Master Thesis
Utrecht University

# Conditional Forecasting of Water Level Time Series with Recurrent Neural Networks

Bart van der Lugt
ICA-4126904

July 3, 2019

**Abstract**

We describe a practical situation in which the application of forecasting models could lead to energy efficiency and decreased risk in water level management. The practical challenge of forecasting water levels in the next 24 hours and the available data are provided by a dutch regional water authority. We formalized the problem as conditional forecasting of hydrological time series: the resulting models can be used for real-life scenario evaluation and decision support. We propose the novel *Encoder/Decoder with Exogenous Variables* RNN (ED-RNN) architecture for conditional forecasting with RNNs, and contrast its performance with various other time series forecasting models. We show that the performance of the ED-RNN architecture is comparable to the best performing alternative model (a feedforward ANN for direct forecasting), and more accurately captures short-term fluctuations in the water heights.

# Contents


**1 Introduction** **4**

**2 Related Work** **5**
2.1 Hydrological Models . . . 5
2.2 Noordoostpolder . . . 5
2.3 Hydrological Time Series Forecasting . . . 5
2.4 Machine Learning . . . 6
2.5 Artificial Neural Networks . . . 6
2.6 Recursive and Direct Forecasting . . . 7

**3 Problem Description** **8**
3.1 Region of Interest . . . 8
3.2 Research Question . . . 9
3.3 Conditional Forecasting . . . 10
3.4 Data Assessment . . . 10

**4 Time Series Analysis** **13**
4.1 Time Series Modelling . . . 13
4.2 Vector Autoregressive Models . . . 15
4.3 Vector Error Correction Model . . . 16
4.4 Exogenous Variables . . . 17
4.5 Conditional forecasting . . . 18
4.6 Application . . . 18

**5 Recurrent Neural Networks** **20**
5.1 Feedforward ANN . . . 20
5.2 RNN Definition . . . 20
5.3 Gated RNNs . . . 21
5.4 Conditional Forecasting using RNNs . . . 22
5.5 Encoder/Decoder RNN with Exogenous Variables . . . 22

**6 Methodology** **24**
6.1 Exogenous Variables . . . 24
6.2 Feature Construction . . . 25
6.3 Forecast Generation . . . 25
6.3.1 Recursive forecasting . . . 26
6.3.2 Direct Forecasting . . . 26
6.4 Performance . . . 27
6.5 VAR and VECM modelling . . . 27
6.6 Feedforward Neural Networks . . . 27
6.7 Recurrent Neural Networks . . . 28
6.8 Conditional Forecasting . . . 29

**7 Results** **30**
7.1 VAR Models . . . 30
7.2 Feedforward Neural Networks . . . 33
7.3 Recurrent Neural Networks . . . 41
7.4 Model Comparison . . . 47
7.5 Conditional Forecasting . . . 51

**8 Conclusion** **57**

**9 Future Work** **58**

# 1 Introduction

In the Netherlands, water is all around us: knowing how to manage this water is key to sustaining our way of life. New technologies and an exponential increase in the amount of data available generate new possibilities in the field of hydrology. Accurate forecasts of weather, water levels and flow rates allow water boards (Waterschappen) to limit risk of flooding, drought damage and energy waste. Water boards are regional government bodies responsible for water quality, water levels and safety.

From this practice, the relevance of accurate *conditional forecasts* of time series data becomes especially clear. Conditional forecasts are a useful means of evaluating the impact of a hypothetical scenario. The goal is to predict the variables of interest conditioned on an *assumed future path* of one or more other variables in the system. These forecasts can be used to guide the decision making process by comparing various scenarios.

Artificial Neural Network (ANN) models have become very popular in forecasting water level time series. They provide a good alternative to the traditional time series models (such as vector autoregressive (VAR) models) because ANNs do not assume linear dependencies. Recurrent Neural Networks (RNNs) were specifically designed to process sequential data, allowing for the model to retain short-term and long-term memory of the input series, thereby improving the performance of traditional feedforward ANNs on many machine learning tasks. Despite their great potential, the application of RNNs to conditional time series forecasting has not been extensively studied in the literature.

The method we propose in this thesis is to use an Encoder/Decoder RNN architecture to generate conditional forecasts of time series data. Our approach distinguishes itself by considering the future path of the variables we conditioned on in the decoding step. It can therefore be characterized as an *Encoder/Decoder RNN with Exogenous Variables*. The resulting architecture is very flexible and can be used to model many real-world time series. In our experiments, we compare the performance of the ED-RNN with various other popular forecasting models in a conditional forecasting scenario. This forecasting problem, as well as the dataset of hydrological variables that is used for training and testing, is provided by the Waterschap Zuiderzeeland.

The rest of this thesis is structured as follows. First, we present relevant literature regarding hydrological time series forecasting. We then provide a more in-depth description of the forecasting problem, traditional time series analysis and RNN modelling of time series. Lastly, we detail our methodology and the results of our experiments and discuss their implications for the field.

# 2 Related Work

## 2.1 Hydrological Models

Traditionally, using domain experts to construct hydrological models is how most of the research in water resource systems was performed. These models usually consist of various pre-defined formulas describing the relations between observable variables. Available data is used to tune the parameters of these formulas. Studies showed that levels [39] [47], flow dynamics [34] and quality [17] of surface water could be effectively modelled with this approach. However, a comparative study of such models for ground water levels was conducted by Konikow and Bredehoeft [29]. In each case, they showed, it was impossible to scientifically verify and validate such models, and argued that calibration procedures generate non-unique solutions with limited predictive accuracy.

## 2.2 Noordoostpolder

Extensive research into the geographical region of interest (the Noordoostpolder in Flevoland, The Netherlands) has been conducted in 2006 [46]. In their evaluation study, hydrologists from the FutureWater agency studied the optimal influx of water to the region. This influx was evaluated for each of the sub-regions in the polder for typical normal, wet and dry years. Additionally, a damage assessment was made in the case of a complete halt of influx, and an influx threshold was determined to guarantee certain quality norms. This was done by constructing several hydrological models that relied heavily on expert knowledge and human data analysis. The researchers showed that there was no historical relation between rainfall and manual water influx from other regions. This was an unexpected result, as it would indicate a waste of precipitation water. The researchers concluded that the current influx management in the region is required only for drought protection and water quality, as opposed to a broader selection of goals that was previously determined.

## 2.3 Hydrological Time Series Forecasting

In this section, we will describe a more analytic approach to modelling hydrological systems, based on statistical methods. Hydrological data is often a sequence of measurements over time, and can thus be characterized as a time series. Time series data is always highly correlated, as each measurement is dependent on a single variable: time [7]. This type of data is well-studied for several purposes, such as the prediction of the future based on knowledge of the past, and to create an understanding of the system underlying the measurements [5]. Besides being a time series, water system data is often multivariate, as there are many observable features which are all potentially relevant. This results in complex relations within the measurement sequences, such as short-term and long-term dependencies between variables, as well as seasonality. Granger [14] describes several possible ways in which multiple time series can be related (*co-integrated* ) and the serious implications this has for the choice of modelling them.

For this study we will focus on the literature concerning time series *forecasting*. Modelling methods based on linear regression [35] can be used for forecasting hydrological time series. The challenge when using these types of models is taking the strong temporal correlation into account. This approach has been well-studied in the development of Auto Regressive (AR) models [18], which fit a linear equation of previous measurements to predict the next value. Important for these models is the lag parameter $p$, which specifies how many time steps from the past are used in the equation, thereby restricting the complexity of the model. Extensions of the AR model involve

adding moving averages, external regressors, analyzing differences and including trend and seasonal components [25]. Additionally, the AR model is the special case of the Vector Auto Regressive (VAR) model [28], which is suited for multivariate instead of univariate time series. Early studies [49] [18] concluded that traditional VAR models are however unable to learn both short- and long-term dependencies between variables in a single model. Still, like the AR model, VAR models can easily be extended to include additional information, trend and seasonal components, as well as a specific lag structure per variable, resulting in a Structured VAR (SVAR) model.

A special case of forecasting multivariate time series is *conditional forecasting*. Conditional forecasts are a useful means of evaluating the impact of hypothetical scenarios. The goal is to predict the variables of interest conditioned on an assumed path of one or more other variables in the system [31]. For VAR models, conditional forecasts for independent variables are generated by adding them as external regressors to the model. Conditional forecasts for dependent variables are typically computed using the algorithm developed by Waggoner and Zha [44].

Epskamp et al. [12] have shown that the parameters of various VAR models can be used to create a graphical representation of linear dependencies within a system very effectively. These relations are categorized as contemporaneous, temporal and between-subject dependencies and are presented as network structures, which can be used for exploratory analysis. In our case, the between-subject dependencies are irrelevant, as they concern different time series for 'subjects', such as questionnaires filled in by different people.

A limitation that remains is that VAR models assume linear dependence over time, as well as linear dependence between the variables. The linear temporal dependence is not guaranteed for hydrological time series. Additionally, water systems often contain non-linear dependencies among the variables, as shown by Heuvelmans et al. [22].

## 2.4 Machine Learning

In time series forecasting, an important focus is creating insight into the system underlying a series of measurements. Studies in the last two decades have drawn attention to *machine learning* methods, which are slightly different in their focus. These models can be characterized as non-parametric and data-driven [33]. Their main goal is to improve on a prediction task. This is accomplished by using only historical data to learn and utilize stochastic dependencies in the underlying system. The field combines knowledge from both computing science and statistics, leading to a wide variety of models and specifications. Common challenges in machine learning solutions include a lack of labeled data, missing or imbalanced data [37] and limited interpretability of the model [43].

## 2.5 Artificial Neural Networks

Artificial Neural Network (ANN) models have become very popular in predicting water levels based purely on historical data. The advantages of the ANN model are that it creates a nonlinear input-output mapping, making it suitable for the estimation of complex systems, and that it derives it's parameters purely from historical data [38]. An important challenge in using a traditional ANN for time series modelling is that it processes each time step separately. Therefore, the information on the temporal correlation is not readily used by the model.

A comparative study of ANN models for groundwater level prediction was conducted by Yoon et al. [48]. Their solution to modelling the time series was to *lag* the input variables, i.e. adding additional features containing measurements from the past. The correlation of observations over

time is therefore turned into a correlation between features within a single observation, which can be modelled using machine learning. This method is the most straightforward, and often involves an autocorrelation analysis to determine the time period over which measurements are delayed.

An ANN was also used by Tiwari et al. [42] to develop a model for accurate hourly flood prediction in an Indian river basin up to 10 hours in advance. Instead of lagging the variables, the authors decomposed every input period as a weighted combination of *wavelet functions*, effectively moving the input space from the time domain to the frequency domain. This method is similar to performing a Fourier Transform on the input space of the ANN, and was first proposed and applied by Wang et al. [45]. This same method was successfully applied by Anctil and Tape [1] for rainfall-runoff forecasting by decomposing into short, intermediate and long wavelet period sub-series.

Recurrent Neural Networks (RNNs) were specifically designed to process sequential data, by considering network output of previous time steps in later iterations [13]. The difficult task of learning both short-term and long-term dependencies without losing efficiency is a well-studied problem, resulting in specialized model architectures. Examples of this are Gated RNN models such as the LSTM [24] and the GRU [9]. These allow for the incorporation of different timescales and informative missing values into the model architecture to achieve better predictive results [8]. Various RNN architectures were compared and successfully applied by Groenen [15] for seasonality extraction, residual learning and accurate prediction of wastewater inflow at municipal wastewater treatment plants. However, a comparative study by Bontsema [6] on wastewater effluent quality showed that regression trees and traditional feature construction might outperform LSTMs for multivariate time series.

## 2.6 Recursive and Direct Forecasting

There are generally two approaches to forecasting multiple subsequent observations of a time series. A recursive forecasting (sometimes also called iterative forecasting) model repeatedly generates a prediction for a single period ahead, using a fixed window of past observations and predictions of past observations if their actual values are unknown. A direct forecasting model generates a prediction multiple time steps ahead that is independent of the observations in between.

Traditionally, statistical time series models such as the VAR model generate only recursive forecasts. They can however be modified to produce forecasts for multiple time steps ahead. McCracken et al. [31] detail how VAR models can be used for both direct and recursive forecasting. They then empirically compare the two methods for conditional forecasting and conclude that the recursive method often yields better results than the direct method, especially for longer forecasting periods.

There have been many studies aimed at determining the difference in forecasting performance when generating recursive or direct forecasts using ANNs. Hill et al. [23] compared the quality of VAR forecasts, recursive ANN forecasts and direct ANN forecasts for the 'well-known' 111 M-competition data. They showed that the performance of the ANN using the recursive forecasting methodology was superior. On the other hand, Mishra et al. [32] showed that in forecasting drought index time series using ANNs, the direct method is superior when predicting more than 2 periods into the future. They argued that this is due to the accumulation of errors when a recursive forecast is generated. In a comparative study by Hamzaçebi et al. [19] it was shown for 6 different time series experiments that direct forecasting using ANNs yields significantly better results than recursive forecasting.

# 3 Problem Description

This section first describes the water management system in the Noordoostpolder. Information from earlier studies in the region, as well as expert knowledge, is presented here. We then present the research question and describe the problems and challenges that we expect, as well as the available data.

## 3.1 Region of Interest

The Noordoostpolder is the northern section of the Dutch province Flevoland. It has a total land surface of about 460 $km^2$ and an average land altitude of about 5 meters below sea level (NAP). This low altitude is a result of Flevolands history: the entire region is a result of the drainage of the Zuiderzee between 1936 and 1942.

The water management system in the Noordoostpolder is governed by employees of the Waterschap Zuiderzeeland. Their job is to manage the influx and efflux of water into the region, and distribution of water to the local farmers. The influx of water is required to counteract drought damage, to supply livestock and greenhouse cultivation, to ensure water quality and the management of nature reserves [46].

Along the borders of the Noordoostpolder, there are 12 locations where water is let into the system using syphons ('hevels') and intakes. An important characteristic of the Noordoostpolder is it's low altitude (-5m NAP), resulting in continuous groundwater seepage ('kwel') from the higher surrounding regions. Therefore, even when all intakes are closed, the water levels in the Noordoostpolder will rise. This water is distributed along the various ditches to accommodate the water demand of the local farmers. Water then flows to one of the 3 major channels, from where it is pumped out by 3 large water pumping stations ('gemalen'). These are operated manually by employees of the Waterschap Zuiderzeeland, according to a standard procedure and various protocols for dry and wet scenarios.

Hydrologist at Waterschap Zuiderzeeland made an assessment of vital locations in the region based on connectivity, distance, flow directions and the way the water system is managed. These are shown in Figure 1. Points A, B and C are the large water pumping stations. Blue lines are the major water channels leading to them. Points 2 (Marknessebrug), 4 (Marknessersluis) and 6 are locations of low altitude at the center of the water management system. These locations are therefore indicative of the general water height of the Noordoostpolder. They are also positioned in large channels, which results in less extreme water level fluctuations and trends. Locations 1, 3 and 5 are positioned upstream: measurements at these points are indicative of impoundment and may predict fluctuations at downstream locations ahead of time.

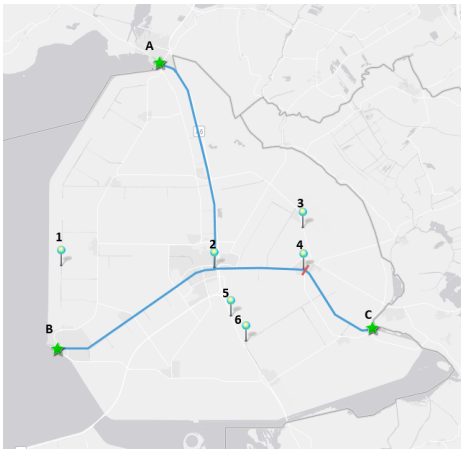



Figure 1: Vital measurement locations in the water management system of the Noordoostpolder

## 3.2 Research Question

The aim of the Waterschap Zuiderzeeland is to reduce the amount of water that is wasted in the Noordoostpolder. Water is pumped out of the system as a precautionary measure when rainfall is expected, to prevent flooding. However, pumping out more water than is required in a given scenario is problematic: if the water levels are too low, the water demand as described in Section 3.1 can not be met. In such cases, an influx of water from nearby regions is required, and the efflux of water is considered waste.

The goal of this study is to provide the water managers at the Waterschap Zuiderzeeland more insight into the possible effects of expected rainfall and drainage given the current state of the Noordoostpolder. Our expectation is that an accurate prediction of how the water management system in the Noordoostpolder will behave in certain situations will lead to a more efficient use of the water resources. The research question of this study is as follows:

*How can we use machine learning to predict the fluctuations in surface water levels in the Noordoostpolder under the influence of precipitation and water management actions?*

For further specificity, we have divided the research question into two sub questions:

1. How can we use statistical methods for time series forecasting, such as vector autoregression (VAR) models, to generate a conditional forecast of the surface water levels?

2. How can we generate a more accurate conditional forecast using machine learning models, specifically, using (recurrent) neural networks?

In answering these questions, we restrict ourselves to predicting the surface water level fluctuations at locations 2, 3 and 5 in Figure 1. Location 2 is chosen because it is very indicative of the central water height in the Noordoostpolder, and therefore a prediction of it's value is very useful. Locations 3 and 5 are more difficult to predict due to their higher variability, while still providing relevant information about water height trends in the polder. The following section describes the available data of the water management system in the Noordoostpolder, and concludes with our research hypothesis based on this data.

## 3.3 Conditional Forecasting

The Waterschap Zuiderzeeland is not merely interested in a prediction of waterheight fluctuations in the next $x$ hours. Rather, they wish to use this prediction to guide their management decision making process. Therefore, it is required that they are able to contrast certain scenario's and compare forecasts, to be able to take better action. That is why the focus in our research question is partially on the *influence of precipitation and water management actions*. The predictions we generate can be characterized as *conditional forecasts*, which are well-studied in the literature. For each of our modelling choices, we will describe in detail how these conditional forecasts can be generated.

## 3.4 Data Assessment

A significant amount of data is being recorded at locations in the Noordoostpolder describing the local characteristics of the water management system. We distinguish the following types of measurements: height of surface water or ground water, volumetric flow rate, evaporation, and weather data. Weather data is recorded at a KNMI weather station [26] in Marknesse, and includes measurements of wind speed and wind direction, precipitation and sun exposure. For this study, we are only interested in values of sensors that are currently still operative: locations without measurements in 2018 were filtered out. For each type, the number of distinct measurement locations in the Noordoostpolder, as well as the sampling frequency, is indicated below.

- **Surface water height**: 189 locations, 15 minute measurements
- **Ground water height**: 58 locations, hourly measurements
- **Flow rate**: 38 locations, 15 minute measurements
- **Evaporation**: 104 locations, daily measurements
- **Weather**: 1 location, hourly measurements

Surface water level measurements, as well as flow rates for the large pumping stations, are available for each of the vital points described in Figure 1. Additionally, ground water level measurements are available for locations near the center of the Noordoostpolder. Evaporation data was extracted from satellite images, which were aggregated over various geographical regions to provide meaningful minimal, maximal and average data per subregion. Almost all of these measurement series contain continuous data from at least 2015 until 2018.

We introduce our own shorthand notation to be able to efficiently reference measurement locations in our dataset. Table 1 below contains the names for some of the most relevant and most often used locations, together with the official identifier used by the Waterschap Zuiderzeeland and an indication of it's location in Figure 1.

| Name | Identifier | Type | Location |
|---|---|---|---|
| WH2 | NOP.PM4810.LT1 | Surface water | 2 |
| WH3 | MP6002 | Surface water | 3 |
| WH5 | NOP.ST4775.LT3 | Surface water | 5 |
| GW1 | 21BN.093.01 | Ground water | near C |
| GW2 | 15HN.018.01 | Ground water | between 1 and A |
| FR2000 | NOP.2000_TOT | Flow rate | A |
| FR2100 | NOP.2100_TOT | Flow rate | C |
| FR2200 | NOP.2200_TOT | Flow rate | B |
| MWSP | KNMI Marknesse | Precipitation | near 4 |

Table 1: Shorthand names introduced to reference important locations in this work, along with their official identifier defined by Waterschap Zuiderzeeland, type of measurement recorded at the location, and an indication of it's position in the Noordoostpolder in Figure 1

We test the level data (ground and surface water) for stationarity and cointegration, since these are relevant properties for the time series modelling approach that we intend to use. Stationarity implies that statistical properties of the data do not vary over time. Many hypothesis tests for stationarity have been developed in the past, and we evaluate them on the most important surface and ground water level series. We report here first of all the results of the augmented Dickey-Fuller (ADF) and Phillips-Perron (PP) tests [7]. Their null-hypothesis is that the time series has a unit root, which is equivalent to non-stationarity. It is believed that these tests have a bias towards stationarity when the data contains structural breaks (i.e., changes in trend). Additionally, it is possible for a series to have no unit root while also being non-stationary. We also report the results of the Kwiatkowski-Philips-Schmidt-Shin (KPSS) [7] and the Priestley-Subba Rao (PSR) tests [36]. Their null-hypothesis is that the time series is (trend-)stationary. This is tested by checking if in the presence of a shock, the process converges back to some (growing) mean.

| | ADF | PP | KPSS | PSR |
|---|---|---|---|---|
| WH2 | **0.01** | **0.01** | 0.01 | 0 |
| WH3 | **0.01** | **0.01** | 0.01 | 0 |
| WH5 | **0.01** | **0.01** | 0.01 | 0 |
| GW1 | **0.01** | **0.01** | 0.01 | 0 |
| GW2 | **0.01** | **0.015** | 0.01 | 0 |

Table 2: Statistical p-values for Dickey-Fuller (ADF), Kwiatkowski-Philips-Schmidt-Shin (KPSS), Phillips-Perron (PP) and Priestley-Subba Rao (PSR) tests for (non-)stationarity of water height time series. In bold: test statistics implying stationarity.

The test results are inconclusive: some tests imply stationarity, while the other tests imply non-stationarity. However, the tests implying stationarity are the ones believed to be biased towards this result under certain conditions. Visual inspection of the various water height series indicates that the series are *most likely not stationary*. Most importantly, because the water levels occasionally move to a significantly different level for several weeks or months, as a result of specific water management actions. These fluctuations can be seen as the structural breaks leading to the mixed results. Since we can not draw inference about the trend changes in water height from the time

series alone, the safest assumption would be to treat the data as non-stationary.

Using various Johansen tests [3] we are unable to determine a good quality cointegration relationship. The tests examine the common stochastic trends in the multiple time series {WH2, WH3, WH5, GW1, GW2}. While the statistical tests imply that there should be one, the cointegration relationships suggested by the Johansen tests appear to be just as non-stationary as the original series. We can argue why this is the case: the trends that are present in the original data are not due to a stochastic process, but are the result of specific, local water management actions. It is unlikely that the resulting trends are present in each of the water height series, and therefore do not cancel out when we take a linear combination of each of these series. Information about the water management actions is present in the flow rates data, which is probably suitable for explaining the trends in the water levels data.

In conclusion, a significant amount of good quality data is available to generate a prediction of the desired water levels using a supervised learning approach. While it is difficult to determine properties of some stochastic process underlying the data, we have reason to believe that the individual time series are strongly related. Thus, our research hypothesis is that we can predict fluctuations in the surface water levels at the target locations using this data very well. 'Very well' in this context means: no more than 5 cm error between the predicted water levels and the actual historical water levels in our test set, for rainfall scenarios that happen at least once every year on average.

# 4 Time Series Analysis

This section expands upon the literature review on Time Series Analysis in Section 2. Below, we provide a more in-depth summary of the research conducted in this field. Our focus is on the research that is potentially relevant for the hydrological problem we are considering. If possible, we try to describe how the subjects in this chapter can be linked to our own data analysis and prediction task.

## 4.1 Time Series Modelling

A time series is a set of observations $x_t$, each one being recorded at a specific time $t$. [7] A *discrete* time series is one where observations are made at instances from a discrete set of times, i.e., at fixed intervals. Each of the time series used for this work (water height, precipitation, evaporation, etc..) are all discrete time series.

The goal of time series analysis is drawing inference from them. Generally, this is done by first choosing a hypothetical probability model to represent the data and then to estimate and validate it's parameters using the observations. The model can then be used to provide a compact representation of the data, extract seasonal patterns, or for noise filtering, forecasting, hypothesis testing and simulation studies [7].

Formally, forecasting of discrete time series using past and present data can be defined as follows. Let $y_t$ denote the value of some variable in period $t$. Then, a forecast $\hat{y}$ of it's value for the next period, made at time $t$, may have the form: [30]

$$\hat{y}_{t+1} = f(y_t, y_{t-1}, ...)$$

where $f$ denotes some function of the past and present observations. To arrive at such a function, we start by defining the probability model underlying our data. An example of this is the AR(1) model [7], an *autoregressive* model of order 1. It can be written as:

$$y_t = c + \phi_1 y_{t-1} + \epsilon_t \tag{1}$$

where c and $\phi_1$ are constant and $\epsilon_t$ is a noise term. Specifically, $\epsilon_t$ is *white noise*, defined as a series of random variables with mean 0 and variance $\sigma^2$. The parameters of this equation can be estimated from data, by performing a least squares regression on each of the observations.

Forecasting one period ahead, at some period $t$ called the *forecast origin*, can be done most easily by calculating:

$$\hat{y}_{t+1} = c + \phi_1 y_t$$

effectively replacing the noise $\epsilon_{t+1}$ with it's expected value, i.e. it's mean, 0. Additionally, we can do this recursively to obtain a forecast $\hat{y}_{t+h}$ for any $h > 0$ if we assume $y_{t+i} = \hat{y}_{t+i}$ for $0 < i < h$. That means we treat the forecasts generated for an earlier interval as the actual values when we need them for a new forecast. The number of periods $h$ into the future for which we generate a forecast is called the *forecast horizon*. Increasing $h$ also increases the number of noise terms we

assume to be 0, meaning that the uncertainty we have about the value of $\hat{y}_{t+h}$ increases when $h$ becomes larger. [7].

It is not difficult to see that the AR(1) model from (1) can easily be extended to also include earlier values of $y_t$. If $p$ earlier values are used in the model equation, this is called an AR model with *lag order* $p$, or an AR($p$) model. Each of these lagged variables is assigned it's own parameter $\phi$ in the equation.

Instead of using past values of $y_t$ to produce a forecast, a moving average model uses the forecast residuals (errors) in a regression model [25]. These errors are similar to the noise term $\epsilon_t$ we have seen earlier, so we write the moving average model with lag order 1, the MA(1) model, as follows:

$$y_t = c + \theta_1 \epsilon_{t-1} + \epsilon_t \tag{2}$$

where c and $\theta_1$ are constant and $\epsilon_t$ is white noise. Similarly to the AR model, the moving average model can be extended to include any number $q$ of previous variables $\epsilon_t$ in the equation, to create an MA($q$) model. Since the past residuals are known at time $t$, generating a forecast $\hat{y}_{t+1}$ at time $t$ is analogous to the AR model case, again assuming that $\epsilon_{t+1} = 0$.

Note that this model is very different from the AR model, since the value of $y_t$ in the MA(1) model is only dependent on the outcome of two random variables. Note however that $y_t$ and $y_{t-1}$ both depend on the variable $\epsilon_{t-1}$, and are therefore strongly correlated. The time series resulting from a moving average process will thus be strongly autocorrelated. More specifically, with respect to certain conditions, an AR($p$) model can be written as an MA($\infty$) model, and an MA($q$) model can be written as an AR($\infty$) model [7].

Combining the two models above creates a new hypothetical probability model from which data could be generated, namely, the ARMA model. It has both parameters $p$ and $q$ as above, such that the ARMA($p$,$q$) model can be written as follows:

$$y_t = c + \phi_1 y_{t-1} + ... + \phi_p y_{t-p} + \theta_1 \epsilon_{t-1} + ... + \theta_q \epsilon_{t-q} + \epsilon_t \tag{3}$$

where c, $\phi_i$ and $\theta_i$ are constant and $\epsilon_t$ is white noise.

More specifically, the model above is a special instance of the ARIMA($p$,$d$,$q$) model, having $d = 0$. This parameter $d$ defines the degree of differencing that is done (using $y'_t = y_t - y_{t-1}$) before modelling the time series. Differencing helps to ensure *stationarity* of the time series $y'_t$, which is an important condition for modelling a linear time series, and intuitively means that it's statistical properties are independent of time [25]. ARIMA models are widely studied and used in time series analysis, providing a general and flexible framework for studying complex univariate processes [7].

Before estimating the parameters of these models using a least-squares regression, we must determine the lag orders $p$ and $q$ of the model that best fits the data. Very often, these can not be theoretically determined from the domain beforehand. In such cases, we can use plots of the autocorrelations (ACF) and the partial autocorrelations (PACF) in the data for this. Significant values in these plots for certain lag orders can be used to argue about what the values of $p$ and $q$ in the data might be [25]. Specifically, we look for certain patterns in the ACF and PACF plots that are often observed for AR($p$) or MA($q$) models. Additionally, we can evaluate and compare estimated models with different $p, q$ values using an Information Criterion (IC). The idea is to assign a score to a model using it's forecasting performance and/or the model complexity, and choosing the best scoring model to represent the data. Examples of such criteria are the Akaike IC (AIC) and the Bayesian IC (BIC) [7].

## 4.2 Vector Autoregressive Models

Section 4.1 describes the foundation of univariate time series analysis. In many applications, significant forecasting improvements can be made when external variables are included. In addition, the relationships between different variables over time can be modelled and analyzed, for example to draw conclusions regarding causality or structure. In the case of the Waterschap Zuiderzeeland, many different data sources are indeed available, and we thus turn to multiple time series analysis.

When dealing with multiple time series, one should beware of discovering *spurious correlations*. This problem of wrongly drawing inference from data arises when a regression model is fitted to two time series that are non-stationary [30]. In the previous section, it was briefly mentioned that differencing the time series is one solution, but this goes at the cost of possibly ignoring long-run relationships between the levels of multiple time series.

Therefore, the concept of *co-integration* is very important and well-studied in multiple time series analysis. First described by Granger [14], co-integration describes in mathematical terms if two time series are related. The two-step Engle-Granger procedure [11] can be used to test this relationship: the underlying intuition is that stochastic trends in two non-stationary series can cancel out if they are co-integrated. Existence of co-integration thus has serious implications for modelling the time series, since we do not have to worry about spurious correlations then.

So, if either stationarity or co-integration are guaranteed, we can safely model the relationship of multiple time series using a regression model. The most often-used model for stationary series is the Vector Autoregressive (VAR) model, which is the multivariate generalization of the AR($p$) model we have seen earlier. The main uses of VAR models are forecasting and structural analysis.[30].

Let $\mathbf{y}_t = \{y_{1,t}, ..., y_{k,t}\}$ be a multiple time series. We define this as a vector of $k$ different time series $y_{i,t}$, each related to a different variable $i \in [1, k]$, with measurements at identical time intervals $t$. Then the VAR model of lag order 1, or VAR(1), is defined as follows:

$$\mathbf{y}_t = \mathbf{c} + \phi_1 \mathbf{y}_{t-1} + \epsilon_t \tag{4}$$

where $\mathbf{c}$ is a $k \times 1$ vector containing constants, $\phi$ is a $k \times k$ matrix containing constants and $\epsilon_t$ is a $k \times 1$ vector containing iid white noise. Note that the $\times$ symbol denotes the general matrix-vector product.

Alternatively, we can write the above definition without the vector notation. Assuming $k = 2$, we write the VAR(1) model as follows:

$$\begin{aligned} y_{1,t} &= c_1 + \phi_{1,1,1} y_{1,t-1} + \phi_{1,1,2} y_{2,t-1} + \epsilon_{t,1} \\ y_{2,t} &= c_2 + \phi_{1,2,1} y_{1,t-1} + \phi_{1,2,2} y_{2,t-1} + \epsilon_{t,2} \end{aligned} \tag{5}$$

where $c_i$ and $\phi_{i,j,k}$ are constants and $\epsilon_{t,i}$ is white noise. Notice how this notation introduces a multitude of subscript symbols, identifying the elements within the vectors and matrices. We use the following subscript ordering: the first subscript variable defines the time-related information of the element, the consecutive subscripts determine the row and column in the element, respectively. Increasing the lag order, thereby defining the general VAR($p$) model, is analogous to the univariate case. This extension increases the number of matrices $\phi_i$ that we use in the computation, but not their dimensions.

Selection of a sensible value for the lag order $p$ of the VAR model is quite different from the univariate case. While some authors have proposed to use multivariate correlograms similar to the

ACF and PACF plots, this approach often does not work well in selecting a good quality VAR($p$) model. In his book, Lütkepohl [30] describes several testing principles, such as the likelihood-ratio (LR) test, Lagrange Multiplier (LM) and Walt tests that can be used instead. These tests can be applied to the data using the hypothesis that a certain lag is irrelevant for the model, and therefore provide a good indication of the desired lag order for the VAR model. Another method that is often used is evaluation and comparison of estimated models using the AIC or BIC, as described for the univariate case.

Parameters of the VAR($p$) model can be estimated in exactly the same way as with the AR($p$) model, namely by simply performing a least-squares regression. This also theoretically minimizes the MSE of the forecast generated by the model, which is often used as a performance metric. The definition of the VAR model provides formulae for the expected MSE and the forecast values that minimize this expected error. From these, it is derived that the optimal long-term forecast is the process mean $\mathbf{c}$, indicating that the past of the VAR process contains no information about developments in the distant future [30]. Additionally, the VAR($p$) definition can be used to derive bounds for confidence intervals of the forecast value.

Once the parameters of the VAR model have been tuned, they can be analyzed to derive certain properties of the VAR process. Specifically, we could look at the roots of the characteristic polynomial [30] and make sure that they are all larger than 1 in absolute value. If this is ensured, the VAR process is called *stable*. Stability implies the absence of trend and seasonality, constant variance, and stationarity of the individual time series. While these are all desirable properties when modelling multiple time series, many real-world series are unstable and therefore do not satisfy this constraint.

Secondly, the VAR($p$) model can be analyzed for causality, for which several definitions exist. Assume we have two time series, $x_t$ and $z_t$. *Granger causality* is often used, intuitively defined as an improvement in prediction accuracy of $z_{t+h}$ if some $x_{t-s}$ is included in the model for $h, s > 0$. *Instantaneous causality* is defined as an improvement in prediction accuracy of $z_{t+1}$ if $x_{t+1}$ is included in the model. Each of these causality relationships can be easily checked by looking at the coefficient matrices $\phi_i$. Both definitions assume that all other relevant information of the process, past and present, is included in the model.

## 4.3 Vector Error Correction Model

Cointegrated processes have quite different properties from stationary ones and this has to be taken into account in the statistical analysis and modelling [30]. Vector Error Correction model (VECM) are specifically designed and often used to model these types of data. The model is based on a VAR($p$) model of a differenced multiple time series.

Rather than modelling $y_t$, the VECM provides a model for $\Delta y_t = y_t - y_{t-1}$, in which the coefficients are functions of the original VAR parameters. The VECM incorporates one (or more) additional feature in the regression functions: the long-run *equilibrium* of the time series. This equilibrium is defined as the cointegration relationship that was mentioned earlier [30]. Again, let $\mathbf{y}_t = \{y_{1,t}, ..., y_{k,t}\}$ be a multiple time series. The cointegration relationship is defined as a linear combination of all variables, and thus has the following form:

$$C_t = \beta_0 + \beta_1 y_{1,t} + ... + \beta_k y_{k,t} \tag{6}$$

in which $\beta_i$ are constant. The definition of the cointegration relationship states that this linear

combination must be stationary, while the individual time series are not stationary. For now, let us assume that there is only one cointegration relationship present in the data. The VECM(1) model is then defined as follows:

$$\Delta \mathbf{y}_t = \mathbf{c} + \theta_1 \Delta \mathbf{y}_{t-1} + \alpha C_{t-1} + \epsilon_t \tag{7}$$

where $\mathbf{c}$ is a $k \times 1$ vector containing constants, $\theta$ is a $k \times k$ matrix containing constants and $\epsilon_t$ is a $k \times 1$ vector containing iid white noise. $C_{t-1}$ is a scalar and $\alpha$ is a $k \times 1$ vector containing constants.

Notice how similar the definitions of the VAR(1) model and VECM(1) model are. Increasing the lag order $p$ of the VECM, defining the VECM($p$), is similar to the VAR model. Additional parameters $\theta_i$ are added to include variables $\Delta \mathbf{x}_{t-i}$ into the regression function, but no earlier values of the cointegration term $C_t$ are used.

The cointegration term thus encodes information about the long-run behavior (equilibrium) of the variables. From the definition of the VECM model, it follows that $\Delta y_t$ is modelled such that the system moves back to this equilibrium. Multiple time series with a common stochastic trend are assumed to occur very frequently in real-world applications. VECMs offer a convenient way to parametrize and specify such cointegrated processes, and can be extended just as easily and in a similar fashion as VAR models.

## 4.4 Exogenous Variables

So far, we have assumed that all variables of the multiple time series are determined within the system. These types of variables are called *endogenous*. In practice, the generation process may be affected by other observable variables which are not affected by the system of interest [7]. Such variables are called *exogenous variables*. For example, rainfall is often considered exogenous in modelling farming and crop output.

Suppose we extend the definition of the VAR(1) model, as in equation 4, with some $l \times 1$ vector $\mathbf{x}_t$ containing $l$ exogenous variables, we would then get the following:

$$\mathbf{y}_t = \mathbf{c} + \phi_1 \mathbf{y}_{t-1} + \beta_0 \mathbf{x}_t + \epsilon_t \tag{8}$$

where the target variable $\mathbf{y}_t$ is a vector of size $k \times 1$ and $\beta_0$ is an $k \times l$ matrix containing constants. The above definition can be extended similarly as before, by including additional lags of the variable $\mathbf{x}_t$ and thereby introducing new coefficient matrices $\beta_i$.

Note however, that the definition dictates that the value of $\mathbf{x}$ at time $t$ is known when forecasting $\mathbf{y}$ at time $t$. In other words, the exogenous variables at the forecast horizon $T+h$ must be observed beforehand, namely, at the time of the forecast origin $T$. Since VAR models were often aimed at modelling low-frequency data, for example economic variables, this was no problem: the value of $\mathbf{x}_t$ could simply be determined in between $T$ and $T+h$, and the forecast would still be generated early enough to be informative. In practice however, it is possible that observations of exogenous variables are not available beforehand, and in that case, the term $\mathbf{x}_t$ is simply omitted from the model definition. Another solution is to not use the observed values of $\mathbf{x}_t$, but to first forecast the series $\mathbf{x}$ and then use the *predicted* values of $\mathbf{x}_t$.

In summary, treating variables as exogenous is important, since it can make time series modelling easier and more efficient. While in practice, the distinction between endogenous and exogenous

variables may not always be clear, it is useful to think of the time series model as a function with input and output: the endogenous variable at time $t$ serves as the output of the function and it's past values serve as input, but the exogenous variable serves purely as input.

## 4.5 Conditional forecasting

With a VAR model, we could forecast all variables in our multiple time series over some period $h$, all at once. This would give us a general idea of how the system's variables will simultaneously behave after some time $t$. We call this an *unconditional* forecast, as each of the multiple time series is unrestricted between periods $t$ and $t+h$. Alternatively, we can also generate a *conditional* forecast, which depends on some predefined future path of one or more variables. Many applications for these types of forecasts exist, especially when one of the variables can be controlled or managed: in such cases, several conditional forecasts can be compared to aid the decision making process.

The easiest way of generating conditional forecasts is by treating the conditioning variables as *exogenous*. In that case, we can define the future paths beforehand, since the exogenous variables are not affected by other variables. We can then forecast the endogenous variables under the influence of the future exogenous variables, using the original time series model. However, when we wish to make assumptions regarding a future path of an *endogenous* variable, things get more complicated.

One way of defining a future path for endogenous variables is by placing restrictions on some of the variables between time $t$ and $t+h$. We distinguish hard restrictions, which completely fix the future path, and soft restrictions that place bounds on the future values. For VAR models, the conditional forecasts are then typically computed using the algorithm developed by Waggoner and Zha [44]. Their methodology involves sampling techniques to draw future paths satisfying the conditions, and from these, a set of conditional forecasts are generated. In their paper, they show that monthly macroeconomic variables can be forecasted over a 4 year period under influence of a restricted federal funds rate. Stock and Watson [40] evaluate this method and show how it can be used effectively for policy analysis and scenario comparison. However, the algorithm by Waggoner and Zha can become unfeasible for higher dimensional time series and long forecast horizons due to the sampling procedure [2].

In response, Banbura et al. [2] proposed a recursive conditional forecasting algorithm that reduces the computational burden. Their approach is based on Kalman filtering techniques: variables for which no future path is assumed are treated as time series with missing data, which are then estimated using the fitted VAR model. The authors show that sizeable improvements in computational performance can be achieved by using this procedure, especially when the number of conditioning variables is large ($> 15$) or the forecast horizon is long ($h \geq 20$). The algorithms in their comparison were evaluated on a dataset with 68 observations of 26 macroeconomic variables, and conditions involved a relative increase of world GDP as well as fixed future paths for 3 variables.

## 4.6 Application

In conclusion, the literature on time series analysis provides us with many tools to model the time series data of the water management system in the Noordoostpolder. Specifically, the methods involving VAR models and conditional forecasting seem perfectly suited for the goals of this project, namely, modelling the relationships between water height measurements, environmental variables and flow rates, and comparing rainfall and flow rate scenarios.

However, one potential problem arises when we compare the data used in most time series

literature with the data from the Waterschap Zuiderzeeland. While the water management data we wish to study contains more than 1 million observations of more than 50 features, the *typical macroeconomic sample involves a limited number of data points, generally, 200 - 250 data points* [2]. The implications of this are unsure, but Hyndman mentions in his book [25]:

*Most time series models do not work well for very long time series. The problem is that real data do not come from the models we use. When the number of observations is not large (say up to about 200) the models often work well as an approximation to whatever process generated the data. But eventually we will have enough data that the difference between the true process and the model starts to become more obvious.*

Therefore, we suspect that time series modelling in the traditional sense is not the preferred approach for the data that we have.

# 5 Recurrent Neural Networks

Traditional machine learning models, such as the ANN, were developed to determine and exploit stochastic dependencies within individual samples of data. Recurrent Neural Networks (RNNs) are an extension to this approach: they are designed to model *sequences* of data. Sequential data has become very common: a video is a sequence of images, a sentence is a sequence of words. The ability to derive information from the sequence context instead of from a single sample has therefore become very important. This sections details the definition of the RNN, various specialized architectures such as the LSTM, and describes the application of RNNs to time series data.

## 5.1 Feedforward ANN

Before we detail the recurrent neural networks, we first consider the traditional feedforward Artificial Neural Network (ANN) [21], also called Multilayer Perceptron (MLP). This generic model architecture is used to learn a non-linear mapping from some input vector $\mathbf{I}$ to output $\mathbf{O}$, and is a generalization of the linear VAR/VECM models. It consists of $k$ layers $L_i$, with $i = 1, \ldots, k$, having $l_i$ neurons in each layer, and each neuron produces a single real-valued activation [38]. Neurons in the input layer are activated using input data, simply using $L_1 = \mathbf{I}$. Neurons in consecutive layers are activated through weighted connections with the previous layer, as follows:

$$L_{i+1} = \alpha(\mathbf{W}_i L_i + b_i) \qquad \text{for } i = 1, \ldots, k-1 \tag{9}$$

where $L_i$ and $L_{i+1}$ are vectors of size $l_i \times 1$ and $l_{i+1} \times 1$ respectively, $\mathbf{W}_i$ is an $l_{i+1} \times l_i$ matrix containing coefficients (weights) and $b_i$ is a vector of size $l_{i+1} \times 1$ containing coefficients. The function $\alpha : \mathbb{R} \to \mathbb{R}$ is some non-linear *activation function* that is applied element-wise. Layers $L_2, \ldots, L_{k-1}$ are called *hidden layers*. The procedure described above is called the *feedforward step*, producing the output $L_k$: using a learning algorithm such as gradient descent backpropagation [21], the error between $L_k$ and the desired output $\mathbf{O}$ is iteratively decreased. For this, a loss function must be defined to quantify the difference between the network output and the desired output: weight updates are then calculated based on the gradient of the loss function.

We apply the generic feedforward ANN to time series forecasting in two ways. Recursive forecasting can be done using $\mathbf{I} = \{\mathbf{y}_{t-p}, \ldots, \mathbf{y}_t, \mathbf{x}_{t-p}, \ldots, \mathbf{x}_{t+1}\}$ as input, and $\mathbf{O} = \{\hat{\mathbf{y}}_{t+1}\}$. We thus have $l_1 = (p+1)(n+m)+m$ and $l_k = n$. This network can now be used recursively, using previous predictions as input, to generate forecasts for time steps $t+1, \ldots, t+h-1$. Alternatively, we can implement an ANN for direct forecasting by choosing $\mathbf{I} = \{\mathbf{y}_{t-p}, \ldots, \mathbf{y}_t, \mathbf{x}_{t-p}, \ldots, \mathbf{x}_{t+h}\}$ as input, and $\mathbf{O} = \{\hat{\mathbf{y}}_{t+1}, \ldots, \hat{\mathbf{y}}_{t+h}\}$. In this case, we have $l_1 = (p+1)(n+m) + hm$ and $l_k = hn$. This network is not used recursively, but instead generates predictions for all desired time steps using a single feedforward operation.

## 5.2 RNN Definition

Recurrent Neural Networks (RNNs) provide an extension to feedforward ANNs, in that they are specifically designed to process sequences of observations. Let us assume a sequence has a fixed length $P$. Instead of concatenating many observations into a single input vector like the ANN, the RNN takes a multivariate time series $\mathbf{I}_t \in \mathbb{R}^m$ for $1 \leq t \leq P$ as input, mapping it to the output

series $\mathbf{O}_t \in \mathbb{R}^n$ by processing each time step $t$ individually. What also distinguishes the RNN architecture is that a layer of RNN neurons uses the layer's activations at the previous time step as additional input. A single RNN layer $L_{i+1}$ thus computes:

$$L_{i+1,t} = \alpha(\mathbf{W}_1\sigma(L_{i+1,t-1}) + \mathbf{W}_2 L_{i,t} + b) \qquad \text{for } t = 1, \ldots, P \tag{10}$$

where the function $\sigma : \mathbb{R} \to \mathbb{R}$ is an element-wise function and the other terms are similar to the feedforward ANN case. The term $\sigma(L_{i,t})$ is called the hidden state of the $i$'th network layer: its initial state $\sigma(L_{i,0})$ can be specified by the user, set to 0 or be learned.

Learning weights of an RNN can be done by using a modified version of the backpropagation algorithm. The idea is that the RNN layer is *unrolled* to create a feedforward ANN with $P$ identical layers, and calculating the gradients of the loss functions for each output [13]. Several problems may arise: *exploding gradients* refers to a large increase in the norm of gradients for long-term components, whereas *vanishing gradients* refers to a sharp decline in these gradients [4]. Both of these events make it very difficult for the model to correctly learn correlations over longer periods of time.

## 5.3 Gated RNNs

To counteract the vanishing or exploding gradient problems, specific RNN architectures were developed. *Gated RNNs* are among the most effective resulting model types, introducing a change in the structure of neurons, allowing the models to consider long-term information more effectively. The intuition behind these architectures is that it might be required that the hidden states in the model are *reset to 0* frequently.

The *Long Short-Term Memory* (LSTM) [13] model was among the first architectures to introduce self-loops in recurrent neurons. These self-loops produce pathways through which information flows: in addition to the hidden state, which serves as the neuron's output, each LSTM neuron has a cell state that is used as memory. Information in these pathways is controlled by various gates. Many variants of the LSTM neuron architecture exist, but we present here the most common one. The *forget gate* uses the current input and previous hidden state to determine if information from the cell state can be discarded. The *input gate* used the same terms to determine if information can be added to the cell state. The *output gate* uses this new cell state, the current input and previous hidden state to generate a new hidden state. Each of these gates is essentially a neuron from a feedforward ANN: they take a vector of input values, multiply with a specific weight matrix that can be tuned during training, and then apply some activation function.

The *Gated Recurrent Unit* (GRU) [10] is another architecture for gated neurons in recurrent neural networks. Similar to the LSTM neuron, the GRU neuron has gates that control the flow of information, however, without the need for a specific cell state as memory. The GRU has only two gates: the *reset gate* and the *update gate*, which both use the current layer input and previous hidden state for computation. The output of these gates is combined in a specific way to generate the new hidden state. Intuitively, the reset gate defines how to combine the previous hidden state and the current input, whereas the update gate defines the portion of the previous hidden state that is retained. The GRU neuron is simpler compared to the LSTM neuron: it has fewer gates and thus fewer parameters, and also applies one less activation function.

## 5.4 Conditional Forecasting using RNNs

The most common solution is to define an RNN that generates 1-ahead predictions, as seen in [16]. For this model, we choose $\mathbf{I}_t = \{\mathbf{y}_{t-1}, \mathbf{x}_t\}$ and $\mathbf{O}_t = \{\mathbf{y}_t\}$. However, for multi-step ahead forecasting, this recursive approach is counter intuitive. The RNN uses as input both its previous prediction and its previous hidden state: this could distort the memory management of the RNN and lead to extreme accumulation of errors for large $h$.

The second approach is to use $\mathbf{I}_t = \{\mathbf{y}_{t-h}, \mathbf{x}_t\}$ and $\mathbf{O}_t = \{\mathbf{y}_t\}$. This way, the RNN is not applied recursively and its memory is efficiently used. However, the input series is now composed of variables with substantial time shifts, which could lead to problems. To prevent the extraction of misleading patterns, we are required to use a moving average series of $\mathbf{y}_t$ instead of the actual measurements, resulting in a loss of information. Additionally, this approach forces us to use input sequences of $\mathbf{y}$ and $\mathbf{x}$ of the same length and we can not use $\mathbf{x}_t$ for $t < T$.

## 5.5 Encoder/Decoder RNN with Exogenous Variables

The above shows that conditional forecasting using RNNs does not work very well, so we propose a novel architecture for this problem. Requirements are that it allows us to use input sequences of $\mathbf{y}$ and $\mathbf{x}$ of varying length without time shifts, and that the memory management of the RNN is used efficiently during forecasting.

Our proposed approach to conditional forecasting with RNNs can be described as an *encoder/decoder approach with exogenous variables*. It consist of two steps: first, given origin $T$, horizon $h$ and lag $p$, we encode the observations in $\mathbf{x}$ and $\mathbf{y}$ at times $T - p \leq t < T$ using an RNN layer of size $l_1$. This can simply be done using $\mathbf{I}_t^1 = \{\mathbf{y}_t, \mathbf{x}_t\}$, since all these observations are known. We only keep the last output of this RNN layer, instead of the entire series. This resulting output, which we call the encoding $E$, is a vector of values of size $l_1 \times 1$.

The second step is to turn the output of the first RNN layer, $E$, into predictions for $\mathbf{y}_{T+1}, \ldots, \mathbf{y}_{T+h}$. We do this by concatenating copies of the encoding to each time step in $\mathbf{x}_{T+1}, \ldots, \mathbf{x}_{T+h}$: the result is a new time series $\mathbf{z}_{T+1}, \ldots, \mathbf{z}_{T+h}$, with $\mathbf{z}_t = \{\mathbf{x}_t, E\}$. We use this combined time series as input to the second RNN layer, thus we write $\mathbf{I}_t^2 = \{\mathbf{x}_t, E\}$. This second RNN layer produces the predictions for $\mathbf{y}_{T+1}, \ldots, \mathbf{y}_{T+h}$, thus we write $\mathbf{O}_t = \{\mathbf{y}_t\}$. While the architecture appears to consist of two separate models, the encoder can not be trained without a decoder. In summary, our architecture generates predictions $\mathbf{O}_T, \ldots, \mathbf{O}_{T+h-1}$ using an encoding of $\mathbf{I}_{T-p}^1, \ldots, \mathbf{I}_{T-1}^1$ and using $\mathbf{I}_T^2, \ldots, \mathbf{I}_{T+h-1}^2$ as additional information during decoding. Figure 2 contains a simplified, schematic representation of this RNN architecture.

The procedure of encoding and decoding a time series using RNNs is also called *sequence to sequence* (seq2seq) modelling, as proposed by Sutskever et al. [41]. In that work, it was not applied to measurement time series, but to words: translating sentences from one language to another, with good results. The most important difference with our application, and therefore the novelty of our approach, is that the initial seq2seq modelling was not conditional: there was no notion of exogenous values $\mathbf{x}_{T+1}, \ldots, \mathbf{x}_{T+h}$, but instead, only the copies of the encoding $E$ were used as input to the second RNN layer.

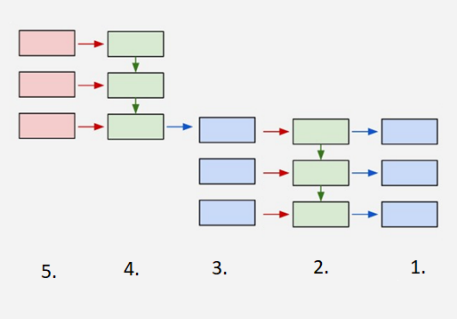



Figure 2: Schematic representation of the Encoder/Decoder with Exogenous Variables RNN structure, with input data before the forecast horizon (5), first RNN layer (4), concatenation of exogenous variables after the forecast horizon and the copies of the encoding (3), second RNN layer (2) and the generated forecasts (1).

# 6 Methodology

This section details our approach in modelling the water management system of the Noordoostpolder. We will describe the data handling process and specifications of the various models that we use, as well as the way we evaluate our results. Additionally, we provide some argumentation about choices that are made in the modelling process.

We begin by selecting a small subset of all variables to incorporate in the models. This is done to allow faster convergence of the models and to prevent spurious regressions. The variables we use are the measurement series shown in Table 1.

## 6.1 Exogenous Variables

The water levels are treated as endogenous variables, the flow rates and precipitation as exogenous variables. The reasoning for this is simple: we are interested in the behavior and dynamics of water levels in the Noordoostpolder. Precipitation is the source of water that enters the Noordoostpolder from the outside. Similarly, the flow rates measure the amount of water that is pumped out of the Noordoostpolder by the pumping stations. Therefore, using domain knowledge, we derive that these two types of measurements can be considered as exogenous, whereas the other time series are dependent upon each other and upon the exogenous variables and should thus be modelled as endogenous variables. Thus, using the notations introduced in earlier sections, we write $\mathbf{y} = \{$WH2, WH3, WH5, GW1, GW2$\}$ and $\mathbf{x} = \{$FR2000, FR2100, FR2200, MWSP$\}$.

To be able to generate conditional forecasts, we must include observations of the exogenous variables between the forecasting origin $T$ and the forecast horizon $T+h$, as described in Section 4.4. For both the precipitation and the flow rates, we assume that at time $T$ the observations between $T$ and $T + h$ are indeed available for forecast generation. From a forecasting perspective, this can be argued by observing that the future values of these exogenous variables are also not affected by the past water levels directly, but instead, purely by the weather or the water management actions.

In a practical, realistic forecasting scenario however, we do not have access to the observations of flow rates and precipitation in the next 24 hours. But we can resort to using their predicted values, quite easily, as follows. First of all, meteorological institutes generate accurate predictions of rainfall multiple days ahead, that are freely available. Secondly, after consultation with hydrologists at the Waterschap Zuiderzeeland, the following is assumed about the flow rates. Flow rate time series are completely dependent on water management actions of the Waterschap Zuiderzeeland. Each pump in the pumping stations can be turned off or on, which is done manually. Pumps create a constant efflux of water when turned on, that equals 0 only when all pumps are switched off. In addition, there is a clear protocol for when pumps should be on or off, as well as centralized coordination in case of an extreme scenario. In conclusion, very clear agreements can be made beforehand about when the pumping stations will be functioning, and from these agreements, the flow rate time series can be immediately and accurately determined.

These assumptions will allow us to generate the desired *conditional forecasts* based on future paths of the exogenous variables. Once the weather forecast is available and the flow rates series are determined, we can forecast the water levels. This is exactly what we are interested in: the behavior of the water management system under the influence of precipitation and management actions. This forecast could aid the decision making process, for example by contrasting the results of various future paths for either flow rates, precipitation, or both.

## 6.2 Feature Construction

We use lags of the endogenous features up to $p = 192$, which is equivalent to using the past two days of measurements. This was determined using domain knowledge of the hydrologists at the Waterschap Zuiderzeeland: they estimated that fluctuations generally persist for about 24 hours in the Noordoostpolder, and that the maximal time over which some fluctuation is still noticeable is 48 hours, equivalent to $p = 192$.

Additionally, we construct a set of features with lags and rolling means of the exogenous variables. These are used as input to all models except the Recurrent Neural Networks, since those models have their own approach to utilizing past exogenous variables. An extensive cross-correlation analysis was carried out to determine the most relevant lag order and rolling mean sizes. Flow rate features were constructed by lagging the variable over 4, 8, 12, 16, 23 and 32 time steps and taking the average of the last 4, 8, 20, 48, 56 and 68 observations. Precipitation features were constructed by lagging the variable over 4, 8, 12, 14, 16, 28, 31, 42 and 61 time steps and taking the average of the last 4, 12, 36, 48 and 96 observations.

## 6.3 Forecast Generation

We have seen in the data assessment of section 3.4 that it is difficult to derive properties of the stochastic process underlying the data. Specifically, we can not draw a single conclusion about stationarity and the cointegration relationship. We also argued that this is most likely caused by the way the data is generated: trends in the data are not stochastic by nature, but are the result of specific water management actions. From the domain experts, we know that the individual time series are strongly related nonetheless. The risk of discovering spurious correlations is therefore not high. Because of this, we will not evaluate the models and their predictions by analyzing their statistical properties, as is common practice in traditional time series analysis. Rather, we will rely heavily on out-of-sample forecasting performance, as is common practice in machine learning methods. Doing so will provide insight into model overfitting and forecast reliability, which is ultimately the goal of this project.

Therefore, we propose and use the following modelling and forecasting approach. We divide the data into two sets: a training set, consisting of all observations in 2015, 2016 and 2017, and the test set, consisting of all observations in 2018. The data in the training set is used to tune the model parameters: in the case of VAR models, for example, by simply performing a least-squares regression of 1-step ahead forecasts using all observations. The test set is then used to estimate the out-of-sample forecasting performance, or in other words, *forecasting performance on unseen data.*

To simulate a realistic forecasting scenario, we use a maximum forecasting horizon $h = 96$, equivalent to a 24-hour period. We also choose a forecast origin $T_i = 12:00:00$ (midday) for each day $i$ in out test set, 2018. Given an origin $T_i$, we generate forecasts for all time steps in the following 24-hour period, which means we use forecasting horizons $h' = 1, \ldots, 96$. For this, we assume that all past observations of endogenous variables $\mathbf{y}_t$, for all $t < T_i$ are known. Similarly, we assume that the values of the exogenous variables $\mathbf{x}_t$, for all $t < T_i + h$ are known, as described in Section 4.4 and Section 6.1.

We explicitly wish to generate a *conditional forecast* of $\mathbf{y}$, conditioned on the observations in $\mathbf{x}$ at times $t > T_i$. We use these known observations as input to the forecasting model, but we do not update the model parameters during forecasting. There are two ways in which we can generate such a conditional forecast for all time steps between $T_i$ and $T_i + 96$. We will describe these in the subsections below.

### 6.3.1 Recursive forecasting

For the recursive forecasting, we estimate the model that generates a prediction of the endogenous values one time step ahead. Using the trained model (with some lag order $p$), the observations $\mathbf{y}_{T-p}, \ldots, \mathbf{y}_T$, the exogenous features as in 6.2, and exogenous variables $\mathbf{x}_{T+1}$, we generate the conditional forecast $\hat{\mathbf{y}}_{T+1}$. Then, using $\mathbf{y}_{T-p+1}, \ldots, \mathbf{y}_T$, $\hat{\mathbf{y}}_{T+1}$, and exogenous variables $\mathbf{x}_{T+2}$, we generate a forecast $\hat{\mathbf{y}}_{T+2}$. Note that in forecasting multiple time steps ahead, the model is applied recursively: we use the previous forecasts and use them as input for the subsequent forecasts. In other words: we treat them as observations. We keep doing this until we have a forecast of $\mathbf{y}_{T+96}$. We then repeat the above procedure for a new $T_{i'} = T_{i+1}$, since we have arrived at the next midday in our data. This procedure results in a time series $\hat{\mathbf{y}}$ containing forecasts of every endogenous variable in the test set.

### 6.3.2 Direct Forecasting

Given an origin $T$, a direct forecasting model generates a forecast $\hat{\mathbf{y}}_{T+j}$ for a future time step $T + j$, without considering or predicting $\mathbf{y}_{T+k}$ for $1 \leq k < j$. This can be done, for example, by training a separate model for each forecasting horizon. Thus, using a trained model (with some lag order $p$), the observations $\mathbf{y}_{T-p}, \ldots, \mathbf{y}_T$, the exogenous features as in 6.2, and exogenous variables $\mathbf{x}_{T+1}, \ldots, \mathbf{x}_{T+j}$, we generate a forecast $\hat{\mathbf{y}}_{T+j}$. This means that in our scenario, a multi-step forecast for all time steps $T + 1$ to $T + 96$ requires 96 separate models.

Alternatively, we can approximate the set of direct forecasts by training a larger single model with multiple outputs. The goal is to get a forecast for all time steps between $T$ and $T+h$ with only one propagation of our model. In that case, the input to the trained model are the observations $\mathbf{y}_{T-p}, \ldots, \mathbf{y}_T$, the exogenous features as in 6.2, and exogenous variables $\mathbf{x}_{T+1}, \ldots, \mathbf{x}_{T+h}$. The model has $h$ different outputs, and each output $O_j$, for $1 \leq j \leq h$, provides one of the predictions $\hat{\mathbf{y}}_{T+j}$.

For our experiments, we decide to implement the latter approach. Firstly, because training and maintaining 96 different models is not feasible in real-life applications. Secondly, the water levels in a 24-hour period are heavily autocorrelated and can therefore probably be estimated using the same set of features in a single model.

One problem arises when we compare the recursive and direct forecasting methodology, which relates to the data that is used when producing a forecast $\hat{\mathbf{y}}_{T+j}$ for some $1 \leq j < 96$. The recursive models use exogenous variables at times $t \leq T + j$ for this; the direct models use all the exogenous variables at once, thus at times $t \leq T + 96$. This not only means that the direct models have access to additional information, it also means that these models can produce forecasts using exogenous variables $\mathbf{x}_{T+j+k}$ for $k > 0$. In other words, this direct forecasting methodology uses exogenous variables at time steps after the time step for which we are generating a forecast. Note that this procedure does not violate the restrictions of our conditional forecasting problem: we assume that all exogenous variables for the next 24 hours are made available for forecast generation in a practical scenario, and that thus, we should train and evaluate the models using this data. However, we do worry that this approach provides an advantage for the direct forecasting models. To assess the significance of this, we compare the forecasting performances at times $T_i + 96$, since for that time step, the direct and recursive models utilize exactly the same information. If the direct forecasting methodology does indeed provide an unfair advantage, this advantage disappears at times $T_i + 96$, resulting in a relatively worse performance of the direct models.

## 6.4 Performance

We choose to evaluate the predicted target variables in $\hat{\mathbf{y}}$ and their actual values in $\mathbf{y}$ by determining the squared correlation coefficient ($R^2$) and Root Mean Squared Error (RMSE). Note that we are evaluating forecasts of $\mathbf{y}$ conditioned on $\mathbf{x}$. While conditional forecasts can be used to compare the predictions for different hypothetical future paths of $\mathbf{x}$, this initial evaluation uses the true values for the observations $\mathbf{x}$. Also, note that we generate forecasts of all endogenous variables, thus for WH2, WH3, WH5, GW1 and GW2, but we evaluate only the predictions of the target variables WH2, WH3 and WH5, since these predictions provide the relevant insight for the Waterschap Zuiderzeeland.

Given the forecasting methodologies detailed in Section 6.3, we evaluate a series of forecasts for times $T_i + 1, \ldots, T_i + 96$ for every day $i$ in our test set. By reporting the average metrics over all $T_i + x$, for $1 \leq x \leq 96$, we provide an indication of the expected error when forecasting some randomly chosen time step in the next 24 hours. Additionally, we present and visually analyze plots of the actual and predicted values over time for certain days in the test set.

To conclude, we will use the Diebold-Mariano test [20] to determine if a model performs significantly better than another model. This significance test is applied to prediction errors of forecasts generated over a specific horizon, since a larger forecast horizon leads to more uncertainty. We must therefore filter the predictions and evaluate the test on predictions only for times $T_i + H$ for fixed values of $H$. The null hypothesis of the one-sided Diebold-Mariano test is that the two forecasts have the same accuracy at the specified horizon $H$, the alternative hypothesis is that either of the two models has a better performance. We use two one-sided significance tests to be able to draw a conclusion about which model performs better, since the two-sided version only states that the model performance is different without telling us which one is better. We use the significance level $\alpha = 0.05$, which is often used in forecasting literature.

## 6.5 VAR and VECM modelling

We decide to fit both VAR and VECM models to the data to be able to compare their performance and choose the best model. An iterative binary search procedure was used to evaluate and determine the best lag order $p$ for the model, up to the previously determined maximal value $p = 192$. Least squares regression is used to tune the parameters. By definition, the VAR and VECM models generate only *recursive forecasts*. Methods for estimating these models, as well as generating their predictions on new data, are provided in the R package 'tsDyn'.

## 6.6 Feedforward Neural Networks

We construct various neural network models to generate a conditional forecast of the water levels. Each input node corresponds to a specific variable at a specific point in time, relative to the forecasting origin $T$. For endogenous variables, these are times $T - i$ for $0 \leq i \leq 192$. For exogenous variables, these correspond to one of the exogenous features as in 6.2, as well as the values at times $T + j$ for $1 \leq j \leq h$.

The neural network models can be trained for both recursive forecasting ($h = 1$) and direct forecasting ($h = 96$). In the case of recursive forecasting, the network has only 5 output nodes: 1 for each target variable in $\hat{\mathbf{y}}_{t+1}$. In the case of direct forecasting, the network has $5 \times 96$ output nodes: each set of 5 output nodes corresponds to some $\hat{\mathbf{y}}_{t+j}$ for $1 \leq j \leq 96$.

The time series data that is used must be transformed accordingly. As each node (input or output) uses the values in a specific column of a data matrix, we copy and shift the time series many times such that each column corresponds to a specific lag or lead of one of the variables.

We experiment with neural networks containing one or two hidden layers. For recursive forecasting, the maximum size for a hidden layer is 48 nodes, as we suspect that more complex networks are not required to model the 1-step ahead forecasts and might be prone to overfitting. For direct forecasting, the maximum size is 700 nodes per hidden layer. Additionally, we use the rectified linear unit (ReLU) activation function for hidden nodes, and no activation function for output nodes, since these are used most often for predicting continuous values. All methods for defining and training these neural networks are implemented in the 'Keras' python package, which can be controlled from the R environment using the R package 'Keras' [27].

Keras splits the training data by default: the last 20% of observations is used as validation set, the rest is used as actual training set. During training, the model performance is evaluated on the validation set after each iteration. This validation loss is used to define an early stopping criterion. Model parameters are saved during training when a minimal validation loss is determined: if, for a fixed number of iterations, this validation loss does not reach a new minimum, the previously saved model is returned. This procedure prevents overfitting on the training data. As a result, however, the NN model parameters are tuned using a smaller dataset compared to the VAR/VECM models. Nonetheless, comparing the forecasts generated by the different models is fair: while the validation set is not used to tune the model parameters, the prevention of overfitting is an equally useful utilization of the data. Therefore, each model architectures utilizes all observations in the training set for fitting the most optimal model, albeit in a different manner.

To be able to efficiently reference the NN model types, we introduce the following notation. 1-$\text{NN}_{i,j}(p)$ is used to describe a neural network generating 1-step ahead predictions, with two hidden layers of sizes $i$ and $j$, and lag order $p$ for the endogenous variables. Neural networks with only 1 hidden layer have no parameter $j$. D-$\text{NN}_{i,j}(p)$ is used to describe a neural network generating predictions for all time steps $T+1, \ldots, T+96$, that can thus be used for direct forecasting, with the same parameters.

Some hyperparameters for the ANNs were chosen using a trial-and-error procedure. The batch size was determined beforehand to be 32, which is a standard value for many ANN applications. We concluded from early plots of the validation loss that a suitable value for the maximum number of iterations is 200, and that a patience parameter of 30 iterations is an effective choice for the early stopping criterion.

## 6.7 Recurrent Neural Networks

We implement and evaluate each of the 3 methodologies described in Section 5.4. We reference these RNN architectures as follows. Firstly, models are referenced by the types of neurons in the layers, namely, "LSTM" or "GRU". 1-$\text{LSTM}_{i,j}$ indicates an LSTM model that is trained for 1-ahead predictions with two layers of size $i$ and $j$. XY-$\text{LSTM}_{i,j}$ is used to reference an LSTM model that implements the second approach described: predicting $\mathbf{y}$ using mainly $\mathbf{x}$ and some extra information on past values of $\mathbf{y}$ to initialize memory. If either of the two models above has only one layer, the variable $j$ is omitted. Lastly, ED-$\text{LSTM}_{i,j}$ is used to reference an LSTM model that implements the encoder/decoder architecture with exogenous variables, which always requires two RNN layers.

All methods for defining and training these RNNs are implemented in the 'Keras' python pack-

age, which can be controlled from the R environment using the R package with the same name [27]. The input and output data must be transformed accordingly: instead of filling columns with lags and leads of exogenous variables, we now transform our dataframes into 3D arrays that can be used with keras. The first dimension references an input sequence, the second dimension references the time step in the sequence, and the third dimension references the variable column. The number of time steps per sample, and the columns selected for input and output vary per model architecture.

Similar to the feedforward ANNs, besides the batch size of 32, we have parameters for the maximum number of iterations and the patience (early stopping). We use the same procedure as before: per architecture, we try different values, we evaluate and choose based on the performance on the test set. By looking at the plots of the loss per iteration on the validation set, we decide that the correct parameter values are 40 maximum iterations with a patience parameter of 20 for the early stopping criterion.

$\mathbf{O}_t = f(\mathbf{I}_t, \mathbf{H}_{t-1})$

## 6.8 Conditional Forecasting

As we have seen above, we explicitly wish to generate conditional forecasts of the water heights in $\mathbf{y}$. This forecast is conditioned on the flow rates and precipitation in $\mathbf{x}$ at time steps between the forecasting origin and forecasting horizon. Once the quality of the generated forecasts has been determined by using the true values for the observations $\mathbf{x}$, as described in 6.4, we wish to get an idea of how well the models can be used to compare different scenario's. For this, we select two exemplary dates from the testing data set. On these dates, we compare the forecast conditioned on the actual values of $\mathbf{x}$ with a forecast conditioned on an alternative, hypothetical future path of $\mathbf{x}$.

We choose these dates and future paths such that the difference between the two predictions can be explained using domain knowledge and experience. The first scenario is a very common day, for which we use an alternative path of $\mathbf{x}$ that has decreased flow rates compared to the actual data. The second scenario contains an alternative path of $\mathbf{x}$ that has a substantial increase in the amount of precipitation. We can not compare the alternative forecasts with some true value, but instead, we plot the original and alternative forecasts in the same figure to get an idea of the difference and the possibility to use these models to guide the decision making process.

# 7 Results

This section contains the results for the various model architectures, that were tuned using the training set and evaluated on the test set. We report the performance only for the target variables, and not for the other forecasted endogenous variables. To be able to place the reported performance metrics in perspective, we first generated a baseline performance. This naive forecast is defined as follows: for each forecasting horizon $T_i$ = 12:00:00 (midday) for every day $i$, we predict that the water levels in the next 24 hours are equal to the average water level of the past 24 hours. In Table 3, we show this baseline performance, expressed as RMSE and $R^2$.

| Model | WH2 | | WH3 | | WH5 | |
|---|---|---|---|---|---|---|
| | RMSE | $R^2$ | RMSE | $R^2$ | RMSE | $R^2$ |
| Baseline | 4.94 | 0.45 | 5.17 | 0.78 | 5.45 | 0.37 |

Table 3: Forecasting accuracy expressed as RMSE and $R^2$ for the baseline predictor evaluated on the test set.

## 7.1 VAR Models

Table 4 contains the RMSE and $R^2$ of various VAR($p$) and VECM($p$) models. We observe that increasing the lag order $p$, and thereby increasing the model complexity, leads to a better performance. We also observe that even with the simplest VAR model, we are able to greatly improve the naive baseline performance. The fact that the stationarity analysis in Section 3.4 was inconclusive does not appear to be a problem: the performance of the VAR models, which are in theory only applicable to stationary data, and the performance of the VECM models, which are especially designed for non-stationary data, is comparable.

We performed the Diebold-Mariano test on the baseline predictions and the predictions of the VECM(192) model. The results are in Table 5. We conclude that the VECM(192) model predictions for every target series are significantly better at horizons up to $h = 48$, and comparable for $h = 96$. Therefore, the VAR/VECM model performance proves to be a reasonable baseline for model comparison.

Additionally, we visually inspect the predicitions generated by the linear models. In Figure 3 we show 4 graphs containing water levels in the 2018 test set, along with the predictions generated by the VECM(192) model. The accumulation of errors for larger forecasting horizons can be clearly identified, especially in situations with moderate or heavy rainfall.

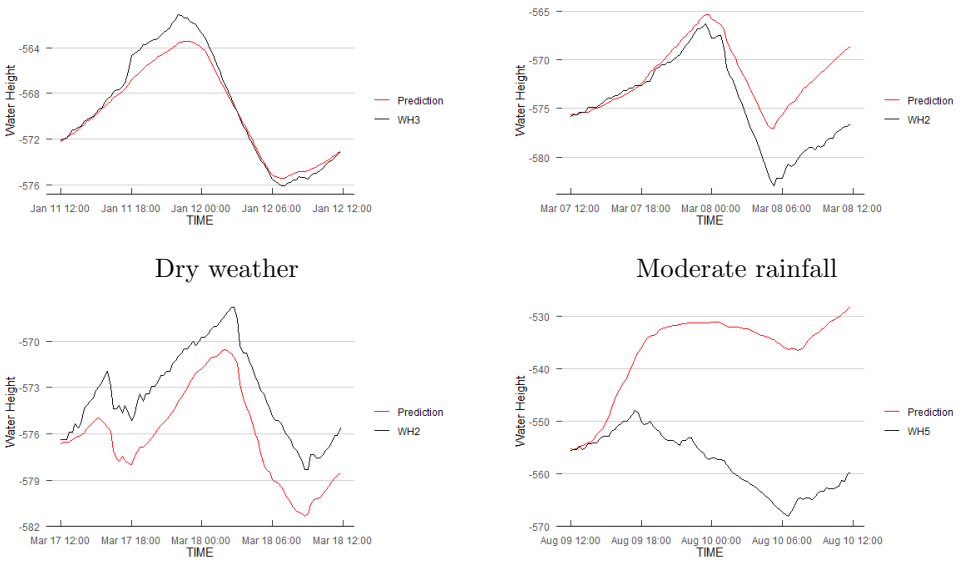
We are also interested in feature importance of both endogenous and exogenous variables. In Table 6, we evaluate this for the VECM(192) model, and for each of the 3 target variables. Rows are indexed by a variable name in the first column: that variable is excluded from the model during training and testing to study the effect on the forecasting performance. This performance is expressed as the ratio of RMSE to the RMSE of the original VECM(192) model.

| Model | WH2 | | WH3 | | WH5 | |
|---|---|---|---|---|---|---|
| | RMSE | $R^2$ | RMSE | $R^2$ | RMSE | $R^2$ |
| VAR(8) | 3.36 | 0.84 | 3.65 | 0.9 | 3.5 | 0.77 |
| VECM(8) | 3.23 | 0.83 | 3.51 | 0.91 | 3.35 | 0.78 |
| VECM(24) | 3.15 | 0.84 | 3.35 | **0.92** | 3.33 | 0.78 |
| VECM(48) | 2.85 | 0.86 | 3.26 | **0.92** | 3.33 | 0.79 |
| VECM(96) | 2.66 | 0.86 | 3.28 | 0.91 | 3.3 | 0.78 |
| VAR(192) | **2.50** | **0.87** | 3.22 | **0.92** | **3.13** | 0.78 |
| VECM(192) | 2.6 | **0.87** | **3.18** | **0.92** | 3.15 | **0.80** |

Table 4: Forecasting accuracy expressed as RMSE and $R^2$ for various VAR($p$) and VECM($p$) models evaluated on the test set. In bold: for each target variable, its highest quality prediction with respect to RMSE or $R^2$.

| Models | | H | **Diebold-Mariano** | | |
|---|---|---|---|---|---|
| | | | WH2 | WH3 | WH5 |
| Baseline | VECM(192) | | | | |
| | | 1 | 0.0000 | 0.0000 | 0.0000 |
| | | 8 | 0.0000 | 0.0006 | 0.0024 |
| | | 24 | 0.0000 | 0.0023 | 0.0069 |
| | | 48 | 0.0000 | 0.0068 | 0.0132 |
| | | 96 | 0.3417 | 0.1183 | 0.3880 |
| VECM(192) | Baseline | | | | |
| | | 1 | 1.0000 | 1.0000 | 1.0000 |
| | | 8 | 1.0000 | 0.9994 | 0.9976 |
| | | 24 | 1.0000 | 0.9977 | 0.9931 |
| | | 48 | 1.0000 | 0.9932 | 0.9868 |
| | | 96 | 0.6583 | 0.8817 | 0.6120 |

Table 5: P-values of the Diebold-Mariano test applied to predictions at specific horizons H of the naive baseline model and the VECM(192) model. The null hypothesis is that the performance is the same, the alternative hypothesis is that the second (rightmost) model has a better performance.




Dry weather | Moderate rainfall

Dry, after rainfall | Heavy rainfall

Figure 3: Exemplary selection of WH2, WH3 and WH5 water levels, and it's predictions generated by VECM(192). Captions indicate the weather conditions on the day for which the forecast was generated.

| Excluded Variables | WH2 | WH3 | WH5 |
|---|---|---|---|
| ∅ | 1 | 1 | 1 |
| WH2 | . | 1.040 | 1.099 |
| WH3 | **0.988** | . | **0.990** |
| WH5 | 1.068 | 1.041 | . |
| GW1 | **0.963** | **0.995** | 1.006 |
| GW2 | 1.001 | 1.010 | 1.047 |
| FR2000 | 1.383 | 1.129 | 1.247 |
| FR2200 | 1.465 | 1.100 | 1.269 |
| FR2100 | 1.018 | **0.999** | 1.010 |
| MWSP | 1.083 | 1.095 | 1.016 |

Table 6: Ratio of VECM(192) forecast RMSE to original RMSE when the variable indicated in the first column is excluded from the model. In bold: forecast RMSE ratio lower than 1, indicating forecast quality improved when removing the indicated variable.

## 7.2 Feedforward Neural Networks

Next, we evaluate the feedforward ANN architectures for recursive forecasting. These ANN models are non-linear and contain many more parameters than the VAR models: the model complexity is therefore much higher. We begin by determining the out-of-sample forecasting RMSE and $R^2$ again, to get an indication of the performance of the models. The first experiment we conduct relates to neural network size. We trained networks of various sizes with lag order $p = 96$ to generate 1-ahead predictions. We then generated recursive forecasts for the entire testing data set. The results are shown in Table 7.

We are also interested in overfitting of the neural network models. For this, the training and validation loss is determined after every iteration of training. We show these in Figure 4 for the 1-ahead models. In 3 out of 4 plots, we clearly see that the validation loss often approaches the training loss, and is sometimes even lower. Thus except for the incredibly simple 1-$NN_4(96)$ model, we conclude that the recursive forecasting ANNs do not suffer substantially from overfitting.

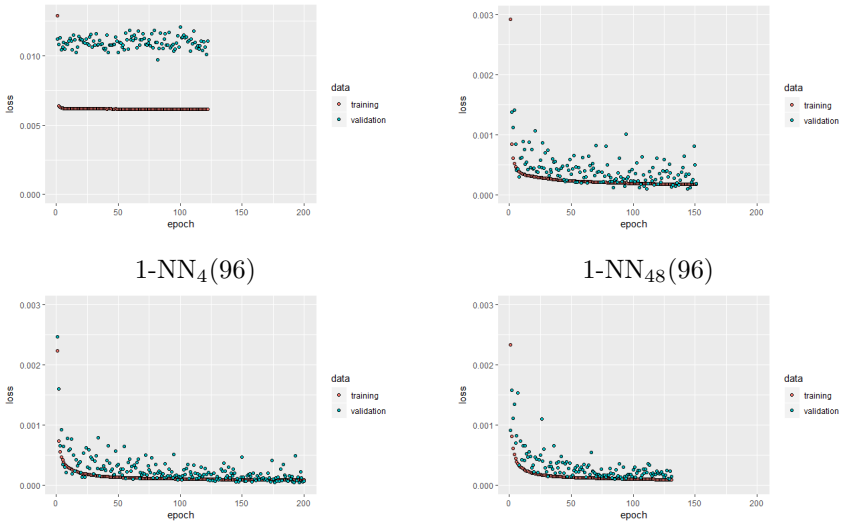
The effect of the lag order $p$ on the neural networks performance was also examined. We trained networks of size [48, 16] with varying lag orders $p$, to generate 1-ahead predictions. We then generated recursive forecasts for the entire testing data set. The results are shown in Table 8.

We observe that generating the recursive forecasts using ANNs yields bad results: it hardly improves the baseline model and appears worse than the linear models. To test the latter statement, we performed the Diebold-Mariano on the predictions of the VECM(192) model and the 1-$NN_{48,16}(96)$ model. The results are in Table 9. We conclude that the VECM(192) model predictions are significantly better at every horizon for WH3 and WH5, and either comparable or significantly better for WH2. This can likely be attributed to the accumulation of errors when forecasting multiple time steps ahead: for NN models, these negative effects are more extreme compared to the VAR/VECM models, due to the increased model complexity.

Finally, we visually inspect the predictions generated by the recursive forecasting neural networks. Figure 5 contains 4 graphics, for the same dates and target variables as before, showing he actual data and the predictions generated by the $NN_{48,16}(96)$ model.

| Model | WH2 | | WH3 | | WH5 | |
|---|---|---|---|---|---|---|
| | RMSE | $R^2$ | RMSE | $R^2$ | RMSE | $R^2$ |
| 1-$NN_4(96)$ | 6.51 | 0.00 | 11.03 | 0.00 | 7.41 | 0.00 |
| 1-$NN_8(96)$ | 16.29 | 0.01 | 24.19 | 0.03 | 16.44 | 0.01 |
| 1-$NN_{24}(96)$ | 7.22 | 0.36 | 16.17 | 0.49 | 8.77 | 0.24 |
| 1-$NN_{48}(96)$ | 4.23 | 0.71 | 6.10 | 0.78 | 5.86 | 0.58 |
| 1-$NN_{48,8}(96)$ | 16.71 | 0.09 | 19.21 | 0.32 | 19.50 | 0.09 |
| 1-$NN_{48,16}(96)$ | **3.28** | **0.78** | **4.73** | **0.87** | **4.59** | **0.70** |
| 1-$NN_{48,48}(96)$ | 4.18 | 0.64 | 5.41 | 0.82 | 5.20 | 0.57 |

Table 7: Forecasting performance expressed as RMSE and $R^2$ for various 1-ahead neural network models with lag order $p = 96$. Performance was evaluated by generating recursive forecasts on the test set. In bold: for each target variable, it's highest quality prediction with respect to either RMSE or $R^2$.




1-NN$_4$(96) 1-NN$_{48}$(96)

1-NN$_{48,16}$(96) 1-NN$_{48,48}$(96)

Figure 4: MSE measured after every training iteration for the training set and validation set, for various neural network architectures that generate 1-ahead predictions.

| Model | WH2 | | WH3 | | WH5 | |
|---|---|---|---|---|---|---|
| | RMSE | $R^2$ | RMSE | $R^2$ | RMSE | $R^2$ |
| 1-NN$_{48,16}$(8) | 6.24 | 0.72 | 7.13 | 0.69 | 7.61 | 0.60 |
| 1-NN$_{48,16}$(24) | 6.74 | 0.50 | 11.50 | 0.78 | 6.68 | 0.47 |
| 1-NN$_{48,16}$(48) | 3.73 | 0.74 | 11.36 | 0.78 | **3.60** | **0.71** |
| 1-NN$_{48,16}$(96) | **3.28** | **0.78** | **4.73** | **0.87** | 4.59 | 0.70 |
| 1-NN$_{48,16}$(192) | 4.61 | 0.65 | 7.55 | 0.71 | 6.91 | 0.58 |

Table 8: Forecasting performance expressed as RMSE and $R^2$ for 1-ahead neural network models of size [48, 16] with varying lag orders $p$. Performance was evaluated by generating recursive forecasts on the test set. In bold: for each target variable, it's highest quality prediction with respect to either RMSE or $R^2$.

| Models | | H | Diebold-Mariano WH2 | WH3 | WH5 |
|---|---|---|---|---|---|
| VECM(192) | 1-$NN_{48,16}$(96) | | | | |
| | | 1 | 1.0000 | 0.9984 | 1.0000 |
| | | 8 | 0.9399 | 0.9972 | 1.0000 |
| | | 24 | 0.8470 | 0.9995 | 0.9999 |
| | | 48 | 1.0000 | 1.0000 | 1.0000 |
| | | 96 | 0.9957 | 1.0000 | 0.9950 |
| 1-$NN_{48,16}$(96) | VECM(192) | | | | |
| | | 1 | 0.0000 | 0.0016 | 0.0000 |
| | | 8 | 0.0601 | 0.0028 | 0.0000 |
| | | 24 | 0.1530 | 0.0005 | 0.0001 |
| | | 48 | 0.0000 | 0.0000 | 0.0000 |
| | | 96 | 0.0043 | 0.0000 | 0.0050 |

Table 9: P-values of the Diebold-Mariano test applied to predictions at specific horizons H of the VECM(192) model and the 1-$NN_{48,16}$(96) model. The null hypothesis is that the performance is the same, the alternative hypothesis is that the second (rightmost) model has a better performance.

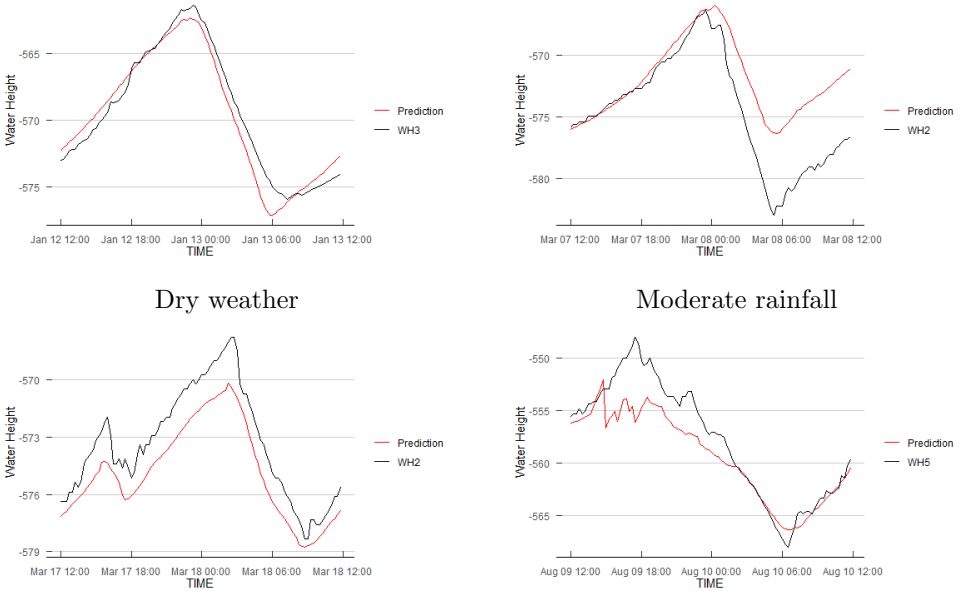



Figure 5: Exemplary selection of WH2, WH3 and WH5 water levels, and the recursive forecasts generated by the $1\text{NN}_{48,16}(96)$ model. Captions indicate the weather conditions on the day for which the forecast was generated.

Let us then consider the ANN for direct forecasting. Again, we first determine the out-of-sample forecasting RMSE and $R^2$ for different model sizes, to get an indication of their performance. We trained networks of various sizes with lag order $p = 96$ using all observations of endogenous variables in the next 24 hours as target variables. Thus, we can generate direct forecasts using these networks, as described in Section 6.3.2. The results of these direct forecasts on the test set are shown in Table 10.

We are interested in the effect of lag order $p$ on the direct forecasting performance. We trained neural networks with 1 hidden layer of size 96 with varying lag order $p$ and evaluated it's performance. The results are shown in Table 11.

To be able to assess overfitting of the direct forecasting models, in Figure 6 we show the history of training and validation loss for various networks. Overfitting occurs for every direct forecasting model, observed as a persistent higher validation loss than training loss. We see that the more complex the model, the more prone it is to overfitting. Early stopping is therefore beneficial, especially for the more complex models.

Additionally, we show in Figure 7 the visualization of the direct forecasts generated by the $\text{D-NN}_{96}(96)$ model. We observe that the minimal and maximal water levels are properly predicted by the model, but the timing and steepness of the fluctuations is not very accurate. The performance metrics and visualizations for the D-NN model were presented to hydrologists at Waterschap Zuiderzeeland: they indicated that the forecasts are accurate enough to be used for decision support if they are generated for up to 24 hours ahead.

The Diebold-Mariano test was performed on the predictions of the VECM(192) model and the D-$NN_{96}(96)$ model. The results are in Table 12. We conclude that the D-$NN_{96}(96)$ forecasts are significantly worse for short horizons ($h = 1, 8, 24$) and comparable to the VECM(192) forecasts for longer horizons ($h = 24, 48, 96$). We observe that the D-$NN_{96}(96)$ model has a slightly better performance for WH3 and WH5 at longer horizons, but this improvement is not significant at the chosen level.

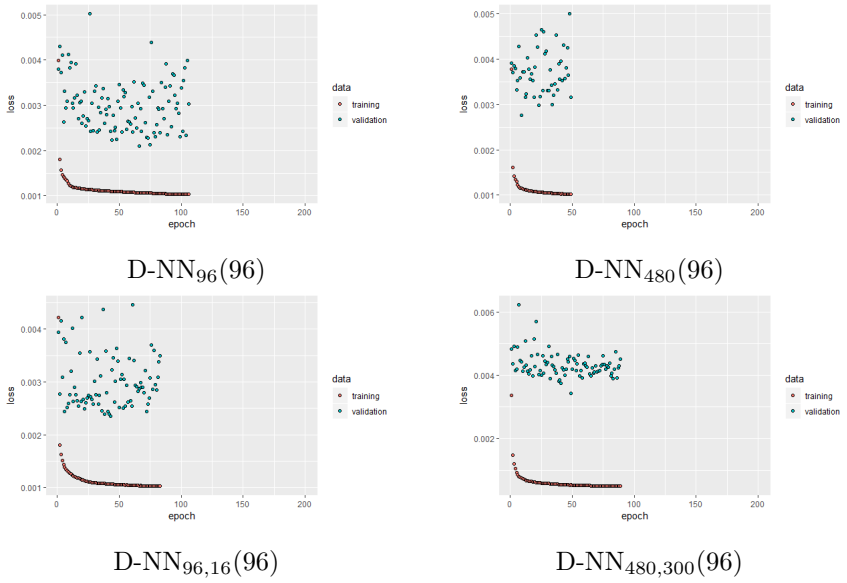
In an attempt to reduce overfitting of the models on the training data, we conducted an experiment with a regularization technique. We are interested in the effect of applying *dropout* on the forecasting performance of the larger networks. We trained neural networks of two sizes: [480] and [480, 300], in which all hidden layers were subjected to the dropout technique. We then evaluated their direct forecasting performance on the test set, and visualized the validation loss of the models: the results are in Table 13, and the history plots are in Figure 8. We conclude that the application of the dropout technique does not yield better results.

| Model | WH2 | | WH3 | | WH5 | |
|---|---|---|---|---|---|---|
| | RMSE | $R^2$ | RMSE | $R^2$ | RMSE | $R^2$ |
| D-$NN_8(96)$ | 3.81 | 0.68 | 4.35 | 0.85 | 4.39 | 0.66 |
| D-$NN_{24}(96)$ | 2.37 | **0.89** | 3.20 | 0.92 | 2.76 | 0.84 |
| D-$NN_{48}(96)$ | **2.35** | **0.89** | 3.15 | **0.92** | 2.85 | 0.84 |
| D-$NN_{96}(96)$ | 2.68 | **0.89** | **3.11** | **0.92** | **2.49** | **0.86** |
| D-$NN_{200}(96)$ | 3.05 | 0.84 | 3.45 | 0.91 | 3.64 | 0.81 |
| D-$NN_{300}(96)$ | 4.20 | 0.83 | 4.22 | 0.89 | 3.72 | 0.80 |
| D-$NN_{480}(96)$ | 3.45 | 0.85 | 4.97 | 0.80 | 4.06 | 0.77 |
| D-$NN_{48,16}(96)$ | 2.63 | 0.86 | 4.23 | 0.91 | 2.76 | 0.85 |
| D-$NN_{48,48}(96)$ | 3.03 | 0.85 | 3.94 | **0.92** | 3.35 | 0.82 |
| D-$NN_{96,16}(96)$ | 2.46 | 0.88 | 4.16 | 0.90 | 2.57 | 0.85 |
| D-$NN_{96,48}(96)$ | 2.98 | 0.82 | 3.46 | 0.91 | 3.39 | 0.80 |
| D-$NN_{96,96}(96)$ | 3.75 | 0.80 | 4.46 | 0.89 | 3.20 | 0.81 |
| D-$NN_{200,96}(96)$ | 3.56 | 0.82 | 3.72 | 0.90 | 3.65 | 0.79 |
| D-$NN_{200,200}(96)$ | 3.72 | 0.81 | 4.18 | 0.87 | 3.48 | 0.79 |
| D-$NN_{300,200}(96)$ | 3.23 | 0.77 | 4.10 | 0.88 | 3.61 | 0.78 |
| D-$NN_{300,300}(96)$ | 3.47 | 0.82 | 4.34 | 0.88 | 3.47 | 0.77 |
| D-$NN_{480,300}(96)$ | 3.57 | 0.80 | 4.09 | 0.87 | 3.58 | 0.76 |
| D-$NN_{480,480}(96)$ | 4.26 | 0.80 | 4.39 | 0.88 | 4.15 | 0.76 |

Table 10: Forecasting performance expressed as RMSE and $R^2$ for neural network models with lag order $p = 96$. Performance was evaluated by generating direct forecasts on the test set. In bold: for each target variable, it's highest quality prediction with respect to either RMSE or $R^2$.

| Model | WH2 | | WH3 | | WH5 | |
|---|---|---|---|---|---|---|
| | RMSE | $R^2$ | RMSE | $R^2$ | RMSE | $R^2$ |
| D-NN$_{96}$(8) | 3.12 | 0.84 | 3.25 | 0.91 | 3.34 | 0.81 |
| D-NN$_{96}$(24) | 3.06 | 0.85 | 3.61 | 0.89 | 2.99 | 0.82 |
| D-NN$_{96}$(48) | 3.39 | 0.85 | 4.49 | 0.85 | 3.87 | 0.81 |
| D-NN$_{96}$(96) | 2.68 | **0.89** | **3.11** | **0.92** | **2.49** | **0.86** |
| D-NN$_{96}$(192) | **2.24** | 0.88 | 4.36 | 0.91 | 3.45 | 0.85 |

Table 11: Forecasting performance expressed as RMSE and $R^2$ for neural network models with varying lag orders $p$. Performance was evaluated by generating direct forecasts on the test set. In bold: for each target variable, it's highest quality prediction with respect to either RMSE or $R^2$.




D-NN$_{96}$(96) D-NN$_{480}$(96)

D-NN$_{96,16}$(96) D-NN$_{480,300}$(96)

Figure 6: MSE measured on the training set and validation set after every training iteration, for various neural network architectures.

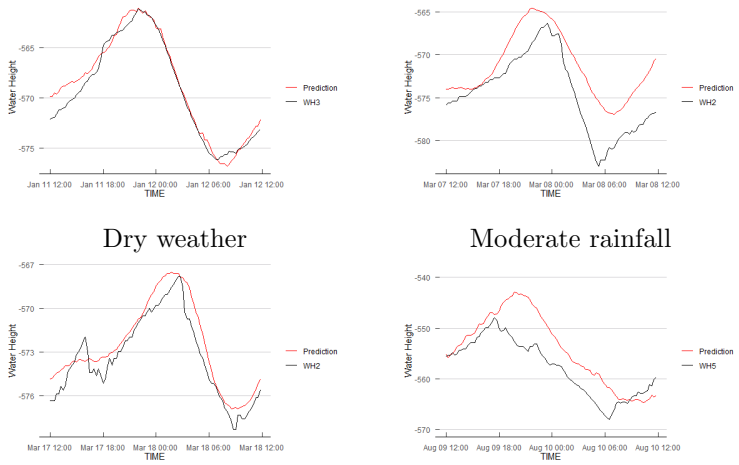


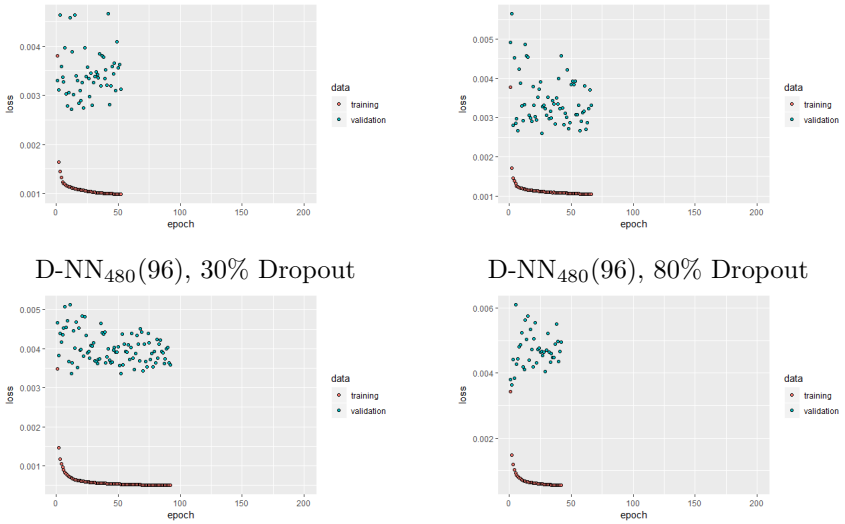

Figure 7: Exemplary selection of WH2, WH3 and WH5 water levels, and the direct forecasts generated by the D-NN$_{96}$(96) model. Captions indicate the weather conditions on the day for which the forecast was generated.

| Models | | H | **Diebold-Mariano** WH2 | WH3 | WH5 |
|---|---|---|---|---|---|
| VECM(192) | D-NN$_{96}$(96) | | | | |
| | | 1 | 1.0000 | 1.0000 | 1.0000 |
| | | 8 | 0.9999 | 1.0000 | 1.0000 |
| | | 24 | 0.9987 | 0.9942 | 0.4341 |
| | | 48 | 0.4596 | 0.0875 | 0.0826 |
| | | 96 | 0.4484 | 0.1384 | 0.0582 |
| D-NN$_{96}$(96) | VECM(192) | | | | |
| | | 1 | 0.0000 | 0.0000 | 0.0000 |
| | | 8 | 0.0001 | 0.0000 | 0.0000 |
| | | 24 | 0.0013 | 0.0058 | 0.5659 |
| | | 48 | 0.5404 | 0.9125 | 0.9174 |
| | | 96 | 0.5516 | 0.8616 | 0.9418 |

Table 12: P-values of the Diebold-Mariano test applied to predictions at specific horizons H of the VECM(192) model and the D-NN$_{96}$(96) model. The null hypothesis is that the performance is the same, the alternative hypothesis is that the second (rightmost) model has a better performance.

| Model | Dropout % | WH2 | | WH3 | | WH5 | |
|---|---|---|---|---|---|---|---|
| | | RMSE | $R^2$ | RMSE | $R^2$ | RMSE | $R^2$ |
| D-$NN_{480}(96)$ | None | 3.45 | 0.85 | 4.97 | 0.80 | 4.06 | 0.77 |
| D-$NN_{480}(96)$ | 30% | 3.37 | 0.84 | 3.61 | 0.90 | 3.47 | 0.83 |
| D-$NN_{480}(96)$ | 50% | 4.17 | 0.78 | 5.86 | 0.72 | 4.05 | 0.74 |
| D-$NN_{480}(96)$ | 60% | 3.19 | 0.86 | 3.56 | 0.90 | 3.12 | 0.83 |
| D-$NN_{480}(96)$ | 70% | 3.34 | 0.85 | 9.06 | 0.41 | 3.35 | 0.81 |
| D-$NN_{480}(96)$ | 80% | 2.81 | 0.84 | 3.79 | 0.90 | 3.80 | 0.82 |
| D-$NN_{480}(96)$ | 90% | 3.37 | 0.82 | 6.38 | 0.67 | 3.38 | 0.81 |
| | | | | | | | |
| D-$NN_{480,300}(96)$ | None | 3.57 | 0.80 | 4.09 | 0.87 | 3.58 | 0.76 |
| D-$NN_{480,300}(96)$ | 30% | 3.76 | 0.76 | 3.69 | 0.89 | 3.64 | 0.75 |
| D-$NN_{480,300}(96)$ | 50% | 3.93 | 0.72 | 3.94 | 0.88 | 3.45 | 0.73 |
| D-$NN_{480,300}(96)$ | 60% | 3.58 | 0.79 | 3.90 | 0.88 | 3.31 | 0.78 |
| D-$NN_{480,300}(96)$ | 70% | 3.46 | 0.78 | 4.21 | 0.87 | 3.58 | 0.76 |
| D-$NN_{480,300}(96)$ | 80% | 4.00 | 0.80 | 4.70 | 0.88 | 3.67 | 0.76 |
| D-$NN_{480,300}(96)$ | 90% | 4.00 | 0.75 | 4.05 | 0.87 | 3.68 | 0.74 |

Table 13: Forecasting performance for neural networks subjected to the dropout technique. Performance was evaluated by generating direct forecasts for the test set.




D-NN$_{480}$(96), 30% Dropout D-NN$_{480}$(96), 80% Dropout

D-NN$_{480,300}$(96), 30% Dropout D-NN$_{480,300}$(96), 80% Dropout

Figure 8: MSE measured on the training set and validation set after every training iteration, for the two neural network architectures subjected to the dropout technique.

### 7.3 Recurrent Neural Networks

We begin by measuring RMSE and $R^2$ of the forecasts generated by each of the 3 RNN architectures. For each architecture, we generated models with various sizes. We also experimented with LSTM neurons and GRU neurons in models of the first architecture with the same size. Conditional forecasts were generated for the data in the test set using the different methods described in Section 5.4. The results are in Tables 14, 15 and 22.

We observe that recursive forecasting using RNNs (the 1-RNN models) yield even worse results than the recursive ANN models. A possible reason for this is that the error accumulation occurs both in the hidden states and in the input data for recursive forecasting RNNs. It also becomes clear that models with GRU neurons consistently yield better results than LSTM neurons.

Additionally, we see that the results of the XY-RNN are slightly worse than the D-NN results on average. Moreover, the XY-RNN models are unable to yield better results than the VAR/VECM models. The results of the Diebold-Mariano test applied the to predictions of the VECM(192) model and the XY-GRU$_{25}$ model are shown in Table 16. We see that for $h = 1, 8, 24$ the VECM(192) is significantly better, for $h = 48$ the performance is comparable, and only for the very long horizon $h = 96$ the XY-GRU$_{25}$ is able to improve the performance. This can probably be explained by looking at the forecasting methods use by the models: the VECM(192) can use the most recent observations as input for short horizons, whereas the performance of the XY-RNN suffers from the large time-shifts in the input data. On the other hand, the XY-RNN does not suffer from an accumulation of error in the input data, resulting in a better performance as $h$ increases.

We also investigate the difference in performance between the ED-RNN architecture and the VAR/VECM models, as well as the D-NN models. The corresponding Diebold-Mariano tests are shown in Table 18. The ED-RNN does not significantly outperform the VECM(192) and D-NN

models. If we compare its performance with the VECM(192), we see that the ED-RNN performs significantly worse for short forecasting horizons but improves the performance at longer forecasting horizons. We also see that the ED-RNN is significantly better than the D-NN model at short forecasting horizons for some target series, and comparable for almost all other horizons. We thus conclude that, similar to all other alternatives considered, the ED-RNN architecture is unable to improve the performance of the VECM(192) model at very short horizons, and that for longer forecasting horizons, the performance is comparable to the D-NN.

To determine if the RNN models overfit during training, we studied the RMSE on the training set and validation set after each iteration. A selection of visualizations of these series are shown in Figure 9. We conclude that the models trained for forecasting multiple time steps into the future suffer from overfitting, and that therefore early stopping of training is beneficial.

Lastly, we are again interested in the shape of the prediction series when compared with the actual series. Figure 10 contains visualizations of the same dates and target series as before, with each plot showing both the actual values and the forecasts generated by the model. From this, we conclude that the timing and steepness of fluctuations are more accurately predicted by the ED-RNN model compared to the D-NN, except for situations with heavy rainfall.

| Model | WH2 | | WH3 | | WH5 | |
|---|---|---|---|---|---|---|
| | RMSE | $R^2$ | RMSE | $R^2$ | RMSE | $R^2$ |
| 1-$LSTM_{25}$ | 7.00 | 0.48 | 20.84 | 0.55 | 6.65 | 0.48 |
| 1-$GRU_{25}$ | 6.47 | 0.57 | **7.32** | **0.72** | 5.01 | 0.64 |
| 1-$LSTM_{50}$ | 12.66 | 0.00 | 24.83 | 0.35 | 17.21 | 0.00 |
| 1-$GRU_{50}$ | **4.04** | **0.70** | 9.90 | 0.41 | **3.97** | **0.69** |
| 1-$GRU_{100}$ | 5.83 | 0.40 | 11.09 | 0.64 | 7.22 | 0.35 |
| 1-$GRU_{50,16}$ | 6.83 | 0.61 | 8.71 | 0.54 | 6.87 | 0.50 |
| 1-$GRU_{50,50}$ | 5.49 | 0.60 | 8.79 | 0.56 | 5.75 | 0.39 |
| 1-$GRU_{100,50}$ | 5.15 | 0.69 | 8.96 | 0.67 | 6.20 | 0.52 |

Table 14: Forecasting performance expressed as RMSE and $R^2$ of RNN models for 1-ahead predictions. Performance was evaluated by generating recursive forecasts on the test set. In bold: for each target variable, it's highest quality prediction with respect to either RMSE or $R^2$.

| Model | WH2 | | WH3 | | WH5 | |
|---|---|---|---|---|---|---|
| | RMSE | $R^2$ | RMSE | $R^2$ | RMSE | $R^2$ |
| XY-$GRU_{16}$ | 3.40 | **0.81** | 3.79 | 0.89 | 3.56 | 0.71 |
| XY-$GRU_{25}$ | 3.35 | **0.81** | **3.47** | **0.91** | **2.83** | **0.81** |
| XY-$GRU_{50}$ | 4.07 | 0.78 | 4.43 | 0.86 | 3.71 | 0.76 |
| XY-$GRU_{100}$ | 3.81 | 0.78 | 3.97 | 0.87 | 3.32 | 0.77 |
| XY-$GRU_{200}$ | 4.18 | 0.71 | 4.68 | 0.83 | 3.76 | 0.71 |
| XY-$GRU_{16,16}$ | 3.20 | 0.80 | 3.58 | 0.89 | 3.65 | 0.73 |
| XY-$GRU_{25,25}$ | 3.83 | 0.78 | 4.03 | 0.87 | 3.47 | 0.74 |
| XY-$GRU_{50,16}$ | 3.68 | **0.81** | 3.83 | 0.90 | 3.21 | 0.77 |
| XY-$GRU_{50,50}$ | 4.45 | 0.74 | 4.17 | 0.87 | 4.04 | 0.70 |
| XY-$GRU_{100,50}$ | **3.01** | **0.81** | 3.94 | 0.90 | 3.27 | 0.78 |
| XY-$GRU_{100,100}$ | 3.73 | 0.80 | 3.89 | 0.88 | 3.40 | 0.74 |

Table 15: Forecasting performance expressed as RMSE and $R^2$ of RNN models with time-shifted input series. Performance was evaluated by generating forecasts on the test set. In bold: for each target variable, it's highest quality prediction with respect to either RMSE or $R^2$.

| Models | | H | Diebold-Mariano | | |
|---|---|---|---|---|---|
| | | | WH2 | WH3 | WH5 |
| VECM(192) | XY-$GRU_{25}$ | | | | |
| | | 1 | 1.0000 | 1.0000 | 1.0000 |
| | | 8 | 1.0000 | 1.0000 | 1.0000 |
| | | 24 | 0.9998 | 0.9987 | 0.9999 |
| | | 48 | 0.8676 | 0.5484 | 0.6229 |
| | | 96 | 0.1271 | 0.0006 | 0.0000 |
| XY-$GRU_{25}$ | VECM(192) | | | | |
| | | 1 | 0.0000 | 0.0000 | 0.0000 |
| | | 8 | 0.0000 | 0.0000 | 0.0000 |
| | | 24 | 0.0002 | 0.0013 | 0.0001 |
| | | 48 | 0.1324 | 0.4516 | 0.3771 |
| | | 96 | 0.8729 | 0.9994 | 1.0000 |

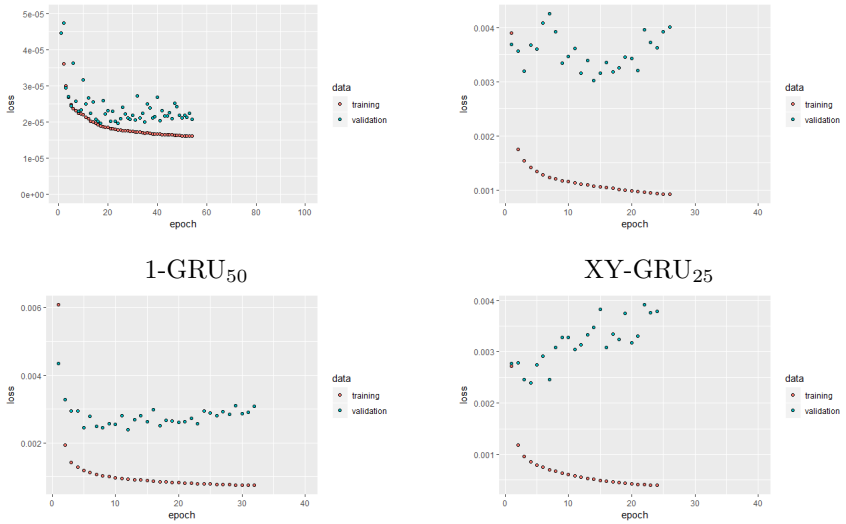
Table 16: P-values of the Diebold-Mariano test applied to predictions at specific horizons H of the VECM(192) model and the XY-$GRU_{25}$ model. The null hypothesis is that the performance is the same, the alternative hypothesis is that the second (rightmost) model has a better performance.

| Model | WH2 | | WH3 | | WH5 | |
|---|---|---|---|---|---|---|
| | RMSE | $R^2$ | RMSE | $R^2$ | RMSE | $R^2$ |
| ED-$GRU_{8,8}$ | 2.98 | 0.84 | 5.39 | 0.78 | 3.01 | 0.80 |
| ED-$GRU_{8,12}$ | 2.51 | **0.87** | 3.31 | 0.92 | **2.74** | 0.83 |
| ED-$GRU_{12,12}$ | **2.37** | **0.87** | 3.24 | 0.92 | 2.79 | 0.83 |
| ED-$GRU_{25,25}$ | 2.85 | 0.85 | 3.23 | 0.92 | 3.00 | 0.79 |
| ED-$GRU_{25,50}$ | 2.95 | 0.86 | **3.04** | **0.93** | 2.85 | **0.84** |
| ED-$GRU_{50,25}$ | 3.28 | 0.85 | 3.31 | 0.92 | 2.84 | 0.83 |
| ED-$GRU_{50,50}$ | 3.04 | 0.84 | 3.49 | 0.91 | 2.87 | 0.82 |

Table 17: Forecasting performance expressed as RMSE and $R^2$ of Encoder/Decoder RNN models with exogenous variables. Performance was evaluated by generating forecasts on the test set. In bold: for each target variable, it's highest quality prediction with respect to either RMSE or $R^2$.

| Models | | H | Diebold-Mariano WH2 | WH3 | WH5 |
|---|---|---|---|---|---|
| VECM(192) | ED-GRU$_{25,50}$ | | | | |
| | | 1 | 1.0000 | 1.0000 | 1.0000 |
| | | 8 | 0.9997 | 0.9946 | 0.9943 |
| | | 24 | 0.9980 | 0.9641 | 0.9764 |
| | | 48 | 0.7205 | 0.1336 | 0.5091 |
| | | 96 | 0.1522 | 0.2375 | 0.0104 |
| ED-GRU$_{25,50}$ | VECM(192) | | | | |
| | | 1 | 0.0000 | 0.0000 | 0.0000 |
| | | 8 | 0.0003 | 0.0054 | 0.0057 |
| | | 24 | 0.0020 | 0.0359 | 0.0236 |
| | | 48 | 0.2795 | 0.8664 | 0.4909 |
| | | 96 | 0.8478 | 0.7625 | 0.9896 |
| D-NN$_{96}$(96) | ED-GRU$_{25,50}$ | | | | |
| | | 1 | 0.4721 | 0.0000 | 0.1162 |
| | | 8 | 0.9083 | 0.0001 | 0.0393 |
| | | 24 | 0.4827 | 0.0388 | 0.8380 |
| | | 48 | 0.7659 | 0.4619 | 1.0000 |
| | | 96 | 0.4266 | 0.5855 | 0.5920 |
| ED-GRU$_{25,50}$ | D-NN$_{96}$(96) | | | | |
| | | 1 | 0.5279 | 1.0000 | 0.8838 |
| | | 8 | 0.0917 | 0.9999 | 0.9607 |
| | | 24 | 0.5173 | 0.9612 | 0.1620 |
| | | 48 | 0.2341 | 0.5381 | 0.0000 |
| | | 96 | 0.5734 | 0.4145 | 0.4080 |

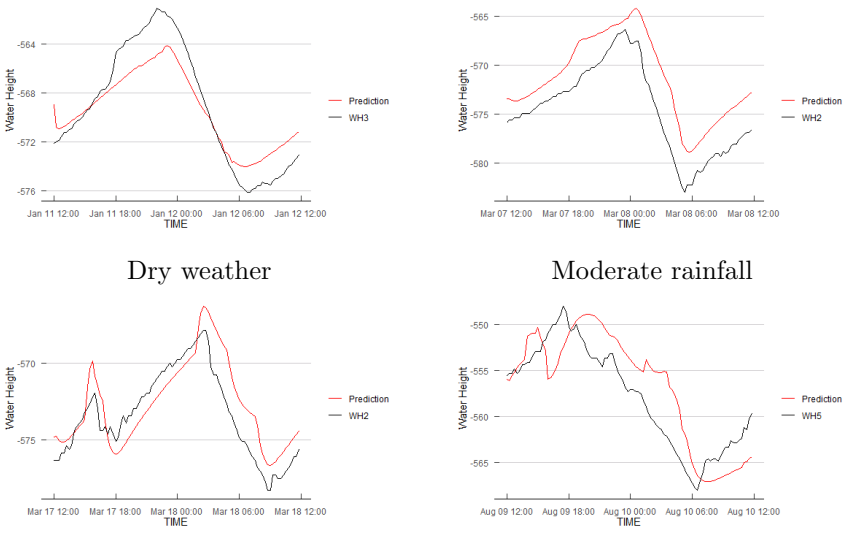
Table 18: P-values of the Diebold-Mariano test applied to predictions at specific horizons H of the VECM(192) model, the D-NN$_{96}$(96) model and the ED-GRU$_{25,50}$ model. The null hypothesis is that the performance is the same, the alternative hypothesis is that the second (rightmost) model has a better performance.




1-GRU$_{50}$ XY-GRU$_{25}$

ED-GRU$_{12,12}$ ED-GRU$_{25,50}$

Figure 9: MSE measured on the training set and validation set after every training iteration, for various RNN architectures.




Dry weather Moderate rainfall

Dry, after rainfall Heavy rainfall

Figure 10: Exemplary selection of WH2, WH3 and WH5 water levels, and the forecasts generated by the ED-GRU$_{25,50}$ model. Captions indicate the weather conditions on the day for which the forecast was generated.

## 7.4 Model Comparison

Since we generate forecasts for multiple time steps into the future, the error is expected to increase as we are further away from the forecast origin. To illustrate this, we report the average RMSE for the WH2 forecasts, conditioned on the hour of the day. Note that Hour = 12 corresponds to forecasts $T+1,\ldots,T+4$ and Hour = 11 corresponds to $T+93,\ldots,T+96$. We report in Table 19 the metrics for the VECM(192) time series model, the recursive- and direct forecasting ANNs, and the ED-RNN model. Our first conclusion is that the two recursive forecasting models have a much better performance for the very short horizons, which we had already concluded from Tables 12 & 18 for the VECM(192) model. Then, for the two recursive forecasting models, we observe a monotonous increase in the error per hour, which can be attributed to the accumulation of errors over multiple time steps into the future. For the D-NN and ED-RNN models however, we occasionally observe a decrease in the error per hour. This could be due to the fact that certain horizons, such as exactly 12 hours into the future, are easier to forecast due to the strong daily pattern in the data. Lastly, we observe that the accumulation of errors leads to a strictly worse performance for the recursive forecasting models for forecasting horizons greater than 19 hours.

Additionally, we are interested in the forecasting performance only at times $T_i+96$, as described in Section 6.3.2. At that forecasting horizon, the direct forecasting methodology does not utilize information that is not used by the recursive methodology. The results are shown in Table 20. We observe that the performance of the 1-$\text{NN}_{48,16}(96)$ is very bad at this long forecasting horizon. We also observe that the performance of the others forecasting models is much worse compared to their performance at times $T_i+1,\ldots,T_i+96$, but relative to each other they remain nearly equal. Comparing the VECM, D-NN and ED-RNN model predictions, it becomes clear that there is no substantial shift in relative model performances when evaluating only $T_i+96$ or when evaluating $T_i+1,\ldots,T_i+96$. This becomes especially clear when we look at the results of the Diebold-Mariano tests for H = 96 in Tables 12 & 18. The performance of these models remains comparable, whereas we were worried that the D-NN would perform significantly worse at times $T_i+96$. Thus, we see no clear indication that the direct forecasting methodology provides a big advantage when it comes to the utilization of exogenous features in this conditional forecasting problem.

| Hour | VECM(192) | 1-$NN_{48,16}$(96) | D-$NN_{96}$(96) | ED-$GRU_{25,50}$ |
|---|---|---|---|---|
| 12 | 0.47 | 0.70 | 1.25 | 1.28 |
| 13 | 0.73 | 0.91 | 1.16 | 1.39 |
| 14 | 1.01 | 1.15 | 1.33 | 1.88 |
| 15 | 1.22 | 1.36 | 1.67 | 2.17 |
| 16 | 1.47 | 1.53 | 1.94 | 2.24 |
| 17 | 1.71 | 1.88 | 2.06 | 2.23 |
| 18 | 1.81 | 2.16 | 2.45 | 2.32 |
| 19 | 1.89 | 2.26 | 2.7 | 2.30 |
| 20 | 1.95 | 2.31 | 2.87 | 2.24 |
| 21 | 2.05 | 2.46 | 2.65 | 2.22 |
| 22 | 2.18 | 2.66 | 2.36 | 2.33 |
| 23 | 2.45 | 2.89 | 2.36 | 2.50 |
| 0 | 2.64 | 3.17 | 2.48 | 2.70 |
| 1 | 2.79 | 3.42 | 2.6 | 2.96 |
| 2 | 2.98 | 3.65 | 2.95 | 3.39 |
| 3 | 3.16 | 3.9 | 3.41 | 3.74 |
| 4 | 3.33 | 4.11 | 3.52 | 3.97 |
| 5 | 3.47 | 4.24 | 3.39 | 3.93 |
| 6 | 3.55 | 4.32 | 2.93 | 3.79 |
| 7 | 3.58 | 4.4 | 2.78 | 3.68 |
| 8 | 3.71 | 4.49 | 2.94 | 3.65 |
| 9 | 3.82 | 4.56 | 3.11 | 3.66 |
| 10 | 3.96 | 4.68 | 3.35 | 3.68 |
| 11 | 4.09 | 4.8 | 3.81 | 3.75 |

Table 19: Forecasting RMSE for the WH2 water levels conditioned on the hour of the day. Forecasts are generated by the VECM(192) model, neural networks for recursive forecasting (1-$NN_{48,16}$(96)) and direct forecasting (D-$NN_{96}$(96)), as well as the ED-$GRU_{25,50}$ RNN.

| Model | WH2 | | WH3 | | WH5 | |
|---|---|---|---|---|---|---|
| | RMSE | $R^2$ | RMSE | $R^2$ | RMSE | $R^2$ |
| Baseline | 4.58 | 0.46 | 5.64 | 0.76 | 5.52 | 0.32 |
| VECM(192) | 4.08 | 0.68 | 4.46 | 0.85 | 4.68 | 0.56 |
| 1-$NN_{48,16}(96)$ | 6.50 | 0.38 | 6.26 | 0.76 | 10.69 | 0.27 |
| D-$NN_{25}(96)$ | 3.97 | 0.71 | 4.13 | 0.87 | 3.50 | 0.67 |
| ED-$GRU_{25,50}$ | 3.77 | 0.74 | 4.27 | 0.86 | 3.80 | 0.69 |

Table 20: Forecasting performance expressed as RMSE and $R^2$ of the predictions at times $T_i + 96$. Forecasts are generated by the naive baseline model, the VECM(192) model, neural networks for recursive forecasting (1-$NN_{48,16}(96)$) and direct forecasting (D-$NN_{96}(96)$), as well as the ED-$GRU_{25,50}$ RNN.

As stated before, the testing and training set are imbalanced when it comes to total precipitation per day. It is interesting to see how this influences the performance of the models. Therefore, we wish to know what the forecast quality is on those days where rainfall was recorded. This is done by taking samples of $\mathbf{y}$ and $\hat{\mathbf{y}}$, for days on which a certain amount of rain has fallen, and determining the forecasting quality for that sample. Table 21 shows the forecasting performance, expressed in RMSE, for days conditioned on this minimal daily precipitation sum.

From Table 21, we conclude that the data is severely imbalanced: only a small portion of 35% of the days has a total precipitation of more than 1 mm, which is not much. The negative impact on the model performance becomes clear: for each of the models, we observe an almost monotonous increase in error as the precipitation sum goes up. We also conclude that the D-NN and ED-RNN models appear to be the most robust to the imbalance in the dataset, and that the recursive models perform much worse in situations with moderate to heavy rainfall.

| Precipitation | VECM(192) | 1-$NN_{48,16}(96)$ | D-$NN_{96}(96)$ | ED-$GRU_{25,50}$ | % of days |
|---|---|---|---|---|---|
| $\geq$ 0 mm | 2.6 | 3.28 | 2.68 | 2.95 | 100% |
| $\geq$ 0.5 mm | 3.5 | 4.07 | 2.98 | 3.00 | 42% |
| $\geq$ 1 mm | 3.72 | 4.34 | 3.13 | 3.07 | 35% |
| $\geq$ 2 mm | 3.65 | 4.05 | 3.18 | 3.22 | 27% |
| $\geq$ 4 mm | 3.98 | 4.1 | 3.32 | 3.24 | 18% |
| $\geq$ 6 mm | 4.48 | 3.82 | 3.32 | 3.38 | 11% |
| $\geq$ 8 mm | 4.69 | 3.75 | 3.39 | 3.46 | 8% |
| $\geq$ 10 mm | 5.68 | 4.13 | 3.77 | 3.81 | 6% |
| $\geq$ 15 mm | 3.41 | 4.58 | 4.02 | 4.12 | 2% |
| $\geq$ 20 mm | 4.12 | 5.05 | 5.08 | 4.01 | 1% |

Table 21: Forecasting RMSE for the WH2 water levels conditioned on days with a minimal total precipitation. Forecasts are generated by the VECM(192) model, neural networks for recursive forecasting (1-$NN_{48,16}(96)$) and direct forecasting (D-$NN_{96}(96)$), as well as the ED-$GRU_{25,50}$ RNN. Last column shows the fraction of days in 2018 that belong to the sample which we evaluated.

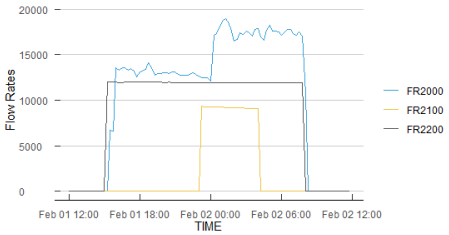
Besides contrasting the predictions generated by each of the model architectures, it might also be useful to see how they could complement each other. For this, we evaluated the forecast performance when the predictions of 2 or more different models are averaged at each time step. The results of this approach are the predictions of an *ensemble model*. The intuition behind an ensemble model is that by combining the predictions of individual models, overfitting is reduced and random errors due to misspecification are to some extent cancelled out, while the predictive power is retained. The ensemble models show that there is still room for improvement.

| Ensemble | WH2 | | WH3 | | WH5 | |
|---|---|---|---|---|---|---|
| | RMSE | $R^2$ | RMSE | $R^2$ | RMSE | $R^2$ |
| ED-GRU$_{12,12}$, ED-GRU$_{25,50}$ | 2.47 | 0.88 | 3.00 | 0.93 | 2.63 | 0.85 |
| D-NN$_{48}$, D-NN$_{96}$ | 2.38 | 0.90 | 2.94 | 0.93 | 2.47 | 0.86 |
| ED-GRU$_{12,12}$, D-NN$_{96}$ | 2.28 | 0.90 | 2.88 | **0.94** | 2.29 | 0.88 |
| ED-GRU$_{25,50}$, D-NN$_{96}$ | 2.65 | 0.90 | 2.79 | **0.94** | 2.34 | 0.88 |
| ED-GRU$_{12,12}$, D-NN$_{48}$ | **2.19** | 0.90 | 2.82 | 0.93 | 2.45 | 0.87 |
| ED-GRU$_{12,12}$, ED-GRU$_{25,50}$, D-NN$_{96}$ | 2.39 | 0.90 | 2.81 | **0.94** | 2.32 | 0.88 |
| D-NN$_{24}$, D-NN$_{48}$, D-NN$_{96}$ | 2.26 | **0.91** | 2.93 | 0.93 | 2.43 | 0.87 |
| ED-GRU$_{12,12}$, ED-GRU$_{25,25}$, ED-GRU$_{25,50}$, ED-GRU$_{50,25}$ | 2.66 | 0.88 | 2.96 | 0.93 | 2.57 | 0.85 |
| ED-GRU$_{12,12}$, ED-GRU$_{25,50}$, D-NN$_{48}$, D-NN$_{96}$ | 2.29 | **0.91** | **2.71** | **0.94** | 2.25 | **0.89** |
| ED-GRU$_{12,12}$, ED-GRU$_{25,25}$, ED-GRU$_{25,50}$, ED-GRU$_{50,25}$, D-NN$_{24}$, D-NN$_{48}$, D-NN$_{96}$ | 2.37 | 0.90 | 2.74 | **0.94** | **2.21** | **0.89** |

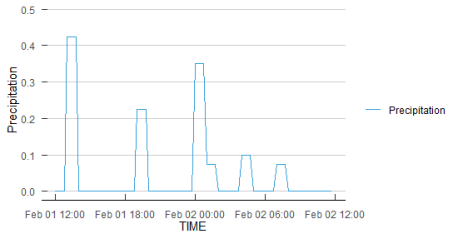
Table 22: Forecasting performance expressed as RMSE and $R^2$ of ensemble models containing one or more Encoder/Decoder RNN models and/or ANN for direct forecasting with lag $p = 96$. Performance was evaluated by generating forecasts on the test set using each of the models in the ensemble, and then taking the average of the predictions at each time step. In bold: for each target variable, it's highest quality prediction with respect to either RMSE or $R^2$.

## 7.5 Conditional Forecasting

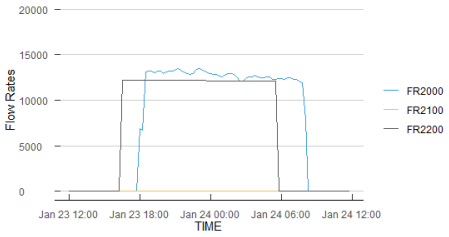
Let us now use the time series models to generate conditional forecasts for hypothetical scenario's. We do this by explicitly defining the future paths for one or more of the exogenous variables. The first scenario we decide to evaluate is a decrease in flow rates. The alternative exogenous future path has flow rates that are lower (less pumping) and with different timing (FR2000 is activated later, FR2200 is de-activated earlier) In a real-life scenario, this would correspond to taking less extreme water management actions, in order to save energy. In Figure 11, we visualize the data subset that was selected from the test set, and the alternative scenario for which we generate a conditional forecast.




Original Flow Rates



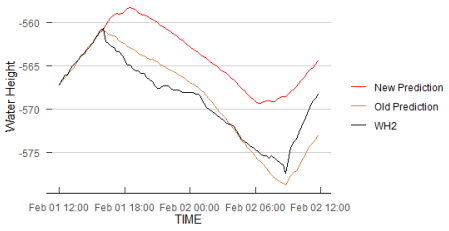

Original Precipitation




Alternative Flow Rates

Figure 11: Visualization of the exogenous variables used to generate the original prediction and the alternative prediction for February 2, 2018.

In Figures 12, 13, 14 and 15 we show the result of this conditional forecasting scenario for each of the modelling architectures. The original water level is plotted to provide an indication of the quality of the original prediction. Hydrologists at Waterschap Zuiderzeeland indicated that, based on their domain experience, this alternative pumping scenario should result in higher water levels with a delayed peak. The alternative D-NN forecasts did not contain the expected delayed peaks, and for WH3 it has mostly lower water levels, which is unrealistic. The alternative VECM and ED-RNN forecasts did contain the expected changes compared to the original scenario. The same holds for the 1-NN model, but we observe that the original forecast was of terrible quality. According to the knowledge and experience of the hydrologists at Waterschap Zuiderzeeland, the alternative forecasts of the VECM and ED-RNN model are therefore realistic and could be used for real-life scenario evaluation.



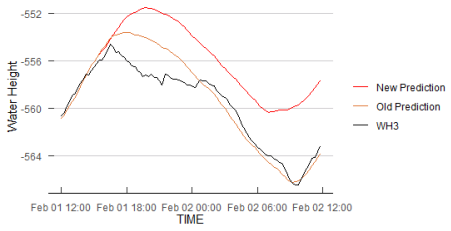




Figure 12: Actual water height, original forecast and conditional forecast on February 2, for target series WH2 and WH3. Predictions are generated by the VECM(192) model.

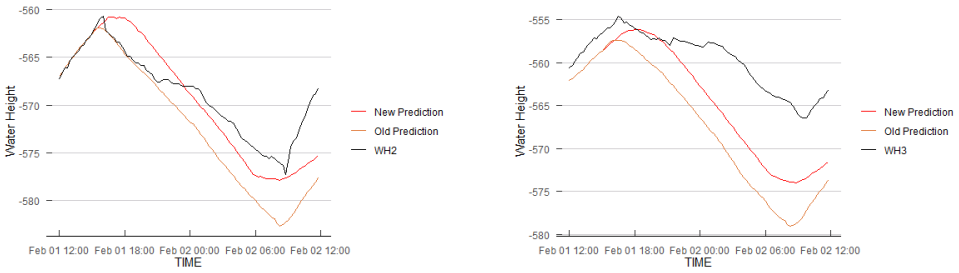



Figure 13: Actual water height, original forecast and conditional forecast on February 2, for target series WH2 and WH3. Predictions are generated by the 1-NN$_{48,16}$(96) model.

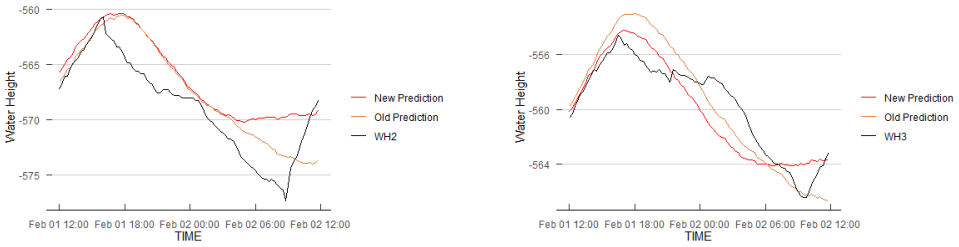



Figure 14: Actual water height, original forecast and conditional forecast on February 2, for target series WH2 and WH3. Predictions are generated by the D-NN$_{96}$(96) model.

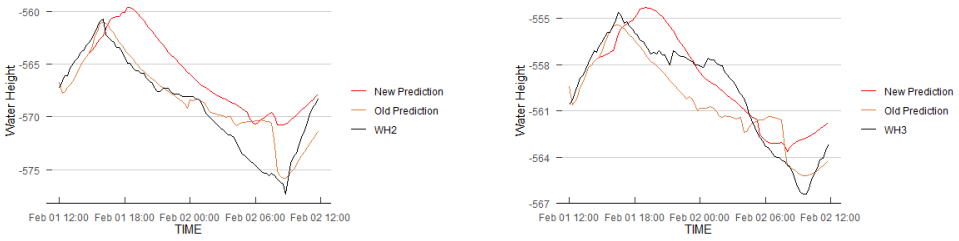
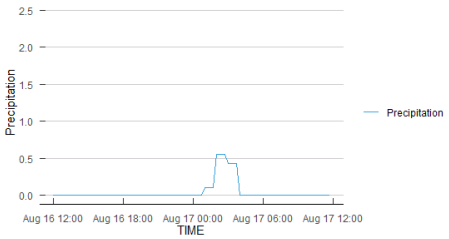
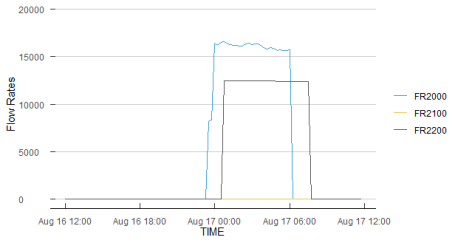



Figure 15: Actual water height, original forecast and conditional forecast on February 2, for target series WH2 and WH3. Predictions are generated by the ED-RNN$_{25,50}$ model.

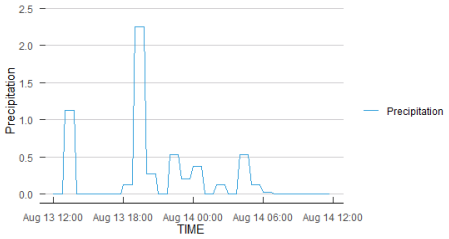
Let us then consider another scenario. This time, we wish to look at a day without significant rainfall and with a generic sequence of flow rates. We are then interested in the effect of alternative, heavy rainfall on such a common day. In a real-life scenario, this would provide the water managers with an indication of the impact of rain on the water levels if they do not take precautionary action. A visualization of the data subset for this scenario is shown in Figure 16.




Original Precipitation




Original Flow Rates




Alternative Precipitation

Figure 16: Visualization of the exogenous variables used to generate the original (old) conditional forecast and the alternative (new) prediction for August 17, 2018.

In Figures 17, 18, 19 and 20 we show the result of this second conditional forecasting scenario. The original water level is plotted to provide an indication of the quality of the original prediction. Hydrologists at Waterschap Zuiderzeeland indicated that, based on their domain experience, this alternative pumping scenario should result in slightly higher water levels. The increase in water levels as predicted by the VECM and 1-NN models is highly unrealistic: this is likely related to the bad performance of these models in situations with moderate to heavy rainfall, as observed in Table 21. The alternative D-NN forecasts does contain the expected increased water height, however, not during the first few hours of the WH3 forecasts, which is unrealistic: this could be caused by the use of exogenous variables at time steps after the forecasting horizon. The alternative ED-RNN forecasts did contain the expected changes compared to the original scenario, however, the forecasts occasionally contains peeks that appear to be too extreme. According to the knowledge and experience of the hydrologists at Waterschap Zuiderzeeland, the alternative forecasts of the ED-RNN model are realistic and could be used for real-life scenario evaluation.

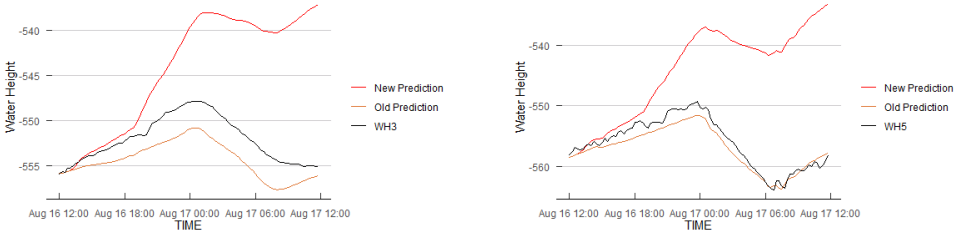



Figure 17: Actual water height, original forecast and conditional forecast on August 17, for target series WH3 and WH5. Predictions are generated by the VECM(192) model.

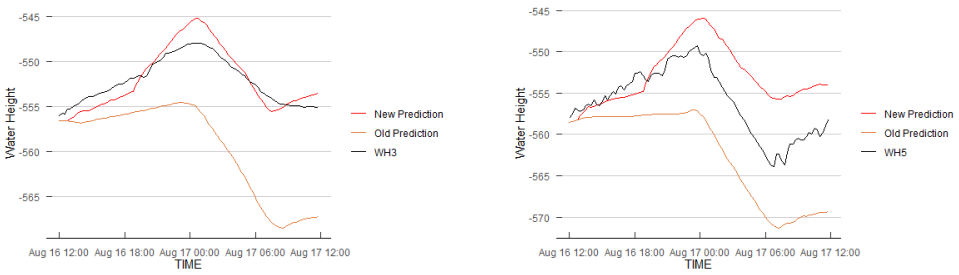



Figure 18: Actual water height, original forecast and conditional forecast on August 17, for target series WH3 and WH5. Predictions are generated by the 1-$NN_{48,16}(96)$ model.

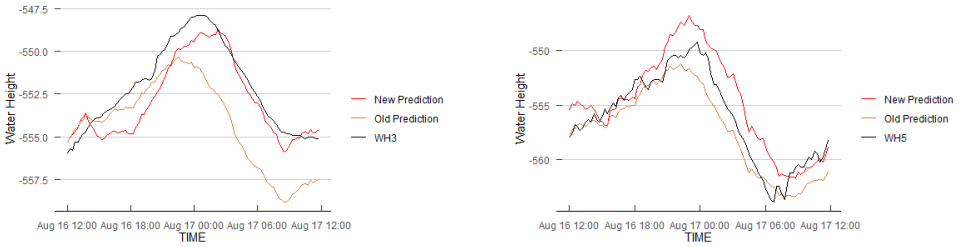



Figure 19: Actual water height, original forecast and conditional forecast on August 17, for target series WH3 and WH5. Predictions are generated by the D-$NN_{96}(96)$ model.

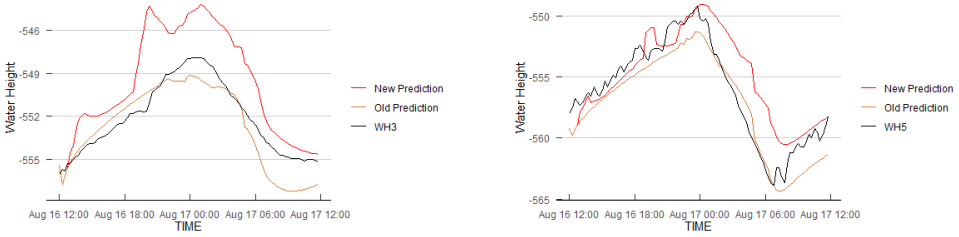



Figure 20: Actual water height, original forecast and conditional forecast on August 17, for target series WH3 and WH5. Predictions are generated by the ED-RNN$_{25,50}$ model.

# 8 Conclusion

This work describes a practical situation in which the application of forecasting models could lead to energy efficiency and decreased risk in water level management. In consultation with the Waterschap Zuiderzeeland, the problem was formalized as conditional forecasting of time series measurements of hydrological variables. Our modelling choices, e.g. the forecasting horizon and the inclusion of future paths of exogenous variables, ensure that the resulting forecasting problem imitates a real-life forecasting task.

We have proposed a novel architecture for conditional forecasting with RNNs. Our approach, described as an *Encoder/Decoder with Exogenous Variables* RNN (ED-RNN) model, is intuitive in its memory management, can be easily extended and is flexible with regard to the inclusion of past observations and future paths of exogenous variables. It combines the ideas of sequence-to-sequence (Seq2Seq) RNN models with the conditional forecasting literature, resulting in a model architecture that can be used for modelling in all sorts of time series problems.

The performance on the conditional forecasting problem was evaluated by fitting various models of the ED-RNN architecture and other time series models in our experiments. We show that on this specific forecasting problem, for the longer forecasting horizons, the results of the ED-RNN are comparable to the best performing time series model (the D-NN) and better than the other alternatives considered. We also show that the predictions generated by the ED-RNN more accurately capture short-term fluctuations in the water heights than the predictions of the D-NN. Additionally, the ED-RNN consistently generates realistic alternative conditional forecasts, unlike the other alternative models considered. Therefore, the model is applicable for real-life scenario evaluation and decision support by the Waterschap Zuiderzeeland.

Balancing the dataset was not investigated. A balanced dataset is important for machine learning models such as the ANN and RNN: if certain patterns (such as wet weather conditions) occur rarely in the data, the models tend to perform poorly in such situations. We showed that this is the case for daily precipitation sum in our data, and concluded that the ED-RNN architecture appears to be among the most robust models under such circumstances.

Our search for good hyperparameter values, such as the model size, lag parameter, number of iterations and stopping criterion, was not exhaustive. Inclusion of more features was also not investigated: this could improve the performance at the cost of longer training times. Hence, there is still room for improvement of the models presented in this thesis. We conclude that our ED-RNN model architecture is a very interesting option for conditional forecasting with RNNs and that more empirical evaluation is needed to compare its performance with that of alternative modelling techniques.

# 9 Future Work

As mentioned before, more empirical evaluation could be done to determine the best forecasting model for practical use by the Waterschap Zuiderzeeland. This search should not only include the hyperparameters of the model, but also the timeframe of the data, its granularity and the number of variables used as either exogenous or endogenous. In addition, a pilot project in which the forecasting models are used in real-life situations could point out the true applicability and create additional opportunities for improvement.

The novel Encoder/Decoder with Exogenous Variables RNN architecture can be further examined in the context of time series analysis and conditional forecasting. For example, the implications of including certain variables in either the encoding or decoding step are an interesting subject for further study. Testing the performance of the ED-RNN model on benchmark conditional forecasting tasks could also provide insight in how the model compares to other architectures. Lastly, the flexibility of the model allows it to model sequence data from different domains: the inclusion of exogenous variable during decoding could potentially also lead to improvements in modelling accuracy in other contexts.

# References


[1] François Anctil and Doha Guy Tape. An exploration of artificial neural network rainfall-runoff forecasting combined with wavelet decomposition. *Journal of Environmental Engineering and Science*, 3(S1):S121–S128, 2004.

[2] Marta Bańbura, Domenico Giannone, and Michele Lenza. Conditional forecasts and scenario analysis with vector autoregressions for large cross-sections. *International Journal of Forecasting*, 31(3):739–756, 2015.

[3] Anindya Banerjee, Juan J Dolado, John W Galbraith, David Hendry, et al. Co-integration, error correction, and the econometric analysis of non-stationary data. *OUP Catalogue*, 1993.

[4] Yoshua Bengio, Patrice Simard, Paolo Frasconi, et al. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[5] Gianluca Bontempi, Souhaib Ben Taieb, and Yann-Aël Le Borgne. Machine learning strategies for time series forecasting. In *European Business Intelligence Summer School*, pages 62–77. Springer, 2012.

[6] A. Bontsema. Master Thesis: Forecasting Ammonium Concentration In Wastewater Treatment Plant Effluent. `https://beta.vu.nl/nl/onderwijs/project-en-stage/stagebureau-wiskunde-informatica/master-project-ba/stageverslagen-online/index.aspx`, 2018. [Not yet online; confidential].

[7] Peter J Brockwell, Richard A Davis, and Matthew V Calder. *Introduction to time series and forecasting*, volume 2. Springer, 2002.

[8] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):6085, 2018.

[9] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[10] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[11] Robert F Engle and Clive WJ Granger. Co-integration and error correction: representation, estimation, and testing. *Econometrica: journal of the Econometric Society*, pages 251–276, 1987.

[12] Sacha Epskamp, Lourens J Waldorp, René Mõttus, and Denny Borsboom. The gaussian graphical model in cross-sectional and time-series data. *Multivariate behavioral research*, pages 1–28, 2018.

[13] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[14] Clive WJ Granger. Some properties of time series data and their use in econometric model specification. *Journal of econometrics*, 16(1):121–130, 1981.

[15] I. Groenen. Master Thesis: Representing Seasonal Patterns in Gated Recurrent Neural Networks for Multivariate Time Series Forecasting. `http://www.scriptiesonline.uba.uva.nl/657906`, 2018. [Online; accessed 20-Oktober-2018].

[16] Tian Guo, Tao Lin, and Yao Lu. An interpretable LSTM neural network for autoregressive exogenous model. *arXiv preprint arXiv:1804.05251*, 2018.

[17] David P Hamilton and S Geoffrey Schladow. Prediction of water quality in lakes and reservoirs.

Part I—Model description. *Ecological Modelling*, 96(1-3):91–110, 1997.
[18] James Douglas Hamilton. *Time series analysis*, volume 2. Princeton university press Princeton, NJ, 1994.
[19] Coşkun Hamzaçebi, Diyar Akay, and Fevzi Kutay. Comparison of direct and iterative artificial neural network forecast approaches in multi-periodic time series forecasting. *Expert Systems with Applications*, 36(2):3839–3844, 2009.
[20] David Harvey, Stephen Leybourne, and Paul Newbold. Testing the equality of prediction mean squared errors. *International Journal of forecasting*, 13(2):281–291, 1997.
[21] Simon S Haykin. *Neural networks and learning machines*, volume 3. Pearson education Upper Saddle River, 2009.
[22] Griet Heuvelmans, Bart Muys, and Jan Feyen. Regionalisation of the parameters of a hydrological model: Comparison of linear regression models with artificial neural nets. *Journal of Hydrology*, 319(1-4):245–265, 2006.
[23] Tim Hill, Leorey Marquez, Marcus O'Connor, and William Remus. Artificial neural network models for forecasting and decision making. *International journal of forecasting*, 10(1):5–15, 1994.
[24] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
[25] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.
[26] Koninklijk Nederlands Meteorologisch Instituut. Hourly weather data. `https://www.knmi.nl/nederland-nu/klimatologie/uurgegevens`, 2018. [Online; accessed 3-Januari-2019].
[27] RStudio Google JJ Alaire, Francois Chollet. R Interface to Keras. `https://keras.rstudio.com/`, 2019. [Online; accessed 05-May-2019].
[28] Søren Johansen. Estimation and hypothesis testing of cointegration vectors in gaussian vector autoregressive models. *Econometrica: Journal of the Econometric Society*, pages 1551–1580, 1991.
[29] Leonard F Konikow and John D Bredehoeft. Ground-water models cannot be validated. *Advances in water resources*, 15(1):75–83, 1992.
[30] Helmut Lütkepohl. *New introduction to multiple time series analysis*. Springer Science & Business Media, 2005.
[31] Michael W McCracken and Joseph T McGillicuddy. An empirical investigation of direct and iterated multistep conditional forecasts. *Journal of Applied Econometrics*, 34(2):181–204, 2019.
[32] AK Mishra and VR Desai. Drought forecasting using feed-forward recursive neural network. *ecological modelling*, 198(1-2):127–138, 2006.
[33] Tom Michael Mitchell. *The discipline of machine learning*, volume 9. Carnegie Mellon University, School of Computer Science, Machine Learning, 2006.
[34] HM Nepf. Drag, turbulence, and diffusion in flow through emergent vegetation. *Water resources research*, 35(2):479–489, 1999.
[35] John Neter, William Wasserman, and Michael H Kutner. Applied linear regression models. 1989.
[36] MB Priestley and T Subba Rao. A test for non-stationarity of time-series. *Journal of the Royal Statistical Society: Series B (Methodological)*, 31(1):140–149, 1969.
[37] Foster Provost. Machine learning from imbalanced data sets 101. In *Proceedings of the AAAI'2000 workshop on imbalanced data sets*, pages 1–3, 2000.
[38] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–

117, 2015.

[39] Marios Sophocleous. Interactions between groundwater and surface water: the state of the science. *Hydrogeology journal*, 10(1):52–67, 2002.

[40] James H Stock and Mark W Watson. Vector autoregressions. *Journal of Economic perspectives*, 15(4):101–115, 2001.

[41] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[42] Mukesh K Tiwari and Chandranath Chatterjee. Development of an accurate and reliable hourly flood forecasting model using wavelet-bootstrap-ANN (WBANN) hybrid approach. *Journal of Hydrology*, 394(3-4):458–470, 2010.

[43] Alfredo Vellido, José David Martín-Guerrero, and Paulo JG Lisboa. Making machine learning models interpretable. In *ESANN*, volume 12, pages 163–172. Citeseer, 2012.

[44] Daniel F Waggoner and Tao Zha. Conditional forecasts in dynamic multivariate models. *Review of Economics and Statistics*, 81(4):639–651, 1999.

[45] Wensheng Wang and Jing Ding. Wavelet network model and its application to the prediction of hydrology. *Nature and Science*, 1(1):67–71, 2003.

[46] Future Water. Rapport Wateraanvoer Noordoostpolder. `https://www.futurewater.nl/downloads/2006_Immerzeel_FW50.pdf`, 2006. [Online; accessed 19-Oktober-2018].

[47] Pei-Hwa Yen, Chyan-Deng Jan, Youe-Ping Lee, and Hsiu-Fang Lee. Application of Kalman filter to short-term tide level prediction. *Journal of waterway, port, coastal, and ocean engineering*, 122(5):226–231, 1996.

[48] Heesung Yoon, Seong-Chun Jun, Yunjung Hyun, Gwang-Ok Bae, and Kang-Kun Lee. A comparative study of artificial neural networks and support vector machines for predicting groundwater levels in a coastal aquifer. *Journal of Hydrology*, 396(1-2):128–138, 2011.

[49] G Peter Zhang. Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, 50:159–175, 2003.