 1409.1259. URL: http://arxiv.org/abs/1409.1259.

[7] Torch Contributors. *ReduceLROnPlateau - Pytorch 1.11.0 documentation*. URL: https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.ReduceLROnPlateau.html (visited on 05/26/2022).

[8] Copernicus. *European State of the Climate 2021 - Flooding in Europe*. 2021. URL: https://climate.copernicus.eu/esotc/2021/flooding-july (visited on 05/04/2022).

[9] G Cybenko. "Approximation by superpositions of a sigmoidal function". eng. In: *Mathematics of control, signals, and systems* 2.4 (1989), pp. 303–314. ISSN: 0932-4194.

[10] Helen Davidson. "Chinese officials arrested for concealing true scale of flood death toll". In: *The Guardian* (Jan. 23, 2022). URL: https://www.theguardian.com/world/2022/jan/23/chinese-provincial-officials-concealed-scores-of-deaths-from-flood-disaster (visited on 01/23/2022).

[11] Lars-Eric Ertlmeier. *Prediction and Adaptive Control of Water Levels in Vejle using Long Short-Term Memory Neural Networks*. Semester project. 2021.

[12] Ken-Ichi Funahashi. "On the approximate realization of continuous mappings by neural networks". In: *Neural Networks* 2.3 (1989), pp. 183–192. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(89)90003-8. URL: https://www.sciencedirect.com/science/article/pii/0893608089900038.

[13] Yarin Gal and Zoubin Ghahramani. *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning*. 2015. DOI: 10.48550/ARXIV.1506.02142. URL: https://arxiv.org/abs/1506.02142.

[14] Sangchul Hahn and Heeyoul Choi. "Gradient Acceleration in Activation Functions". In: *CoRR* abs/1806.09783 (2018). arXiv: 1806.09783. URL: http://arxiv.org/abs/1806.09783.

[15] Michael Hauser. *On Residual Networks Learning a Perturbation from Identity*. 2019. DOI: 10.48550/ARXIV.1902.04106. URL: https://arxiv.org/abs/1902.04106 (visited on 05/13/2022).

[16] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. DOI: 10.48550/ARXIV.1512.03385. URL: https://arxiv.org/abs/1512.03385 (visited on 05/13/2022).

[17] Kaiming He et al. *Identity Mappings in Deep Residual Networks*. 2016. DOI: 10.48550/ARXIV.1603.05027. URL: https://arxiv.org/abs/1603.05027 (visited on 05/13/2022).

[18] Geoffrey E. Hinton et al. *Improving neural networks by preventing co-adaptation of feature detectors*. 2012. DOI: 10.48550/ARXIV.1207.0580. URL: https://arxiv.org/abs/1207.0580.

[19] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* (1997). URL: https://www.bioinf.jku.at/publications/older/2604.pdf (visited on 03/22/2022).

[20] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(89)90020-8. URL: https://www.sciencedirect.com/science/article/pii/0893608089900208.

[21] Caihong Hu et al. “Deep learning with a long short-term memory networks approach for rainfall-runoff simulation”. eng. In: *Water (Basel)* 10.11 (2018), pp. 1543–. ISSN: 2073-4441.

[22] Intergovernmental Panel on Climate Change. *Technical Summary*. Oct. 1, 2021. URL: https://www.ipcc.ch/report/ar6/wg2/downloads/report/IPCC_AR6_WGII_FinalDraft_TechnicalSummary.pdf (visited on 05/04/2022).

[23] Stanisław Jastrzębski et al. *Residual Connections Encourage Iterative Inference*. 2017. DOI: 10.48550/ARXIV.1710.04773. URL: https://arxiv.org/abs/1710.04773 (visited on 05/13/2022).

[24] K.L.S. Gunn J.S. Marshall Walter Hitschfeld. “Advances in Radar Weather”. In: *Advances in Geophysics* 2 (1955), pp. 1–56.

[25] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[26] V. Klemeš. “Dilettantism in hydrology: Transition or destiny?” In: *Water Resources Research* 22.9S (1986), 177S–188S. DOI: https://doi.org/10.1029/WR022i09Sp0177S. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/WR022i09Sp0177S. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/WR022i09Sp0177S.

[27] Andrei Nikolaevich Kolmogorov. “On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition”. In: *Doklady Akademii Nauk*. Vol. 114. 5. Russian Academy of Sciences. 1957, pp. 953–956.

[28] Frederik Kratzert et al. “NeuralHydrology – Interpreting LSTMs in Hydrology”. eng. In: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Lecture Notes in Computer Science 11700 (2019), pp. 347–362. ISSN: 0302-9743.

[29] Jung Hwan Lee et al. “Integrated flood forecasting and warning system against flash rainfall in the small-scaled urban stream”. eng. In: *Atmosphere* 11.9 (2020), pp. 971–. ISSN: 2073-4433.

[30] Ke Li et al. “The applicability of ASCS_LSTM_ATT model for water level prediction in small- And medium-sized basins in China”. eng. In: *Journal of hydroinformatics* 22.6 (2020), pp. 1693–1717. ISSN: 1464-7141.

[31] Hongzhou Lin and Stefanie Jegelka. *ResNet with one-neuron hidden layers is a Universal Approximator*. 2018. DOI: 10.48550/ARXIV.1806.10909. URL: https://arxiv.org/abs/1806.10909 (visited on 05/13/2022).

[32] Yu Liu et al. "Short term real-time rolling forecast of urban river water levels based on lstm: A case study in Fuzhou city, China". eng. In: *International journal of environmental research and public health* 18.17 (2021), pp. 9287–. ISSN: 1661-7827.

[33] G. G. Lorentz. "Metric Entropy, Widths, and Superpositions of Functions". eng. In: *The American mathematical monthly* 69.6 (1962), pp. 469–. ISSN: 0002-9890.

[34] Marco Luppichini et al. "Deep learning models to predict flood events in fast-flowing watersheds". eng. In: *The Science of the total environment* 813 (2022), pp. 151885–151885. ISSN: 0048-9697.

[35] Eric Nalisnick, José Miguel Hernández-Lobato, and Padhraic Smyth. *Dropout as a Structured Shrinkage Prior*. 2018. DOI: 10.48550/ARXIV.1810.04045. URL: https://arxiv.org/abs/1810.04045.

[36] *Population 1. January by urban and rural areas, population, area and population density and time*. 2021. URL: https://m.statbank.dk/Data?lang=en (visited on 04/19/2022).

[37] Maryam Rahimzad et al. "Performance Comparison of an LSTM-based Deep Learning Model versus Conventional Machine Learning Algorithms for Streamflow Forecasting". eng. In: *Water resources management* 35.12 (2021), pp. 4167–4187. ISSN: 0920-4741.

[38] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. *On the Convergence of Adam and Beyond*. 2019. DOI: 10.48550/ARXIV.1904.09237. URL: https://arxiv.org/abs/1904.09237 (visited on 05/23/2022).

[39] Benjamin Refsgaard and Thomas Aagaard Jensen. "Oversvømmelsesvarsling i Vejle - Prognosticering af vandstande med neurale netværk". MA thesis.

[40] *Scalgo Live*. Available at https://scalgo.com/live/denmark?res=25.6&ll=9.502569%2C55.745734&lrs=datafordeler_skaermkort_daempet%2Cdenmark%2Fdenmark%3A25832%3Acurrent%3Arain%3Aflooded-edgeflow-dfs%3Adhm2015&watershed=9.535087%2C55.715197, last accessed on 2022-04-19.

[41] PETER J. SCHMID. "Dynamic mode decomposition of numerical and experimental data". In: *Journal of Fluid Mechanics* 656 (2010), 5–28. DOI: 10.1017/S0022112010001217.

[42] Anton Maximilian Schäfer and Hans Georg Zimmermann. "Recurrent Neural Networks Are Universal Approximators". eng. In: *Artificial Neural Networks – ICANN 2006*. Vol. 4131. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 632–640. ISBN: 9783540386254.

[43] Guy Shalev et al. *Accurate Hydrologic Modeling Using Less Information*. 2019. DOI: 10.48550/ARXIV.1911.09427. URL: https://arxiv.org/abs/1911.09427.

[44] Muhammed Sit et al. "A comprehensive review of deep learning applications in hydrology and water resources". eng. In: *Water science and technology* 82.12 (2020), pp. 2635–2670. ISSN: 0273-1223.

[45] David Sprecher. "On the Structure of Representations of Continuous Functions of Several Variables as Finite Sums of Continuous Functions of One Variable". eng. In: *Proceedings of the American Mathematical Society* 17.1 (1966), pp. 98–105. ISSN: 0002-9939.

[46] Nitish Srivastava et al. "Dropout: A simple way to prevent neural networks from overfitting". eng. In: *Journal of machine learning research* 15 (2014), pp. 1929–1958. ISSN: 1532-4435.

[47] Andreas Veit, Michael Wilber, and Serge Belongie. *Residual Networks Behave Like Ensembles of Relatively Shallow Networks*. 2016. DOI: 10.48550/ARXIV.1605.06431. URL: https://arxiv.org/abs/1605.06431 (visited on 05/13/2022).

[48] Jin Wang, Bo Peng, and Xuejie Zhang. "Using a stacked residual LSTM model for sentiment intensity prediction". eng. In: *Neurocomputing (Amsterdam)* 322 (2018), pp. 93–101. ISSN: 0925-2312.

[49] World Meteorological Organization. *WMO Atlas of Mortality and Economic Losses from Weather, Climate and Water Extremes (1970–2019)*. 2021. URL: https://library.wmo.int/doc_num.php?explnum_id=10989 (visited on 05/05/2022).

[50] Shuzhi Yu and Carlo Tomasi. *Identity Connections in Residual Nets Improve Noise Stability*. 2019. DOI: 10.48550/ARXIV.1905.10944. URL: https://arxiv.org/abs/1905.10944 (visited on 05/13/2022).

[51] Alireza Zaeemzadeh, Nazanin Rahnavard, and Mubarak Shah. *Norm-Preservation: Why Residual Networks Can Become Extremely Deep?* 2018. DOI: 10.48550/ARXIV.1805.07477. URL: https://arxiv.org/abs/1805.07477 (visited on 05/13/2022).