Sensitivity Analysis of a Deep Learning Model for Discharge Prediction in the Regen Catchment

Master's Thesis (M.Sc. Environmental Engineering)
Department of Civil, Geo and Environmental Engineering
Technical University of Munich

Supervised by Dr. phil. Jorge Eduardo Teixeira Leandro
Chair of Hydrology and River Basin Management, TUM

Dr. Wolfgang Kurtz
Leibniz Supercomputing Centre of the Bavarian Academy of Sciences and Humanities

Ivana Jovanovic Buha, M.Sc. (hons)
Chair of Scientific Computing, TUM

Submitted by Leon Fiedler (03642227)
Friedrich – Ebert – Straße 5
82256 Fürstenfeldbruck

Place and Date Munich, 27.11.2020

Declaration of Authorship

I, Leon Fiedler, hereby declare that this Master Thesis, titled "Sensitivity Analysis of a Deep Learning Model for Discharge Prediction in the Regen Catchment” and the work presented in it are my own, except where stated otherwise.

The information and statements asserted in this master’s thesis are based on technical data, as well as available information and materials which are part of arbitration processes. All references have been quoted, and all sources of information have been specifically acknowledged. This thesis as not previously presented to another examination board and has not been published.

Munich,

Ort, Datum, Unterschrift

Acknowledgment

This thesis was supported by many people and various researchers of both, the Technical University of Munich and the Leibniz Supercomputing Centre of the Bavarian Academy of Sciences and Humanities.

First of all, I would like to send special thanks to Alexander von Ramm, who, although no longer with us, continues to inspire me by his example and dedication. I am particularly thankful that he supported me in my decision to choose this challenging – and to me completely new – field of studies for my master thesis.

Huge thanks to Ivana Jovanovic Buha, M.Sc. (hons) and Dr. Wolfgang Kurtz, who took over the intensive consultation, continuous motivation, and endless discussions on theoretical and practical programming concepts. Furthermore, I would like to offer huge thanks to Dr. phil. Jorge Eduardo Teixeira Leandro at the Chair of Hydrology and River Basin Management TUM, who has not only made this innovative master thesis topic possible, but also guided me patiently and faithfully through the process of writing my thesis.

The support of these four dedicated supervisors had a huge impact on the completion of this thesis – not only because of their profound professional knowledge, but also due to their emotional support. Finally, I would like to thank my friends and family for being my backup system when things get tough.

Abstract

Estimating rainfall-runoff relationship and streamflow modeling is a significant element in operational flood management. In order to be conducted successfully the rainfall-runoff relationship and discharge behavior in a watershed have to be predicted accurately. Thus, simplified physically based or conceptual models are still routinely applied for operational purposes. However, the development of accurate rainfall-runoff and streamflow models is still a challenging task since hydrological processes inherently exhibit nonlinear and complex behavior. In recent years data-driven approaches have increased hydrologists' attention due to their modeling simplicity and high accuracy for rainfall-runoff modeling. This thesis explores the idea of using a Long Short-Term Memory (LSTM) network model to predict the runoff in daily and hourly resolution within the Regen catchment, Germany. This is a special type of neural network that proved to be a powerful tool in learning long-term dependencies between provided input and output and in finding abstract patterns within long sequences. As model input point-based measurement series of 9 different meteorological parameters is used, recorded over a period of 14 years. A method is provided, which enables the LSTM to not only handle missing data gaps but also to potentially enhance its prediction accuracy. In the scope of a sensitivity analysis various combinations of input features are examined to evaluate their impact on the model performance and to identify the most appropriate sets for both temporal resolutions. The ability of LSTMs to forecast multiple time steps ahead regarding different lead times is explored, whereby also a novel approach is introduced to bring LSTMs closer to a real-world operational forecasting scenario. This approach essentially decompose the prediction process of an LSTM into individual steps through which model's output can be fed back into itself and predictions can potentially be made conditioned on forecasted meteorological data. Throughout the thesis, the prediction accuracy of the LSTM is compared to the deterministic, physically based Large Area Runoff Simulation Model (LARSIM), which serves as a benchmark. The results indicate that LSTMs are principally able to predict discharge with reasonable accuracy in a multi-step scenario with lead times of up to 24 hours. In a single step scenario the LSTM model, when trained on suitable input feature sets, proved to replicate the discharge hydrograph almost perfectly on unseen data, outperforming the physically based benchmark model. By tuning crucial hypermeters of the model with respect to a hydrological understanding improved the accuracy of the LSTM further. The findings in this thesis suggest that that LSTM models have the capability to learn the complete rainfall-runoff process purely from a properly selected data basis, underlining the potential of data-driven approaches for hydrological modelling applications.

Table of Contents

Declaration of Authorship i
Acknowledgment ii
Abstract iii
Table of Contents iv
List of Tables vi
List of Abbreviations xii
1. Introduction 1
- 1.1. Machine Learning Basics 3
- 1.2. Machine Learning in Hydrology 4
- 1.3. Objectives and Outline 9

2. Deep Learning Fundamentals 10
- 2.1. Introduction to Artificial Neural Networks 11
  - 2.1.1. Anatomy of a Neural Network 11
  - 2.1.2. Training, Backpropagation and Optimization 13
  - 2.1.3. Evaluation and Regularization 16
- 2.2. Deep Learning for Time Series 18
  - 2.2.1. Long-term Short Memory Networks 19
  - 2.2.2. Development of a multivariate forecasting model 23

3. Methodology 29
- 3.1. Overview of Investigation Steps 30
- 3.2. Study Area – Regen Catchment 32
- 3.3. Data Overview 33
- 3.4. Data Preparation 35
  - 3.4.1. Data Cleaning 35
  - 3.4.2. Imputation techniques for Missing Data 36
  - 3.4.3. Feature Scaling Techniques 39
  - 3.4.4. Final Dataset preparation 40
- 3.5. Model Architectures 41
- 3.6. Validation of the different Models 46
- 3.7. Methods for Multi-Step Forecasting 56

4. Results 59
- 4.1. First Model Test 60
- 4.2. Preliminary Analysis 64
  - 4.2.1. Standard versus Robust Scaling 64
  - 4.2.2. Methods for Missing Values 67
  - 4.2.3. Imputation techniques for Precipitation Data 67
- 4.3. Model Sensitivity to Feature Selection 70

4.3.1. Daily Resolution 71
4.3.2. Hourly Resolution 79
4.4. Hyperparameter Tuning 83
4.5. Multi-step Predictions 89
5. Discussion 93
5.1. Scaling 93
5.2. Missing Data 93
5.3. Imputation of Precipitation 94
5.4. Sensitivity analysis 95
5.5. Hyperparameter Tuning 98
5.6. Multi-step Forecasting 99
6. Conclusion and Outlook 101
Appendix 106
A. Tables 106
A. Figures 116
Publication bibliography 120

List of Tables

Table 1: Overview of available meteorological and hydrological parameter in the dataset 34

Table 2: Defined hyperparameters based on hyperparameter optimization conducted by Unnikrishnan (2019)........ 45

Table 3: Overview of Training, Validation and Test set size regarding the normal Holdout Method 47

Table 4: Overview of the used performance metrices in this thesis 52

Table 5: Resulting performance metric scores calculated for Training, Validation and Test Period, respectively. Scores highlighted in red indicate best score per metric within the complete table........ 62

Table 6: Resulting performance metric scores calculated for Training, Validation and Test Period, respectively. Scores highlighted in red indicate best score per metric within the complete table........ 66

Table 7: Resulting performance metric scores calculated for Training, Validation and Test Period, respectively. Scores highlighted in red indicate best score per metric within the complete table........ 68

Table 8: Resulting performance metric scores calculated for Training, Validation and Test Period, respectively. Scores highlighted in red indicate best score per metric within the complete table........ 69

Table 9: Resulting performance metric scores calculated for Training, Validation and Test Period, respectively. Scores highlighted in red indicate best score per metric across all input feature sets (including Table 10) with respect to daily (left) or hourly (right) data basis. 77

Table 10: Resulting performance metric scores calculated for Training, Validation and Test Period, respectively. Scores highlighted in red indicate best score per metric across all input feature sets (including Table 9) with respect to daily (left) or hourly (right) data basis. 78

Table 11: Overview of tested hyperparameter configuration with respect to temporal resolution. Bolt values indicate tested settings, whereby the other values remain constant........ 84

Table 12: Complete list of available measurement stations. "Parameters" refers to observed parameters at corresponding station. "Parameters_considered" refers to the parameters actually considered in the input dataset. Some parameters had to be discarded due to a large amount of missing data. 106

Table 13: Pseudo Code of the customized prediction loop (left) and corresponding array dimensions (right) 109
Table 14: Resulting performance metric scores calculated for Training, Validation and Test Period, respectively. Scores highlighted in red indicate best score per metric within one input feature set. 110
Table 15:Resulting performance metric scores calculated for Training, Validation and Test Period, respectively. Scores highlighted in red indicate best score per metric within the complete table..111
Table 16: Resulting performance metric scores calculated for Training, Validation and Test Period, respectively. Scores highlighted in red indicate best score per metric within the complete table..111
Table 17: Resulting performance metric scores calculated for Training, Validation and Test Period, respectively. Scores highlighted in red indicate best score per metric within the complete table..111
Table 18: Resulting performance metric scores calculated for Training, Validation and Test Period, respectively. Scores highlighted in red indicate best score per metric within one input feature set. 112
Table 19: Resulting performance metric scores calculated for Training, Validation and Test Period, respectively. Scores highlighted in red indicate best score per metric within the complete table..113
Table 20: Resulting performance metric scores calculated for Training, Validation and Test Period, respectively. Scores highlighted in red indicate best score per metric within the complete table...114
Table 21: Resulting performance metric scores calculated for Training, Validation and Test Period, respectively. Scores are calculated by averaging over the respective lead time. 12 Steps, 24 Steps and 36 Steps refer to multi-step one shot LSTM models. The customized prediction loop was only applied to the Test period. 115

List of Figures

Figure 1: Scheme of an Artificial Neural Network (La Fuente et al. 2019) 12

Figure 2: Illustration of a neuron in a feedforward network. The *j_{th}* neuron receives inputs from neurons (a_1, a_2, … , a_i) of the preceding layer multiplied by their respective weights and applies a nonlinear transformation (*f*). The output of the neuron, called activation, is passed to the neurons of the succeeding layer. (Shen 2018) 12

Figure 3: Schematic flow chart of a full training cycle in a Neural Network (Chollet 2018) 14

Figure 4: Training History of a Neural Network, showing the learning curves on the training and validating dataset according to (Géron 2019) 17

Figure 5: A recurrent neuron (left) unrolled through time (right) (Géron 2019) 19

Figure 6: A layer of recurrent neurons (left) unrolled through time (right) (Géron 2019) 19

Figure 7: Visualization of the standard LSTM cell as defined by Eq. (1-6) (Kratzert et al. 2019c) 21

Figure 8: Visualization of the sliding window method for single step predictions 24

Figure 9: Schematic illustration of how features within one window in one batch are processed by a LSTM unrolled through time predicting the next time step................. 25

Figure 10: Visualization of the sliding windows method for multi-step one shot predictions.. 26

Figure 11: Schematic illustration of how features within one window in one batch are processed by a LSTM unrolled through time predicting the next *n* time step (in this example the next 3 time steps)... 27

Figure 12: Illustration of the individual views along the axis of the 3D input array. The three dimensions are: (i) batch size (number of simultaneously processed windows), (ii) window size (number of consecutive time steps considered in the lookback window), (iii) number of available input features 28

Figure 13: Flow chart of the consecutive investigation and analysis steps carried out in this thesis 31

Figure 14: Overview of study area.. 32

Figure 15: Architecture of the LSTM model with corresponding Layers (left) unrolled over time (right) according to Unnikrishnan (2019) (except Masking Layer). In case missing values *Method 3* is applied, a masking layer is added to the model architecture. In the other both cases (*Method 1* & M*ethod 2*) the input features are passed directly to the 1st LSTM layer... 42

Figure 16: Computational graph that visualizes the array shape manipulation during a forward pass through the LSTM model. In this example 32 input features are processed with a window size 60 time steps. The defined number of LSTM units is 120. The “?” corresponds to the batch size but is indicated a variable sized within the network, since the batch size can be varied between model run calls. 44

Figure 17: Visualization of the normal Holdout Method. The dataset is split into three subsets. 47

Figure 18: Normal Holdout Method visualized on the discharge hydrograph at Marienthal station (left) and corresponding Boxplots (right) based on the underlying data distribution (log-transformed) of the three different subsets 48

Figure 19: Visualization of the Rep-Holdout Method. Split point p is randomly selected within a specified window. Percentages correspond to the total number of samples of the combined Training and Validation set and define the range of the window. For each training of the model a new split point is selected. 49

Figure 20: Visualization of the k-fold cross validation. In this example the training and validation set is split in 3 folds, which results in three different validation sets. The three model runs are evaluated on the same test set. 50

Figure 21: Schematic illustration of the customized prediction loop. As an example, the procedure of the loop is demonstrated for two iterations, whereby the model predicts 3 steps ahead (*n)* for each iteration based on cached input windows from the measurement series (inner loop). For illustration purposes, an example data frame is depicted, which consists of random samples from three input features (X1,X2,X3) and one target variable(Y). Green shaded rows imply measured values, whereas blue shaded rows are assumed to be forecasted values. After each time step the predicted value is replace with the target value of the next input window. The second iteration is restarted, e.g. every 3 hours (update step *u) (outer loop)* and the forecasted values could be replaced by true observations within the measurement series before getting cached again to compute the next n steps ahead (inner loop). 57

Figure 22: Study Area, only showing stations that are within or close (<2km) to the catchment boundary. The Orange square frames the *Leming* gauge (blue rhombus) and precipitation station at *Neukirchen bei Heiligen Blut* (gray circle) 60

Figure 23: Comparison between measured, predicted (LSTM) and simulated discharge (LARSIM) in daily resolution. LSTM predictions are based on the input of one precipitation station (*Neukirchen bei Heiligen Blut).* 62

Figure 24: Comparison between measured, predicted (LSTM) and simulated discharge (LARSIM) in daily resolution. LSTM predictions are based on the input of one precipitation station (Neukirchen bei Heiligen Blut) & one discharge gauge (*Leming* = target station) 63

Figure 25: Kernel Density Estimation plots for different tested scaling techniques. For each plot one example feature (station) per meteorological parameter was selected, that showed a high number of outliers.......65
Figure 26: Comparison between measured, predicted (LSTM) and simulated discharge (LARSIM) in daily resolution. LSTM predictions are based on input feature set 1.......73
Figure 27: Comparison between measured, predicted (LSTM) and simulated discharge (LARSIM) in daily resolution. LSTM predictions are based on input feature set 5.......74
Figure 28: Comparison between measured, predicted (LSTM) and simulated discharge (LARSIM) in daily resolution. LSTM predictions are based on input feature set 4.......75
Figure 29: Comparison between measured, predicted (LSTM) and simulated discharge (LARSIM) in hourly resolution. LSTM predictions are based on input feature set 1. ...80
Figure 30: Comparison between measured, predicted (LSTM) and simulated discharge (LARSIM) in hourly resolution. LSTM predictions are based on input feature set 5. ...81
Figure 31: Comparison between measured, predicted (LSTM) and simulated discharge (LARSIM) in hourly resolution. LSTM predictions are based on input feature set 4. ...82
Figure 32: Comparison between measured, predicted (LSTM) and simulated discharge (LARSIM) in daily resolution. LSTM predictions are based on input feature set 5 and model is trained with MSE as objective function.......85
Figure 33: Comparison between measured, predicted (LSTM) and simulated discharge (LARSIM) in daily resolution. LSTM predictions are based on input feature set 5, window size 60 steps and 256 LSTM units.......87
Figure 34: Comparison between measured, predicted (LSTM) and simulated discharge (LARSIM) in hourly resolution. LSTM predictions are based on input feature set 5, window size 120 steps and 256 LSTM units.......88
Figure 35: Comparison between measured, predicted (LSTM) and simulated discharge (LARSIM) in hourly resolution. LSTM predictions are based on input feature set 3, window size 120 steps and 256 LSTM units.......89
Figure 36: Progression of the NSE score on Test Data over future time steps for all tested models (indicated by different line styles). In order to have more zoomed in view, the graph of the single-shot model with a lead time of 36 steps (yellow triangles), is cut off. For the 36th step this model reached a NSE score of 0.654.......91
Figure 37: Progression of the KGNP score on Test Data over future time steps for all tested models (indicated by different line styles). In order to have more zoomed in view, the graph of the single-shot model with a lead time of 36 steps (yellow triangles), is cut off. For the 36th step this model reached a KGNP score of 0.912.......92
Figure 38: Training history of the trained LSTM model based on one precipitation station as input feature.......116

Figure 39: Rep-Holdout Method visualized on the discharge hydrograph at Marienthal station (left) and corresponding Boxplots (right) based on the underlying data distribution (log-transformed) of the three different subsets. Here, as example, 6 model iterations are shown, each having a slightly different split point between Training and Validation set. 117

Figure 40: K-fold cross validation visualized on the discharge hydrograph at Marienthal station (left) and corresponding Boxplots (right) based on the underlying data distribution (log-transformed) of the three different subsets. Here, as example, 3 model iterations are shown, where each model is trained on different training subsections within the measurement series. 118

Figure 41: Progression of the MAE/ RSME score on Test Data over future time steps for all tested models (indicated by different line styles). 119

List of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Networks
AVwC	Arithmetic Averaging Nearer Stations Based on Conditions
B	Batch size
CNN	Convolutional Neural Network
CONUS	Continental United States
DL	Deep Learning
DNN	Deep Neural Network
ERNN	Elman Recurrent Neural Network
ESN	Echo State Network
F	Features
FDC	Flow Duration Curve
GRU	Grouped Response Units
HU	Hydraulic Units
IDW	Inverse Distance Weighting
IQR	Interquartile Range
KDE	Kernel Density Estimation
KGE	Kling-Gupta-Efficiency
KGNP	Nonparametric Kling-Gupta-Efficiency
LARSIM	Large Area Runoff Simulation Model
LR	Linear Regression
LSTM	Long-Short Term Memory
Max-Error	Maximum Error
MeAE	Median-Absolute-Error
ML	Machine Learning
MLP	Multilayer Perceptrons

MSE	Mean-Squared-Error
MTS-LSTM	Multi timescale LSTM
NARX	Nonlinear Autoregressive Exogenous Neural Network
NN	Neural Networks
NRM_D	Normal Ratio Method with Respect to Distance
NSE	Nash-Sutcliff-Efficiency
NWM	US National Water Model
NWP	Numerical Weather Prediction
PUB	Prediction in Ungauged Basins
ReLU	Rectified Linear Units
REV	Relative Error in Volume
RMSE	Root-Mean-Squared-Error
RNN	Recurrent Neural Network
SAC-SMA	Sacramento Soil Moisture Accounting Model
SVM	Support Vector Machines
W	Window size

1. Introduction

Artificial Intelligence (AI) is the field of computer science that aims to make intelligent computer systems, by trying to mimic human behavior. This specific type of intelligence includes the ability to interact with the environment by learning from and modeling it to make predictions, even in the case of unexpected scenarios. AI techniques are adopted for resolving a huge area of science and engineering problems. One area, in which the application of AI-based methods increased in recent years is the field of hydrology. Especially flood management and flood forecasting are popular scenarios (Fotovatikhah et al. 2018).

In flood forecasting, describing the relationship between rainfall and runoff is one of the central tasks. This involves the prediction of river discharge using meteorological observations of a river basin. The basin or catchment of a river is defined by the area from which all (surface) runoff drains to a common outlet. Streamflow forecasting, i.e. predicting the discharge of a river, especially in a long-term perspective, is of great importance to water resource management. Short-term streamflow forecasting is crucial for flood defense, whereas medium-range forecasting is highly beneficial for reservoir operation.

In general, a better understanding of the streamflow process is fundamental for improving the skill of streamflow forecasting (Wang 2006). In order to accurately predict streamflow, the components of hydrological processes in a catchment have to be modeled in some way as well. According to Freeze and Harlan (1969) two basic approaches can be distinguished: a physical model approach and a hydrologic system investigation. Physical hydrology should examine the fundamental process in the hydrophilic cycle in a systematic and scientific way. Freeze and Harlan (1969) state that, if each part of this process are formulated by a physical law in an implicit mathematical way, it should be possible to model the complete process within an entire watershed. Alternatively, the system approach to hydrological investigations strives to develop explicit relationships between parameters that can be determined from direct measurements in the watershed. Therefore, mathematical models to describe watershed hydrology can be derived from physically-based mathematical methods or by parametric or stochastic methods of system investigation (Freeze and Harlan 1969).

All rainfall-runoff models can be applied in order to model streamflow processes that are influenced by different runoff sources (Wang 2006). Streamflow processes are usually dominated by rainfall events. However, also other streamflow processes resulting from the drainage of the groundwater storage or other delayed sources, like snow melt, play an important role (Wang 2006).

In the last decades, these basic modelling concepts have been continuously developed by integrating physically-based process concepts into the model formulations (Kratzert et al. 2018). Some of these concepts encompass directly addressing the spatial variability of processes, boundary conditions and physical properties of the catchments (Kratzert et al. 2019a). There is a variety of available methods - based on the aforementioned concepts - to model streamflow, which may fall into two general categories: process-driven methods (also known as white-box, physical or conceptual models) and data-driven methods (also called black-box, meta-models or surrogate models) (Wang 2006). Process-driven models, or physically-based models, can further be divided into lumped, semi-distributed and distributed models (Wang 2006).

In process-driven models, rainfall-runoff processes are described by combining physically-based equations, empirical relationships and parametric assumptions (Wang 2006). This type of models abstract a streamflow process as the output of a watershed system, by mathematically approximate the internal processes within a catchment that guide the streamflow behavior (Wang 2006). These internal processes involve two fundamental sub-processes: rainfall-runoff transformation and channel routing. Thus, a typical physically-based conceptual rainfall-runoff model is composed of two modules. One of them simulates the process of rainfall to runoff transformation; the other module simulates the process of channel routing (Wang 2006).

The result of a rainfall-runoff model is a hydrograph in response to a precipitation event and to the internal storage leakages of the catchment. Although physical models show great capabilities for predicting a diverse range of flooding scenarios, they often require various types of hydro-geomorphological monitoring datasets (Mosavi et al. 2018). Furthermore, the development of physically-based models often requires in-depth knowledge and expertise regarding hydrological processes, since internal model parameters need to be cautiously calibrated.

By contrast, data-driven methods are fundamentally black-box methods, which mathematically characterize the relationship between the inputs and the outputs, without considering the internal underlying physical mechanism in the watershed (Wang 2006). Hence, data-driven models have the advantage of representing almost indefinitely complex processes (Wang 2006). What is more, data-driven models can also sometimes reveal useful information on the dynamics of the subject of analysis. For example, Machine Learning (ML) can support with revealing distinct relationships between landcover, climate, soil, and geomorphology (Shen 2018).

Due to the development of modern measuring techniques (e.g. remote sensing), more and more data has become available at a high spatial and temporal resolution. Moreover, because of the expanded computational capabilities of increasingly advanced computer techniques, data driven-model applications in the field of hydrology have grown progressively within the last decades (Wang 2006). Furthermore, the growing amount of available discharge data in combination with the enhanced development of more sophisticated ML algorithms utilized in data-driven models have accelerated the improvement of rainfall-runoff models (Seo et al. 2018).

1.1. Machine Learning Basics

Over the past decades data availability increased explosively, especially Earth System Data (Reichstein et al. 2019). Machine learning (ML) plays an increasingly important role in analyzing these large datasets. According to Reichstein et al. (2019), the major tasks in the upcoming years are extracting knowledge from these datasets and utilizing models that are able to learn much more from the data than traditional data assimilation methods, while still complying with nature’s laws. As the name suggests, data-driven methods try to learn complex correlations within large datasets (Cerqueira et al. 2019). Thus, these type of methods can be classified into symbolic approaches (rules, trees, and logical data representations) and statistical approaches. The latter include methods such as k-nearest neighbors, Bayesian methods, Neural Networks (NN), and Support Vector Machines (SVM), among others (Fotovatikhah et al. 2018). Unlike other statistical approaches, ML is able to extract new, nontrivial information from large, complex databases, for which classical statistical analysis – such as Bayesian methods – would be impractical (Chollet 2018). As mentioned in Goodfellow et al. (2016), ML can be broadly categorized into supervised learning and unsupervised learning. The former is trained to predict some observed target variable, either categorical or continuous, given some input attributes. Alternatively, the target variable is also called dependent variable or labeled data. Supervised learning is suitable for classification or regression tasks. During classification procedures, the model tries to identify, which previously defined categories a new instance belongs to, for example, whether an image shows a cat or a dog. In case of regression tasks, a continuous prediction of the target variable is produced (Géron 2019).

In contrast, in unsupervised learning environments, a target value is not given to the model. Unsupervised learning is used in descriptive data analysis, i.e. the model seeks to learn how to represent input data efficiently and meaningfully by looking for hidden patterns or clusters (Goodfellow et al. 2016). Unsupervised learning is thus often employed for dimensional reduction or feature extraction, because – after removing less important details – the remaining key bits of information are the most important ones to characterize the data (Shen 2018).

One main reason for the popularity of ML models in hydrology is their ability to numerically formulate the flood’s non-linearity, solely based on measurement data without requiring knowledge about the underlying physical processes (Mosavi et al. 2018). Data-driven prediction models using ML are promising tools, as they are easier to develop with minimal inputs. Another advantage over physical models is, that ML models often provide a more straightforward implementation due to reduced complexity (Mosavi et al. 2018). Thus, ML models offer faster testing and evaluation without the usually intensive process of calibration. Within the last two decades, the continuous advancement of ML methods demonstrated their suitability for flood prediction by outperforming conventional approaches at an acceptable rate (Mosavi et al. 2018). Numerous studies have compared the prediction performance between physical and ML models, showing at least equal performances of ML methods.

1.2. Machine Learning in Hydrology

This thesis focuses on a special subfield of ML, called Deep Learning (DL). DL methods aim at learning representations from data with multiple levels of abstraction by focusing on learning successive layers of increasingly meaningful representations (Hu et al. 2018). All these layers learn automatically when exposed to training data. In almost every case these layered representations are gathered via models called Neural Networks (NN), structured in stacked layers (Chollet 2018). According to Shen (2018), some potential applications include modeling land management decisions in response to water constraint or flooding risks, irrigation and consumptive water demands, water saving strategies, ecosystem responses, and urban water flows. For these problems, a process-based model could involve too many interdependent processes and parameters without clear ways of obtaining reliable values. Both, the mathematical structure and the boundary conditions of these problems, as well as the relationships between model-internal variables could also be too complex and interdependent to fully resolve (Shen 2018).

In hydrology, there are three types of DL applications so far: (i) extracting hydrometeorological and hydrological information from images, (ii) dynamic modeling of hydrologic variables or data collected by sensor networks, and (iii) learning and generating complex data distributions (Shen 2018). A common conclusion reported in these studies is that, as could have been expected, DL models have surpassed traditional statistical methods (Shen 2018). One study investigated how the uncertainty of flood predictions can be reduced to support early flood warning systems (Doycheva et al. 2017).The study assessed the correct weighting of meteorological ensemble forecast members with the aim to incorporate them in rainfall-runoff simulations performed by three different ML methods in the Mulde river basin in Germany.

SVM, NN and Rotation Forest were used as ML methods, while the latter method showed the best performance in classifying the best ensemble of member pairs. Thus, the uncertainty range in flood risk prediction was reduced and the overall performance of the ensemble system was improved (Doycheva et al. 2017).

Another study used Artificial Neural Networks (ANN) for accuracy enhancements in real-time flood forecasting scenarios. Coupling Numerical Weather Prediction (NWP) and hydrological models allows a connection between meteorology and hydrology to generate real-time flood forecasting (Jabbari and Bae 2018). The aim of the study was to evaluate real-time bias correction of precipitation data from a hydro-meteorological point of view. Jabbari and Bae (2018) used a mesoscale and operational NWP model to forecast heavy rainfall predictions (72h ahead, every 6h) at high spatial (1km x 1km) and temporal (10 min) resolution in real-time. Additionally, in order to consider rainfall loss and to stimulate the streamflow, a soil moisture accounting model was applied. Finally, the ANN model was utilized for bias correction of the real-time precipitation produced by the NWP model, which resulted in a significantly improved rainfall forecast in all statistical terms and in an overall improved real-time flood forecast in the Imjin basin in North/South Korea (Jabbari and Bae 2018).

A similar study used DL to develop an effective weather-runoff forecasting system to support real-time decision making in the Metropolitan Region of Chile (La Fuente et al. 2019). For this purpose, a coupled model of a near-future global meteorological forecast with a short-range runoff forecasting system was implemented. The hydro-meteorological and geomorphological variables were obtained through statistical scaling of the Global Forecast System (GFS), thus enabling near-future prediction. Furthermore, two data-driven approaches were implemented to predict the entire hourly flow time series up to three days into the future all at once. The first model was a simple ANN and the second approach was based on Long Short-Term Memory (LSTM) cells, which outperformed the prediction performance of the ANN by far, since the temporal capacity of remembering sequences for long periods of LSTM-based algorithms allows the more accurate prediction of temporal changes in the flow. The study concluded that the whole system is capable of predicting the streamflow in a finer temporal resolution (1h) than the input time-series (6h) with high accuracy enabling a reliable flow forecast (La Fuente et al. 2019).

The main disadvantage of the “black box-ness” of NNs is their lack of ability to adhere the mechanistic understandings of the underlying physical processes in hydrological modeling. To circumvent this issue, Karpatne et al. (2018) introduced a novel framework, in which a NN is coupled with a physic-based model. The so-called physics-guided NN uses simulated outputs of a physic-based numerical model as one additional input. Additionally, a physic-based objective function achieves better generalization performance by ensuring physical consistency of model outputs. Karpatne et al. (2018) applied the model to simulate the temperature of water in a lake at varying depths and time. It was shown that this model is in fact closer to the actual observations than SVM, normal NN and other non-linear regression models, always revealing physically consistent results (Karpatne et al. 2018).

There are further studies that developed hybrid models: Tian et al. (2018) applied a lumped hydrological model (GR4J model) in the Xiangjiang and Qujiang River basins in China to conduct a rainfall-runoff simulation. Further, four modifications of a standard Recurrent Neural Network (RNN) were applied to predict discharges. These are namely: the Elman Recurrent Neural Network (ERNN), Echo State Network (ESN), Nonlinear Autoregressive Exogenous inputs neural network (NARX) and a LSTM network. After a performance assessment, the best two RNNs (LSTM and NARX) and the integrated GR4J model were selected to forecast streamflow. The performance of the GR4J model coupled with an LSTM model showed best performance and reduced the uncertainty intervals further, especially in high flow conditions (Tian et al. 2018).

In general, it seems that LSTMs outperform normal ANNs in almost every experimental setting for rainfall-runoff simulation. Hu et al. (2018) confirmed this in a paper, where 98 flood events in the Fen river basin in China were simulated on an hourly basis by ANNs and LSTMs models. In contrast to conceptual and physical-based models, these black-box models worked well, simulating rainfall-runoff processes with excellent performance evaluation criteria. However, the LSTM model outperformed the ANN model considering different lead times and better stability, holding better overall simulation performance (Hu et al. 2018).

The most intensive research so far, reflecting the applicability and interpretability of LSTMs in the context of rainfall-runoff simulations was carried out in various studies by Kratzert et al. (2018, 2019a, 2019b, 2019c, 2020). In the first study, Kratzert et al. (2018) explored the potential of LSTMs to describe the rainfall-runoff behavior and further investigated the ability of LSTMs of regionalization. In this work, the authors trained LSTM models to predict the streamflow for 241 catchments across the Continental United States (CONUS) and compared the results to the Sacramento Soil Moisture Accounting Model (SAC-SMA) as a benchmark. Their results show that LSTMs are able to predict runoff (on daily resolution) solely from grided meteorological observation data (averaged daily precipitation, solar radiation, min/max air temperature, vapor pressure) with accuracies comparable to the benchmark model. Moreover, the authors suggest that pre-trained knowledge can be transferred into different catchments, possibly reducing the data demand and implying the applicability of these models in ungauged basins (Kratzert et al. 2018).

Kratzert et al. (2019a) also investigated the question, if there is some sort of correlation between the internal memory cells of a LSTM network and the hydrological states (snow-water equivalent, upper zone storage and lower zone storage) in the catchment. They compared two different catchments, one that is snow-influenced and one that is not influenced by snowmelt. Thus, they were able to show that the hydrological understanding of preceding days influencing the runoff signal at specific succeeding days matches quite well. In general, the detailed interpretability of all internal memory cell states of an LSTM was limited, but nevertheless, some cells showed high correlation to other hydrological system states, indicating a strong relation between them. This suggests that the LSTM realistically represents short- as well as long-term dynamics in snow cell storage in the catchment. Overall, the processes learned by the LSTM matches the hydrological comprehension of the real-world environmental system, thus taking away some of the "blackbox-ness" (Kratzert et al. 2019a).

In another paper, Kratzert et al. (2019c) demonstrate that – by the use of large-sample hydrology data – a regional LSTM rainfall-runoff model that attaches importance to additional data in form of catchment attributes produces more accurate streamflow estimates compared to several existing hydrologic models. This was achieved by proposing a new form of LSTM, in which a static set of catchment characteristics is used as additional input data to provide the network with some information for discriminating between different catchments. The newly introduced entity aware LSTM links catchment characteristics to the dynamics of specific sites and is able to learn from the combined data of all catchments. In terms of performance, the regionally trained entity aware LSTM outperformed not only the regionally calibrated benchmark models but also the models calibrated separately for individual catchments.

The biggest advantage though is the possibility of a level of interpretability about how the model learns to differentiate between different catchment-specific behaviors (Kratzert et al. 2019c). To corroborate their findings so far and to show that they can leverage the aforementioned capability of LSTMs concerning prediction in ungauged basins (PUB), Kratzert et al. (2019b) demonstrated a ML strategy for PUB in a technical note. Their results show that – in ungauged basins (on average) – out-of-sample LSTMs outperform both a conceptual model (SAC-SMA) calibrated independently for each catchment, as well as a distributed, process-based National Water Model (NWM). Further, they show that the LSTM is able to extrapolate to new catchments from catchment attributes, advocating the hypothesis that such ML methods can synthesize information from multiple sites and situations into a single model Kratzert et al. (2019b).

In a recent study, Kratzert et al. (2020) investigated the ability how LSTMs can leverage different precipitation products in a spatio-temporally way to improve discharge predictions.

This is interesting in view of the fact that many hydrological models usually require gridded data fields, which are products gained from spatial interpolation and/or satellite retrieval algorithms. Those are, in turn, based on different sets of assumptions potentially introducing different types of error and information loss (Kratzert et al. 2020). In their experiments, Kratzert et al. (2020) showed that there are systematic, location- and time- specific differences between different precipitation products. However, the proposed LSTM trained simultaneously on multiple precipitation products, outperformed all LSTMs trained on single/pairwise precipitation products and all classical hydrological models. Hence, LSTMs are able to learn and leverage these differences mitigating uncertainty in data products (Kratzert et al. 2020). Overall, the results imply that DL models not only have the capability to use a larger number and a broader variety of input data than classical hydrology models, but are also able to learn spatio-temporally variable interactions between distinct multi data products (Kratzert et al. 2020).[1]

In the most recent paper published by Gauch et al. (2020), they examined whether LSTM-based modeling generalizes to multiple timescales, since flood forecasting requires subdaily predictions. They introduced a specific configuration of LSTMs, called multi-timescale LSTM (MTS-LSTM), which is able to generate daily predictions and the corresponding 24 hourly predictions during the same model run. The key insight of this model is that the hourly prediction phase utilizes the internal LSTM states from the daily prediction phase, which act as a summary of long-term information up to that point. This setting emphasizes the idea that for rainfall-runoff modelling the history of total mass and energy inputs to the watershed are in fact important. However, the impact of high-frequency variations becomes less significant at long lead times (Gauch et al. 2020). By using this specific configuration Gauch et al. (2020) are able to provide multiple output timescales through processing long-past input data at coarser temporal resolution than recent time steps. As a result, the size of the input sequences can be substantially shortened, because high-resolution inputs are only necessary for the last few time steps. Testing the newly introduced LSTM model configurations, the results suggest that the shared state between daily and hourly LSTM predictions holds as much information as the naïve model (trained only on hourly resolution data). Besides that, this MTS-LSTM achieves a performance beyond the best single forcing models. One small drawback of the MTS-LSTM is that, at daily and hourly target timescales, it predicts 24 hourly steps once a day. For general analysis settings this might be sufficient, but in an operational forecasting setting, it is essential to generate these 24 hourly prediction steps more frequently within one day (Gauch et al. 2020).

---

[1] Paper is still open for discussion (last checked: 12.11.2020)

1.3. Objectives and Outline

Overall, the previous section shows that a diversity of ML methods has already been applied across several major subdomains of hydrology. The focus is on LSTM models and their potential modifications as they have been increasingly integrated into operational schemes and used to discover patterns, to improve our understanding and to evaluate traditional physical-based models in terms of rainfall-runoff forecasting. In general, the most accurate precipitation data are collected by point-based measurement stations within a catchment, which are in situ gauges to detect complex spatial processes (Kratzert et al. 2020).

However, there is a significant problem in this context: measurement stations hardly ever capture observational data on a complete continuous timescale due to e.g. system failures or maintenance suspensions. This leads to commonly missing data, so called data gaps, in the available measurement datasets. However, most hydrological models typically require a complete time series of observation data. Thus, data gaps have to be filled during preprocessing, which potentially can be a time-consuming issue. It would be advantageous to have a type of model that is able to actually deal with missing values in a dataset in an unrestrained way.

Furthermore, process-based models often require information on physical characteristics (e.g. soil characteristics) of the catchment as inputs, which are frequently missing and highly heterogeneous in space and time. Hence, these models are determined by spatially and temporally distributed system states and physical parameters, which are often unknown (Kratzert et al. 2019a). Moreover, hydrologic information is required at different timescales depending on the application area. Flood forecasting needs streamflow predictions in a considerably higher temporal resolution than for example reservoir management for hydropower operation. Yet, as described in the previous section, much of the research of DL in runoff prediction has been conducted at a daily timescale. All the aforementioned aspects lead to several objectives that are investigated in this thesis, which are the following:

- It is examined if a LSTM model achieves reasonable accuracy in rainfall-runoff predictions based on raw point-based observation data (non-gridded data) in the Regen catchment in Germany with respect to both, daily and hourly temporal data resolution. For this purpose, the performance of the LSTM model is compared to a conceptual physical-based model (LARSIM), which is used in flood forecasting operation throughout Germany and Switzerland.
- Then, the LSTM model performance is optimized regarding the scaling of the input features, the imputation technique of missing precipitation data and the used hyperpa-

rameter configurations. Regarding the latter, the focus is on tuning specific hyperparameters, that are likely - according to the author's impression - to be a decisive factor for discharge prediction from a hydrological perspective.

- Further, three different strategies are explored, in order that the LSTM model can cope with incomplete datasets. One of these strategies explores the idea of a model that could learn from missingness patterns in the data.
- Next, it is investigated in how far the LSTM model is sensitive to the type and volume of input data, or, in other words, how the model performs on different input feature sets. More precisely, these input feature sets are supposed to independently test the influence of specific parameter groups on the model performance. This analysis includes the scenario in which the LSTM is trained exclusively on meteorological forcing data to examine if the LSTM can model rainfall-runoff relations within the catchment without any prior knowledge of discharge.
- To approach a real-world scenario of operational flood forecasting, tests of predicting discharge at multiple (hourly) time steps ahead are conducted. It is investigated whether the LSTM model is able to forecast streamflow with different lead times with reasonable accuracy – compared to the physical-based LARSIM simulation.

In the beginning, an introduction to DL is given with emphasis on the conceptual ideas of ANNs and LSTMs along with the fundamental aspects of using LSTM models in the context of multivariate streamflow forecasting. In the next section, the methodology of preprocessing the datasets with respect to data cleaning, scaling methods and imputation techniques is presented, followed by a detailed description of the used LSTM model architecture. On top of that, the evaluation strategies for assessing the performance of the LSTM model and a short summary of the concept of the LARSIM model are provided. In the next step, the results are presented and discussed in detail. Finally, suggestions are made – on further research directions and opportunities for even more advanced investigations.

2. Deep Learning Fundamentals

Deep Neural Networks (DNNs) have proven to be quite successful in recognizing complicated patterns in the field of image processing, language models, handwriting recognition and sequential data being applied in complex real-world systems (Ogunmolu et al. 2016). Among these, three general groups of NNs, which have been frequently applied in the aforementioned fields, are: (1) Multilayer Perceptrons (MLPs), (2) Recurrent Neural Networks (RNNs) and (3) Convolutional Neural Networks (CNNs). These types of networks are referred to as "deep", since they are created by accumulating multiple layers of non-linear operations on top of each other (Chollet 2018). On the one hand, MLPs, also called Artificial Neural Networks (ANNs), have been applied to examine static and dynamic simple non-linear systems; on the other hand, RNNs and their variants have been used to analyze timeseries or sequential optimization problems. (Ogunmolu et al. 2016). In this section, the most important DL fundamentals with their underlying principles regarding ANNs and recurrent neural networks are examined. Furthermore, it is explained which steps are essential for developing a multivariate forecasting model, when utilizing RNN structures for discharge predictions.[2]

2.1. Introduction to Artificial Neural Networks

ANNs are models that are data-driven and work with a flexible mathematical structure, which enables them to identify complex non-linear relations between input and output datasets, while not requiring an understanding of the phenomena's nature (Wang 2006). Thus, ANNs use a supervised learning approach to approximate functions by interconnecting units called neurons across various layers. The network consists of three different types of layers: input, hidden and output layer, which are all composed of an arrangement of neurons (Chollet 2018).

2.1.1. Anatomy of a Neural Network

NNs are typically called "feedforward", because the information flows through the network in only one direction, from the input layer to the output layer. There are not any feedback connections (cycles or loops) between the layers, in which outputs of the model are fed back into itself. An example of a multilayer network is shown in Figure 1. NNs are referred to as fully connected, as every neuron on one layer is connected to every neuron on the adjacent layer. In principle, NNs try to learn deterministic mappings from an input to an output (La Fuente et al. 2019). The training process consists of finding the weights and biases that will minimize an objective loss function between the network predictions and the corresponding targets (Goodfellow et al. 2016). The input and output layers have direct connections to inputs and outputs and thus are called visible layers – as opposed to hidden layers in the middle. The number of neurons in one layer is called width, while the number of layers is called the depth of the network. Hence, a deep network implies a large depth dependent on the complexity of the problem (Géron 2019).

---

[2] All the proposed conceptual principals and fundamental ideas described in the sections 2.1 to and including 2.2.1 are predominantly summarized based on Chollet 2018, Géron 2019 and Goodfellow et al. 2016. For further and more in-depth information, please refer to them.

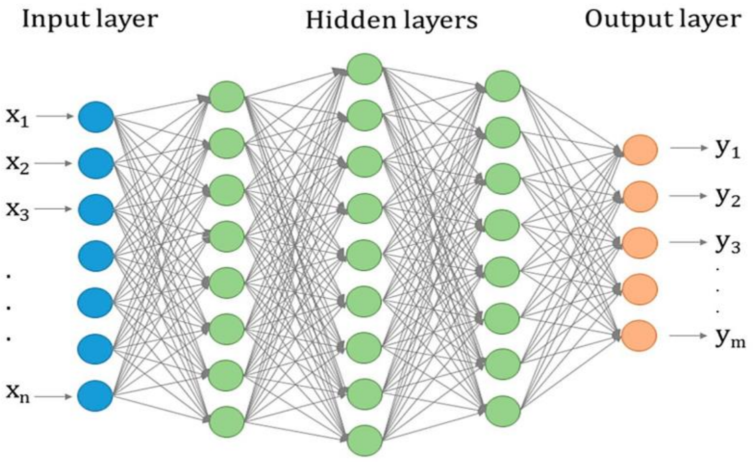


Figure 1: Scheme of an Artificial Neural Network (La Fuente et al. 2019)

The units of connection are called “neural”, because they are inspired by neuroscience. The human brain is composed of nerve cells called neurons that are connected with each other by axons. Analogous to this, ANNs are composed of multiple nodes, which imitate the biological neurons of a human brain. In Figure 2 an illustration of a standard artificial neuron within a feedforward network is shown.

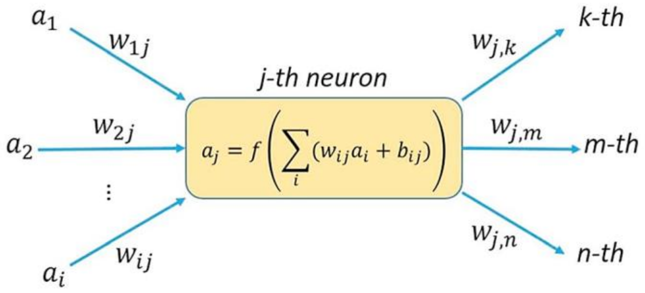


Figure 2: Illustration of a neuron in a feedforward network. The j_{th} neuron receives inputs from neurons (a_1, a_2, … , a_i) of the preceding layer multiplied by their respective weights and applies a nonlinear transformation (*f*). The output of the neuron, called activation, is passed to the neurons of the succeeding layer. (Shen 2018)

Since each hidden layer in the network is typically vector-valued, the layer can be described as not representing a single vector-to-vector function, but rather as consisting of many interconnected neuron units that act in parallel; each representing a vector-to-scaler function. Each unit resembles a neuron in the sense that it receives input from many other units and computes its own activation value (Goodfellow et al. 2016). According to Shen (2018), a neuron on layer ***L*** receives inputs from the input layer or from multiple neurons on layer ***L-1**,* where all the inputs are stored in vectors or tensors; i.e. the ***j^{th}*** neuron receives inputs from neurons (*1, 2, … , i)* of the upstream layer. The connections are represented by linear weights between neurons, e.g. ***w_{ij}*** denoting the connection between the ***i^{th}*** neuron and the ***j^{th}*** neuron, which are in layer ***L-1*** and ***L***, respectively. Further, to each neuron, a bias parameter ***b_{ij}*** is added (Shen 2018). According to Goodfellow et al. (2016), a forward pass of input data through the network can be described stepwise as follows: The input tensors to an artificial neuron contain either raw data values or may also comprise outputs from preceding artificial neurons (e.g. in hidden layers), which are then multiplied by their respective weights. Afterwards, the bias parameter is added and a linear or non-linear transformation θ is applied to the linearly combined inputs of all preceding connections. This transformation function squashes the linearly combined inputs to produce a desired bounded and constant non-linear output. The output of a neuron, called its activation, is then sent to the succeeding layer.

Common non-linear activation functions used in practice include the logistic sigmoid function, the hyperbolic tangent function (tanh), or the point-wise Rectified Linear Units (ReLU) (Géron 2019). The logistic sigmoid function is relevant to functions that map into probability output spaces [0,1], while the secondly mentioned, hyperbolic tangent function, maps to the output range $[-1, 1]$. The output of the latter, the ReLU function, however, is equal to the input if the input is greater than zero or zero otherwise. It has the advantage of being easier to optimize and providing faster convergence in network (Ogunmolu et al. 2016). After a forward pass, the corresponding network weights are updated by the use of a specific algorithm called *Backpropagation*. This tuning process, which “trains” the network, is described in more detail in the next section. To summarize it in an abstract way: By consolidating a large number of neurons, a nonlinear function is obtained, which uniformly approximates the continuous function between the input and the output space to an acceptable bounded error (Ogunmolu et al. 2016).

2.1.2. Training, Backpropagation and Optimization

In the training process, the weights and biases of a network are adjusted to approximate an optimal solution to a problem. For this optimization procedure, i.e. during training, a quality measure for the performance of the neural networks is needed to compare the ground truth (label/target) with the prediction of the network and to calculate the error signal in the network. Optimization refers to the task of either minimizing or maximizing some function f(x) by altering x (Goodfellow et al. 2016). The measure of performance that is used in neural networks is called objective function. This function should typically be minimized and is thus often referred to as the cost function, loss function, or error function.

The different procedure steps of training inside a neural network are illustrated in Figure 3 in a simplified form. The different layer transformations in the network are mapping the input data to predictions. The loss function then compares these predictions to the true targets, producing a scalar loss value. The optimizer, which determines how the network will be updated based on the loss function, uses this loss value to update the network's weights (Chollet 2018).

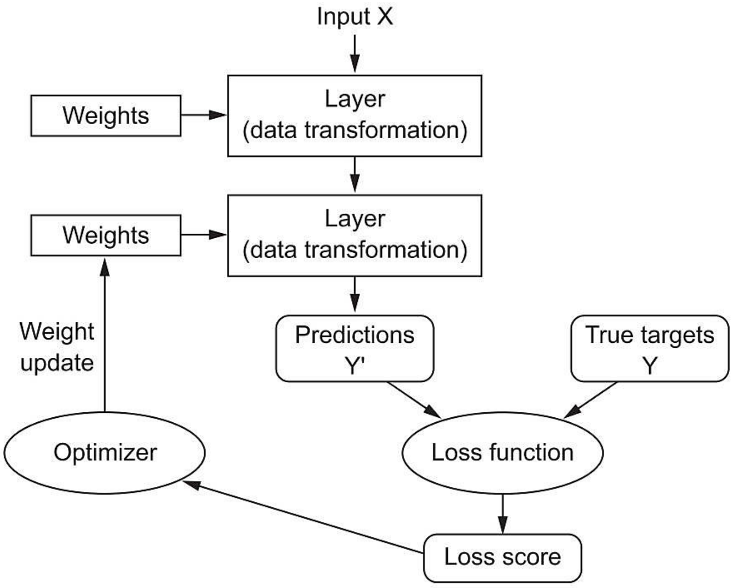


Figure 3: Schematic flow chart of a full training cycle in a Neural Network (Chollet 2018)

According to Goodfellow et al. (2016), a linear model, mapping from features to outputs via matrix multiplication, can by definition represent only linear functions. As a consequence, many loss functions, while applied to linear models, result in convex optimization problems. This has the advantage that linear models are easy to train since for convex functions, any local minimum is guaranteed to be a global minimum (Goodfellow et al. 2016). However, the nonlinearity of a neural network causes most loss functions to become non-convex in an n-dimensional space and its error surface is often chaotic and possesses several local minima (Goodfellow et al. 2016).

In order to adjust the weights, the following approach is commonly used. At first, the weights for feedforward NNs are initialized to small random values, whereas the biases may be initialized to zero or to small positive values. This defines the starting point for the optimization. The gradient is then calculated for this point. The gradient gives the direction of the steepest descent on the error surface and a step is taken in this direction. Hereafter, the gradient is estimated again and a new point is determined in the direction of the gradient. These steps are repeated until (ideally) a global minimum is reached. The algorithm for finding the minimum is called gradient descent (Goodfellow et al. 2016). With the gradient descent algorithm described before, only the weights of the single neurons in the last layer are adjusted. However, the neurons in the hidden layers have no value, which can be compared to the ground truth. For this reason, the so-called backpropagation algorithm is used, which is a form of gradient descend method that propagates the loss as a signal into the network and updates network weights of the hidden layers based on the chain rule (Chollet 2018). The algorithm computes the error derivatives, which depend on the activation of the single neurons in each layer through the whole network, starting from the output layer. From this, the influence of a single weight on the global error can be concluded. With the backpropagation algorithm, the gradient-descent based optimization can be performed for large networks.[3] Compared to other optimization methods, such as evolutionary algorithms, backpropagation is highly efficient thanks to the differentiability and linear nature of the network connections (Ogunmolu et al. 2016). Since gradient descent applied to non-convex and complex error spaces has no convergence guarantee, the algorithm is sensitive to initial parameter values and thus is not able to always reach the global minimum. This means it can get stuck on local minima. In order to prevent these drawbacks, specific variants of training algorithms exist, which improve or refine the use of the gradient to minimize the loss function in any possible way. The version of the gradient descent discussed before is called batch gradient descent (Goodfellow et al. 2016). In this approach, the error is calculated with the whole dataset. The model is updated after all the training examples have been evaluated.

As the data set size grows very large, the time to take a single gradient step becomes prohibitively long. In order to reduce training time, the stochastic gradient descent (SGD) algorithm can be used. The perception of stochastic gradient descent is that the gradient is an expectation, which is approximately estimated using a small set of samples, partitioned into batches of definable size. These batches of samples, typically chosen to be a relatively small number of samples ranging from one to a few hundred, are also called minibatches, and they are drawn uniformly from the training set (Goodfellow et al. 2016). For each minibatch, an approximation of the gradient is computed, and the weights of the model are updated.

One cycle across all batches in a training dataset is called epoch. All batches are evaluated once in an epoch, so the weights and biases are adjusted in each epoch $\frac{numer\ of\ samples}{batch\ size}$ times, before the whole training process starts all over again (Chollet 2018). Moreover, the efficiency of batch gradient descent is combined with preventing getting stuck in a local minimum and minimizing training time by the SGD algorithm (Géron 2019). In this thesis, an algorithm called *Adagrad* is used for optimization; it is described in section 3.5.

---

[3] A detailed mathematical description of the backpropagation algorithm can be found in Goodfellow et al. 2016.

2.1.3. Evaluation and Regularization

The central challenge in ML is that we must perform well on new, previously unseen inputs — not only those that our model was trained with. The ability to perform well on previously unobserved inputs is called generalization (Chollet 2018). Thus, not only the error on trained data, but also the error on new input data, called generalization error, should be as low as possible.

Typically, the used dataset is divided into three parts: (1) training dataset, (2) validation dataset and (3) test dataset. The training set is used to adapt the weights and biases during the training process. Besides the weights and biases, NN models have several settings that can be used to control their behavior during training and define the training process of the networks. These settings are called hyperparameters and their values are usually not adapted by the learning algorithm itself. Furthermore, if the model were able to learn the hyperparameter on the training set, it would always choose the maximum possible model capacity, which would lead to overfitting (Goodfellow et al. 2016). Thus, the hyperparameters have to be adjusted before training. In DL, the model's capacity is often referred to as the number of learnable parameters, which is determined by the number of layers and the number of units per layer.

To solve this problem, a validation set of examples is needed, which the training algorithm does not observe (Goodfellow et al. 2016). Hence, a validation subset is used to evaluate the error during training on that validation data. With a subroutine, the hyperparameters can be tuned accordingly to the validation error. In order to tune the hyperparameters, there are different approaches. A popular approach is to use a grid search, in which sets of various combinations of hyperparameters are tested and the best one is chosen.

The third subset of the complete dataset, the test set, is used for the final model evaluation. It is of utmost importance that the test examples are not used in any way to make choices about the model, including its hyperparameters. This is crucial to avoid that the model gets any information about the data, on which its generalization ability should be tested. For this reason, no sample from the test set can be used in the validation set. Therefore, the validation set is always constructed from the training data (Chollet 2018). During the final evaluation of the model on the test set, the generalization error is estimated. This error, also named testing error, refers to errors incurred when the algorithm is applied to new data, neither included in the training nor in the validation set [Goodfellow et al. 2016, Géron 2019]. During the training process, the expected validation error is usually greater than or equal to the expected value of training error. Generally, there are two factors determining how well a NN model performs. On the one hand, the training error should be reduced as much as possible. On the other hand, the gap between training and validation error should be considerably small as well. These two factors correspond to the two central challenges in ML: underfitting and overfitting. The former occurs when the model is not able to obtain a sufficiently low error value on the training set, whereas the latter occurs when the gap between the training error and the test error is too large (Goodfellow et al. 2016). As mentioned before, an epoch corresponds to one full training cycle, in which the weights and biases are adapted on the training dataset. As shown in Figure 4, in the beginning of training, optimization and generalization are correlated.

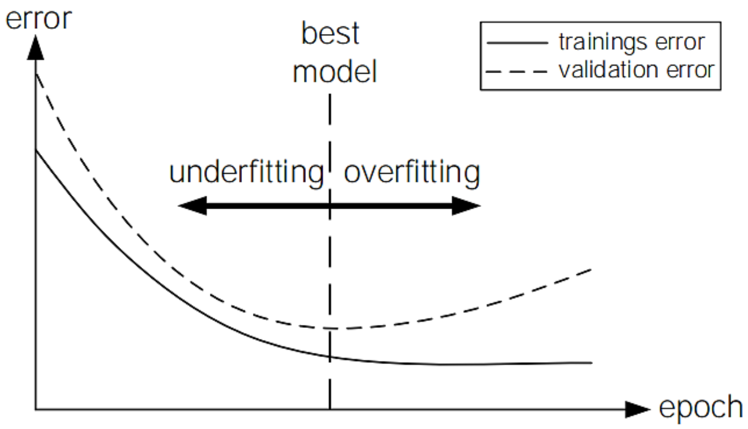


Figure 4: Training History of a Neural Network, showing the learning curves on the training and validating dataset according to (Géron 2019)

The training error as well as the validation error usually decreases during training (along the epoch number). While this is happening, the model is said to underfit. This means the network has not yet modeled all relevant patterns in the training data. However, after a certain epoch generalization stops improving. At that point, the validation loss begins to degrade, which implies that the model is starting to overfit. Overfitting is the result of the model describing a particular dataset too accurately. This intents that the model is beginning to learn patterns that are specific to the training data, but that are misleading or irrelevant when it comes to new data, so it fails to generalize (Chollet 2018). The optimal number of epochs – after which the training is stopped – should be chosen in a way that, on the one hand, the model converges and, on the other hand, overfitting does not take place.

The procedure in order to reduce or prevent overfitting and to reduce generalization errors is called regularization. For an effective regularization, several techniques have been developed. The best solution is to get more training data, since a model trained on a broader dataset will naturally generalize better (Chollet 2018). However, the availability of data is often limited. Then, the second-best solution is to modulate the quantity of information that the model is allowed to store or to add constraints to the learning process. To accomplish this, the simplest way is to reduce the size of the model, i.e. the number of learnable parameters in the model (Chollet 2018).

A deep model with more parameters has more capacity and therefore it can – with its higher degree of freedom – easily learn a virtually perfect mapping between samples and their targets. Nevertheless, if the network has limited memorization resources, it will not be able to learn this mapping effortlessly. Hence, the model capacity should not be reduced too much to avoid an underfitting of the model. A compromise between too much capacity and not enough capacity must be found. Another method to mitigate overfitting is to constrain the complexity of the model by forcing its weights to generate a more regular distribution of weight values, which is called weight regularization. This is achieved by adding a cost to the objective function, which is associated with having large weights (Chollet 2018).

Another frequently used strategy to cope with high generalization errors are ensemble methods. This strategy employs the idea of training a few different models independently and then averaging the results on test examples (Goodfellow et al. 2016). In general, NNs can have multiple model outcomes even if trained on the same dataset, which is due to random initialization, random shuffled batches and/or disparity in hyperparameters (Goodfellow et al. 2016). According to Kratzert et al. (2019b), this implies that a certain amount of uncertainty in these models derives from randomness, rather than from systematic model structural error. Therefore, evaluating models by averaging across several model runs is a robust and proven approach for reducing generalization error (Goodfellow et al. 2016).

Nevertheless, the most commonly used regularization technique is called dropout. The method randomly blocks a fraction (defined by the dropout rate) of the connections in a network. This operation is implemented via a randomly generated mask with the same dimension as the connections between two layers (Chollet 2018). This mask is 0 for the blocked connections and 1 for those kept open. The dropout is only implemented during training, while during testing the output values of the layer are scaled down by a factor that is equivalent to the dropout rate. This is done in order to balance the surplus of active units during testing compared to training time (Chollet 2018). Dropout introduces some noise into the optimization, which improves robustness (Shen 2018). Another commonly used regularization is early stopping, whereby the training of the model is stopped before it fully achieves its best performance and the validation error reaches the minimum (Géron 2019).

2.2. Deep Learning for Time Series

Despite the advantages of ANNs, there are a number of drawbacks associated with using them. These types of networks are not exactly adequate for the analysis of sequential data (Ogunmolu et al. 2016). In order to process sequential data and to make predictions, it is necessary that the model is able to memorize previous states of the system, thus taking advantage not only from the present information but also from previous states (La Fuente et al. 2019). However, ANNs do not have that concept of temporary storage integrated in their model structure. That is why it is difficult for them to perceive temporal dynamics in sequential data. Thus, their ability to approximate sequential data may not be robust, which is reflected in a greater error when predicting floods and therefore they tend to underestimate the flow (La Fuente et al. 2019). One very successful method to circumvent the mentioned shortcomings is the use of Long-Short Term Memory (LSTM) cells, which are based on Recurrent Neural Networks (RNN) (Hochreiter and Schmidhuber 1997). This specific type of model architecture is explained in the following section in more detail, since it was applied in this thesis.

2.2.1. Long-term Short Memory Networks

RNNs are one of the most powerful types of NNs, capable of processing sequences of arbitrary input patterns. RNNs incorporate cells with content addressable memory, thus being able to capture entire sequences of information (Hochreiter and Schmidhuber 1997). Simple recurrent neurons are similarly constructed to the units used in ANNs, except that they have a feedback connection for each neuron in the hidden layer(s) (Ogunmolu et al. 2016). In principle, a recurrent neuron can utilize these feedback connections to store representations of previous inputs in form of activations (Hochreiter and Schmidhuber 1997). This principle of one recurrent neuron is illustrated on the left side in Figure 5.

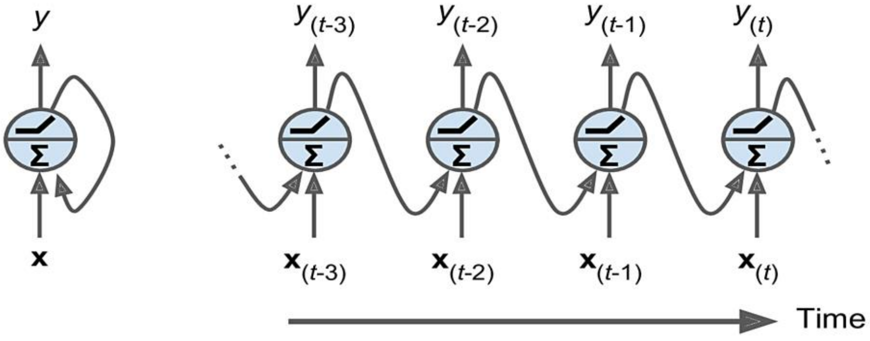


Figure 5: A recurrent neuron (left) unrolled through time (right) (Géron 2019)

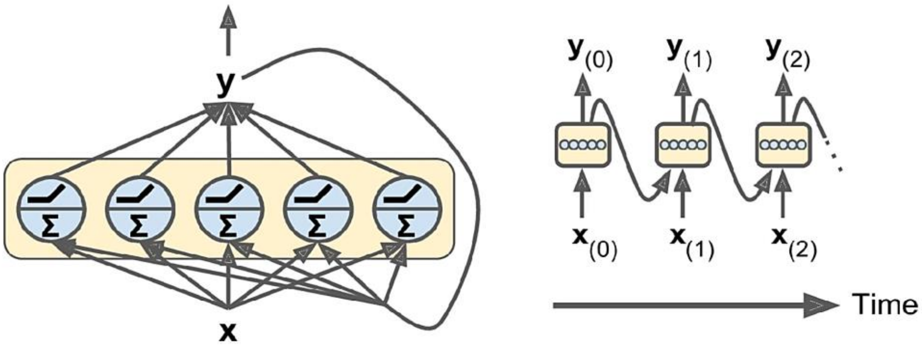


Figure 6: A layer of recurrent neurons (left) unrolled through time (right) (Géron 2019)

At each time step $\boldsymbol{t}$, the recurrent neuron receives the input vector $\boldsymbol{x_t}$ as well as the output from the previous time step $\boldsymbol{y_t}$**.** Unrolling the feedback connection of one recurrent neuron through time, as shown on the right side in Figure 5, the idea of how the network can benefit from a form of memory becomes explicit. With this architecture, the RNN can preserve some states across multiple time steps. One single or multiple of these recurrent neurons can be concatenated in one cell or unit, called memory cell, as depicted on the left side in Figure 6. Again, unrolling these memory cells through time, as depicted on the right side in Figure 6, the RNN is capable of remembering short patterns by passing its memory cell state to the next cell (Géron 2019). On the contrary, the traditional RNN can only learn dependencies around 10 or less time steps (Hochreiter and Schmidhuber 1997). The reason for this is the so-termed “vanishing or exploding gradients” phenomenon, which becomes apparent in an error signal that either diminishes towards zero or grows against infinity during Backpropagation, thus preventing the effective learning of long-term dependencies (Hochreiter and Schmidhuber 1997). Nevertheless, as stated in Kratzert et al. (2018), from the perspective of streamflow modeling, the discharge in a catchment is influenced by various hydrological processes with dependencies well above 10 hours or even well above 10 days (when assuming hourly or daily data basis). Therefore, the memory of standard RNNs with a maximum 10 time steps is too short for considering catchment processes including groundwater recharge, snow accumulation during winter or even glacier storages, with lag times between precipitation and discharge of several months or even a few years. However, to accurately predict the discharge, hydrological models need to somehow replicate these processes as correctly as possible (Kratzert et al. 2018). To overcome the weakness of the limited amount of memory in basic RNN cells, alternative designs of these recurrent cells have been developed. The most promising configuration is called the Long-Term Short Memory (LSTM) cell, which was first proposed by Hochreiter and Schmidhuber in 1997. An LSTM is a special kind of RNN structure, that is deep in time and can learn when to forget or retain the state information of previous time steps. Thus, a specific configuration of operations in this network, so-called gates, control the information flow within a memory cell over long time periods within the LSTM. The gates help the LSTM network to decide what to forget and what to remember, so it avoids error signal decay by keeping the errors in memory (Hochreiter and Schmidhuber 1997). To explain how an LSTM works, it is helpful to look at the unfolded acyclic graph of one LSTM layer (see Figure 7).

Given an input sequence $\boldsymbol{x} = [\boldsymbol{x}[\mathbf{1}], .., \boldsymbol{x}[\boldsymbol{T}]]$ with $\boldsymbol{T}$ consecutive time steps, in which each element $\boldsymbol{x}[\boldsymbol{t}]$ is a vector or tensor containing input features for the model at time step $\boldsymbol{t}$ ($1 \leq t \leq T$), the following equations describe the forward pass through the LSTM to predict the discharge at time step $\boldsymbol{t}$:

$$\mathbf{i_t} = \boldsymbol{\sigma}(\mathbf{W_i x_t} + \mathbf{U_i h_{t-1}} + \mathbf{b_i}) \tag{1}$$

$$\mathbf{f_t} = \boldsymbol{\sigma}(\mathbf{W_f x_t} + \mathbf{U_f h_{t-1}} + \mathbf{b_f}) \tag{2}$$

$$\mathbf{g_t} = \mathbf{tanh}(\mathbf{W_g x_t} + \mathbf{U_g h_{t-1}} + \mathbf{b_g}) \tag{3}$$

$$\mathbf{o_t} = \boldsymbol{\sigma}(\mathbf{W_o x_t} + \mathbf{U_o h_{t-1}} + \mathbf{b_o}) \tag{4}$$

$$\mathbf{c_t} = \mathbf{f_t} \odot \mathbf{c_{t-1}} + \mathbf{i_t} \odot \mathbf{g_t} \tag{5}$$

$$\mathbf{h_t} = \mathbf{o_t} \odot \mathbf{tanh}(\mathbf{c_t}) \tag{6}$$

$$\acute{\mathbf{y}} = \mathbf{W_d h_T} + \mathbf{b_d} \tag{7}$$

Here, $\boldsymbol{i_t}$, $\boldsymbol{f_t}$ and $\boldsymbol{o_t}$ are the input gate, the forget gate and the output gate, respectively (Kratzert et al. 2019c). $\boldsymbol{c_t}$ is the cell output (cell state) at time step $\boldsymbol{t}$, and $\boldsymbol{g_t}$ is the cell input computed by the recurrent input (hidden state) $\boldsymbol{h_{t-1}}$from the previous time step $\boldsymbol{t}$**-1**. In the above equations, $\boldsymbol{W}$ and $\boldsymbol{U}$ are the adjustable weight matrices for each gate, whereby subscripts indicate to which particular gate they belong. Together with the adjustable bias vector $\boldsymbol{b}$, these metrices represent the learnable parameters of the network. Furthermore, two different activation functions are used: the sigmoid-function ($\boldsymbol{\sigma}$) and the hyperbolic tangent function ($\mathbf{tanh}$). The symbol $\odot$ refers to the element-wise multiplication of metrices/vectors.

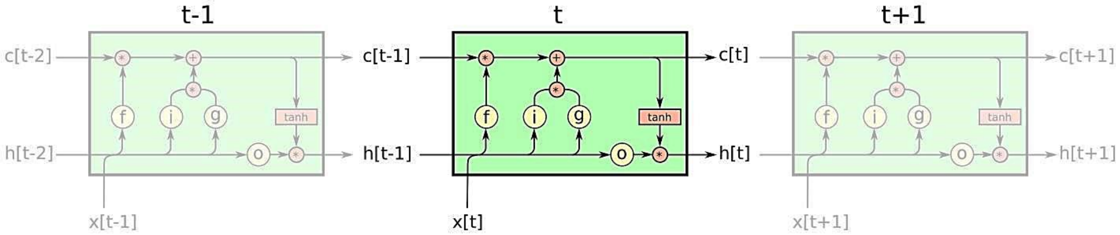


Figure 7: Visualization of the standard LSTM cell as defined by Eq. (1-6) (Kratzert et al. 2019c)

In the first time step, the hidden state and the cell state are each initialized as a vector of zeros or a vector with very small random values with specific distribution (Chollet 2018). The size of the hidden state is directly related to and defined by the chosen number of LSTM units, which is a user-defined hyperparameter of the network. As with other NNs, the number of LSTM units can be interpreted as the number of neurons inside one LSTM memory cell.

The following description of the calculations within the forward pass through a LSTM is outlined in Kratzert et al. (2018) and developed by Hochreiter and Schmidhuber (1997). The first gate is the forget gate, which controls, based on recurrent hidden state, to which degree the elements of the cell state vector will be forgotten. This is computed by equation *(1)*, which decides how long and what past state memory should be retained. In the next step, a potential update vector $\boldsymbol{g_t}$ for the cell state is computed from the current input $\boldsymbol{x}[\boldsymbol{t}]$**,** and the last hidden state $\boldsymbol{h_{t-1}}$ (equation *(3)*). Simultaneously, the input gate, defining which (and to what degree) information of $\boldsymbol{g_t}$ is used to update the cell state in the current time step is calculated by equation *(2)*. With the results of equations *(1)* – *(3)* the cell state $\boldsymbol{c_t}$ is updated by the use of equation

(5). Since both the vectors $\boldsymbol{i_t}$ and $\boldsymbol{f_t}$ have entries in the range [0, 1], equation *(5)* can be interpreted in the way that it defines which information stored in the previous cell state $\boldsymbol{c_{t-1}}$ will be forgotten (values of approx. 0) and which will be kept (values of approx. 1). Likewise, $\boldsymbol{i_t}$ determines which new information stored in $\boldsymbol{g_t}$ will be added to the cell state (values of approx. 1) and which will be ignored (values of approx. 0). The dimension of the cell state vector always corresponds to the dimension of the hidden state vector. The third and last gate is the output gate, calculated by equation *(4)*, which controls the information of the cell state $\boldsymbol{c_t}$ that flows into the new hidden state $\boldsymbol{h_t}$ at time step $\boldsymbol{t}$. As the last step, the new hidden state is calculated by the use of the vector of the output gate and the cell state (equation *(6)*). It is in particular the cell state $\boldsymbol{c_t}$ that allows for an effective learning of long-term dependencies. Due to its very simple linear interactions within the LSTM cell, it can store information unchanged over a long period of time steps (Hochreiter and Schmidhuber 1997). During training, this characteristic helps to prevent the problem of the exploding or vanishing gradients in the backpropagation step (Hochreiter and Schmidhuber 1997). In order to get predicted values, the output from a LSTM cell is passed through a traditional dense layer (normal layer as in a feed-forward NN) to a single output neuron, which computes the final prediction (equation *(7)*) (Kratzert et al. 2018)). This final dense layer has again its own learnable parameters. The user, i.e. model architect, can decide whether either the network should produce predictions after each time step, or only the hidden state at the last time step $\boldsymbol{h}[\boldsymbol{T}]$ should be outputted. This kind of setting is application-specific and depends on the research purpose. For the prediction of discharge in future time steps, as it is done in this thesis, only the hidden state of the last time step is used to compute the predicted values. As for normal feed forward NNs, multiple LSTM layers can be stacked on top of each other in order to increase the model depth.

Regarding the training of a LSTM or a RNN, the computational principle works the same way as for traditional NNs, meaning that the learnable parameters of the network, the weights and biases, are updated depending on a given objective function (loss function) for each iteration step. Equivalent to regular backpropagation, there is first a forward pass through the unrolled network. Then, the output is evaluated using an objective function (loss function) and the loss, which is the cumulative loss of each time step, is computed. Dependent on the model setting, the loss function will ignore some outputs, for instance when only the last hidden state of the network at the last time step is calculated. After that, the gradients of that loss function are computed by backpropagation through time. Finally, the learnable parameters are updated after a complete sequence of forward and backward passes by the use of these gradients (Géron 2019).[4]

---

[4] For a very detailed description see e.g. Goodfellow et al. (2016.

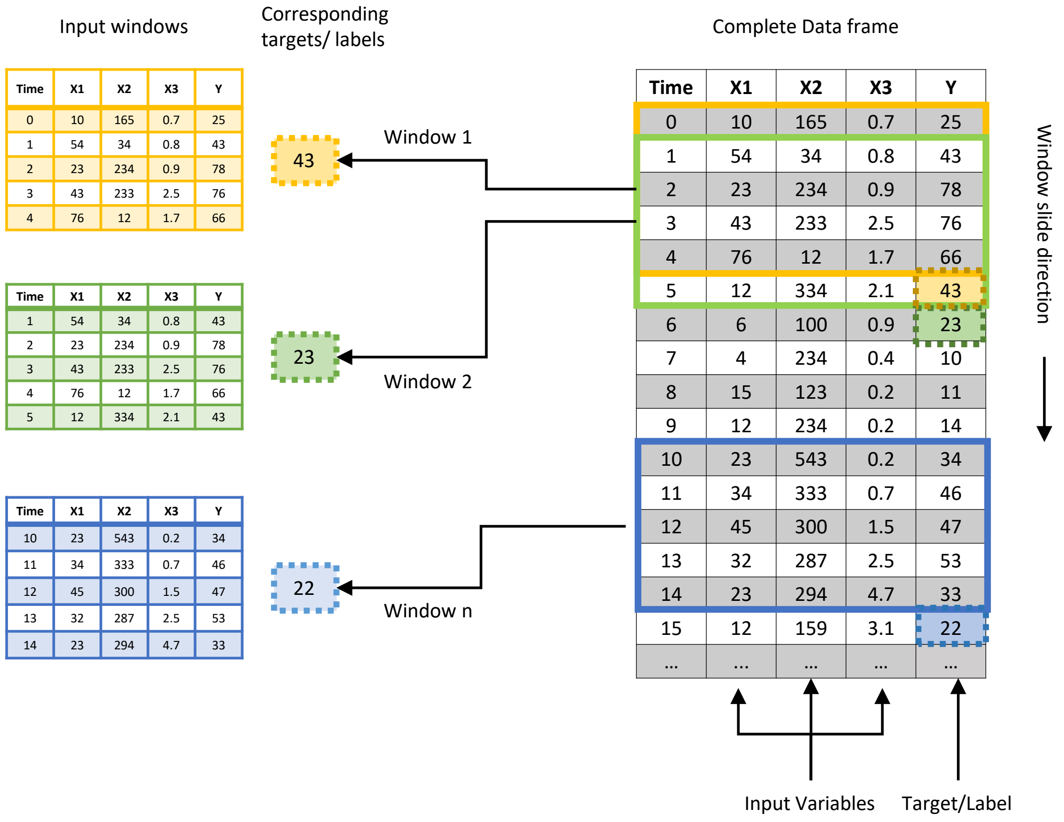
The advantages of a LSTM network are that the constant error backpropagation within memory cells results in LSTM's ability to bridge very long time lags and that it can handle noise, distributed representations and continuous input sequences (Hochreiter and Schmidhuber 1997). The gradient-based algorithm assures constant error flow through internal states of special gates, thus eliminating exploding or vanishing gradients (Hochreiter and Schmidhuber 1997).

2.2.2. Development of a multivariate forecasting model

Time series forecasting involves developing and using a predictive model on data, in which there is an ordered relationship between observations. A time series forecasting problem, in which one or more future numerical values should be predicted in a quantitative way, is called a regression type modeling problem (Géron 2019). If for a forecast more than one input variable is used as input for a NN model, the forecasting is said to be multivariate (Jason Brownlee 2018). That is the case in this thesis, since several meteorological measurement parameters are fed into the model. This classification can also be transferred to the output, meaning that a model can also be used to predict several different variables (Jason Brownlee 2018). However, in this thesis, the output is univariate as the discharge is the only variable to be predicted.

Within a basin of a river, various hydrological processes take place that influence and lead to the river discharge, including, e.g., evapotranspiration, snow accumulation and snow melt, water movement in the soil or groundwater recharge and discharge. These processes have highly non-linear correlations and are predominantly contingent on the states of the system, which represent the memory of a river basin (Kratzert et al. 2019a). As a consequence the discharge at a given time step ***t*** is driven by these system states and by meteorological events of the preceding time steps (Kratzert et al. 2019a). One of the central questions of the forecasting problem is how many of the preceding time steps should be considered, i.e. how long the observation history fed to the model should be. This time window should be chosen in such a way that the long time lag in discharge generation in snow-influenced basins can be captured by the model. Therefore, from a hydrological point of view, the precipitation during all winter months might be influential for the correct prediction of the discharge. In contrast, in arid periods or in less snow-influenced and purely precipitation-driven watersheds, the discharge likely depends only on the response time of the catchment, thus requiring far fewer time steps of the meteorological observation history (Kratzert et al. 2019a). Hence, the appropriate length of time steps within an input window is dependent on the research domain and on the catchment characteristics.

In general, time series forecasting can be framed as a supervised learning problem. Given a sequence of numbers in a time series dataset, the data can be restructured in such a way that previous time steps of the variables in the dataset are used as input, and the target variable, i.e. the predicted variable of the next time step, is used as output (Chollet 2018). This method is called *window method, sliding window method* or *lag method* – depending on the literature. The number of previous time steps that are used as model input is called the window length, size or width. In Figure 8, this technique is visualized for an example dataset containing random values to demonstrate the idea.



Time	X1	X2	X3	Y
0	10	165	0.7	25
1	54	34	0.8	43
2	23	234	0.9	78
3	43	233	2.5	76
4	76	12	1.7	66

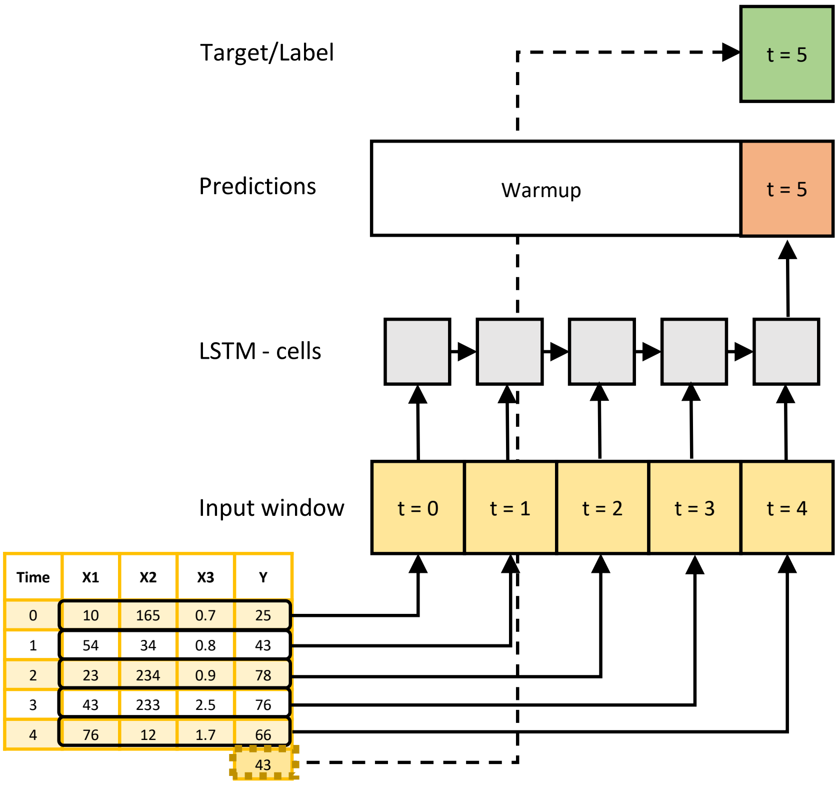
Time	X1	X2	X3	Y
1	54	34	0.8	43
2	23	234	0.9	78
3	43	233	2.5	76
4	76	12	1.7	66
5	12	334	2.1	43

Time	X1	X2	X3	Y
10	23	543	0.2	34
11	34	333	0.7	46
12	45	300	1.5	47
13	32	287	2.5	53
14	23	294	4.7	33

Time	X1	X2	X3	Y
0	10	165	0.7	25
1	54	34	0.8	43
2	23	234	0.9	78
3	43	233	2.5	76
4	76	12	1.7	66
5	12	334	2.1	43
6	6	100	0.9	23
7	4	234	0.4	10
8	15	123	0.2	11
9	12	234	0.2	14
10	23	543	0.2	34
11	34	333	0.7	46
12	45	300	1.5	47
13	32	287	2.5	53
14	23	294	4.7	33
15	12	159	3.1	22
…	…	…	…	…

Figure 8: Visualization of the sliding window method for single step predictions

The window of past input time steps can contain any number of different input variables. However, the size of the window, i.e. the number of considered time steps, is the same for each input variable. Figure 8 visualizes that input arrays are created based on the input window size of for example five time steps. These windows are shifted by one time step along the time axis of the input data frame. In this example, each of the created input windows contains three individual variables (X1-X3) and the target variable (Y). The corresponding target value – that the model tries to predict based on the input window – is the target variable of the next time step subsequent to the last time step of the input window. The time index of the target value is thus compared to the input window shifted by one time step. Another important aspect is that the order of the observations within a window is preserved and must continue to be preserved when using this dataset to train the model. To understand how the model processes the times steps inside an input window, maintaining an internal state from time step to time step, the following Figure 9 is provided.



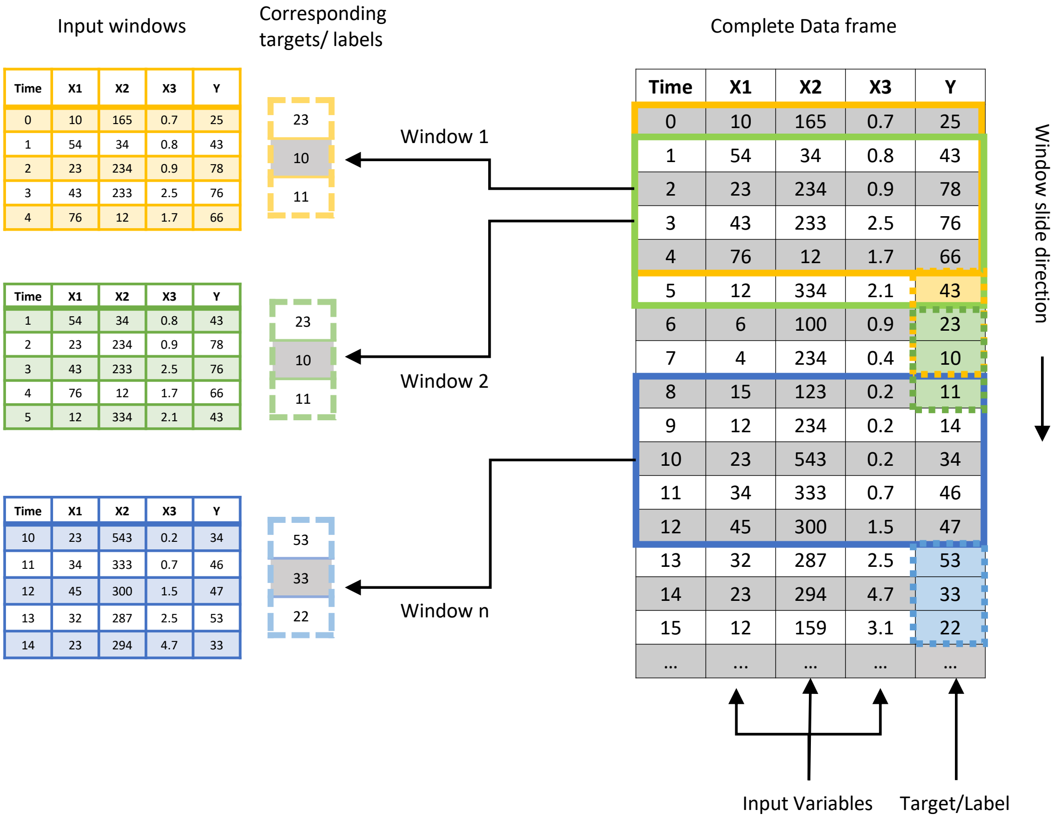
Time	X1	X2	X3	Y
0	10	165	0.7	25
1	54	34	0.8	43
2	23	234	0.9	78
3	43	233	2.5	76
4	76	12	1.7	66
				43

Figure 9: Schematic illustration of how features within one window in one batch are processed by a LSTM unrolled through time predicting the next time step

This model setting is called many-to-one setting, in which – at each time step – a set of specific input variables is processed (black squares in Figure 9), but the model will only output the very last hidden state at the last time step of the window. This output is then compared to the true measurement, i.e. the target, of the subsequent time step. After one window is processed, the next window with the same size is fed into the network. There is no need to have the windows consecutively ordered along the time axis, since the model updates its learnable parameters only based on the continuous time steps within one window. As a result, the individual windows can be randomly shuffled when being passed to the network (Chollet 2018).

Furthermore, it can be seen that in the beginning of the time series, when the model has not received the complete sequence length of the predefined window size yet, the model is not able to compute any predictions. This is called the warmup phase. Therefore, at the beginning of a time series no predictions can be made by the model for the number of time steps of the defined window size. Similarly, at the end of the time series, the model can only predict as close to the last time step of the complete sequence as there are target values available for the comparison between true observations and model predictions. In other terms, for the very last time step in a time series (in case of single step predictions), the model cannot output any value, since at the end of a time series there is no next true observation for comparison available.

The scenario described above can be referred to as a single step forecast, in which only the target value of the very next time step should be predicted. However, it is also possible for the model to predict multiple time steps ahead based on one single past window. In this multi-step scenario, the only difference is that the output consists not only of the value of the next time step, but also of the values of the next ***n*** time steps, which the model should predict. The number of future time steps, i.e. how far the model should predict ahead, corresponds to the target steps, which is a user-defined setting. Hence, in the further course of this thesis, a model with a target step number greater than one serves as a multi-step prediction model. During training, a multi-step model does not compare one output value with the corresponding true measurement, but it compares a vector of ***n*** outputs with the corresponding ***n*** observations of the subsequent target steps. In a multi-step scenario, the arrangement of the windows with their corresponding targets is illustrated in Figure 10. In this example, the next three time steps of the target (Y) should be predicted, based on the input windows of size 5, containing 4 different variables (X1, X2, X3, Y).



Time	X1	X2	X3	Y
0	10	165	0.7	25
1	54	34	0.8	43
2	23	234	0.9	78
3	43	233	2.5	76
4	76	12	1.7	66

23
10
11

Time	X1	X2	X3	Y
1	54	34	0.8	43
2	23	234	0.9	78
3	43	233	2.5	76
4	76	12	1.7	66
5	12	334	2.1	43

23
10
11

Time	X1	X2	X3	Y
10	23	543	0.2	34
11	34	333	0.7	46
12	45	300	1.5	47
13	32	287	2.5	53
14	23	294	4.7	33

53
33
22

Time	X1	X2	X3	Y
0	10	165	0.7	25
1	54	34	0.8	43
2	23	234	0.9	78
3	43	233	2.5	76
4	76	12	1.7	66
5	12	334	2.1	43
6	6	100	0.9	23
7	4	234	0.4	10
8	15	123	0.2	11
9	12	234	0.2	14
10	23	543	0.2	34
11	34	333	0.7	46
12	45	300	1.5	47
13	32	287	2.5	53
14	23	294	4.7	33
15	12	159	3.1	22
…	…	…	…	…

Figure 10: Visualization of the sliding windows method for multi-step one shot predictions

In contrast to the many-to-one model setting, the model is now said to have a many-to-many setting, since several inputs are processed, and several predictions are produced. The model predicts the entire output sequence of future values at once, i.e. in one step. Thus, it is called a single shot model. This model behavior is shown in Figure 11 by way of illustration.

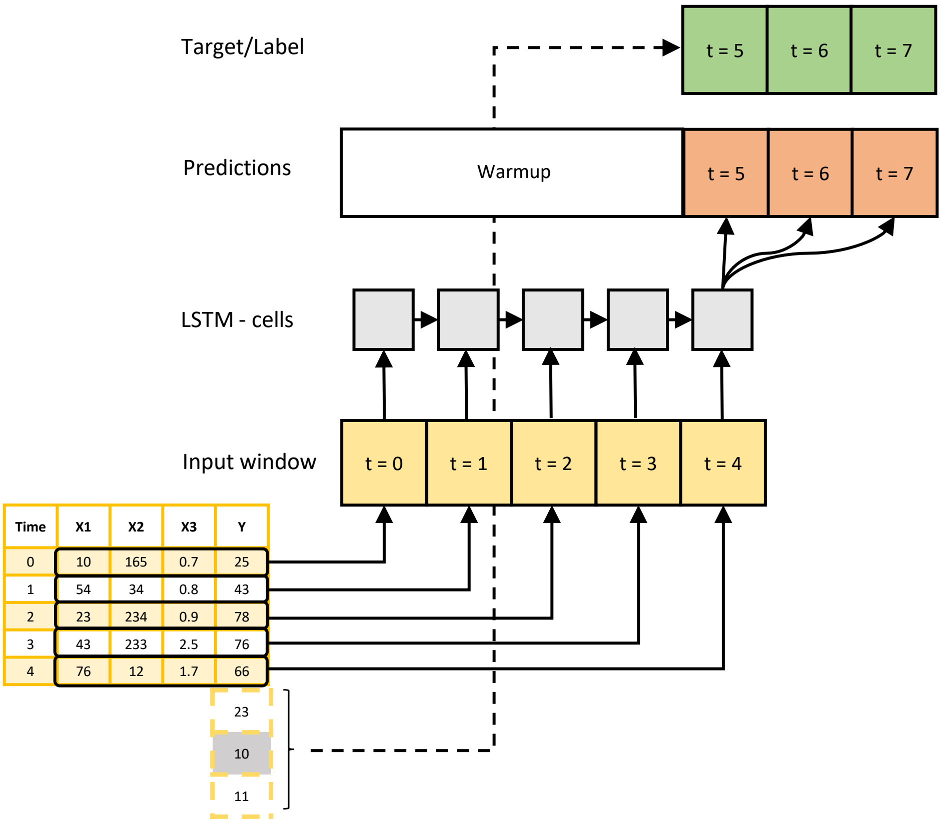


Figure 11: Schematic illustration of how features within one window in one batch are processed by a LSTM unrolled through time predicting the next *n* time step (in this example the next 3 time steps)

One general drawback of the single shot approaches mentioned above is the fact that the model predicts future values of the next ***n*** time steps without additional incorporation of input variables during the forecasting period of these ***n*** target steps. To put it differently: the predictions are only based on the observation of the past window and not on additionally estimated input variables for the forecasting period.

To project this onto the scenario in this thesis, in which the discharge should be predicted based on meteorological input parameters, meteorological forcing data (e.g. precipitation) is not considered during the ***n*** prediction steps the model conducts (in one shot). It should be mentioned here that this does not necessary imply a possibly wrong assumption of zero precipitation during the target steps, but rather the model does not receive *any* meteorological data. In other words: from a computational perspective, there is a significant difference between the model being provided with precipitation values of zero during the future prediction steps and the model not receiving any information about the future precipitation at all. From a hydrological forecasting perspective, it might be advantageous for the model to receive meteorological information during the prediction steps. Hence, to somehow evade the aforementioned drawback a customized prediction loop is provided in this thesis, which suggests a prediction algorithm for the LSTM model to incorporate forecasted data during multi-step predictions.[5]

In order to create either the single-step or the multi-step scenario, the dataset of the complete time series has to be split in a training, validation and test set, as described in section 2.1.3. Before these three subsets are transformed into the right format for the model, the user has to define the number of target steps, the size of the lookback window and the batch size. The next step is to reshape each of the three subsets (training, validation and test set), so that it can be processed by the model. This results in an input array for each subset that is three-dimensional, whereby one of the dimensions indicates the number of batches (B) in the array. Each batch consists of a specific number of input variables, i.e. features (F), which create another dimension in the input array. Finally, each of these considered features are processed in the network simultaneously step by step for a sequence of ***n*** consecutive recorded time steps, which matches the predefined lookback window size (W). Thus, in total, there are three independent views along the input array with the size: [B, W, F] as displayed in Figure 12. The total number of windows the network processes within one subset can be calculated by: *number of time steps – window size – target steps*.

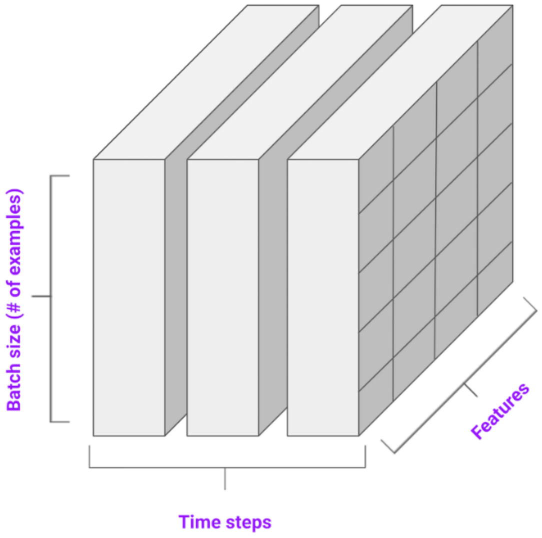


Figure 12: Illustration of the individual views along the axis of the 3D input array. The three dimensions are: (i) batch size (number of simultaneously processed windows), (ii) window size (number of consecutive time steps considered in the lookback window), (iii) number of available input features

However, before the input arrays are fed into the network, the features are scaled. Many ML algorithms perform better or converge faster when features are on a relatively similar scale

---

[5] The customized prediction loop is described in detail in section 3.7.

and/or are close to normally distributed (Goodfellow et al. 2016). Scaling tends to make the training process work better by improving the numerical condition of the optimization problem and by ensuring that various default values involved in initialization are appropriate (Kratzert et al. 2018). It is of high importance that each feature is individually scaled and that the scalers are fitted to the training data only, not to the full dataset (including the test set). After that, the scalers can be applied to scale the validation and the test set, ensuring that there are no information leaks from the training data into the validation or test set (Géron 2019, Chollet 2018). The scaling techniques that were applied in this thesis are described in section 3.4.3.

After the scaling process, the data is in the right format to be handled by the model. Now the model has to be created and initialized, with its desired settings of hyperparameters. The batch size is one hyperparameter that has already been defined at this point. Within traditional hydrological model calibration, the number of iteration steps defines the total number of model runs performed during calibration (Kratzert et al. 2018). The corresponding term for NNs is called an epoch, as already mentioned in section 2.1.2, and it also has to be specified before fitting the model to the data. In addition, other hyperparameters, e.g. the number of LSTM units and the used objective function, need to be set. A full list of the applied hyperparameters in this thesis is provided in section 3.4.

After the model is fitted or calibrated to the training data for the selected number of epochs, the model with its optimized weights can be used to predict an individual sequence of the target variable. Finally, these predicted values can be compared to the true observations, and the model performance can be evaluated using individual performance metrices. These performance metrices should give an indication of how accurate the model predictions are and how well the model performs on unseen data. The choice of performance metrices is strongly domain-specific and depends on the research purpose. For regression tasks such as runoff prediction, the Mean-Squared-Error (MSE) is commonly used. Hydrologists also commonly use the Nash-Sutcliff-Efficiency (NSE) because it has an interpretable range (Kratzert et al. 2019c). Further details on which performance metrices are used for evaluating the model in this thesis are given in section 3.5.

3. Methodology

The topic of this chapter involves providing the reader with an overview over the concrete methods that are applied to investigate on the different objectives of this thesis. At the very beginning the general procedure of the sensitivity analysis is explained. This is accompanied by a flow chart to visualize also all the intermediate analysis steps. Afterwards an overview over the catchment characteristics and its meteorological stations is provided. Then all the necessary information about the meteorological measurements together with a basic statistical overview is given. Furthermore, the basic workflow and necessary preprocessing steps of the input data are described. This section explains how the data is cleaned and further prepared to serve as input for the neural network. In the next section, the specific LSTM architecture with selected hyperparameter is explained in detail. As next step, different fundamentals of validating the model are presented. Finally, the principle of the customized prediction loop is explained.

3.1. Overview of Investigation Steps

To get an overview, about all aspects of investigation within this thesis and to be able to follow along the different analysis steps in a structural manner a flow chart is provided in Figure 13. After the Raw Data is preprocessed, the analysis starts with testing very basic scenarios, in which the model with its primary model settings - as a result of the research in (Unnikrishnan 2019) - is used to predict the discharge at one example station. Hereby, only one close by precipitation station is considered as meteorological input and the effect of additionally including the target discharge gauge, as part of the input data, is investigated. Since the model might behave differently with respect to input data scaling applied during preprocessing and to the chosen method of dealing with missing values, a grid search of model runs is carried out. This should give the most suitable combination of both aforementioned methods regarding model performance. This preceding analysis is performed on daily data basis only and the most suitable setting is used in further course of analysis steps. The succeeding research steps are conducted in both, daily and hourly resolution. As a next step, the influences of the different imputation techniques on the model performance are identified. The best imputation techniques for precipitation data is then used in the further course of the thesis. The approved dataset configurations based on the aforementioned preliminary analysis are then applied in the sensitivity analysis. In this analysis the impact on model performance of having various input feature combinations of meteorological stations and/or discharge gauges in the input data is examined. Considering the most promising configuration of input features, it should be investigated, if the model performance can further be enhanced by basic hyperparameter tuning. Finally, the overall optimal model configuration on an hourly basis is used to explore the LSTM's performance of predicting discharge multiple time steps ahead with different methods. To round the investigations off a customized prediction loop is introduced, in which a pre-trained single step LSTM model is used to predict several time steps ahead. Within this prediction loop the LSTM model is fed continuously with new data for each prediction step.

Since no truly forecasted meteorological data was available, instead true observations of meteorological input features - naively assumed as "forecasted" – are considered.

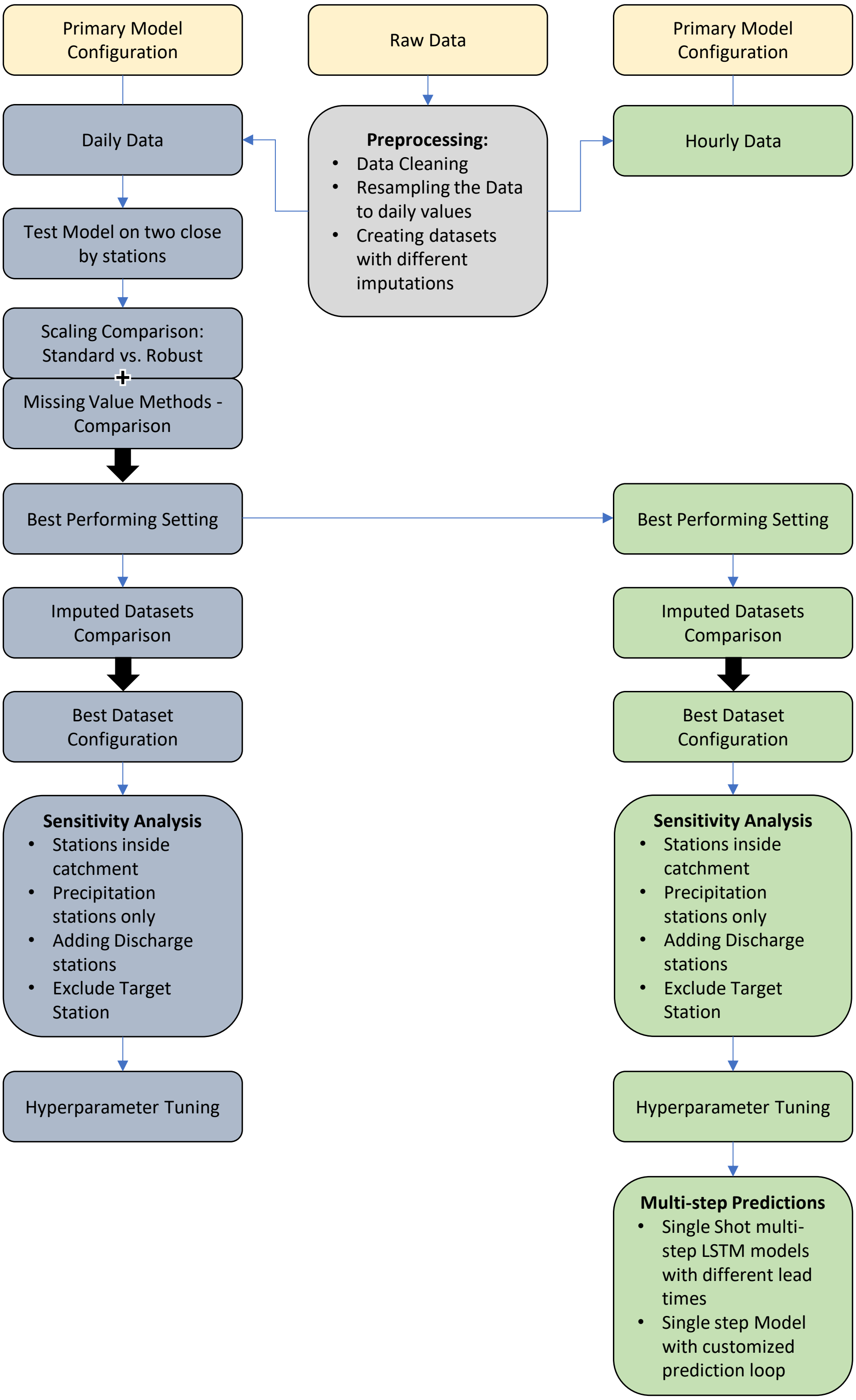

Primary Model Configuration
Raw Data
Primary Model Configuration
Daily Data
Preprocessing:
Data Cleaning
Resampling the Data to daily values
Creating datasets with different imputations
Hourly Data
Test Model on two close by stations
Scaling Comparison: Standard vs. Robust
+
Missing Value Methods - Comparison
Best Performing Setting
Best Performing Setting
Imputed Datasets Comparison
Imputed Datasets Comparison
Best Dataset Configuration
Best Dataset Configuration
Sensitivity Analysis
Stations inside catchment
Precipitation stations only
Adding Discharge stations
Exclude Target Station
Sensitivity Analysis
Stations inside catchment
Precipitation stations only
Adding Discharge stations
Exclude Target Station
Hyperparameter Tuning
Hyperparameter Tuning
Multi-step Predictions
Single Shot multi-step LSTM models with different lead times
Single step Model with customized prediction loop

Figure 13: Flow chart of the consecutive investigation and analysis steps carried out in this thesis

3.2. Study Area – Regen Catchment

The Regen catchment has an area of 2878,13 km² and is located in the upper Palatinate in Bavaria, Germany.[6] The Western boundary of the watershed area follows the border to the Czech Republic, where the main headstream of the river *Regen* has its source. The *Regen* is the second longest left sided tributary of the Danube. The river's total length, including its headstreams, the *Great Regen* and *Black Regen*, is 191 km. The riverhead of the *Great Regen*, is in the Bohemian Forest on the territory of the Czech Republic, near Železná Ruda.[7] The river crosses the border after a few kilometers, at Bayerisch Eisenstein. At Zwiesel, the *Great Regen* is joined by the *Little Regen* to form the *Black Regen*. The *Black Regen* flows through the villages Regen and Viechtach and is joined by the *White Regen* near Bad Kötzting. Beyond this confluence, the river is called *Regen*. The outlet of the Regen catchment is the point where the *Regen* flows into the Danube in Regensburg. In Figure 14 an overview of the catchment area and the available meteorological stations as well as the gauging stations of the biggest tributaries is provided.

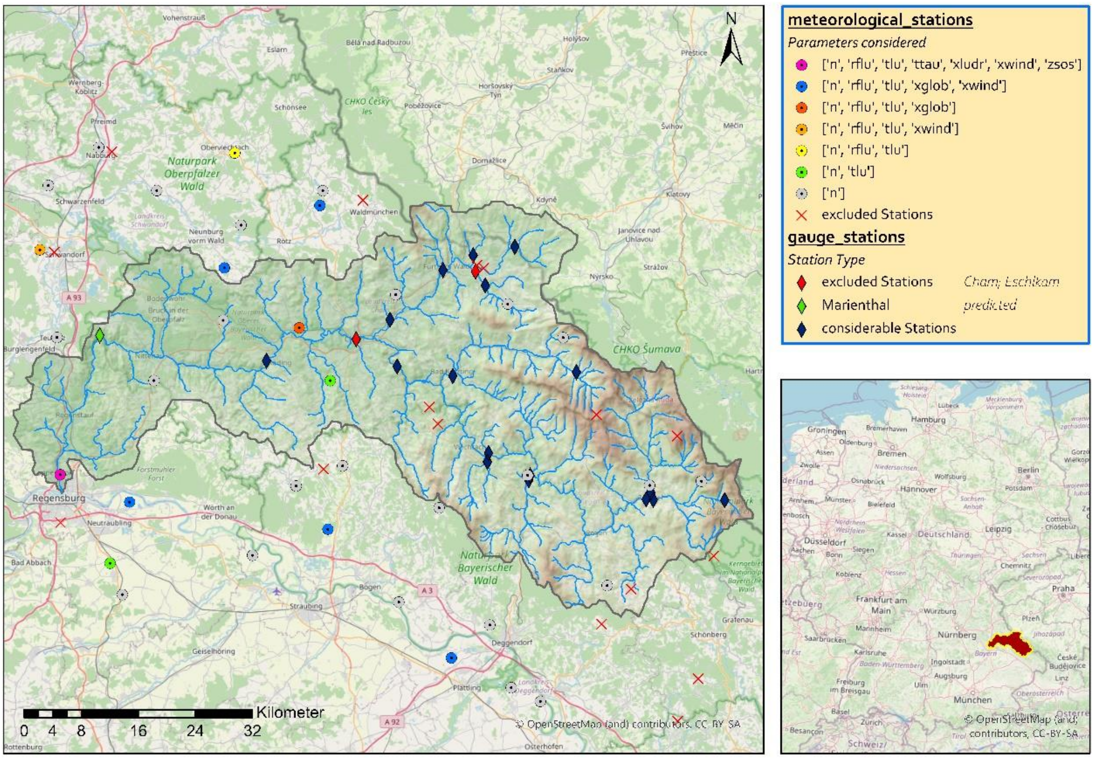


Figure 14: Overview of study area

[Sources of shape files: *river network* from Ecrins (https://www.eea.europa.eu/data-and-maps/data/european-catchments-and-rivers-network); the *catchment outline* from HydroSheds (www.hydrosheds.org), the *DEM* from EU_DEM v1.1 (https://land.copernicus.eu/imagery-in-situ/eu-dem/eu-dem-v1.1?tab=download)]

---

[6] http://www.lfu.bayern.de/wasser/gewaesserverzeichnisse/doc/tab_alle.xls by the Bavarian State Office for the Environment

[7] https://en.wikipedia.org/wiki/Regen_(river)

The elevation of the watershed area extends between 316 m and 1446 m above sea level.[8] The annual mean precipitation in across all available stations is about 793.19 mm and the annual mean temperature is around 8.81°C with temperature ranges roughly between -25°C and + 38°C.[9] The maximum rainfall amount in 24h was 104.06 mm at the Station in *Neukirchen bei Heiligen Blut*.[9] In total there are 20 available discharge gauges and 55 meteorological stations, which measure several numbers of meteorological parameters. Some of the meteorological stations are actually located outside the catchment boundaries. However, since the weather patterns are no strict local phenomenon, these stations may still capture useful input data. This especially applies to the stations close to the watershed boundaries. Additionally, these stations can be used for interpolation purposes. The symbolization of the meteorological stations in Figure 14, is dependent on the particular parameters, that are recorded. The green rhombus in Figure 14, indicates the gauge at *Marienthal*. This is the target station at which the discharge of the *Regen* should be predicted. At this station, the mean annual discharge is 37.70 m³/s and the annual peak discharge is around 304.00 m³/s. The highest ever recorded value is 720.00 m³/s.[10] However, the highest measured discharge in the available time period (2003 – 2018) is 410.17 m³/s, which corresponds to a return period between five and 10 years. The two red rhombi as well the red crosses in Figure 14 indicate gauges and meteorological stations, that had to be excluded from the dataset due to a high amount of missing data. The remaining blue rhombi specify discharge gauges, that have enough records and can thus be potentially considered as additional input data for the model.

3.3. Data Overview

The dataset that is used in this thesis was provided by the Flood Forecast Center from the Bavarian Water Authorities *(Landesamt für Umwelt (LFU)).* As mentioned above there is data from 55 meteorological stations and 20 discharge gauges available. Principally, all the meteorological stations measure the precipitation, but 29 of these stations additionally measure several other meteorological parameters (e.g. temperature). In Table 1 an overview of all existing parameters is given.

---

[8] Source: EU_DEM v1.1 (https://land.copernicus.eu/imagery-in-situ/eu-dem/eu-dem-v1.1?tab=download)

[9] Source: LARSIM input files provided by Flood Forecast Center from the Bavarian Water Authorities; Reference period: 2003 - 2018

[10] https://www.gkd.bayern.de/de/fluesse/abfluss/naab_regen/marienthal-15207507/hauptwerte; Reference period: 1901 - 2013

Table 1: Overview of available meteorological and hydrological parameter in the dataset

Parameter	Abbreviation	Unit	Number of Stations	Percentage of missing values	
				hourly	daily
Precipitation	n	[mm]	55	12.11	11.29
Discharge	q	[m³/s]	20	4.73	4.64
Relative humidity	rflu	%	28	23.36	22.61
Air temperature	tlu	[°C]	29	19.57	18.67
Dew point temperature	ttau	[°C]	4	6.62	6.07
Global radiation	xglob	[w/m²]	15	27.97	27.35
Air pressure	xludr	[hPa]	4	26.67	26.45
Wind speed	xwind	[m/s]	23	26.81	25.03
Sunshine duration	zsos	[min]	7	33.31	27.40

An input feature describes on specific measurement parameter at one particular station. Altogether this results in a total of 185 input features. The station data of all 9 available parameters are recorded on hourly basis between the 1st of November 2003 and the 1st of January 2018. However, for necessary preprocessing steps as well as for some sections of the sensitivity analysis, the temporal resolution is resampled to daily resolution in order to reduce computational intensity (less amount of data) and to investigate the difference of some other temporal resolution on the performance of the model. The resampling procedure is conducted for the different parameters individually. For precipitation, the cumulative sum is calculated to get the total amount of rain in one day. Also, the global radiation and sunshine duration it makes sense to use the total amount measured over the period of one day. The parameters discharge, relative humidity, air pressure, dew point temperature and wind speed are resampled by taking the mean value over 24 hours, respectively. The approach to calculate the daily air temperature is differently. With the intention to not lose the valuable information of the minimum and maximum temperature for one day, the parameter air temperature is duplicated. Instead of taking the mean of the air temperature, now two parameters are considered as input, which are minimum and maximum temperature in one day, respectively. This increases the available number of input features for the dataset based on daily resolution by 29. There are now 214 accessible input features in the daily dataset. Nevertheless, it is often the case, that for such a long measurement period the available data comes along with missing data gaps due to e.g. measuring system failures or maintenance work. This holds also true for the dataset in this thesis. In total there are 18.3% of all possibly available measurements missing (hourly values). After resampling to daily values this percentage drops to 17.5%. The percentage of missing values compared to the amount of available records per parameter is shown in Table 1. The amount of missing values is quite significant and requires preparing a strategy for dealing with these missing values. Thus, one essential step before the data is fed into any type of hydrological or data-driven model, the input data have to go through necessary cleaning and further preprocessing steps, which are described in the following section. A complete list of all available stations including station IDs and recorded parameters can be found in Table 12 in the Appendix.

3.4. Data Preparation

Data preparation and preprocessing is usually designed to serve different purposes, for instance removing some variability in the input data, determining extreme outliers, filling in data gaps, changing the underlying data formats, etc. In this thesis some basic preprocessing seems reasonable and necessary due to a high percentage of missing data, as pointed out in the previous section. Various approaches have been developed to address missing values in time series. Probably the simplest solution is to discard missing samples and to perform the analysis only on the available measurements. However, when the rate of missing data is high and faulty observations are conserved, this method does not provide good performance (Che et al. 2018). In this thesis reducing the quantity of missing values is accomplished by disregarding complete features from the dataset and imputing missing gaps by applying appropriate techniques. The following subsections explain the necessary steps to prepare the dataset to be fed into the LSTM model.

3.4.1. Data Cleaning

As mentioned above, there are two datasets with two different temporal resolutions available: one on hourly basis with 185 possible input features and one with 214 possible input features with daily records, which incorporates both, min and max temperature at each station. The former data contains 22,981,440 samples and the latter one has 1,107,664 samples. Since the recorded resolution is on hourly basis, the cleaning process is applied to the hourly dataset and then adopted to the resampled daily dataset. All the decisions on disregarding specific features is done after viewing and examining the data manually. Even though data has been available since 1st of November 2003, the chosen start date is 11th of January 2005, because since this date continuous measurements have been available from all meteorological stations. This ensures that a maximum number of features are available. Furthermore, some stations show obvious errors (e.g. constant values over several months) in their recordings and therefore their measured parameters are excluded completely from the dataset. This is done to make sure to not introduce any systematic bias. All the remaining missing values are either present as larger gaps of up to a few months or show up as occasionally scattered intervals of few hours across the measurement period. As a next step, the remaining percentages of missing values compared to the available samples per features is examined. To be able to have a sufficient time span over 10 years of available data per feature to train the model, a threshold percentage for missing samples is defined to be 10%. Consequently, all features showing a higher amount of missing values than this threshold are dropped from the input dataset. Regarding the discharge gauges, two stations have a noticeable quantity of missing data. The discharge gauge at *Cham/Regen*, began its operation on 24th of November 2009, whereas the recordings at *Eschlkam /Chamb* have only been available since 22nd of June 2007. Due to their short measurement series both of aforementioned gauges are not considered as possible input features, although these stations could have been of importance. Overall, the potentially available input features based on hourly resolution were reduced from a total of 185 to 92 features. Hereby, the number of meteorological features dropped by 91 features, whereas the number of discharge gauges are reduced by two. Hence, after the cleaning procedure the total size of both datasets (hourly & daily resolution) changed: the daily dataset consists of 488,117 samples with 103 considerable input features, while the hourly dataset comprises 10,463,712 samples from 92 considerable input features. An overview of all included stations together with its meta data can be found in Table 12 in the Appendix.

3.4.2. Imputation techniques for Missing Data

The two cleaned datasets still contain a not neglectable amount of missing values. To reduce the amount of missing measurement even further, it is common practice to either substitute them by utilizing information from surrounding measurement stations, also known as data imputation, or use interpolation and spline methods between adjacent measurement points within the measurement series (Che et al. 2018). There are also more modern and sophisticated methods to fill data gaps in time series. Especially ANNs and SVMs have proven to be effective, when approximating non-linear relations. However, these methods require intensive calculations with high computational cost (Campozano et al. 2014). This thesis focuses on the application of deterministic methods, since they are computational efficient, easy to implement and robust in settings with high spatial variability (Campozano et al. 2014). According to Campozano et al. (2014), deterministic methods are mathematical models that provide the same output from a given initial condition, and neither contemplates the existence of randomness nor attribute a probability of occurrence. The choice of imputation methods in general is highly dependent on the catchment characteristics, the number of available precipitation stations in the catchment and their spatial arrangement. Since after the cleaning process the majority of the remaining percentage of missing data concentrates on precipitation data, the focus is on applying different imputation techniques only to the precipitation samples. For imputing missing precipitation values, 4 different techniques are examined and evaluated: (i) simple arithmetic averaging, (ii) normal ratio method, (iii) inverse distance weighting and (iv) linear regression analysis.

ARITHMETIC AVERAGING NEARER STATIONS BASED ON CONDITIONS (AVWC)

A simple deterministic method is the imputation by the arithmetic mean of the corresponding precipitation measurement of the stations near the station of concern. This method is suitable for areas where the variable under consideration possesses small spatial variability. This can be calculated via equation *(8)* (Caldera et al. 2016),

$$\mathrm{p}_{\mathrm{x(t)}} = \frac{1}{\mathrm{m}} \sum\nolimits_{\mathrm{i=1}}^{\mathrm{m}} \mathrm{p}_{\mathrm{i(t)}} \tag{8}$$

where $\boldsymbol{m}$ is the total number of nearby stations, $\boldsymbol{i}$ indicates the $\boldsymbol{i_{th}}$ considered station and $\boldsymbol{t}$ is the time stamp. Which surround precipitation stations for imputing one specific target station are selected is based on two conditions that both have to be met: (1) a threshold for the Pearson correlation coefficient between the target station and the surrounding stations of at least 0.8 for daily values and 0.6 for hourly values, respectively; (2) only nearby precipitation stations are considered, that are within a radius of 15 km.

NORMAL RATIO METHOD WITH RESPECT TO DISTANCE (NRM_D)

This method uses the ratio of normal annual precipitation of the target station ($\boldsymbol{N_x}$) to the normal annual precipitation of the selected surrounding stations ($\boldsymbol{N_i}$) as additional weight for computing the arithmetic mean for these stations, calculated by equation *(9)* (Caldera et al. 2016).

$$\mathrm{p}_{\mathrm{x(t)}} = \frac{1}{\mathrm{m}} \sum\nolimits_{\mathrm{i=1}}^{\mathrm{m}} \left(\frac{\mathrm{N_x}}{\mathrm{N_i}}\right) \mathrm{p}_{\mathrm{i(t)}} \tag{9}$$

The additional parameter $\boldsymbol{m}$ is the total number of nearby stations, $\boldsymbol{i}$ indicates the $\boldsymbol{i^{th}}$ considered station and $\boldsymbol{t}$ is the time stamp. In this case only nearby stations are selected that are within a radius of 15 km, but no threshold for the person correlation coefficient is considered.

INVERSE DISTANCE WEIGHTING (IDW)

Inverse distance weighting (IDW) is probably the most commonly used method to estimate missing data in hydrology and geographical sciences. The success of this method depends on the existence of a positive spatial autocorrelation and is be calculated by equation *(10)* (Caldera et al. 2016).

$$p_{x(t)} = \frac{\sum_{i=1}^{m} \frac{1}{d_i^2}\, p_{i(t)}}{\sum_{i=1}^{m} \frac{1}{d_i^2}} \qquad (10)$$

The parameter ***m*** is the total number of nearby stations with verified conditions, ***i*** indicates the ***i^{th}*** considered station and ***t*** is the time stamp. This method basically weights each precipitation record according to the inverse proportion to its squared distance ***d*** of the neighboring station to the target station. Here, it is chosen to have no requirement of conditions to be fulfilled.

LINEAR REGRESSION (LR)

This technique uses a linear regression model to estimate the missing precipitation data. The adjacent station for the regression model input is chosen based on the highest correlation coefficient between the surrounding stations and the target station (Caldera et al. 2016).

$$p_x = c_1 p_i \qquad (11)$$

The regression model is obtained by the use of the scikit-learn library for python[11]. In order to be able to generate zero values together with non-zero values, the regression line is forced through the origin, as stated in equation *(11)*, where the parameter ***c_1*** is the regression coefficient.

For some time stamps, the selected nearby stations have missing observations as well, thus not all missing precipitation values in the input dataset can be eliminated. Furthermore, regarding the *AVwC-Method*, the desired conditions are not always met and thus measurements of these surrounding stations are not considered to impute missing values of the target stations. Overall, 10 different input datasets for both daily and hourly measurements have been created: Two dataset with no imputed precipitation data (one hourly dataset & one daily dataset) and eight additional datasets for each applied imputation technique (four hourly datasets & four daily datasets). In the further course of this thesis it is analyzed, which of these different datasets works best for LSTM model with respect to both temporal resolutions. In addition, in order to train the LSTM model successfully, the target feature (discharge measurements at Marienthal station) must not contain any missing values. Therefore, these missing values are linearly interpolation between adjacent records along the time axis. Overall the total amount of missing values for the daily measurements have been reduced to 0.71% after data cleaning and further to 0.48% after the applied imputation. For the dataset on hourly basis, the total per-

---

[11] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

centage drops to 1.07% after data cleaning and further to 0.65% after the imputation of precipitation data.[12] The next step before the data can be fed into the network is feature scaling, which is described in the following.

3.4.3. Feature Scaling Techniques

It is common practice to scale data before feeding it into a NN model, due to several reasons. The main reason is, that ML algorithms commonly, with a few exceptions, do perform better when numerical input attributes have similar scales. To "scale" generally means to change the range of the values of one feature, whereby the shape of the distribution stays the same, unlike using non-linear transformations (Goodfellow et al. 2016). The input datasets contain features highly varying in magnitudes, unit and range. Hence, to regularize the variance of the features to be in the same range, feature scaling is applied to the data. In this thesis two methods are applied and comparted: *Standard Scaling*, commonly called standardization, and *Robust Scaling*.

STANDARD SCALING

Standard scaling standardizes a feature by subtracting the mean and then dividing by standard deviation and is given by following equation *(12)* (Chollet 2018):

$$x_s = \frac{x_i - \bar{x}}{s} = \frac{x_i - \bar{x}}{\sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{x})^2}} \qquad (12)$$

The Parameter ***N*** is the number of samples, $\bar{x}$ is the sample mean of one feature and x_i is the ***ith*** value of one feature. The parameter ***s*** is the sample standard deviation of one feature. The standard scaling results in a distribution, which has zero mean and unit variance.

ROBUST SCALING

If the input data contains many outliers, scaling using the mean and variance of the data is likely to not work very well. Since some of the available features in the dataset used in this thesis contains outliers, robust scaling is tested as an alternative method. This method uses more robust estimates for the center and range of the underlying data and thus reduce the effects of outliers and prove to work better for data, that have a heavy-tailed distribution.[13] Robust scaling can be achieved by the following equation *(13)*:

---

[12] How the remaining missing values are treated is explained in section 3.4.4.

[13] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html#sklearn.preprocessing.RobustScaler

$$x_r = \frac{x_i - \tilde{x}}{IQR} = \frac{x_i - \tilde{x}}{Q_3(x) - Q_1(x)} \tag{13}$$

A robust scaler transforms the feature data by subtracting the sample median $\tilde{x}$ from each value of one feature and then dividing by the interquartile range (IQR) of the data. The IQR is a measure of variability, based on dividing a data set into quartiles. The interquartile range is the difference between the third quartile (Q_3) the first quartile (Q_1) of the data, corresponding to the 75th and 25th percentile, respectively.[13] However, this quantile range can be adapted and is set to the 95th and 5th percentile in this thesis, to address only the very large outliers.

3.4.4. Final Dataset preparation

Much of missing data was substantially reduced to this point. Besides that, the data was scaled. However, there is still missing data in the different input datasets, as previously mention in section 3.4.2. As mentioned in section 1.3, one objective of this thesis is to analyze three basic methods of how a LSTM networks could deal with missing data. Since the LSTM model cannot handle missing values (indicated as *NaN* values) from a computational perspective, a final dataset preparation has to be done. The three basic methods applied in this thesis to deal with the last bit of missing data are explained in the following:

METHOD 1 – DROP MISSING VALUES

This method is the simplest one to implement. All remaining missing values are dropped. However, since the developed model cannot deal with input features of unequal sample size per feature, the complete row (all features along on time stamp) for one missing data entry is dropped additionally. Therefore, a lot of usable information get lost and the dataset shrinks quite a bit. For example, in case of the not imputed dataset based on daily records around 39.5% of the total data is lost. If only a dataset with a small sample size is available, this method may not be ideal, since too much data gets lost in order to train the model properly.

METHOD 2 – IMPUTATION BY UNPHYSICAL VALUES

The idea behind this method is to let the model try to learn by its own, that a sample is considered missing. This should be achieved by filling the remaining missing records by an unphysical value or by a value that does not occur naturally in this region. During the training process if these unphysical values run through the model, the model should teach itself to recognize this specific value as being missing. Since the imputation by unphysical values is done after the feature scaling, just one specific defined value can be used for all unique parameters. It must be assured, that this specific fill values result in an unnatural or unphysical value after inversing the parameters back to their original ranges. Of course, the fill value must not match with any other measurement value in the scaled dataset and should be small or big enough to be within the scaling range of the dataset. As the two scaling methods produce different dataset ranges, the fill value has to be chosen individually for the two scaling techniques. For the investigations in this thesis the fill value is set to -7.0 for standard scaling and to -3.0 for robust scaling, respectively.

METHOD 3 – USE MASKING LAYER

In this method the LSTM model is "told" beforehand, which values it should consider as missing. This is achieved by adding a masking layer as the very first layer of the LSTM model. For each time step in the input data, if all values in the input tensor at that time step are equal to one specific mask value, the timestep will be skipped (masked out)[14]. Again, it has to be assured, that the chosen mask value does not match any other non-missing data value, which, if true, could get accidentally masked out. As an appropriate mask value, the corresponding fill values from *Method 2* are chosen with respect to the applied scaling technique. Because this method requires a slightly different model architecture, two model configurations are used in the first steps of the analysis. In the later course of this thesis the model architectures are slightly configured to optimize the model in response to changed input data. The exact model architectures are explained in more detail in the next section.

3.5. Model Architectures

As mentioned earlier there are two slightly different LSTM model architectures used in this thesis with respect to the method for dealing with missing values. Both of them are illustrated in Figure 15. The right hand side in Figure 15, shows the network unrolled through time. This architecture is identical to the one used in the study conducted by Unnikrishnan (2019). Regarding *Method 1* and *Method 2*, this LSTM model is composed of two layers containing LSTM cells. In between these two layers are Dropout layer is added for regularization purposes, as described in section 2.1.3. Concerning *Method 3*, the only modification is an additional masking layer applied before input data is processed by the first LSTM layer. In this case, the missing samples per input features are masked out for the corresponding time step, which implies that they are skipped by the model during training.

---

[14] https://www.tensorflow.org/api_docs/python/tf/keras/layers/Masking

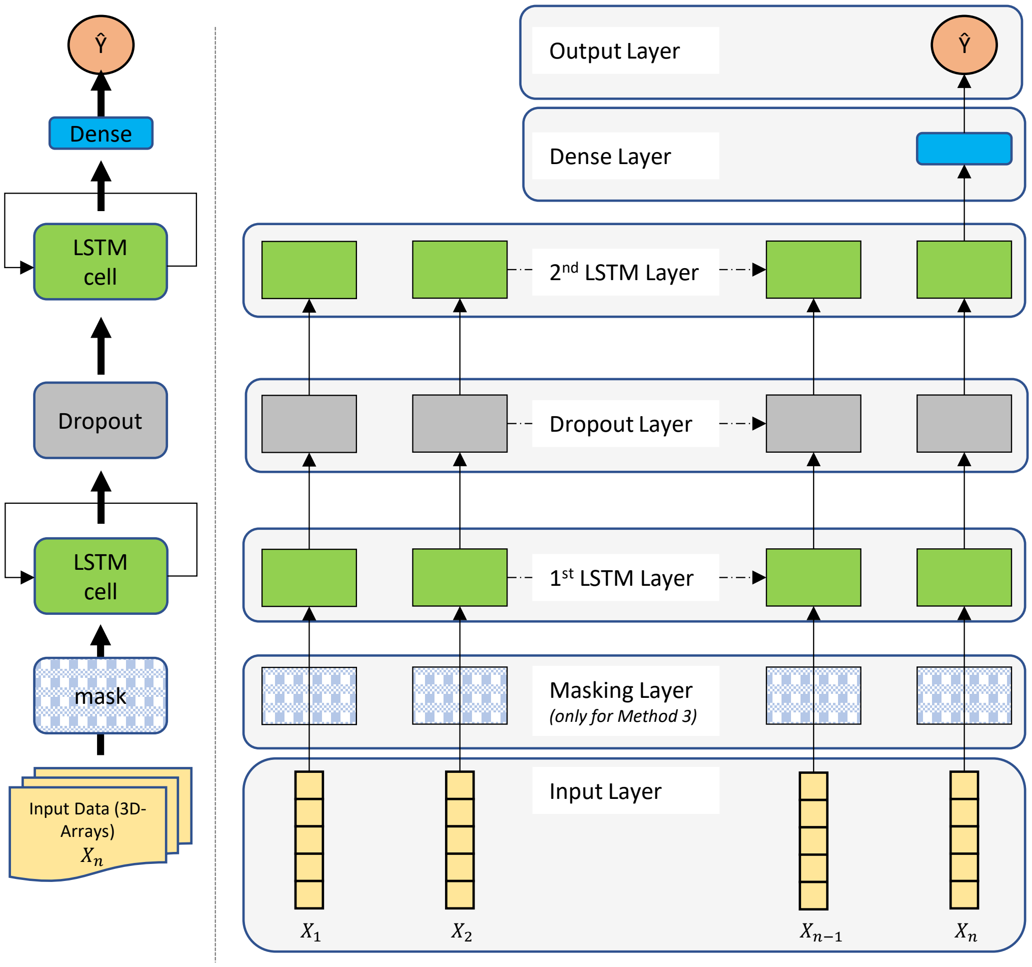


Figure 15: Architecture of the LSTM model with corresponding Layers (left) unrolled over time (right) according to Unnikrishnan (2019) (except Masking Layer). In case missing values *Method 3* is applied, a masking layer is added to the model architecture. In the other both cases (*Method 1* & *Method 2*) the input features are passed directly to the 1st LSTM layer.

In the input layer of the LSTM model the data is fed into the network in a specifically shaped array, as described in section 2.2.2 (see Figure 9). According to Figure 12 in section 2.2.2, this input array is three-dimensional with the size [B, W, F]. The first dimension in this array is related to the batch size *B*. Each batch considers *F* input features. Within one batch all considered features are fed simultaneously into the network are processed in the network step by step for a sequence of ***n*** consecutive recorded time steps, which matches the predefined look-back window size *W*. During model training each batch is shown exactly once per epoch to the model. During the training steps in the first LSTM layer, the individual outputs, i.e. hidden states, are returned by the LSTM cells after each time step and passed to the Dropout layer to prevent overfitting, as explained in section 2.1.3. The remaining outputs are then passed to the second LSTM layer, which contains LSTM cells with the identical number of LSTM units as the cells in the first layer. However, the second LSTM layer, does not return the cell state for each time step, but only conveys the one of the very last time step (***n***) before passing it to a dense layer. This dense layer is a layer as found in an ANN (see section 2.2.1) consisting of neurons, which finally compresses the last hidden state of the LSTM cell to one scalar value. This value represents the predicted discharge value of the consecutive time step.

Figure 16 displays how the shape of the input array changes, while beeing process by the individual layers and being forwarded through the network. This is especially useful to understand, how the different mathematical operations (see equation *(1)* to *(7)* in section 2.2.1) modify each individual batch when computing the predictions. In the exampel, shown in Figure 16, 32 individual feautres are included in the input data, for which each a measurement series of 60 consecutive time steps is conserdered (W = 60). In general, the number of LSTM units (e.g. 120), the window size (w) and the number of input features (F) must be defined before the model is initialized. The question marks in first positions of the shape vector of each layer indicates the batch size. When the graph is compiled it is not denoted as a specific size in the figure, because the batch dimension is effectively variable sized and could be varied between run calls. Overall the input array of size [B, W, F] is reduced to a single predicted value for each batch, as describe in the previous paragraph.

Apart from the three parameters (B, W, U) the model needs additional internal parameters to be defined, i.e. hyperparameters, as mentioned in section 2.2.1. The preliminary hyperparameter setting for the LSTM model used in this thesis, are listed in Table 2. Unnikrishnan (2019) carried out an extensive hyperparameter optimization on a daily data basis and these settings proved to provide the best fit within the problem domain in his analysis. On daily basis, the length of the input sequence fed to the network to predict the next discharge value is set to 120 time steps corresponding to 4 months. This value is chosen to capture possible internal storage processes in the catchment that are present over a four month period (e.g. snow accumulation) and could potentially result in a delayed release into the receiving waters. In the hourly case, however, the lookback is substantially reduced to 5 days (i.e. 120 time steps) to focus more on capturing potentially short response times within the catchment during and after heavy precipitation events.

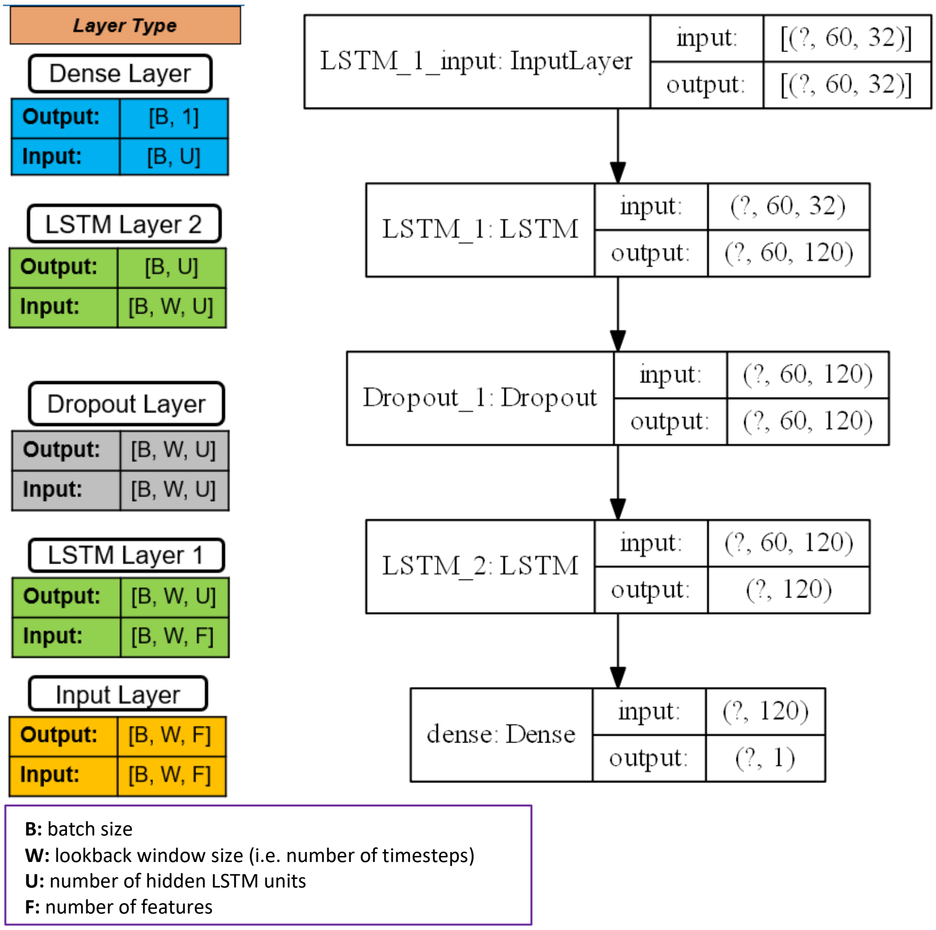


Figure 16: Computational graph that visualizes the array shape manipulation during a forward pass through the LSTM model. In this example 32 input features are processed with a window size 60 time steps. The defined number of LSTM units is 120. The "?" corresponds to the batch size but is indicated a variable sized within the network, since the batch size can be varied between model run calls.

The batch size is set to 20 for daily values and to 200 for hourly resolution, respectively. This increase in batch size while lowering the temporal resolution is due to higher computation efficiency and reduced training time of the model on hourly data. The number of internal LSTM units, that defines the capacity of cell states in the model, as mentioned in section 2.2.1 is set to 120 for daily and hourly resolution. With respect to the iterations the model is trained, 25 Epochs turned out to be adequate in terms of run time and convergence without leading to an overfit of the model (Unnikrishnan 2019). In order to determine to which degree the LSTM model should be regularized to be less susceptible to overfitting, is defined by dropout rate. As mentioned in section 2.1.3, this rate denotes a certain percentage of values within the output of the cell states, that are set to zero during training. This forces the network into a more robust feature learning (Goodfellow et al. 2016). In both model configurations this value is set to 45%. The model is trained to minimize the Mean Absolut Error (MAE) between observed and predicted discharge. This objective function is explained in detail in the next section.

Table 2: Defined hyperparameters based on hyperparameter optimization conducted by Unnikrishnan (2019)

Hyper- Parameter	daily	hourly
Lookback (Number of timesteps/ Window size)	120 (-> 4 months)	120 (-> 5 days)
Batch Size	20	200
Number of internal LSTM units in each Layer	120	120
Epochs	25	25
Number of LSTM Layers	2	2
Dropout-Rate	45%	45%
Loss Function	Mean Absolut Error	Mean Absolut Error
Kernel & Bias - Initializer	Random Uniform	Random Uniform
Optimization Algorithm	Adagrad	Adagrad

To speed up training and to break symmetry between LSTM units, kernel and bias initializer are commonly used, which pre-set the weights with a specific distribution instead of setting them to zero (Géron 2019). According to Unnikrishnan (2019), the best performance was achieved with a random uniform initialized distribution. As mentioned in section 2.1.3, a significant fraction of the uncertainty in in LSTM models is introduced by these random weights/bias initializations. In order to eliminate the source of randomness in the model, an identical seed value is set for every model configuration before training. This should ensure that the weights/bias vectors are initialized with the same values for every model run. Additionally, since the lookback windows within one batch are randomly shuffled, there is also a seed value determined, ensuring that the windows are passed to the network always in the same order. As a result a LSTM model produces consistently the exact same results when trained with identical model configurations. However, the issue that the defined seed values might not lead to the optimal model outcome is neglected, since finding the optimal seed value is not part of the objectives. Finally, the optimization algorithm has to be chosen, which the model uses during the training phase to compute the gradients and weight updates. The chosen *Adagrad* algorithm allows the learning rate parameter to be adaptive based on model parameters. As stated in Goodfellow et al. (2016), the mechanism in this algorithm allows, that the learning rate is decrease relatively to the values of the partial derivatives of the loss function. The larger the partial derivatives the more rapid the decrease in their learning rate. The effect is a that greater progress of finding the global minimum is made in the more gently sloped directions of parameter space. In contrast, this is also a drawback, since the learning rate is always decaying, with the result, that the model converges more slowly (Goodfellow et al. 2016).

To implement the above described LSTM model *Keras*[15] is used, which is an open-source neural-network library written in Python[16]. It is a high-level neural network API capable of running on top of many different machine learning frameworks. *TensorFlow*[17] is the framework of choice in this thesis. It is a free and open-source software library for dataflow and differentiable programming across a range of ML tasks. It is developed by Google and is a useful tool for research work conducted in any fields of machine learning. *TensorFlow* also provides GPU (graphical processing unit) support, which enables the user to rapidly reduce training time of NNs while increasing model complexity. To evaluate the performance of the models there are different validation methods used in this thesis, which will be examined in the following section.

3.6. Validation of the different Models

As already mentioned in section 2.2, the normal way of evaluating your model is to split the data into training, validation and test set, since the goal in machine learning is to achieve models that generalize well on data, which has the model never seen before (Chollet 2018). There are numerous ways of doing this, but in this thesis the following three methods are applied.

NORMAL HOLDOUT

The Out-of-samples approach, also called *Normal Holdout method*, is the simplest evaluation protocol and is used as a standard method in machine learning. This method has been traditionally used to estimate predictive performance in time-dependent data. Essentially, normal holdout methods split the time-series into two parts: an initial training period in which a model is fitted to the data, and a testing period of the last part of the time series, which is held out for estimating the model performance on new data. In order to prevent information leaks into the test dataset while tuning your model, it is common practice to further subdivide the training dataset and reserve a validation set (Chollet 2018).

To visualize this principle, Figure 17 is provided. An overview of how the daily and hourly data are subdivided in each individual set is given in Table 3. This table also shows the number of available samples within each subset.

---

[15] https://keras.io/; Version: 2.3.0-tf

[16] https://www.python.org/; Version: 3.8.5

[17] https://www.tensorflow.org/; Version: 2.2.0

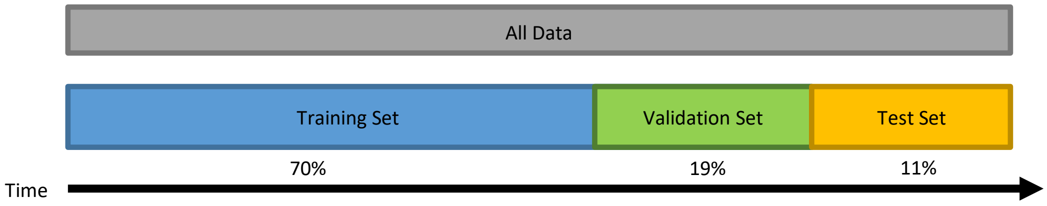


Figure 17: Visualization of the normal Holdout Method. The dataset is split into three subsets.

To get an idea on how these splits appear on the dataset that is used in this thesis, in Figure 18 the normal holdout method is visualized on the hydrograph of daily averaged values of the Marienthal station. This is the target station at which the model should predict the discharge. The training, validation and test set are colored individually. On the right-hand side of the Figure 18 boxplots for the discharge of each particular set is shown. Boxplots display the variation in samples of a statistical population without making any assumptions of the underlying statistical distribution. Boxplots are composed of a rectangular box, a horizontal line inside the box and two horizontal lines above and below the box, which are connected to box boundaries. These lines outspreading form the boxes, called whiskers, indicating the variability outside the interquartile range, which is derived by the 75th percentile (upper bound) and the 25th percentile (lower bound) of the data, respectively. The line inside the box corresponds to the median of the dataset. In addition, outliers are plotted as individual points. The spacings between the different parts of the box show the degree of dispersion and skewness in the data. For improving the interpretability and the appearance of the boxplot, a log-transformation was previously applied to the discharge data.

Table 3: Overview of Training, Validation and Test set size regarding the normal Holdout Method

Data		Training	Validation	Test
Daily	*samples*	3317	900	522
	[%]	70	19	11
Hourly	*samples*	79615	21609	12512
	[%]	70	19	11

The *Normal Holdout method* preserves the temporal order within the measurement series, which might be an important property to cope with the dependency among measurements and attributes the potential temporal correlation between consecutive time steps (Cerqueira et al. 2019). Nevertheless, it suffers from on disadvantage: if little data is available, then the validation and test sets may contain too few samples to be statistically representative of the data at hand. As a result, this implies statistical uncertainty around the estimated average test error (Goodfellow et al. 2016).

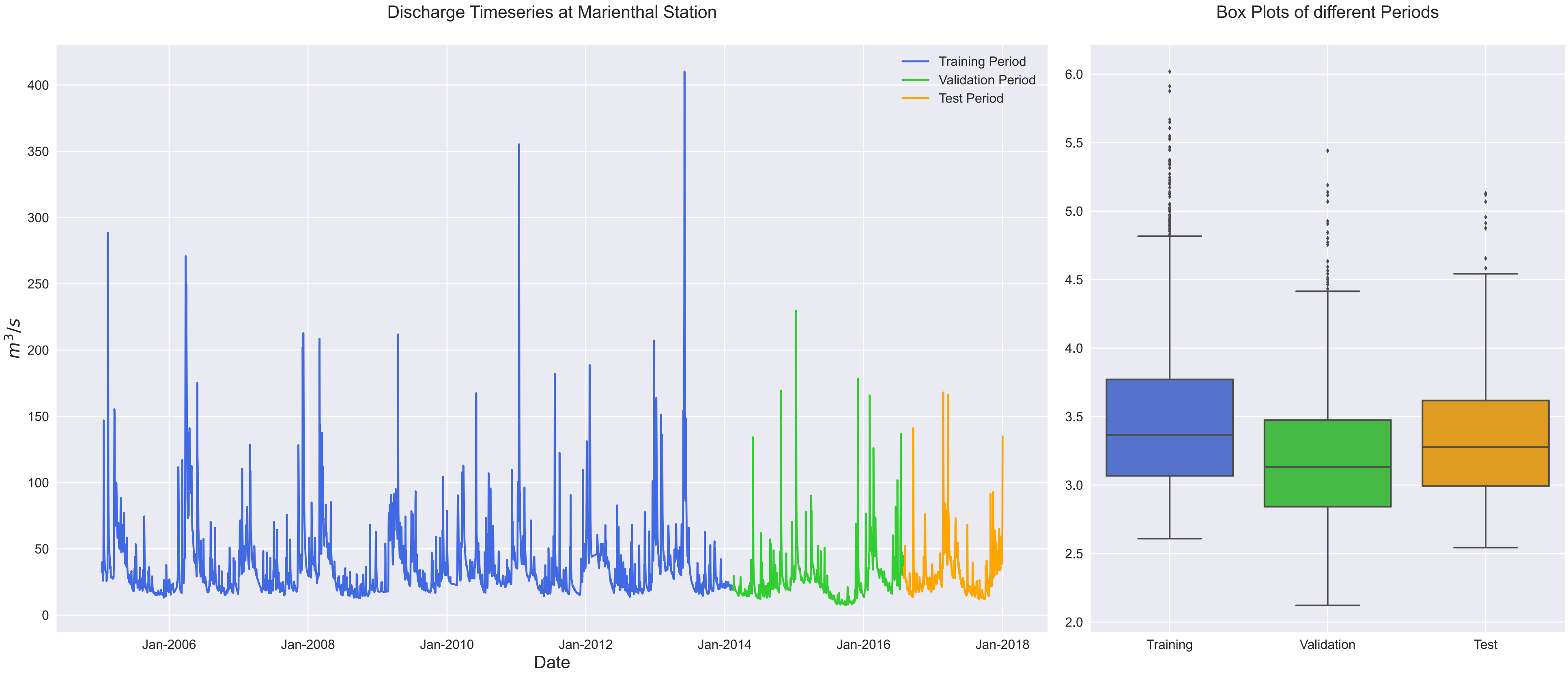


Figure 18: Normal Holdout Method visualized on the discharge hydrograph at Marienthal station (left) and corresponding Boxplots (right) based on the underlying data distribution (log-transformed) of the three different subsets

This problem is intensified, if the split points of the validation and test set are not properly chosen. The split point should be selected in a way that the variability and the frequency distribution of the individual sets are similar. As a consequence, the performance on the validation set might change a lot depending on which time period is used for validation and which for training. Hence the model loss on the validation set might have a high variance with respect to the validation split point (Chollet 2018). The following two methods are based on the idea repeating the training and validation computation on different chosen subsets or splits of the complete dataset.

REP-HOLDOUT

In order to produce a robust estimate of predictive performance, it is recommended to employ validation strategies, regarding training, validation and test set separation, on multiple training and test periods. Among the strategies, the split point could be altered during multiple training runs, for instance by selecting the point within a growing or sliding window. For a more general setting one can also adopt a randomized approach, indicating a randomly selected split points (Cerqueira et al. 2019).

In this thesis, a strategy is used, in which random sub-sampling is repeatedly applied to the dataset. This is implemented by utilizing the N*ormal Holdout method* several times using different sized and possibly overlapping time periods for training and validation. This approach is called *rep-holdout method* and its principle is illustrated in Figure 19, that shows one iteration of training the model on chosen split point ***p***.

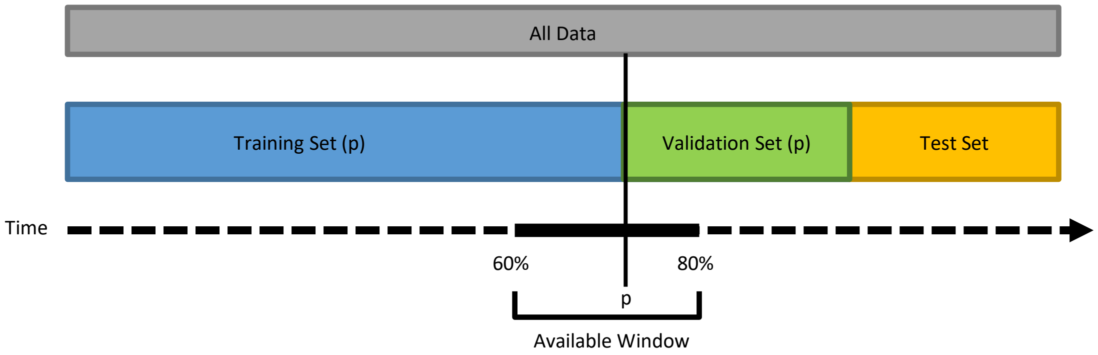


Figure 19: Visualization of the Rep-Holdout Method. Split point p is randomly selected within a specified window. Percentages correspond to the total number of samples of the combined Training and Validation set and define the range of the window. For each training of the model a new split point is selected.

In this validation method, for one iteration, a point ***p*** is randomly chosen from a defined window within the time series constrained by the size of the training and validation set. The available window is set to a range from 60% to 80% of the total training data size (training + validation set). This should assure, that both the training and validation set have still sufficient length. The split point ***p*** then divides the data into two sets, while the preceding part is used for training and the subsequent one for validation. Furthermore, only an even number of iterations are used in order to split the data into training and validation subsets. Hereby, half of the iterations are splited below the 70% mark (which is the default for the normal holdout method) and the other half is splited above the 70% mark, but still within the defined window range. This is done to verify that some of the resulting training and validation sets contain more or less samples compared to the original normal holdout method. Moreover, the resulting subsets should not be too similar in size and the independence between the sets should be increased. After the model is trained on the individual training sets, its performance is evaluated on a test set. In this thesis, this test set is chosen to stay the same for all iterations, for a better comparison to models validated by the other two methods. In order to get the overall performance across all iterations, the performance indicators are averaged. The *Rep-Holdout method* is particularly applied in the preliminary analysis (see section 4.2). The method applied to the hydrograph at Marienthal station is pictured in Figure 39 with a total of 6 iterations, which is attached to the Appendix. In this figure the different training and validation sets for all iterations are displayed along with the specific frequency distributions, illustrated by the boxplots on the right.

K-FOLD CROSS VALIDATION

Another validation approach that is applied in a specific step in the sensitivity analysis (see section 4.3.2) is called *k-fold cross validation*. The typical approach when using *k-fold cross-validation* is to split the data in ***k*** equally sized and non-overlapping folds or blocks. Each fold is a subset of the data comprising of training and validation set. For each fold ***i***, the model is trained on the remaining ***k – 1*** folds and fitted to remaining fold ***i*** for validation**.** Thus, after splitting the data into K-folds, each fold is iteratively picked for validation. The performance obtained by *k-fold cross-validation* is then the average of the performance estimate computed on the final evaluation on a left-out test set across ***k*** trails (Cerqueira et al. 2019, Goodfellow et al. 2016). Schematically, *k-fold cross-validation* looks like depicted in Figure 20.

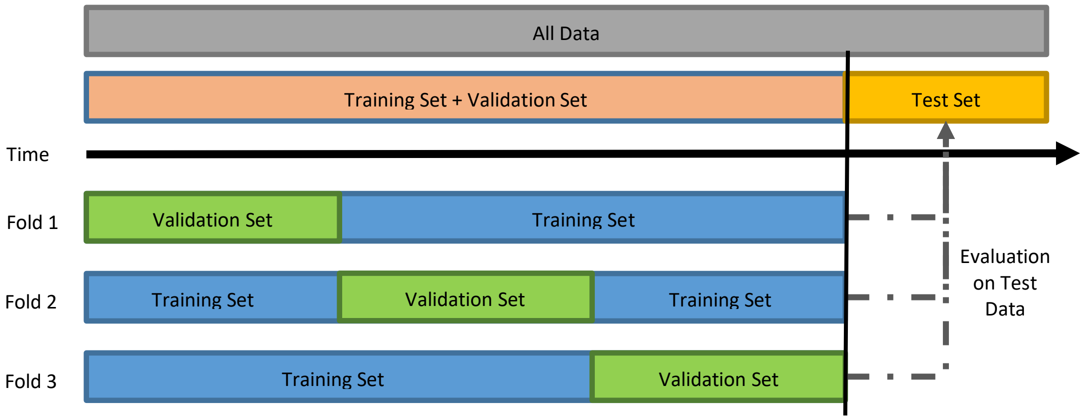


Figure 20: Visualization of the k-fold cross validation. In this example the training and validation set is split in 3 folds, which results in three different validation sets. The three model runs are evaluated on the same test set.

In Figure 40, attached to the Appendix, the *k-fold cross validation* with three folds is shown on the discharge hydrograph for Marienthal station along with the boxplots of the individual subsets of data. It can be seen that the validation set is shifted to the right throughout the training period. The test set remains the same for each iteration. The size the validation and training subsets is defined by the total number of available samples and the desired number of folds. This method is helpful when the performance of the model shows significant variance based on your train-test split (Doycheva et al. 2017). Furthermore, cross-validation is commonly applied to mitigate any bias caused by the particular subset of sample (Chollet 2018). In case of limited training and test datasets, the cross-validation guarantees that the results obtained for the specified test set would be the same as results obtained by independent test sets. However, one drawback in this approach is, that it can be computationally expensive (Doycheva et al. 2017, Chollet 2018).

PERFORMANCE METRICS

After the model is fitted, i.e. calibrated to the validation data using the training set, the model with its adjusted weights is used to predict the discharge hydrograph for the test period. Since the model still operates on scaled data, the predicted values have to be inverse transformed. Then the performance of the model can be evaluated. In general, the application of rainfall-runoff models requires some form of performance estimation to ensure reliable discharge simulations for the catchment of interest. Performance estimation is usually based on comparing simulated and observed discharge using a goodness-of-fit measure (Pool et al. 2018). Because no single evaluation metric can fully capture the consistency, reliability, accuracy, and precision of a streamflow model, a variety of performance metrics for model benchmarking are used (Kratzert et al. 2019c). Evaluation metrics utilized in this thesis to compare models are listed in Table 4. Some of the metrics focus specifically on assessing the ability of the neural network to model high-flows and low-flows, as well as assessing overall performance using a decomposition of the standard squared error metrics that is less sensitive to bias (Gupta et al. 2009).

Table 4: Overview of the used performance metrices in this thesis

Metric	Equation	Reference
Maximum Error (Max Error)	$Max\ Error = max\left(\left\|Q_{sim,t} - Q_{obs,t}\right\|\right)$	[18]
Relative Error in Volume (REV)	$REV = \frac{\sum_{t=1}^{n}\left(Q_{sim,t} - Q_{obs,t}\right)}{\sum_{t=1}^{n} Q_{obs,t}} * 100\%$	(Jabbari and Bae 2018)
Median-Absolute-Error (MeAE)	$MeAE = median\left(\left\|Q_{sim,1} - Q_{obs,1}\right\|, \dots, \left\|Q_{sim,t} - Q_{obs,t}\right\|\right)$	[19]
Mean-Absolute-Error (MAE)	$MAE = \frac{1}{n}\sum_{t=1}^{n}\left\|Q_{sim,t} - Q_{obs,t}\right\|$	(Wang 2006)
Mean-Squared-Error (MSE)	$MSE = \frac{1}{n}\sum_{t=1}^{n}\left(Q_{sim,t} - Q_{obs,t}\right)^2$	(Gupta et al. 2009)
Root-Mean-Squared-Error (RMSE)	$RMSE = \sqrt{MSE}$	(Wang 2006)
Nash-Sutcliff-Efficiency (NSE)	$NSE = 1 - \frac{\sum_{t=1}^{n}\left(Q_{obs,t} - Q_{sim,t}\right)^2}{\sum_{t=1}^{n}\left(Q_{obs,t} - \mu_{obs}\right)^2} = 1 - \frac{MSE}{\sigma_{obs}^2}$	(Gupta et al. 2009)
Nonparametric Kling-Gupta-Efficiency (KGNP)	$\alpha_{NP} = 1 - \frac{1}{2}\sum_{k=1}^{n}\left\|\frac{Q_{sim}(I(k))}{n\mu_{sim}} - \frac{Q_{obs}(J(k))}{n\mu_{obs}}\right\|$ $\beta = \frac{\mu_{sim}}{\mu_{obs}}$ $r_s = \frac{\sum_{t=1}^{n}(R_{obs,t} - \bar{R}_{obs})(R_{sim,t} - \bar{R}_{sim})}{\sqrt{\left(\sum_{t=1}^{n}(R_{obs,t} - \bar{R}_{obs})^2\right)\left(\sum_{t=1}^{n}(R_{sim,t} - \bar{R}_{sim})^2\right)}}$ $KGNP = 1 - \sqrt{(\beta - 1)^2 + (\alpha_{NP} - 1)^2 + (r_s - 1)^2}$	(Pool et al. 2018)

In Table 4 ***n*** is the total number of samples, i.e. time steps, $Q_{sim,t}$ is the simulated discharge value at time step ***t***, $Q_{obs,t}$ is the observed discharge value at time step ***t***, and μ_{obs} and σ_{obs} are the mean and standard deviation of the observed values, respectively. Further, $I(k)$ and $J(k)$ are the time steps when the k^th^ largest flow occurs within the simulated and observed time series. The Spearman rank correlation (r_s) was calculated on the ranks of the observed R_{obs} and simulated R_{sim} discharge time series, where $\bar{R}_{sim}$ and $\bar{R}_{obs}$ are the mean ranks.

The max error function computes the maximum residual error, which is the worst-case error between the predicted and measured discharge. Another performance metric is the relative error of volume (REV), which is an indicator the total model accuracy of predicting discharge volume over the complete time series. It is calculated by ratio of the absolute error of the simulated and observed discharge. Negative percentages indicate that the total runoff volume is underestimated, whereas the ideal value is zero (Jabbari and Bae 2018). The median absolute error (MeAE) is calculated by taking the median of all absolute differences between the target and the prediction. It is particularly interesting since it is robust to outliers. It is unit-dependent,

---

[18] https://scikit-learn.org/stable/modules/model_evaluation.html#max-error

[19] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.median_absolute_error.html#sklearn.metrics.median_absolute_error

and the optimal value is zero. An additional very important performance metric is the mean absolute error (MAE) function, since it is used as the loss function for the LSTM model. It can be seen as a risk metric corresponding to the expected (average) value of the absolute errors. Again, the optimal value is zero and the MAE has the same unit as the underlying data.

The mean-squared-error (MSE) metric and its related normalization, the Nash–Sutcliffe efficiency (NSE) are the two criteria most widely used for calibration and evaluation of performance of hydrological models. With respect to optimization MSE is subject to minimization, whereas NSE is subject to maximization. In this thesis the main focus is on the NSE, but the results can be generalized to MSE and similar criteria such as RMSE (Gupta et al. 2009). The values of MSE and RMSE are unit-dependent and vary on the interval [0 to ∞], whereas NSE is dimensionless, being scaled onto the interval [-∞ to 1]. Further, NSE can be interpreted as a classical skill score, where skill is interpreted as the comparative ability of a model with respect to a baseline. According to Gupta et al. (2009) the baseline of NSE is considered to be the mean of all observations, which implies that if the NSE is less than 0 the model is no better than using the observed mean as a predictor. As a consequence, this can lead to overestimation of model skill for highly seasonal variables such as runoff in snowmelt dominated basins (Gupta et al. 2009). The MSE, as well as the NSE, consist of three components. On the one hand MSE can be decomposed into a mean, variability and dynamics term, whereas on the other hand NSE can be splited in terms of correlation, bias and a measure of variability (Pool et al. 2018). Estimating model parameters by optimizing the MSE or the NSE is critical in several ways. Gupta et al. (2009) demonstrated that a high model performance for discharge dynamics is directly linked to an underestimation of discharge variability. Furthermore, the decomposition shows that in order to maximize NSE the variability and thus the runoff peaks have to be underestimated. Additionally, this means that in basins with high runoff variability the bias component will tend to have a smaller impact on the computation and optimization of NSE, possibly leading to model simulations having large volume balance errors (Pool et al. 2018, Knoben et al. 2019). Because of these reasons Gupta et al. (2009) suggest an objective function, the so called Kling-Gupta efficiency (KGE), that is based on an improved combination of the three meaningful components of the MSE, which are the variance ratio α, the mean ratio β and the linear correlation coefficient *r*. The alternative model performance criterion KGE is easily formulated by computing the Euclidian distance of the three components from the ideal point (Gupta et al. 2009). Nevertheless, Knoben et al. (2019) showed that the interpretation of the KGE should not be guided by the understanding of NSE values, since these two metrics cannot be compared in a straightforward manner. However, since assumptions on linearity and normality of the data and on the absence of outliers were made during formulation of the KGE, a modification was proposed by Pool et al. (2018) to take into account that discharge time series and model simulation errors are often highly skewed. The modification comprises adopted efficiency decompositions by reformulating the variability and the correlation term of the KGE in a non-parametric form. This is achieved by using the flow–duration curve (FDC) as a non-parametric alternative to the standard deviation. The FDC describes the relationship between the frequency and magnitude of streamflow and is an indicator of flow variability across all flow magnitudes of a catchment, whereas the standard deviation only reveals the variability of flows around the mean flow. Besides this adaption, instead of the Pearson correlation coefficient the Spearman rank correlation is used to describe discharge dynamics, which is less sensitive to extreme values and hence providing a more robust characterization of the correlation (Pool et al. 2018). Further, it has to be mentioned, that the KGE as well as the non-parametric KGE (KGNP) are both unit-independent and are scaled onto the same interval as the NSE ([-∞ to 1]).

Overall, with these 8 different performance metrics the ability of the model to capture different aspects of the discharge hydrograph can be evaluated. Nevertheless, to get an impression of how well the LSTM model is able to predict the discharge for the *Regen* catchment compared to another type of hydrological rainfall-runoff models, a benchmark model is used for additional validation. In this thesis, the Large Area Runoff Simulation Model (LARSIM) serves as benchmark, which was specifically calibrated for the study area. This physically based model is described in the following.

LARGE AREA RUNOFF SIMULATION MODEL[20]

The LARSIM model is a water balance model, which describes hydrological processes in a specific catchment area. The application is not limited to the simulation of larger areas but can be applied to a whole range of different catchment scales. This scale includes subarea sizes ranging from a few hectares up to several hundred square kilometers. However, it is usually operating on a meso-scale level. LARSIM represents a deterministic and conceptual modeling approach, while considering special dimensions during modeling. Thus, the distributed model quantifies the spatial and temporal distribution of important hydrometeorological data and hydrologic conditions like precipitation, evaporation, seepage, water storage in the catchment and runoff. Water balance models, as an extension of conventional precipitation-runoff models, allow continuous, process-oriented simulations and forecasts for the entire runoff process. LARSIM describes the following water balance subprocesses: interception, evapotranspiration, snow accumulation, compaction and melt, soil water retention, storage and lateral water transport, as

---

[20] This section is summary based on Leibundgut et al. 2006.

well as flood-routing in channels and retention in lakes. On the one hand water balance models use catchment specific data like elevation, land use, soil parameters and channel geometry. On the other hand, LARSIM uses hydrometeorological time series data including precipitation, air temperature, air pressure, air humidity, wind speed, global radiation, duration of sunshine, water temperature and discharge. The model can be operated either on grid-based subareas or on subareas according to hydrologic sub-catchments, where interception, evapotranspiration, snow processes and soil water storage are modelled separately within a subarea. These output results of hydrologic sub-models for different types of land use and field capacities (referred to as grouped response units (GRU)) without regarding their spatial allocation within the subarea. The runoff resulting from the different GRU of a subarea is separated into three soil storages: on for direct runoff, one for interflow and one for groundwater runoff. The water release from these three storages forms the total runoff from a subarea. Besides the use of LARSIM as a water balance model with a continuous simulation, it can also be used as an event-based flood forecast model. In case of LARSIM is used as a flood forecast model while snow-influence is irrelevant, it requires only precipitation as meteorological input. As flood forecasting model LARSIM can have different possible time intervals ranging from 5 min up to 1 day. Several German flood forecasting centers enhance the LARSIM model for an operational continuous forecast of discharge. The operational calculation mode differs from offline simulation runs insofar, as it comprises a combination of simulation and forecast in each run. The computed period can thus be separated into two phases, whereby during the first one model-parameters are optimized by minimizing the deviation between simulated and measured data to improve the quality of the forecast, and the latter one, whereby the actual discharge is forecasted. During automated operational forecast mode, gaps in hydrometeorological data input will be automatically identified and filled by using suitable interpolation techniques. For the operational forecast, measured discharge time series (up to two days) at a gauge with respect to low, mean or high flow conditions is analyzed if data is available and of good quality, which is verified by the model itself. The automated model optimization within LARSIM evaluates MS-differences and subsequently applies data assimilation and other different types of correction methods depending on the actual range of discharge forecasting (Leibundgut et al. 2006).

For the LARSIM model, that is used as a benchmark in this thesis, exactly the same data basis is used. This means the LARSIM model is provided with the same meteorological input data on hourly basis for the same time period. The model is used in operational forecast mode and it had received true observations of meteorological input data, which was assumed to be “forecasted” withing the lead time. Real forecasted meteorological data, e.g. by the German Meteorological Service, was not considered in order to eliminate the uncertainties within this forecast. For each forecast run the LARSIM model predicted the discharge at Marienthal station with a lead time of 24 hours. A new forecast run started every day at 5:00 am. Prior to each start of a new 24h discharge simulation, an ARIMA correction procedure was applied based on an internal calibration period over 53 hours. During this calibration period internal parameter of the LARSIM model are adjusted and the predicted discharge hydrograph is shifted into the measured discharge at the start of each new forecast. The LARSIM input data, which serve as data basis in this theses, as well as the LARSIM simulation data was provided by the Flood Forecast Center from the Bavarian Water Authorities[21].

3.7. Methods for Multi-Step Forecasting

In general, forecasts on a daily basis are reasonable in a medium- to long-range forecast perspective, however, daily input resolution mutes sub daily dynamics in the data that might be influential on the temporal fluctuations in the discharge hydrograph. Hence, predictions on the daily basis are often too coarse for a short-range forecast to reliably provide indications for action measures in the view of flood management (Gauch et al. 2020).

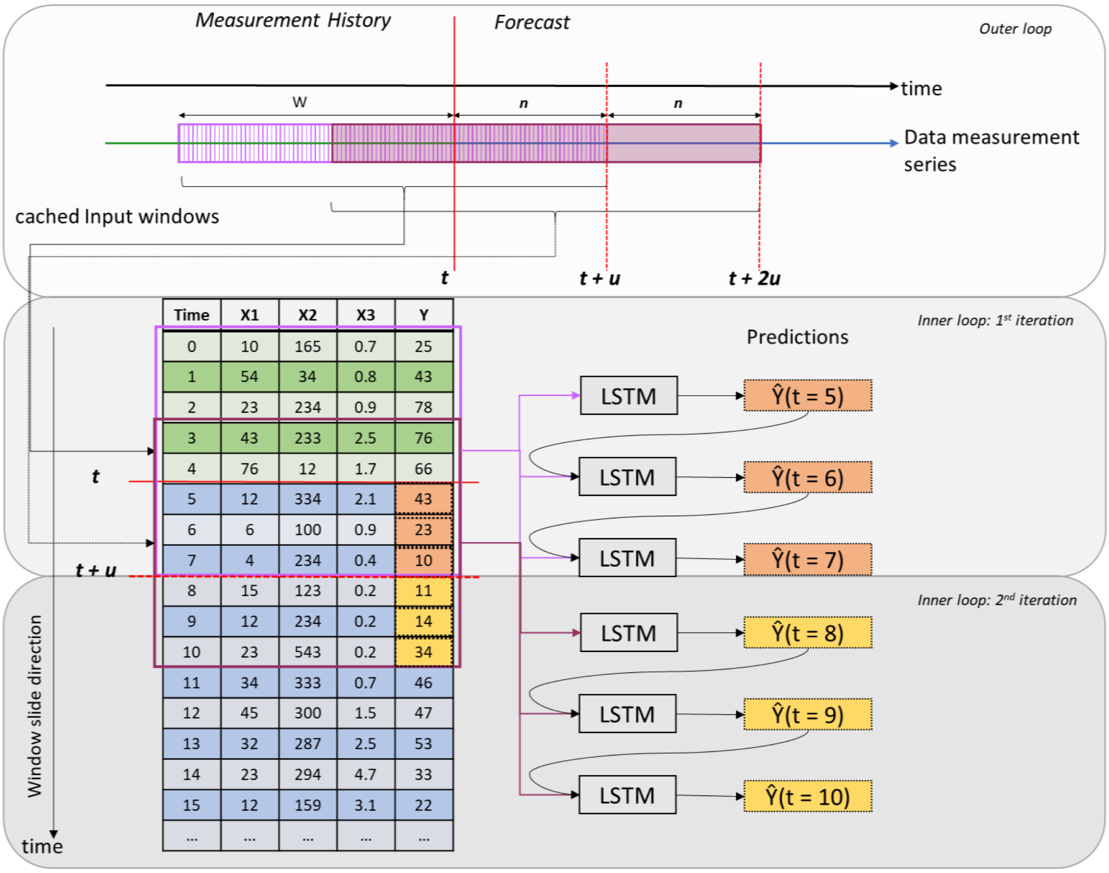
As a consequence, rainfall-runoff models used for forecasting discharges in a real-world operational scenario need to be able to predict the potential runoff not in single, but with multiple time steps ahead. For example, the LARSIM model used as a benchmark model in this thesis, is applied in such operational flood forecasting scenarios. As part of the objectives within this thesis, it is investigated if LSTMs reach comparable accuracy compared to the LARSIM model, when forecasting multiple time steps ahead.

To test this, two methods, which potentially reflect two contrasting prediction principles, are applied in this thesis to forecast discharge for different lead times (12, 24, 36 hours). The first method makes use of multi-step single shot (or one shot) LSTM models. The principle here, as explained in section 2.2.2, is that the model predicts the runoff for e.g. the next 12 time steps, based *only* on the input data of the past 120 hours (size of the lookback window) in one shot. Regarding each tested lead time, one LSTM model is specifically trained to predict one particular number of time steps. In other words, three different LSTM models are created, individually trained to predict a particular number of steps into the future (one for each lead time). In general, one major difference in the prediction procedure between the LARSIM and the LSTM model, despite the fundamental conceptual structure, is the fact that the LARSIM model does incorporate forecasted meteorological data. The LARSIM model, as noted in section 3.5, had received measured data during the forecasting period of 24 hours.

---

[21] *Bayrisches Landesamt für Umwelt (LFU)*

In contrast, the single shot LSTM models do not consider *any* additional data during the forecasting timesteps, i.e. the models get absolutely no information about precipitation or other meteorological conditions during the prediction period. Therefore, as the second method a customized prediction loop for LSTM models is introduced. This customized prediction loop should allow the model to also consider forecasted (or measured) meteorological data during the prediction process. Hence, this should provide a method to better compare the prediction accuracy of the LSTM model against the LARSIM model and further provide one possible approach towards an operational flood forecasting setting. The principle of this customized prediction loop as visualized in Figure 21.



Time	X1	X2	X3	Y
0	10	165	0.7	25
1	54	34	0.8	43
2	23	234	0.9	78
3	43	233	2.5	76
4	76	12	1.7	66
5	12	334	2.1	43
6	6	100	0.9	23
7	4	234	0.4	10
8	15	123	0.2	11
9	12	234	0.2	14
10	23	543	0.2	34
11	34	333	0.7	46
12	45	300	1.5	47
13	32	287	2.5	53
14	23	294	4.7	33
15	12	159	3.1	22
...	...	...	...	...

Figure 21: Schematic illustration of the customized prediction loop. As an example, the procedure of the loop is demonstrated for two iterations, whereby the model predicts 3 steps ahead (*n)* for each iteration based on cached input windows from the measurement series (inner loop). For illustration purposes, an example data frame is depicted, which consists of random samples from three input features (X1,X2,X3) and one target variable(Y). Green shaded rows imply measured values, whereas blue shaded rows are assumed to be forecasted values. After each time step the predicted value is replace with the target value of the next input window. The second iteration is restarted, e.g. every 3 hours (update step *u) (outer loop)* and the forecasted values could be replaced by true observations within the measurement series before getting cached again to compute the next n steps ahead (inner loop).

The introduced prediction loop separates the training procedure and prediction process of a LSTM model. To put it differently, the LSTM model is used for another scenario as it is trained for. Hence, in order to apply the customized prediction loop, first a LSTM model needs to be trained to predict single steps on measured data (see Figure 9 in section 2.2.2). Then this trained model is used to predict a defined number of steps ahead (hereafter referred do as: prediction steps ≡ lead time ***n***) in a customized prediction loop, which can be subdivided into two separate loops. The inner loop, starting at time ***t,*** basically cache the number of future lookback windows in consecutive order from the measurement series (in this thesis: test data) according to the defined lead time (***n***). Then for each of these windows the model predicts the discharge of the next hour **(*t* + 1[h])**. This predicted value is used to replace the true observed discharge value of the last time step within the next window. This is repeated for the number of defined lead time steps (***n***). Once the inner loop has finished, the outer loop shifts the new starting point of the forecast within the test period by an independently defined number (hereafter referred do as: update steps ***u***). Basically, this defines how frequently the model restarts a new forecast of ***n*** steps ahead. This principle is demonstrated in an example, in which the customized prediction loop with a defined lead time and a defined updated step of 3 hours (***n***, ***u* = 3**) for the first two iterations is shown (see Figure 21). The first iteration of the inner loop predicts the discharge for time steps ***t* + 1[h], *t* + 2[h], *t* + 3[h]** (starting at ***t = 0***), whereas the second iteration starts at ***t* + 3[h]** (= ***t + u***) and forecasts the discharge for the time steps ***t* + 4[h], *t* + 5[h], *t* + 6[h],** respectively. If the update step is lower than the defined lead time the customized prediction loop provides multiple discharge predictions for the *same* time step. Moreover, right before each iteration of the inner loop starts, in which new data is cached, the herby used true observations could be potentially exchanged with forecasted meteorological data. However, such data was not available and further, to maintain the comparability to the LARSIM model, true measurements were considered as theoretically “forecasted” input data during, just as it was done for the benchmark model. [22]

In order to have an indication of how well the different models predict the discharge for several time steps ahead and to validate model performance changes, the same performance metrices as presented in section 3.5 are applied. The scores are calculated by the same functions, however the measured and predicted values that are compared one by one are different for this multi-step prediction scenario. The performance metric scores are calculated separately for every individual prediction step in the future. In other words, e.g. for a 12 step ahead prediction, the observed discharge value at ***t* + 1[h]** is compared to the predicted discharge of the LSTM model at ***t* + 1[h],** where ***t*** corresponds to the starting time stamp of a new prediction. However, calculating this only once would result in the performance metric scores for only the first prediction step. Hence, the calculation is repeated for the remaining prediction steps (hours) from ***t* + 2[h]** up to ***t* + 12[h].** Therefore, 12 single scores per performance metric

---

[22] For a more detailed understanding please refer to the pseudo code, provided in Table 13, attached to the Appendix.

are obtained (one score for each time step ahead). The performance metric scores are calculated analogously for different lead times.

UNCERTAINTY ESTIMATION

Hydrological flood forecasting involves the prediction of complex processes, occurring at a variety of scales (Leandro et al. 2019). Hence, streamflow flood forecasting is a complicated task with multiple sources of uncertainty. These uncertainties may arise from various sources like, for example, model parameters, model structure or hydrological input data. Discharge prediction estimates are commonly given in ranges, indicated by upper and lower bounds, which may result from a reasonable fit of several distributions of multiple model runs to the observed data (Viglione et al. 2013). In operational flood forecasting decision making is more difficult under large uncertainty, making it essential to somehow incorporate methods, based on forecast performance statistics or other principles, into hydrological discharge modeling (Leandro et al. 2019). Thus, estimating the predictive uncertainty is important for assessing how much to trust the forecast produced by a model. However, there are no standard tools in the field of deep learning to capture model uncertainties within neural networks (Gal and Ghahramani 2015). Therefore, integrating available methods in the field of deep learning with respect flood statistics and/or probabilistic modeling into this study would go beyond the objectives and scope of this thesis. Nonetheless, some efforts were made to reduce at least some sources of randomness within the LSTM model, as mentioned in section 3.4. Additionally, true measurements were assumed as “forecasted” observations during the multi-step prediction analysis, which eliminates the uncertainty within the meteorological input data considered during the lead time of the forecast. Of course, this uncertainty in the input data is not entirely eliminated, since there could potentially remain sources of uncertainty introduced by the measuring device and/or technique.

4. Results

The structure of this chapter follows the visualized procedure (see Figure 13: Flow chart of the consecutive investigation and analysis steps carried out in this thesis) in section 3.1. It starts off with the first model test on two stations close together, followed by presenting the results of the preliminary analysis, in which different scaling and missing value methods are compared. After identifying the optimal configurations of model settings, the influence of input feature selection on model performance is highlighted. Additionally, the improvement in performance by tuning specific hyperparameters is described. All model tests up this point are conducted as single step predictions (see section 2.2.2). Hence, as a final step, the performance of the LSTM with respect to predicting multiple steps into the future is presented. It is important to mention once again that for the LSTM (in contrast to the LARSIM model) meteorological data during the forecasting period (time steps the model predicts ahead) was not considered within section 4.1 till section 4.4.

4.1. First Model Test

Within a catchment, the contribution of the direct runoff is the most influential factor during the rainfall-runoff process on the temporal dynamics of a discharge hydrograph. The direct runoff is mainly generated by highly intensive precipitation events. Hence, this relation should be reflected in the hydrograph signatures in a way. The effect is more easily observable in stations that are close to one another. To test whether the model with its preliminary settings is capable of capturing such short-term changes in the hydrograph, two stations are selected, which are located close to each other. As the precipitation measurement location, the station at *Neukirchen bei Heiligen Blut* is selected, whereas the target discharge gauge is the *Leming* station, which is located approximately 4.1 km further downstream near the river *Freybach*. The location of both stations is depicted in Figure 22, framed by an orange square.

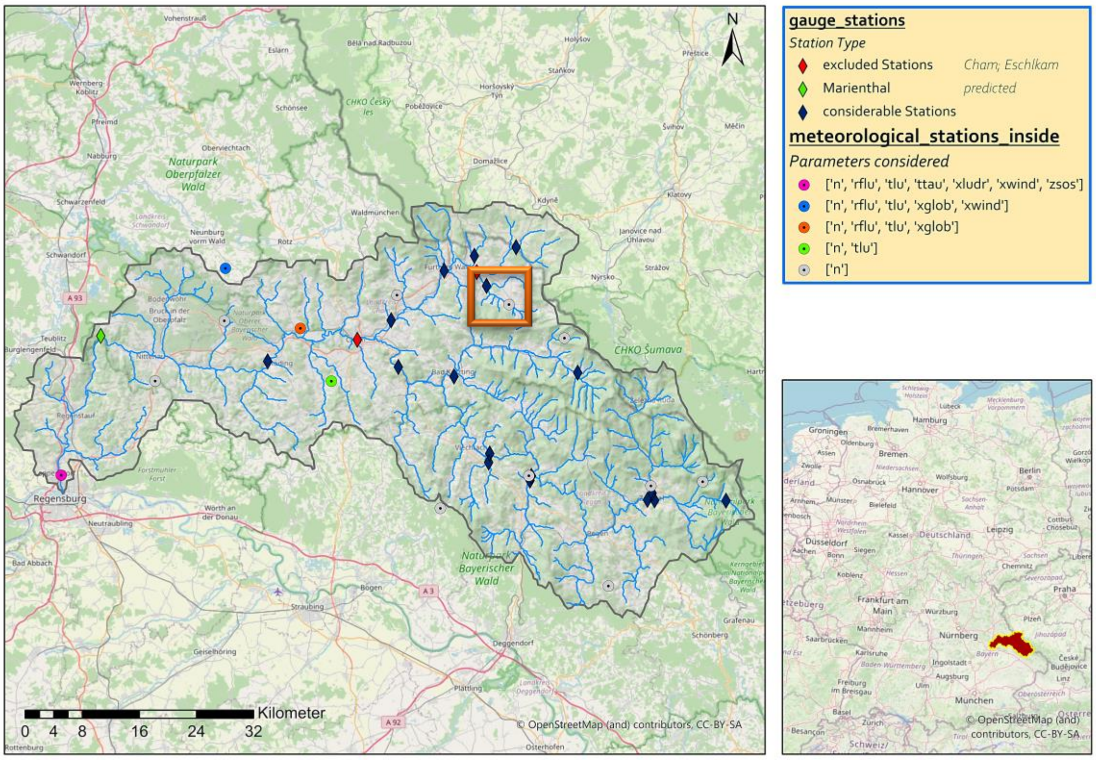

N
gauge_stations
Station Type
♦ excluded Stations Cham; Eschlkam
♦ Marienthal predicted
♦ considerable Stations
meteorological_stations_inside
Parameters considered
● ['n', 'rflu', 'tlu', 'ttau', 'xludr', 'xwind', 'zsos']
● ['n', 'rflu', 'tlu', 'xglob', 'xwind']
● ['n', 'rflu', 'tlu', 'xglob']
● ['n', 'tlu']
● ['n']
Kilometer
0 4 8 16 24 32
© OpenStreetMap (and) contributors, CC-BY-SA
Regensburg
Naturpark Oberpfälzer Wald
Naturpark Bayerischer Wald
CHKO Šumava
CHKO Český les
Hamburg
Berlin
Frankfurt am Main
Nürnberg
Praha
München

Figure 22: Study Area, only showing stations that are within or close (<2km) to the catchment boundary. The Orange square frames the *Leming* gauge (blue rhombus) and precipitation station at *Neukirchen bei Heiligen Blut* (gray circle)

It is common practice in DL to utilize the preceding information within the lookback window of the same feature that the LSTM model is designed to predict. In other words, past discharge observations at the target station are incorporated in the input data for the model. This is also practiced in the LARSIM model to a certain extent, as stated in section 3.5. Thus, taking advantage of available discharge measurements is feasible and considered to be reasonable. Therefore, one objective, which should be tested within this first model test, is the effect of the aforementioned practice on the prediction accuracy. In order to accomplish this, two scenarios are investigated. In one scenario, the only input features for the model are the observations of the single precipitation station. In contrast, in the other scenario, the model additionally receives the past discharge measurements at the gauge *Leming.* For both scenarios, the data on a daily basis is used, and the model is tested with its preliminarily selected hyperparameters as described in section 3.4. As the optimal scaling method is yet to be found at this stage, the initial methods had to be selected in advance according to common practices in literature. Hence, the data was scaled using standardization, as explained in section 3.4.3. Furthermore, the missing values are imputed with unphysical values (*Method 2*, see section 3.4.4). The model is evaluated by the *normal Holdout method* (see section 3.6).

The results of the model runs for both scenarios are presented in Table 5. Within each row, one particular performance metric score (see section 3.6) for each scenario is displayed. These individual scores are calculated separately on the training, validation and test data, respectively. Comparing the performance metric scores, the scenario that includes the discharge at the target station shows significantly better scores with respect to each metric. This is unsurprising, since the model receives more input data, and it could probably learn from patterns within the lookback window of four months of the past hydrograph. Thus, the LSTM model can use information about past discharge observations to update its internal parameters potentially better during training. This finding is confirmed when examining Figure 23 and Figure 24, which show the predicted discharge hydrograph for the test period compared to the true observations and to the simulations of the LARSIM model. Each of these figures displays the aforementioned hydrographs for one of the scenarios. Comparing both figures, it can clearly be seen that the model indeed captures the discharge dynamics better if the discharge at the target station is part of the input features.

Table 5: Resulting performance metric scores calculated for Training, Validation and Test Period, respectively. Scores highlighted in red indicate best score per metric within the complete table.

Dataset Comparison	without discharge at target station			with discharge at target station		
	Training	Validation	Test	Training	Validation	Test
Max Error	10.368	7.768	6.620	10.716	7.361	5.875
MAE	0.509	0.446	0.505	0.306	0.279	0.294
MSE	1.248	0.916	0.880	0.771	0.712	0.525
RMSE	1.117	0.957	0.938	0.878	0.844	0.725
MeAE	0.224	0.228	0.243	0.071	0.069	0.069
REV	-30.661	-14.928	-34.753	-20.761	-19.153	-19.835
NSE	0.090	0.143	-0.017	0.438	0.334	0.393
KGNP	0.325	0.463	0.301	0.735	0.750	0.758

In contrast, evaluating the MAE, the almost constant values across the three different periods (in both scenarios; see Table 5) suggest that the model does not actually “learn” from the input data, as its scores hardly decrease. It must be noted that in Figure 23 and Figure 24, the lack of discharge predictions of the model at the beginning of the test period (until the start of December 2016) is due to the chosen window size of the model. Since the model has learned to make prediction based on the lookback window, it can only start to make discharge predictions after having seen the complete time series within the first lookback window.

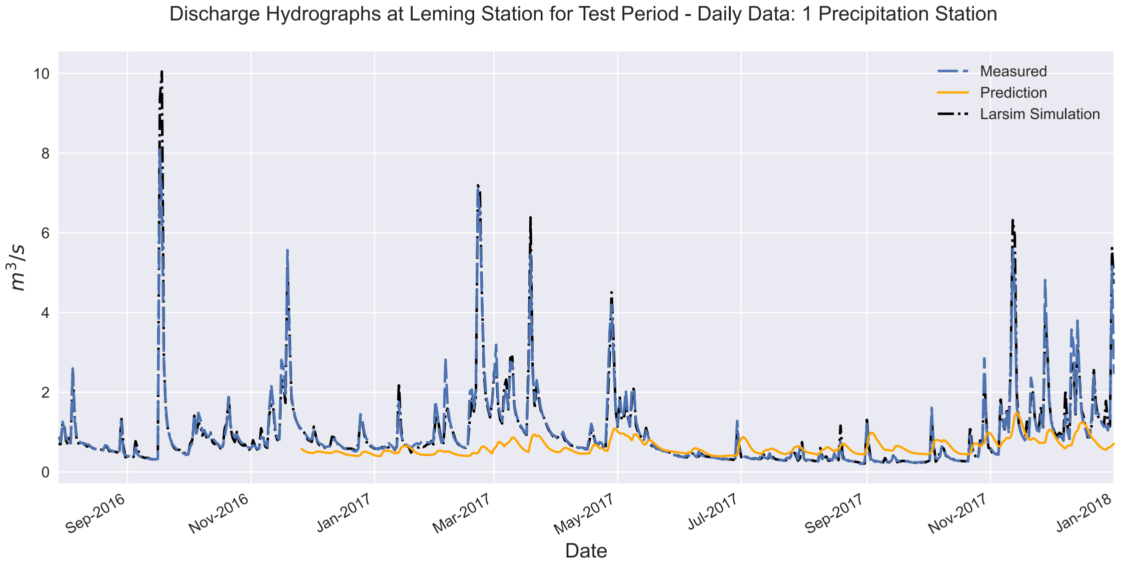

Discharge Hydrographs at Leming Station for Test Period - Daily Data: 1 Precipitation Station

Figure 23: Comparison between measured, predicted (LSTM) and simulated discharge (LARSIM) in daily resolution. LSTM predictions are based on the input of one precipitation station (*Neukirchen bei Heiligen Blut*).

As the model is fitted to the training data, the metrics of the loss function should usually be substantially lower for that period (and the validation period) than for the testing period. This conjecture is confirmed by examining the training history of one scenario. Figure 38 (see Appendix) shows the training history for the scenario without the discharge as input. In Figure 38, it can be observed that both, the training and validation loss during the learning process over 25 epochs stagnate after the first two epochs. This implies that the model is not able to properly converge to a more optimal solution after the second epoch. In addition, it seems as though the validation set is not as “challenging” to learn as the training set, which results in the validation loss graph laying below the training loss graph.

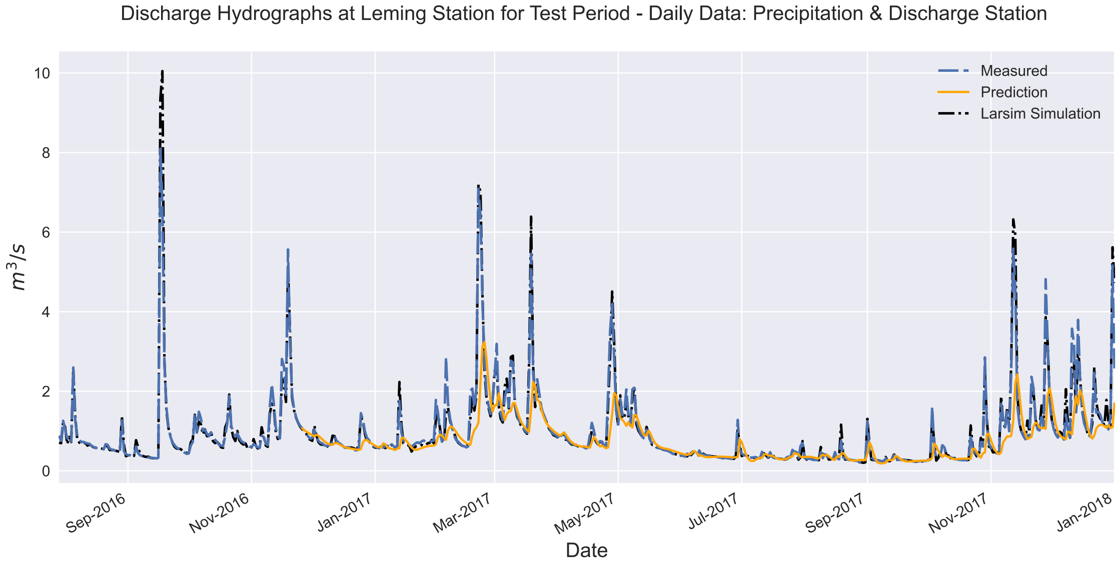


Figure 24: Comparison between measured, predicted (LSTM) and simulated discharge (LARSIM) in daily resolution. LSTM predictions are based on the input of one precipitation station (Neukirchen bei Heiligen Blut) & one discharge gauge (*Leming* = target station)

It can be assumed that the limitation of the LSTM model to properly learn the dependencies of precipitation and discharge for these two scenarios might be due to the not exactly optimal selection of both stations. By taking a closer look at the map, some tributaries to the river *Freybach* can be recognized in between the target gauge and the precipitation station. As a result, the rainfall-runoff relation between these two stations might get blurred by the inflows of these tributaries, inhibiting the model from properly learning the rainfall-runoff relations between the selected stations.

Nevertheless, these results from the first model test suggest that incorporating the past discharge measurements at the target stations lead to a meaningful increase of performance and could enhance the prediction accuracy of the model. Thus, in the further course of this thesis, the discharge measured at a target station is included in the input data, unless stated otherwise. Furthermore, throughout the rest of this thesis, the target station is not selected to be in *Leming*, but in *Marienthal,* which is the actual station of interest (target station) for predicting the discharge.

4.2. Preliminary Analysis

4.2.1. Standard versus Robust Scaling

As already mentioned before in this thesis, scaling the input features is important in order to achieve better and faster convergence during training. The input features potentially have different units and value ranges. Therefore, by scaling them, it can be assured that especially input features with high values and ranges will not dominate, and that the algorithm is able to learn from all different features. Thus, feature scaling ensures that every feature is brought to the same basis, whereby any prior accentuation of a feature should be reduced.

Since the meteorological parameters used as input features contain some outliers (even when log-transformed), it should be tested whether using a robust scaler, which is less sensitive to outliers, makes a difference compared to standard scaling. Figure 25 visualizes the characteristic influence of the distinct scaling technique on the density distribution of the various input parameters. Figure 25 shows the Kernel Density Estimation (KDE) plots of one example feature (one specific station) per parameter (containing outliers), in order to illustrate the effect of different scaling techniques. A KDE plot visualizes the distribution of samples in a dataset, similar to a histogram[23]. In this thesis, the Standard Scaler (upper righthand figure) is compared to the Robust Scaler with a specifically selected IQR (0.05 – 0.95) (lower righthand figure). Neither of the two scaling techniques bound the values to a specific range (in contrast to e.g. normalization), nor are the underlying distributions changed during scaling. After the application of standard scaling, the values for all features concentrate around -5 and +5, which is a slightly broader range than for robust scaling, in which the values cluster between -2 and + 2.

In order to detect, which of the scaling techniques indeed produces more accurate predictions, the model is used with its primary model settings as described in section 3.5. The model is fed with data on daily basis and is evaluated using the *Rep-Holdout method* (see section 3.6), with 10 iterations of model training. In this case the input dataset contains all the available meteorological input data after the cleaning processes (see section 3.3) as well as the discharge observations at *Marienthal* gauge (86 input features). The resulting performance metrices are averaged across the 10 iterations.

---

[23] https://seaborn.pydata.org/generated/seaborn.kdeplot.html

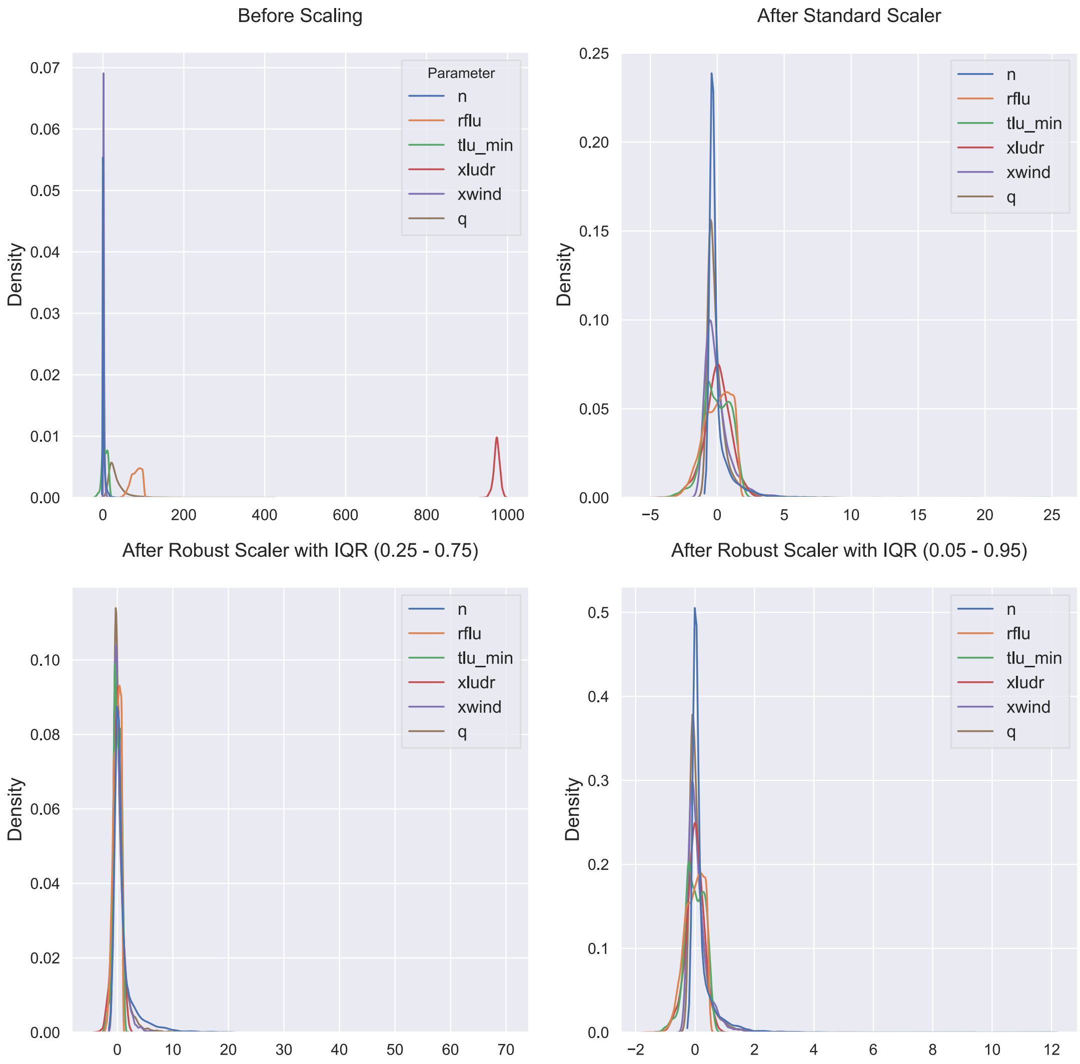


Figure 25: Kernel Density Estimation plots for different tested scaling techniques. For each plot one example feature (station) per meteorological parameter was selected, that showed a high number of outliers.

Furthermore, to investigate if the scaling technique has also an effect on performance with respect to the choice of the method for dealing with missing values, a small grid search has been carried out. In this grid search, the two scaling techniques are evaluated individually for each of the three different missing value methods. Thus, six individual model configurations are tested, and the performance scores are averaged across 10 runs for each model. The results of the averaged performance metrices are presented in Table 6. It becomes apparent that both scaling techniques generally lead to a very similar performance regarding each individual missing value method. The model does not perform consistently better across all performance metrics throughout the three different missing values methods. With respect to some performance metrices, the Robust Scaler actually performs better, whereas for others, the opposite applies. However, focusing on the NSE and KGNP score for the individual missing value methods, standardized values (applied Standard Scaler; left column in Table 6) lead to a slightly better performance on average during the test periods compared to robust scaling. Therefore, in the further course of this thesis, the Standard Scaler is applied to the input data.

Table 6: Resulting performance metric scores calculated for Training, Validation and Test Period, respectively. Scores highlighted in red indicate best score per metric within the complete table.

Scaling vs. Missing Values Methods		Standard Scaler			Robust Scaler		
		Training	Validation	Test	Training	Validation	Test
Method 1	Max Error	241.667	144.798	96.431	241.874	152.094	100.837
	MAE	4.767	6.342	7.226	5.381	5.670	6.718
	MSE	180.392	147.302	172.023	198.015	145.314	176.021
	RMSE	13.424	12.125	13.111	14.067	12.044	13.265
	MeAE	1.585	3.900	4.222	2.022	3.377	3.706
	REV	-4.955	7.690	-3.489	-4.940	4.369	-6.188
	NSE	0.770	0.670	0.666	0.747	0.675	0.658
	KGNP	0.926	0.874	0.880	0.914	0.903	0.874
Method 2	Max Error	207.845	196.061	94.536	209.609	185.002	99.359
	MAE	4.283	5.787	5.814	4.943	5.763	5.751
	MSE	127.490	159.550	109.508	149.112	156.956	114.708
	RMSE	11.289	12.291	10.462	12.209	12.314	10.707
	MeAE	1.647	3.645	3.745	1.912	3.492	3.930
	REV	-4.082	4.094	-0.071	-4.458	3.004	1.354
	NSE	0.835	0.738	0.746	0.807	0.733	0.734
	KGNP	0.940	0.880	0.906	0.926	0.888	0.888
Method 3	Max Error	207.479	196.159	94.840	209.559	184.879	99.231
	MAE	4.277	5.802	5.839	4.947	5.780	5.724
	MSE	127.049	160.213	110.218	149.295	156.993	114.405
	RMSE	11.269	12.313	10.494	12.217	12.316	10.693
	MeAE	1.641	3.659	3.766	1.913	3.515	3.898
	REV	-4.062	4.135	0.058	-4.390	3.093	1.198
	NSE	0.836	0.737	0.744	0.807	0.733	0.734
	KGNP	0.941	0.879	0.904	0.926	0.888	0.889

At this point, it should be mentioned that in the further course of this thesis, an improvement or deterioration of performance of a model mostly refers to only some of the performance metric scores and not to all eight of them. For instance, when the performance of a model is said to increase, it refers to the fact that some performance metric scores might improve, but some might deteriorate at the same time. For some performance metrices, the change might be quite marginal (differences in the second to third decimal place), so that difference is assumed to be neglectable. The focus is on the NSE score, since it is the most commonly used performance metric, as already mentioned in section 3.6. Thereby arising, for example a model with higher NSE score is said to perform better, even though some other performance metrices might show slightly worse scores. This holds true for the complete course of this thesis.

4.2.2. Methods for Missing Values

As mentioned in section 3.4.2, there is still a small percentage within the input dataset left after the preprocessing steps, which raises a computational problem for the LSTM model, since it cannot handle *NaN* values during training. Hence, in this section, three different methods (see section 3.4.4) are tested; each one reflects a separate approach to deal with any remaining missing values in the input data. As mentioned in the previous section, the examination of the three different methods was also part of the grid-search, carried out to find the better performing scaling technique. Thus, the same model settings, the same input data and the same evaluation method (Rep-Holdout) are used as for the scaling comparison. Looking at the different performance metrices, shown in Table 6 in the previous section, the model prediction capabilities seem to be distinctively depending on the choice of the missing values method. Again, the model does not perform consistently better with respect to every performance metric throughout one of the three different missing values methods. Because standardization was identified to the more appropriate scaling technique, looking at the left column in Table 6 (standard scaling), the NSE as well as the KGNP metric for *Method 1* shows slightly worse scores compared to the other two methods during the test period. This was to be expected, since during *Method 1,* in which missing values are dropped from the input data*,* the input dataset is substantially reduced, as described in section 3.3.4. Thus, the model has less data available to learn from than the other methods have. Comparing *Method 2* and *Method 3* (for standard scaling), the quantity of all performance metrices, except the REV score, is slightly greater on average (across 10 iterations) for the test period than for *Method 2*. Additionally, this method does not need an extra masking layer as *Method 3* does (see section 3.5), which slightly simplifies the model architecture. The imputation of missing values by unphysical values shows, that the model seems to draw useful information from the imputed unphysical values or their distribution pattern (to a certain extent), as it performs at least as good as the *Method 3*, which provides the model with the information of which values to ignore during computation. The model setting, used in the further course of the thesis, is applying standardization in combination with the missing values *Method 2*.

4.2.3. Imputation techniques for Precipitation Data

In contrast to the aforementioned idea of a model, which can handle the interpretation of missing data on its own, for the parameter precipitation it is nevertheless desirable to have a complete observation series. This is due to the fact, that precipitation is the most influential factor on discharge generation. As an exception, the model actually should not try to learn from missing data patterns in precipitation time series, but rather increase the prediction accuracy even further. Hence, to increase the accuracy it is desirable to train the model on data, in which missing precipitation records are fill in with reasonable estimates. For addressing this purpose four different imputation techniques, as described in section 3.3.2, should be assessed in this section. This analysis is carried out on both, daily and hourly basis, given that the correlation of rainfall among the precipitation stations is different dependent on the temporal resolution. Thus, the imputation procedure has to be done separately for daily and hourly data, resulting in a total of eight different datasets, each comprising potentially different estimates for missing rainfall observations. To be able to assess the model's generalization capability more accurately, the performance was evaluated using the *Rep-Holdout method* again (see section 3.6), in which performance metric scores are averaged over 10 individual model runs. The averaged performance metric scores are given in Table 7 (daily data) and Table 8 (hourly data), respectively. As a reference, these tables also include the model performance on the original dataset without imputed precipitation samples.

Table 7: Resulting performance metric scores calculated for Training, Validation and Test Period, respectively. Scores highlighted in red indicate best score per metric within the complete table.

Imputation Methods (daily resolution)	No Imputation			AVwC		
	Training	Validation	Test	Training	Validation	Test
Max Error	207.845	196.061	94.536	205.906	188.441	92.450
MAE	4.283	5.787	5.814	4.229	5.352	5.327
MSE	127.490	159.550	109.508	122.578	136.078	96.665
RMSE	11.289	12.291	10.462	11.069	11.385	9.831
MeAE	1.647	3.645	3.745	1.627	3.315	3.346
REV	-4.082	4.094	-0.071	-3.989	5.016	1.706
NSE	0.835	0.738	0.746	0.842	0.774	0.775
KGNP	0.940	0.880	0.906	0.941	0.895	0.922

	NRM_d			IDW		
	Training	Validation	Test	Training	Validation	Test
Max Error	205.577	189.132	91.810	203.593	188.868	95.832
MAE	4.235	5.362	5.295	4.195	5.230	5.323
MSE	122.374	136.755	95.558	119.221	134.496	100.869
RMSE	11.060	11.412	9.774	10.917	11.316	10.039
MeAE	1.634	3.324	3.314	1.616	3.251	3.340
REV	-3.943	5.082	1.719	-3.857	4.360	-0.445
NSE	0.842	0.773	0.778	0.846	0.777	0.766
KGNP	0.941	0.896	0.924	0.942	0.900	0.915

	LR		
	Training	Validation	Test
Max Error	204.281	188.700	96.321
MAE	4.204	5.251	5.284
MSE	119.864	135.033	101.018
RMSE	10.946	11.339	10.047
MeAE	1.623	3.263	3.331
REV	-3.847	4.441	-0.660
NSE	0.845	0.776	0.765
KGNP	0.942	0.900	0.916

Investigating the imputation strategies for daily datasets, the metrices across the different imputed datasets, including the not imputed dataset, do show only minor differences. However, it seems that the performance of the model does generally increase slightly when using imputed precipitation datasets. This is expectable, since the model potentially receives accurately reproduced precipitation estimates for the missing records. On the average of 10 model runs the *Normal Ratio method* with respect to distance (NRM_d) *d*, reveals the greatest number of best performance metrics scores on the test period compared to the other four different input datasets. In other terms the model trained on the imputed dataset based on the *NRM_d method* produces slightly better scores for the Max Error, MSE, RMSE, NSE and KGNP metric on the test period, respectively, across all other analyzed datasets.

Table 8: Resulting performance metric scores calculated for Training, Validation and Test Period, respectively. Scores highlighted in red indicate best score per metric within the complete table.

Imputation Methods (hourly resolution)	No Imputation			AVwC		
	Training	Validation	Test	Training	Validation	Test
Max Error	67.576	82.693	45.477	60.354	64.206	47.636
MAE	1.387	2.587	2.425	1.315	2.032	1.697
MSE	7.156	24.967	17.067	6.110	13.693	10.265
RMSE	2.675	4.997	4.131	2.472	3.700	3.204
MeAE	0.812	1.260	1.383	0.763	1.111	0.954
REV	-0.436	4.816	0.984	-0.270	2.613	-0.295
NSE	0.992	0.944	0.959	0.993	0.969	0.975
KGNP	0.990	0.917	0.953	0.991	0.959	0.982
	NRM_d			IDW		
	Training	Validation	Test	Training	Validation	Test
Max Error	61.653	64.640	44.886	57.961	62.270	45.336
MAE	1.288	2.023	1.661	1.282	1.989	1.638
MSE	6.081	13.454	10.402	5.893	13.293	10.041
RMSE	2.466	3.668	3.225	2.428	3.646	3.169
MeAE	0.739	1.108	0.932	0.736	1.082	0.909
REV	-0.298	2.652	0.012	-0.285	2.514	0.078
NSE	0.993	0.970	0.975	0.994	0.970	0.976
KGNP	0.991	0.959	0.983	0.991	0.961	0.984
	LR					
	Training	Validation	Test			
Max Error	58.327	65.516	43.374			
MAE	1.278	1.983	1.640			
MSE	5.875	13.163	9.915			
RMSE	2.424	3.628	3.149			
MeAE	0.734	1.082	0.913			
REV	-0.321	2.561	0.068			
NSE	0.994	0.970	0.976			
KGNP	0.991	0.961	0.984			

Similar observations can be made for the models trained on hourly datasets. Likewise to the results for daily resolution, imputing missing precipitation data minimally improves the performance scores regardless of the used method. However, the differences in performance metric scores are even smaller than in the daily scenario. There is one imputed dataset, that is just slightly better for at least some of the performance metrices on average (10 iterations) with respect to test period. In hourly scenario it is not the *NRM_d method*, but the applied *LR method*, that lead to the best model results. Moreover, for the first time it can be observed, that the overall performance of the model trained on hourly data is significantly better than trained on resampled daily data. This is verified by comparing e.g. the NSE scores for the different temporal resolutions. On the one hand the highest NSE score (*NRM_d method*) is 0.778 on daily data basis and on the other hand the highest NSE score (*LR method*) is 0.976 based on hourly resolution. This is an increase of 25%. This is definitely not attributable to the unequal number of imputed missing precipitation samples, but rather to the in general higher amount of received input data in case of hourly observations. This issue is investigated in more detail in the further course of this thesis. On the premise of this analysis, the NRM_d-imputed dataset is used for daily discharge predictions, whereas the LR-imputed dataset is utilized for hourly predictions throughout the rest of this thesis.

4.3. Model Sensitivity to Feature Selection

After the preliminary analysis, the best model settings with respect to the scaling technique of the input data, the method for dealing with missing values and the most suitable approach of imputing missing rainfall observations are identified at this point. At this stage of the thesis, the complete set of input features in the dataset is utilized. Since there are a lot of different meteorological parameters in the dataset, it might be difficult for the model to learn the potential non-linear correlations among all of them, since these correlations are somehow fuzzy. Furthermore, some input parameters might be more decisive for the rainfall-runoff process within this specific study area than others and some might be not relevant for the model at all. To test this hypothesis a sensitivity analysis is carried, in which the number of input features is reduced stepwise and the change in model performance is evaluated. Since generally precipitation is the most dominating parameter in a rainfall-runoff process, the focus lays of excluding other parameters than rainfall measurements. Moreover, the model performance of the LSTM model has not been compared to performance of the calibrated LARSIM model yet, which is assessed as well in the following sections. This sensitivity analysis is conducted separately for the daily and hourly resolution datasets. First the analysis is carried out the daily datasets to test if there is a considerable difference in model performance, while reducing the number of input features. Afterwards the identical analyses steps are conducted for the hourly dataset. The sensitivity analysis section, despite the division in daily and hourly resolution is further organized in two parts. In the first part of the analysis, the performance of the model based on three different groups/sets of input features for the datasets are investigated. In the second part of the analysis, the effect of including or excluding discharge gauges in the input data is examined. Hence, once more three different groups/sets of input features are tested. The model performance in both parts is evaluated based on the *normal Holdout method* (see section 3.5).

4.3.1. Daily Resolution

Influence of Meteorological Forcing Data

For each of the three analyzed input datasets the number of input features is different. The first group of input features excludes all stations that are de facto located outside the catchment boundary (hereafter referred to as: *input feature set 1*). Hence, this dataset only contain input features measured by stations, which are situated within the catchment boundary or have at least a distance of less than 2km to the boundary (but still outside the boundary; see Figure 22 in section 4.1). By excluding the aforementioned stations, the number of input features is reduced in this first group by a total of 53 features from 86 (complete dataset) to 33 (dataset containing only features measured inside the catchment boundaries). The second set of input features does only contain the rainfall observation stations from the previous group (stations inside the catchment). Hereby, the number input features is further reduced from 33 to 15 input features (hereafter referred to as: *input feature set 2*). For the sake of completeness, it is important to note that during this first section of the analysis the discharge at the target station is always part of the input data, thus counted as one input feature. Moreover, there are the performance metric scores given of the model with unchanged input data (86 features), which should serve as a reference (hereafter referred to as: *reference set)*.

In the summary Table 9 (first part) and Table 10 (second part) the results of both parts of the analysis procedures are presented regarding both temporal resolutions. The black dashed line in the middle divides both tables in two bigger columns. The left column comprises the results based on the daily datasets, whereas the right column shows the results based on the hourly resolution. Table 9 also contains the performance scores of the LARSIM model. For each input feature set each performance metric score is separately calculated for three time periods (Training, Validation and Test subset). Comparing the original model performance (*reference set*) and *input feature set 1*, in which all features located outside the catchment are disregarded, a slight performance increase can be observed with respect to all performance metric scores (except REV) on the test period. The model receives data of 53 less input features but performs slightly better. The model trained on *input feature set 1* shows the second highest KGNP score (0.925) and the lowest MAE score (4.309) on test data for daily resolution. This suggests that some of the original 86 input features are less or even of zero importance for the model, i.e. the internal model weights corresponding to these features are close to zero. This result seems reasonable since some of the measuring stations are positioned quite far outside the catchment boundaries and the parameters observed at these stations might not have a direct influence on the rainfall-runoff processes within the watershed at all. It must be noted that there is still at least on input feature for each available parameter present in the input data (see section 3.2). In other words the model still receives the complete variety of meteorological parameters (precipitation, air temperature, air pressure, etc.), even though the number of features per parameter (stations) is substantially reduced. However, within this group of recued input features the most frequent input parameter is precipitation, which constitutes with 14 measurement stations the biggest portion (almost the half) of input features. At this stage it is still too early to make assumptions about, which specific parameter is most affecting the model performance. To investigate the importance of parameter precipitation alone, in the second set of input features, just the 14 precipitation are included (as a reminder: the discharge at Marienthal station is still additionally considered). As a result, the model predictions with *input feature set 2* are slightly worse (except KGNP and MeAE) compared to *input feature set 1.* As expected, precipitation is the most significant parameter. However, this reveals that at least a few other parameters than precipitation does actually contribute valuable information to the model, even though the effect on the performance seems to be almost negligible. Which of these parameters affects the model performance most besides precipitation could be part of future investigations. Moreover, none of the above mentioned dataset configurations does reach nearly the performance of the LARSIM simulation (see Table 9), even though the performance metric scores regarding the LARSIM simulation are resampled from the natural output resolution of hourly values to daily values (see section 3.6).

Influence of Discharge Observations

Besides meteorological data there are discharge measurements at other gauges within the catchment available, which have not been utilized yet. In this second part of the analysis based on daily data these runoff observations should be considered as additional input features. Moreover, the effect of including or excluding the runoff observations at the target (*Marienthal*) station as an extra input feature should be investigated. As a reminder, in the very first model test (see section 4.1) this influence of incorporating the discharge at the target (*Leming*) station as added input feature was significant. However, conceivably the model was provided with too less meteorological forcing data (only one close by precipitation station) at that stage to predict the discharge accurately. Hence, once more three different sets of input data are examined to test the impact of added discharge features as part of the input on the model performance in more detail. One of these datasets does solely contain the meteorological station data (within the catchment), but the observed discharge at the target station is excluded as a feature (hereafter referred to as: *input feature set 5*). Another dataset does consider the meteorological input features plus all available additional discharge stations, including the target station (hereafter referred to as: *input feature set 4*). The third dataset holds the meteorological features and the additional discharge gauges as well, but the discharge at the target station is excluded (hereafter referred to as: *input feature set 3*). This set is tested to assess exclusively the influence of the target station on the model performance.

In general, there is data of 17 supplementary discharge gauges (in addition to *Marienthal* station) available (see section 3.1 and 3.2). In all the previous investigations, the runoff at the target station was regularly an included feature in the input data. Thus a first logical step would be to drop the target station form the input. As a result, the LSTM model experienced quite a significant drop in performance, when trained purely on meteorological forcing data (*input feature set 5*), compared to *input feature set 1*. In this case not only the NSE score drops from 0.787 (33 input features) to 0.590 (32 input features), but also all other performance scores deteriorate on the test period. This drop in performance can additionally be observed by comparing the predicted discharge hydrographs of the model with the observations on the test period. In Figure 26 the discharge predictions of the model trained *on input feature set 1* is shown, whereas Figure 27 displays the discharge predictions of the same model but trained without the runoff at the target station (*input feature set 5*).

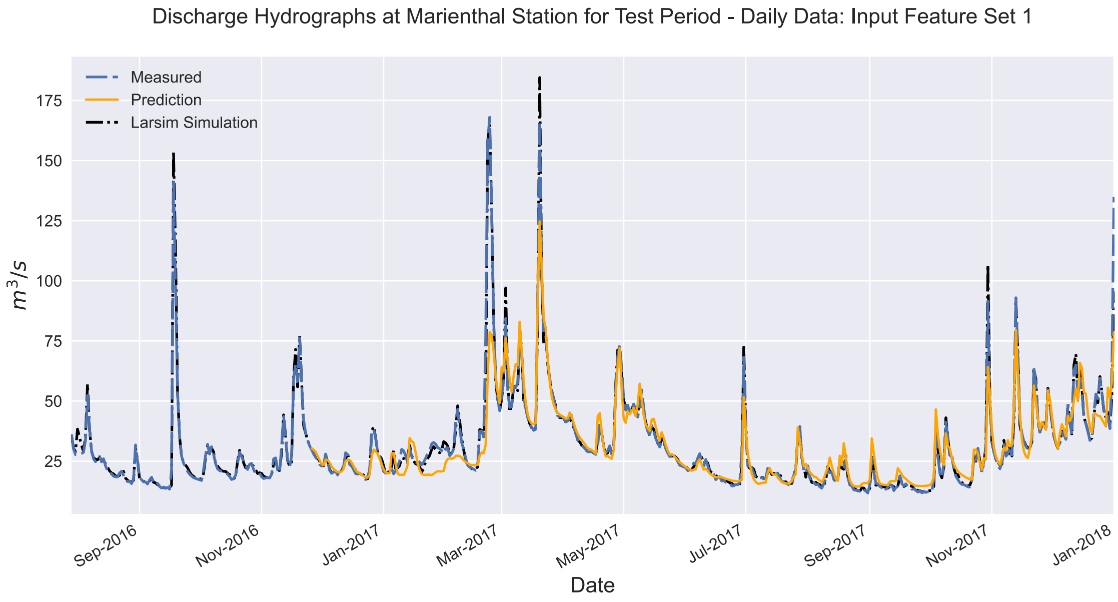


Figure 26: Comparison between measured, predicted (LSTM) and simulated discharge (LARSIM) in daily resolution. LSTM predictions are based on input feature set 1.

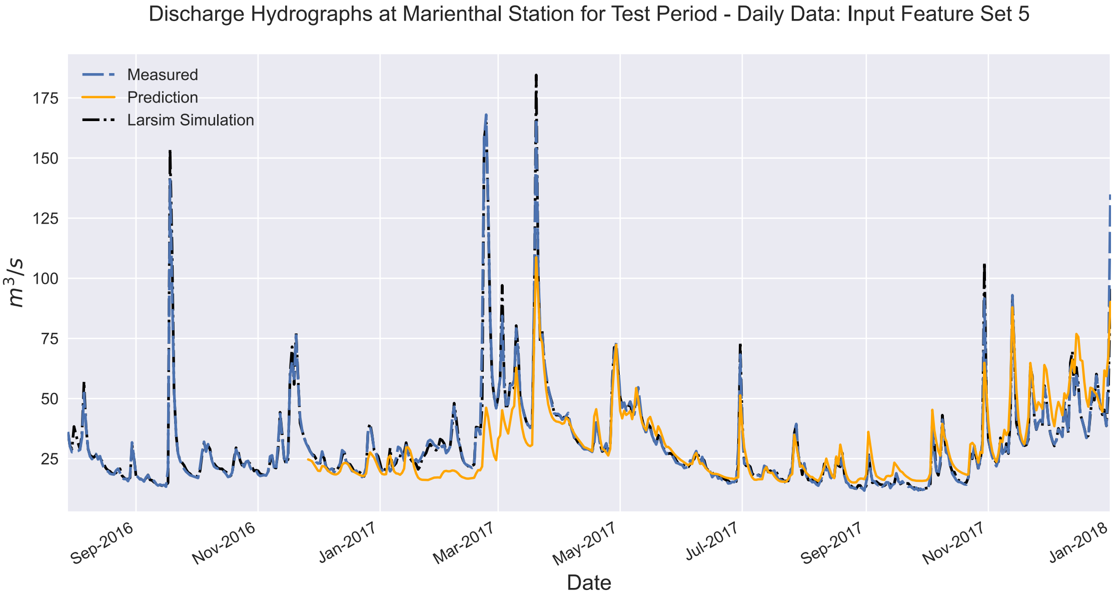


Figure 27: Comparison between measured, predicted (LSTM) and simulated discharge (LARSIM) in daily resolution. LSTM predictions are based on input feature set 5.

In Figure 27, during the period from February till April in 2017 the discharge is heavily underestimated, whereas the discharge during the end of the test period (starting around December 2017) is slightly overestimated by the model. When incorporating the discharge in the input data, these effects are well reduced (see figure x). Along with the first model test presented in section 4.1, this result proves the fact, that considering the discharge at the target station improves the model performance by a noteworthy amount, as expected. Nonetheless, flood peaks with a steep rising limp above 75 m³/s in the test period are hardly captured (with few exceptions) by either of both models (Figure 26 and Figure 27). The most likely reason for this might be choice of the MAE as objective function for training the LSTM model, which has a lower responsiveness, i.e. gives less weight, to high values. In contrast during baseflow dominated periods in the prediction hydrograph match largely well with the observations. The aforementioned outstanding performance of the LARSIM simulation (NSE = 0.976) is reflected in the almost perfect fit between prediction and observations, merely flood peaks are slightly overestimated in a few cases (Figure 26 and Figure 27).

When enlarging the number of input features by utilizing the discharge observations of all other discharge gauges except from *Marienthal* Station (*input feature set 3*), the performance metric scores increase fairly compared to the use of *input feature set 5* (see Figure 27). This seems reasonable, since the information of the discharge rate at Marienthal station, even though the target station is excluded, is partially comprised of the discharge dynamics at all other discharge gauges. In other words all the runoff generated in the catchment is routed through Marienthal station to a certain extent. Thus, the model might be able to exploit useful information about the non-linear correlations among other discharge gauges. The NSE score increases from 0.590 (*input feature set 5*) to 0.860 (*input feature set 3*) (see Table 10), which is higher compared to *input feature set 1* and *reference set*. Indeed, not only the NSE scores are better, but all performance metric scores show a superior value. This suggests that the model can harness useful information of the temporal dynamics of the runoff at other discharge gauges to improve its overall prediction accuracy.

The accuracy with respect to the performance metrices can even slightly be improved when the discharge at target station is included as well (*input feature set 4*). When the data of all measuring stations inside the catchment boundaries is utilized (50 input features), namely meteorological data and discharge data, the model achieves its best performance on test data for a daily resolution so far. This manifests in the highest performance metric scores for daily data with respect of the MAX-ERROR, MSE, RMSE and NSE score, respectively (see Table 10). This outcome can further be approved by looking at the predicted discharge hydrograph of the model during the test period depicted in Figure 28. In contrast to the discharge hydrographs in Figure 26 (*input feature set 1*) and Figure 27 (*input feature set 5*), the model in this scenario seems to capture the temporal dynamics especially during high flow conditions better (see Figure 28). This coincides with the improved scores of the performance metrices, which are more sensitive to high values, i.e. MAX-ERROR, MSE, RMSE and NSE. On the contrary, during low flow periods (e.g. August till November in 2017) the discharge seems to be systematically overestimated (see Figure 28), when adding the observation of all discharge gauges to the input data (*input feature set 4*).

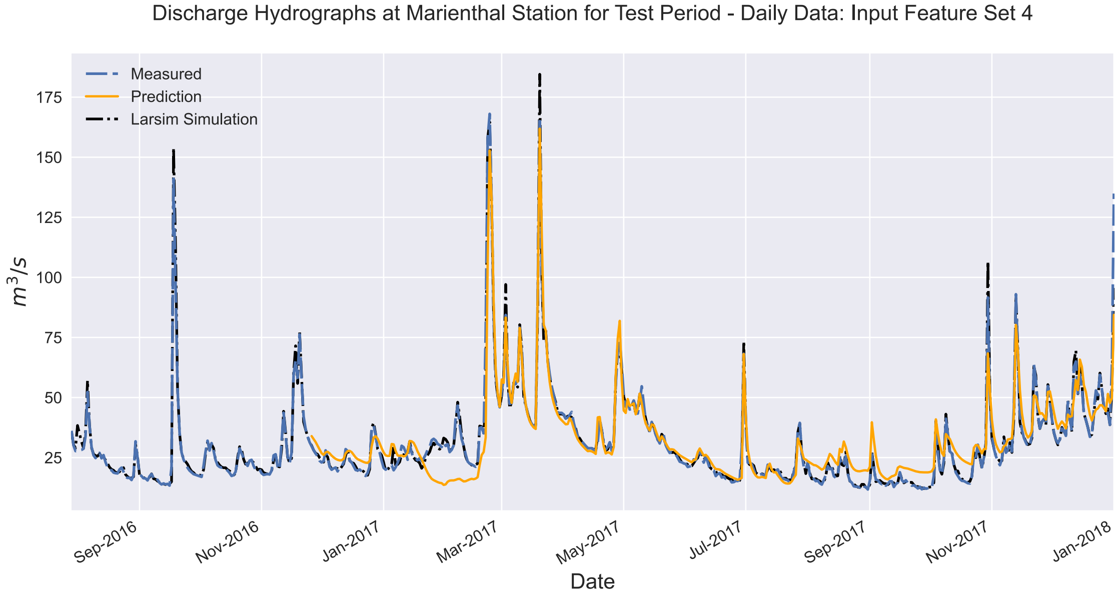


Figure 28: Comparison between measured, predicted (LSTM) and simulated discharge (LARSIM) in daily resolution. LSTM predictions are based on input feature set 4.

Overall, not a single LSTM-model trained on any of the input datasets in daily resolution reached nearly the accuracy of the LARSIM simulation. Every individual performance metric of the LARSIM simulation showed exceptional good scores. Nevertheless, it has to be mentioned that the LARSIM model actually never predicts on a daily timescale. Thus, the model performance of the LARSIM and LSTM model are only comparable with some reservations.

Table 9: Resulting performance metric scores calculated for Training, Validation and Test Period, respectively. Scores highlighted in red indicate best score per metric across all input feature sets (including Table 10) with respect to daily (left) or hourly (right) data basis.

Dataset Comparison	Daily Datasets (with NRM_d imputation of precipitation features)						Hourly Datasets (with LR imputation of precipitation features)					
	Reference: all meteorological Stations & target station ***[86 input features]***			**Input feature set 1:** Stations inside Catchment & target station ***[33 input features]***			**Reference:** all meteorological Stations & target station ***[75 input features]***			**Input feature set 1:** Stations inside Catchment & target station ***[29 input features]***		
	Training	Validation	Test	Training	Validation	Test	Training	Validation	Test	Training	Validation	Test
Max Error	198.844	138.931	102.739	200.747	138.856	98.098	58.327	65.516	43.374	58.499	57.786	31.889
MAE	4.063	5.271	5.452	4.554	4.528	4.309	1.278	1.983	1.640	1.114	1.414	1.291
MSE	117.322	95.609	111.445	136.049	93.884	91.759	5.875	13.163	9.915	4.396	7.724	5.000
RMSE	10.832	9.778	10.557	11.664	9.689	9.579	2.424	3.628	3.149	2.097	2.779	2.236
MeAE	1.584	3.683	3.524	1.686	2.792	2.189	0.734	1.082	0.913	0.608	0.795	0.754
REV	-3.858	7.698	0.212	-4.638	2.045	-2.518	-0.321	2.561	0.068	-0.249	1.231	-0.105
NSE	0.859	0.786	0.741	0.837	0.789	0.787	0.994	0.970	0.976	0.995	0.983	0.988
KGNP	0.943	0.902	0.906	0.931	0.940	0.925	0.991	0.961	0.984	0.993	0.980	0.988

	Input feature set 2: Precipitation Stations (inside catchment) & target station ***[15 input features]***			LARSIM Simulations			**Input feature set 2:** Precipitation Stations (inside catchment) & target station ***[15 input features]***			LARSIM Simulations		
	Training	Validation	Test	Training	Validation	Test	Training	Validation	Test	Training	Validation	Test
Max Error	180.545	146.770	99.638	59.833	25.131	39.277	179.904	62.874	35.436	103.620	80.570	74.100
MAE	4.884	3.887	4.419	1.392	1.105	1.239	1.732	1.487	1.376	2.137	1.800	1.878
MSE	134.319	97.305	98.426	9.888	4.897	10.120	36.153	14.257	7.957	28.395	18.951	21.045
RMSE	11.590	9.864	9.921	3.145	2.213	3.181	6.013	3.776	2.821	5.329	4.353	4.587
MeAE	1.746	1.817	2.113	0.611	0.542	0.582	0.575	0.607	0.672	0.786	0.773	0.904
REV	-4.272	-1.201	-3.770	0.411	0.321	-0.054	-0.582	0.664	0.262	0.325	0.367	0.606
NSE	0.839	0.782	0.771	0.988	0.989	0.976	0.960	0.968	0.981	0.969	0.957	0.948
KGNP	0.927	0.954	0.927	0.991	0.991	0.990	0.989	0.987	0.989	0.984	0.983	0.979

Table 10: Resulting performance metric scores calculated for Training, Validation and Test Period, respectively. Scores highlighted in red indicate best score per metric across all input feature sets (including Table 9) with respect to daily (left) or hourly (right) data basis.

Dataset Comparison	Daily Datasets (with NRM_d imputation of missing precipitation samples)						Hourly Datasets (with LR imputation of missing precipitation samples)					
	Input feature set 3: No Target station & with additional Discharge gauges & meteorological stations inside catchment ***[49 input features]***			**Input feature set 4:** With target station & with additional Discharge gauges & meteorological stations inside catchment ***[50 input features]***			**Input feature set 3:** No Target station & with additional Discharge & meteorological stations inside catchment gauges ***[45 input features]***			**Input feature set 4:** With target station & with additional Discharge gauges & meteorological stations inside catchment ***[46 input features]***		
	Training	Validation	Test	Training	Validation	Test	Training	Validation	Test	Training	Validation	Test
Max Error	166.486	61.248	54.738	163.798	62.612	51.462	71.547	62.998	65.484	61.028	49.083	51.973
MAE	3.014	3.626	4.957	2.986	3.535	5.087	1.410	2.788	5.532	1.008	1.704	3.932
MSE	54.836	41.507	60.401	53.833	41.542	58.183	6.837	23.182	60.272	4.113	11.943	33.361
RMSE	7.405	6.443	7.772	7.337	6.445	7.628	2.615	4.815	7.764	2.028	3.456	5.776
MeAE	1.319	2.348	3.499	1.299	2.292	3.858	0.878	1.864	3.817	0.595	0.947	2.236
REV	-2.458	-1.430	-0.348	-2.212	-1.249	1.724	0.205	-0.658	4.507	0.056	0.414	3.902
NSE	0.934	0.907	0.860	0.935	0.907	0.865	0.992	0.948	0.856	0.995	0.973	0.920
KGNP	0.963	0.937	0.820	0.965	0.940	0.819	0.988	0.943	0.716	0.993	0.971	0.820

	Input feature set 5: Meteorological Stations only (inside Catchment) ***[32 input features]***		
	Training	Validation	Test
Max Error	213.863	142.739	121.848
MAE	4.870	6.342	6.477
MSE	147.327	120.476	176.354
RMSE	12.138	10.976	13.280
MeAE	1.933	4.273	3.269
REV	-5.058	8.829	-5.594
NSE	0.823	0.730	0.590
KGNP	0.926	0.870	0.832

	Input feature set 5: Meteorological Stations only (inside Catchment) ***[28 input features]***		
	Training	Validation	Test
Max Error	134.433	157.244	168.279
MAE	5.771	12.828	9.569
MSE	96.674	448.474	294.303
RMSE	9.832	21.177	17.155
MeAE	3.254	8.004	5.499
REV	-4.600	24.870	12.641
NSE	0.893	-0.008	0.297
KGNP	0.842	0.482	0.707

4.3.2. Hourly Resolution

Influence of Meteorological Forcing Data

The model sensitivity analysis procedures based on hourly data is repeated in the same way as for daily resolution. Similarly, in the first part of the analysis, the performance of the model based on hourly resolution is investigated on the same three groups/sets of input features as in the daily scenario. The only difference is the number of features each of these sets consists of compared to the examined daily input feature sets. Since the hourly datasets do not consider minim and maximum air temperature separately as input features, as described in section. 3.2, the analyzed hourly datasets contain generally less features as the daily ones. The first set of input features excludes all stations that are located outside the catchment boundary (*input feature set 1*). By excluding the stations, the number of input features is reduced in this first group by a total of 46 features from 75 (complete dataset) to 29 (dataset containing only features measured inside the catchment). The second set of input data does only contain rainfall observation data (*input feature set 2*). Hereby, the number input features is further reduced from 29 to 15 input features. Again it is worth mentioning, that in this part of the analysis the discharge at the target station is always part of the input data and is counted as one feature. For comparison purpose the performance metric scores of the model with unchanged input data (*reference set*) with 75 features is provided as well.

The results of both parts of the analysis are shown in Table 9 and Table 10. When reducing the amount of input features in the *input features set 1*, the model performance metric scores increase by a minor amount compared to the *reference set*. These findings coincide with the results from the daily scenario, supporting the hypothesis, that a major of the original input features (*reference dataset*) have less or even zero influence on the model accuracy and rather obscuring the model with irrelevant information. The resulting model trained on *input feature set 1* results in the overall highest model performance on the test period so far. With this specific input feature set the best performance metric scores are reached (except for MeAE and KGNP) within the complete sensitivity analysis. For example the model predictions show a remarkable NSE score of 0.988, which suggests almost a perfect fit between predicted and observed discharge hydrograph. This assumption is revealed in Figure 29, which depicts the predicted discharge of the model. Furthermore, it can be observed in Figure 29, that the LARSIM simulation overestimates particularly the peak flows, as already noticed for the daily scenario. This might be an indication for the worse performance compared to the LSTM model with respect to each individual performance metric score. The LSTM model trained on *input feature set 1* is able to capture the complete range of hydrograph dynamics (see Figure 29). However, at this stage it is unclear to which extent the discharge at the target station, as part of the input features, has an impact on this high prediction accuracy.

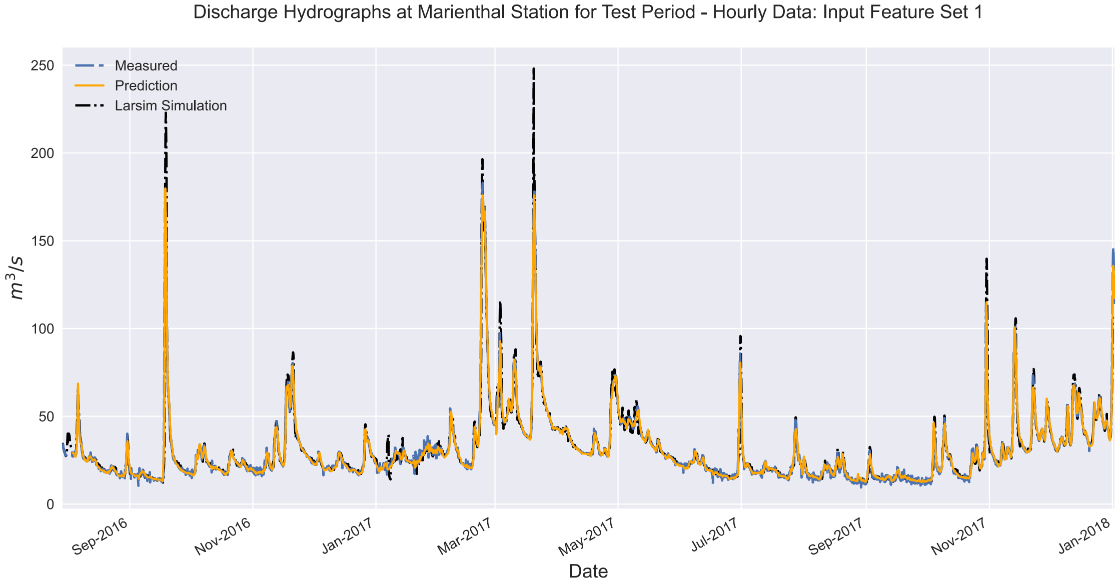


Figure 29: Comparison between measured, predicted (LSTM) and simulated discharge (LARSIM) in hourly resolution. LSTM predictions are based on input feature set 1.

Reducing the amount of input features further to only 15 precipitation stations (*input feature set 2*), the model performance drops faintly (except to the KGNP and MeAE score), as it does, when considering the same procedure step for the daily resolution dataset. This implies once more that at least some other parameters than precipitation does actually contribute valuable data to the model. However, this positive effect on the performance seems to be almost negligible, as it is the case for the daily scenario. It should be mentioned that actually each LSTM-model, independent of the input feature group, showed superior performance compared to the LARSIM simulation in this first part of the sensitivity analysis.

Influence of Discharge Observations

As also in the daily scenario the first test in the second part of the sensitivity analysis is to exclude the discharge observations at the target gauge from the input dataset and train the model solely on meteorological input data (*input feature set 5*). After training the model on the new *input feature set 5*, the performance experienced a substantial drop comparted to the i*nput feature set 1*. In this case, it seems that the model is as not able to learn the rainfall-runoff relation at all. This is reflected in the lowest performance metric scores on test period ever observed throughout the thesis, which is far beyond the lowest performance of any model trained on daily input data. The relative drop in performance in the hourly scenario is exceptionally higher than for the same analysis step in daily resolution. Regarding the NSE score for the validation period, which in fact drops below zero (see Table 10), the model shows a poorly prediction accuracy. This is confirmed by looking at the predicted hydrograph (see Figure 30).

It can be clearly seen that the model is not able to capture the temporal dynamics simply from meteorological forcing data, which results in a deficient fit between observed and predicted hydrograph.

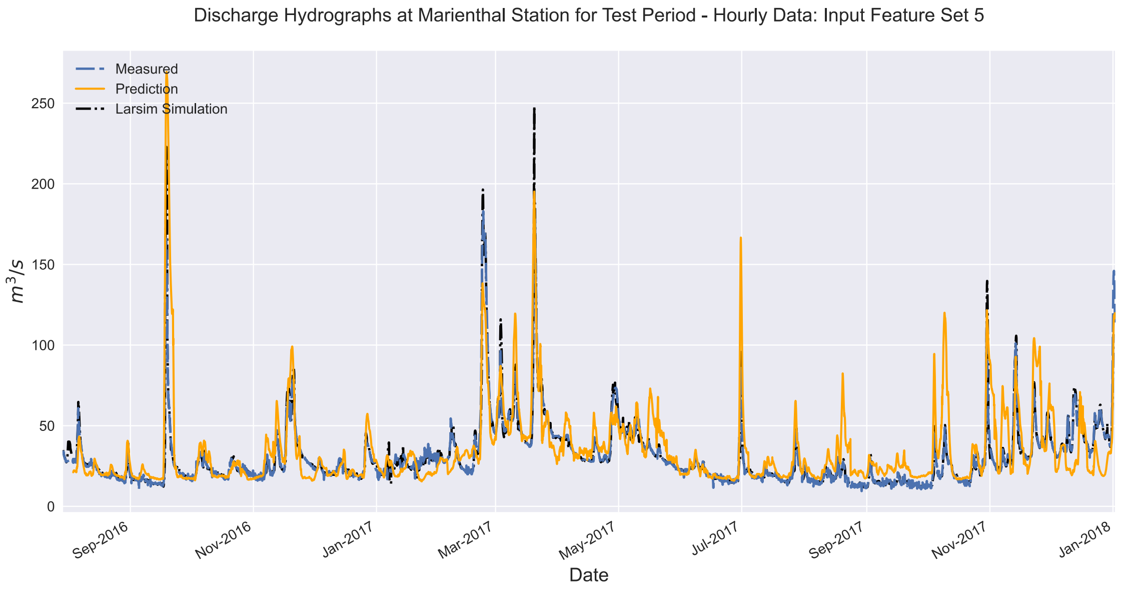


Figure 30: Comparison between measured, predicted (LSTM) and simulated discharge (LARSIM) in hourly resolution. LSTM predictions are based on input feature set 5.

Doing the opposite by adding all available discharge stations (*input feature set 4*) to the input data the performance significantly increases again. This is expected, since the model gets potentially relevant information about the discharge rate at other stations, as previously shown for the daily scenario. However, with the additional information of all upstream located discharge gauges, the model trained on input *feature set 4* does not reach the performance scores as the model that considers solely the target station as added discharge feature (*input feature set 1*). This is proven by comparing the performance metric scores, while the former mentioned model has e.g. a NSE score of 0.920 (see Table 10), the latter one has a NSE score of 0.988 (see Table 9). Thus, incorporating all available discharge observations in the input data seems to be counterproductive for the “learning” process of the model. On top of that the model shows less accurate predictions than the LARSIM simulation. These findings contrasts with the results for the same scenario in daily resolution, in which the model performance actually increased, when including more information from other discharge gauges additionally to meteorological input data. Looking at the predicted discharge hydrograph in Figure 31 during the test period for this scenario, it appears that the model misses to accurately predict the baseflow conditions for particular time periods. From August till October in 2016 and 2017 the model overestimated the baseflow constantly, whereas around February 2017 the streamflow is underestimated for a period of two months. To make sure that these issues are not due to an adversely selected point to split the data into a training, validation and test subsets, the both models either trained on *input feature set 4* or *input feature set 1* are once more evaluated by the k-fold cross validation method (see section 3.6). A badly chosen point might have the effect that the training or validation dataset do have a lack of periods with specific flow conditions, which would prevent the model to properly learn the necessary correlations.

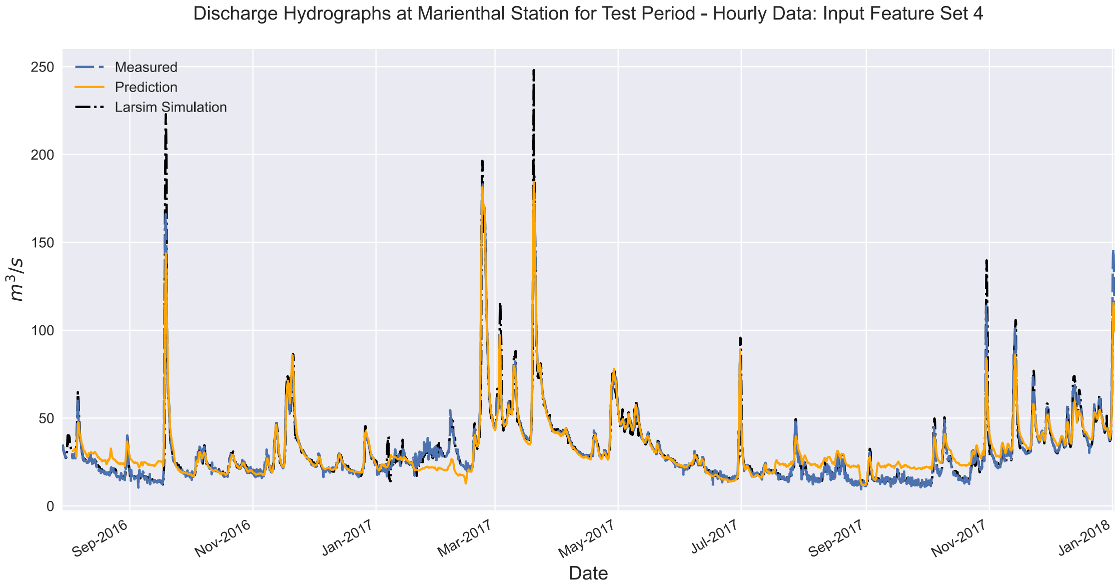


Figure 31: Comparison between measured, predicted (LSTM) and simulated discharge (LARSIM) in hourly resolution. LSTM predictions are based on input feature set 4.

However, the k-fold cross validation with three iterations, lead to the same results, which indicate that additional information of other discharge gauges have rather a negative impact on model performance. Hereby, another reason could be, that the concentration time of the runoff within the catchment tributaries might be shorter than the window size, i.e. 5 days. As a consequence, the discharge information at *Marienthal* is “overlapped” and blurred by the runoff measured at other gauges that reaches the gauge at *Marienthal* in a time delay. Because of that the model might be unable to learn the complex non-linearities among the 46 input features.

If the discharge information only at the target station is dropped form the model input dataset (*input feature set 3*) the LSTM-model performs slightly worse than with the discharge at the target station included (*input feature set 4*). This is also the case for the same scenario in daily data resolution. However, the drop in performance metric scores is relatively seen higher (see Table 9). For the performance of the model trained on daily input data the difference between *input feature set 3* or *4* for e.g. the NSE score is about 0.6%, whereas this difference the hourly scenario is ca. 7.2%. This suggests that an hourly model is more responsive to the presence of the target station as an included feature in the input data. Moreover, the model accuracy trained on *input feature set 4* decreases slightly compared to *input feature set 3.* Subsequently, analyzing the predicted discharge and the observed one shows especially during high flow conditions, i.e. the peak flows, and during low flow conditions a poor fit.

Overall it seems, that the model trained on data with hourly data is quite a lot more sensitive to the presence of discharge measurements of the target station in the input data than the model trained on daily basis. When the discharge of the target station is considered as part of the input features, the LSTM model in hourly resolution clearly outperforms the model trained on daily data. Furthermore, an hourly LSTM model shows superior discharge prediction accuracy compared to the LARSIM model, as long as no additional discharge features are incorporated in the input, except the target gauge.

4.4. Hyperparameter Tuning

As it turned out in the preceding section, the LSTM model trained on daily resolution data does not reach the performance metric scores of the LARSIM model. Specifically, this is also valid for a model in hourly resolution, when trained exclusively on meteorological forcing data. Once there are not any discharge measurements incorporated in the input, the forecasting accuracy of the LSTM model drops significantly resulting in an incommensurate fit of the predicted hydrograph. As a consequence, in this scenario the model is not to make any predictions with reasonable accuracy. One main reason might be a not ideal choice of hyperparameters. The previously presented sensitivity analysis was carried out on the preliminary setting of hyperparameters (see Table 2 in section 3.5), based on the hyperparameter optimization performed by Unnikrishnan 2019). However, the hyperparameter selection suggested by Unnikrishnan (2019) might not be completely appropriate for all the scenarios examined in this thesis, since Unnikrishnan (2019) used an input dataset to optimize the hyperparameter with not neglectable differences (e.g. a dissimilar input feature combination). Moreover, the search space of some specific hyperparameters was limited. These individual hyperparameters include the number of LSTM units, the size of the lookback window and the choice of the objective function. The LSTM units is directly related to the number of trainable parameters of one LSTM cell, whereas the size of the lookback window determines the time period the model can learn from input features. From a hydrological perspective in a snow-influenced catchment, where potentially quite a big time delay exists between accumulation of snow and released runoff, the model might profit from a long lookback window size to capture such processes. However, in the *Regen* Catchment, which is not located in mountain regions is not snow-influenced. Thus, a the original chosen window size of four months (120 days) might not be necessary in case of daily data and might actually have a negative impact on model performance. For hourly resolution data, a window size of five days (120 hours) seems reasonably long to capture the rainfall concentration and flood routing processes within the catchment. Furthermore, models operating on this window size (120 hours) already proved to provide very good result for some input datasets (*e.g. input feature set 1*, see section 4.3.2). The choice of the loss-function used to train the model might also have an influence on the performance of the model, as the MSE or the NSE respond more sensible to high values (as explained in section 3.6). Hence, a change of the objective function is expected to increase the accuracy of the predicted runoff for specific discharge ranges (especially high flow conditions).

To test how much the changed hyperparameters affect the model performance separately, the tuning is conducted in two parts, each carried out for both temporal resolutions. In the first part only the objective model function is varied, whereas in the second part different combinations of window size and number of LSTM units analyzed, respectively. Each model run is evaluated by the normal holdout method. In Table 11 the selected hyperparameter settings regarding the two parts are shown. The specific LSTM-unit numbers (256/512) in part 2 are chosen according to commonly used values in literature, e.g. (Kratzert et al. 2018). To verify all the following results of the hyperparameter tuning process, the tables containing the computed performance metric scores for both parts can be found in Table 14, Table 15, Table 16, Table 17, Table 18, Table 19 and Table 20 in the Appendix.

Table 11: Overview of tested hyperparameter configuration with respect to temporal resolution. Bolt values indicate tested settings, whereby the other values remain constant.

Hyperparameter Configuration		**Hourly Data**	**Daily Data**
Change objective function (Part 1)	*Loss-function*	**MAE | MSE | NSE**	**MAE | MSE | NSE**
	LSTM units	120	120
	Window Size	120 (5 days)	120 (4 months)
Change LSTM units and Window size (Part 2)	*Loss-function*	MAE	MAE
	LSTM units	**120 | 256**	**120 | 256 | 512**
	Window Size	120 (5 days)	**120 | 60 | 30**

Switching the objective function

In the first part of the hyperparameter tuning, the model is trained on two different input datasets, namely *input feature set 5* and *input feature set 3*. The latter one is picked to see if having discharge observations in the input data does have a noticeable impact on performance, while the model is trained with distinct objective functions. The target station is generally excluded from the input during the tuning, since, as mentioned above, the model performance (independent of the temporal resolution) is poor and the biggest improvements in performance are expected on these two input feature sets. In general the choice of the loss-function does not seem to have a big influence on the performance, as the metric scores vary insignificantly on the test period throughout the tests on the three different loss-functions. When switching to the MSE or NSE as objective function, the performance metrices scores particularly susceptible to high discharge values, i.e. the MAX-ERROR, MSE, RMSE and NSE improved slightly on test data. This is observed for both of the two different input datasets. With respect to the NSE score the biggest jump in model performance is found when trained to minimize the MSE rather than the MAE on *input feature set 5*. In this case the NSE increase from 0.590 (trained with MAE as loss-function) to 0.686 (trained with MSE as loss-function). At large the model with the MAE as objective function does capture conditions around low flow periods better (see Figure 27 in section 4.3.1), whereas a model trained using the MSE as objective function does capture conditions around high flow periods more accurately, see Figure 32 below. Nonetheless, it can be observed in Figure 32 that the predicted discharge peaks still do not fit very well to the measured peaks. Yet, the discharge at high peaks is sporadically underestimated, whereas at smaller peaks the discharge seems to be systematically overestimated.

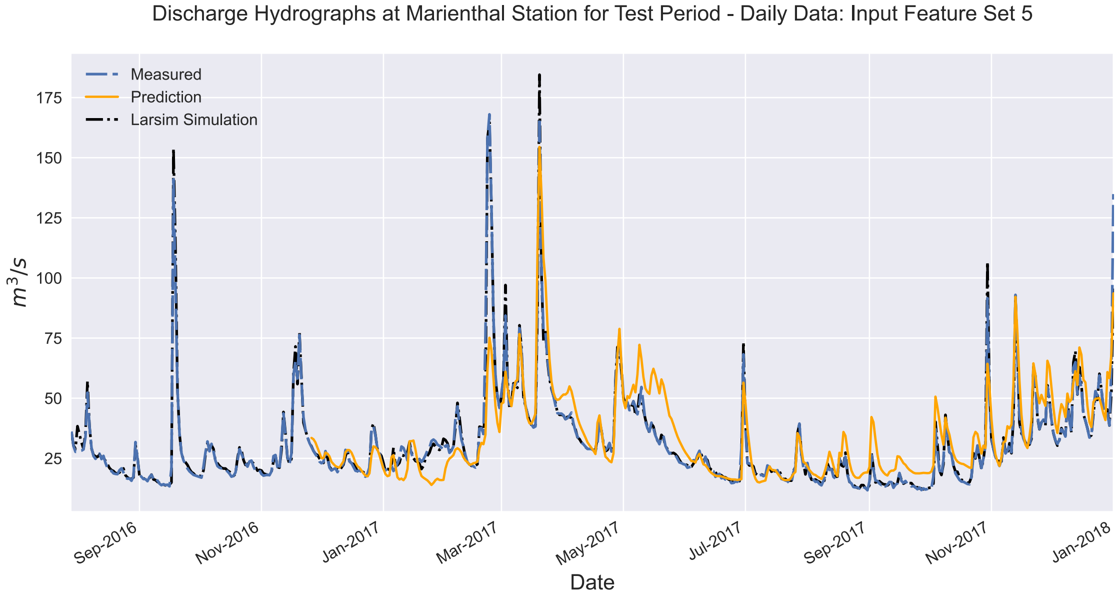


Figure 32: Comparison between measured, predicted (LSTM) and simulated discharge (LARSIM) in daily resolution. LSTM predictions are based on input feature set 5 and model is trained with MSE as objective function.

The results for the first part of the hyperparameter tuning in case of a daily data basis reveal that the model solely trained on meteorological forcings can still not be considered as a model with reasonable prediction accuracy when switching the objective function. When the 17 discharge gauges within the catchment are added (*input feature set 3*) changing the objective function to NSE or MSE, the model performance does hardly show any significant differences (compared to the MAE function) regarding the performance metric scores.

Looking at the same hyperparameter tuning procedure but for models trained on hourly data a converse picture can be drawn. For models trained on either of the two different input feature sets, changing the loss-function to MSE or NSE results in a performance drop for either of these two objective functions. Therefore, models trained on input data with hourly resolution show the highest accuracy, when trained using the preliminary chosen MAE function.

Changing model capacity and lookback window size

In the second part of the hyperparameter tuning the same two datasets as in part 1 are used. For each model run the window size is reduced and simultaneously the model capacity is increased by changing the number of hidden LSTM units. In daily scenario, a 3x3 grid-search (three different LSTM units vs. three different window sizes) is carried out for both datasets (*input feature set 5 & 3*). In an hourly scenario, the model capacity is increased to 256 LSTM units when trained on both of the two datasets. Unfortunately, the model capacity in this experiment was limited to 256 units, because increasing the number further resulted consistently in a crash during the training execution of the model. This might has happened due to memory issues of the GPU of the local machine. However this is only an assumption and the true underlying problem has not been found and could not be solved throughout working on this thesis.

Regarding the input *feature set 5* in daily resolution, the grid-search revealed that generally decreasing the window size improved the model performance slightly (compared to window size 120), regardless the number of used LSTM units. Further, a reduced window size of 30 time steps do not increase the performance metric scores with a higher number of LSTM units. In contrast, a reduced window size of 60 time steps results in a slight increase in performance metric scores while increasing the number of LSTM units simultaneously. Models trained with a window size of 60 time steps (compared to other window sizes) generally showed the best performance metric scores aside from the number of hidden LSTM units. For a window size of 120 time steps the performance metric scores first drop when changing the LSTM units to 256, but then increase again for 512 LSTM units. The highest NSE score provides the model trained with a lookback window size of 60 time steps and 256 LSTM units. Changing the window size and the LSTM units from the original setting (120/120) to the optimal setting (60/256) results in a significant increase of the NSE-score from 0.590 (original model) to 0.771 (model with tuned hyperparameters). Looking at the predicted discharge hydrograph (see Figure 33) for the model with optimized hyperparameters trained only on meteorological data (*input feature set 5*), it can be observed that the model predictions increased a little in accuracy compared to the model with original hyperparameter settings (see Figure 27 in section 4.3.1). This is especially true during the high flood peaks around March 2017.

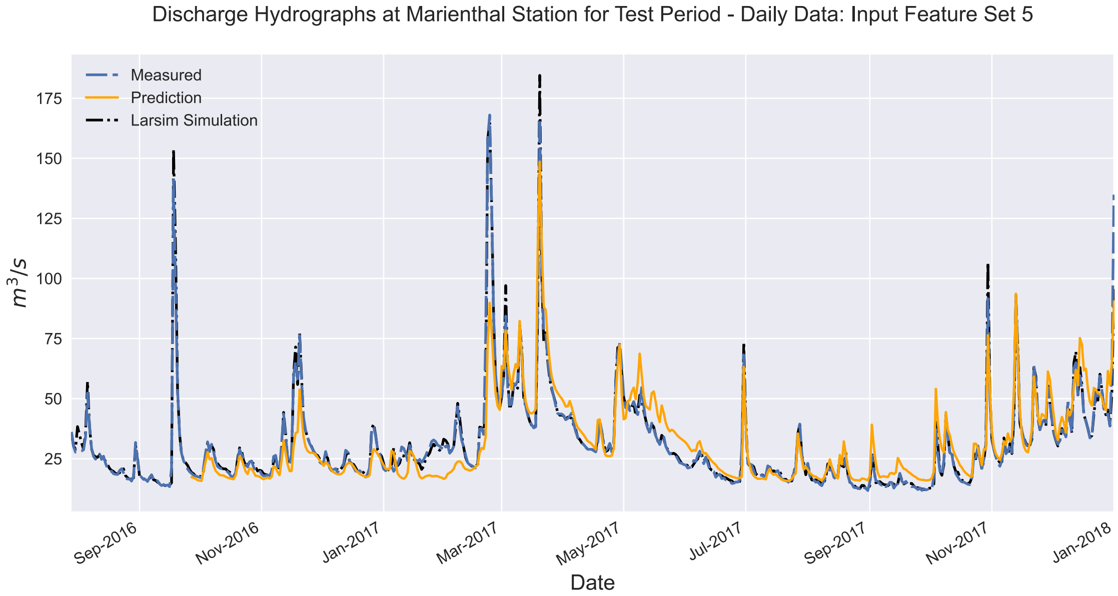


Figure 33: Comparison between measured, predicted (LSTM) and simulated discharge (LARSIM) in daily resolution. LSTM predictions are based on input feature set 5, window size 60 steps and 256 LSTM units.

Switching to the other daily dataset, in which the 17 discharge gauges are added to the input set (*input feature set 3*), the hyperparameter tuning lead to contrasting findings. On the one hand generally decreasing the window size resulted in negligible small decrease of performance metric scores (regardless the three different LSTM units). On the other hand generally increasing the LSTM units with respect to one particular window size the model resulted once more in negligible small decrease of performance metric scores. These are rather opposite results to the first input feature set. The effect of the hyperparameter tuning on the model performance seems to be barely different. In this case the best performing hyperparameter combination on *input feature set 3* is the original setting (Window size: 120/ LSTM units: 120).

As a next step, it should be analyzed if a scaled up model capacity, i.e. increasing the LSTM units to 256, results in a higher model performance when trained on hourly resolution. It turns out, that the model performance with respect to the performance metric scores indeed increase for both datasets. When the model is trained on meteorological data only (*input feature set 5*), the NSE score jumps from 0.297 (120 LSTM units) to 0.520 (256 LSTM units), which is quite a significant increase. However, the prediction accuracy of this model for the runoff is still very loose, as it can be seen in the Figure 34.

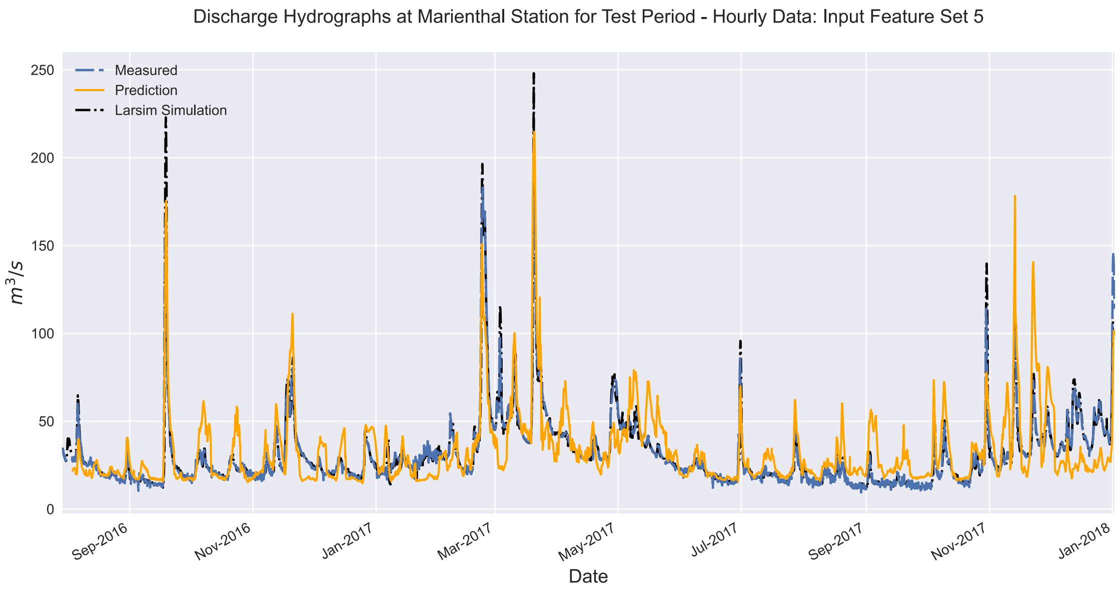


Figure 34: Comparison between measured, predicted (LSTM) and simulated discharge (LARSIM) in hourly resolution. LSTM predictions are based on input feature set 5, window size 120 steps and 256 LSTM units.

Thus, just increasing the model capacity seems not enough for the model to learn the non-linear relations in the rainfall-runoff processes for this catchment alone from meteorological measurements, even though the performance metric scores indicate a significant improvement. Subsequently, adding the 17 discharge gauges located somewhere upstream of Marienthal station (*input feature set 3*) and train the model while increasing the LSTM units, the model performance increase as well. However, the relative increase in this case compared to the other input feature set is less significant. As an example, the NSE score advance slightly from 0.856 to 0.890. This small difference is mainly due to the fact, that the model trained on *input feature set 3* with 120 LSTM units is quite accurate in the first place. Nonetheless, this minor increase of performance, affects the prediction accuracy of the model to a certain extent. Comparing the discharge predictions of the model with original hyperparameter settings (Figure 30 in section 4.3.2) with the model with increased capacity (256 LSTM units) (see Figure 35), it can be observed that specifically the baseflow during low flow periods is captured more accurately. There is one exception to that: during December 2017 the discharge is consistently underestimated.

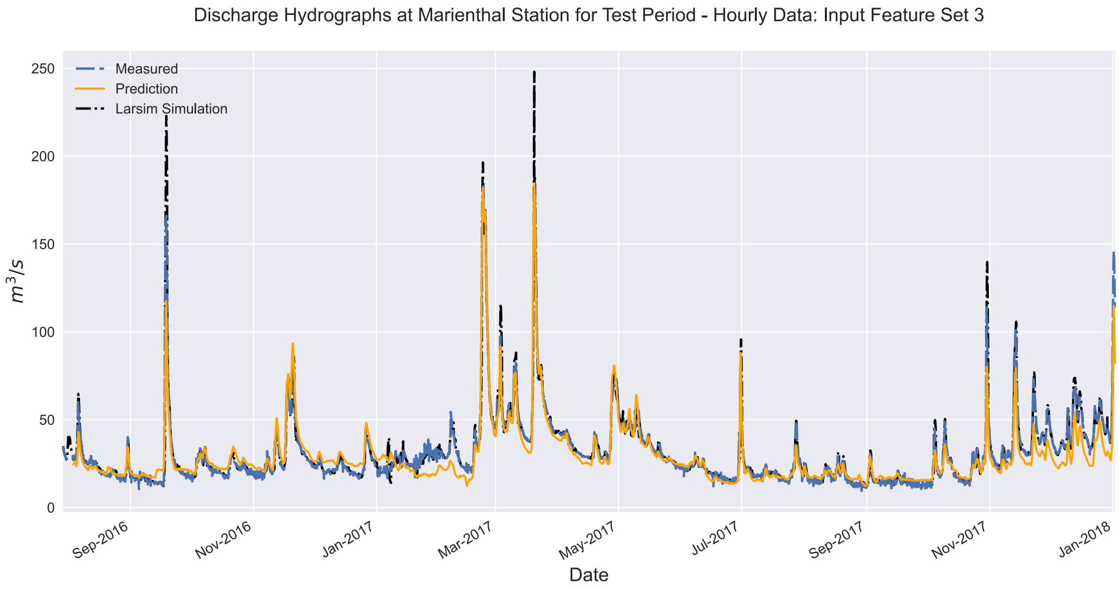


Figure 35: Comparison between measured, predicted (LSTM) and simulated discharge (LARSIM) in hourly resolution. LSTM predictions are based on input feature set 3, window size 120 steps and 256 LSTM units.

As an add on for the hourly resolution data: If the model with optimized hyperparameter settings is not trained on *input feature 5 or 3* but rather *on input feature set 1*, the model performance can even be further enhanced. As a reminder: This is the dataset, which includes measurement data from meteorological stations plus the discharge at the target station (29 input features). This data was originally not part of the hyperparameter tuning procedure, since the model with the preliminary hyperparameter settings showed already extremely good performance. However, when training the model with increased model capacity (256 LSTM units) on this *input feature set 1*, the model performance can further be increased with respect to all performance metric scores (except MAX-Error). Nevertheless, this increase in performance is hardly noticeable in the view of prediction accuracy, when comparing the predicted hydrograph to the measured one.

4.5. Multi-step Predictions

To this point all LSTM models predicted the discharge in single steps, which means, that a model predicts only the very next time step (see section 2.2.2). However, this operational mode is more used in a specific analysis environment, like in this thesis, rather than actually performing real-time streamflow forecasts. In this section it should be investigated if the LSTM model can also predict multiple time steps (hours) ahead with reasonable accuracy compared to the LARSIM model. The single step model that reached the highest accuracy during the previous analysis, is the LSTM model trained on *input feature set 1* with 256 hidden LSTM-units considering the preceding 120 time steps. This input feature set includes meteorological measurement data (inside the catchment) plus the discharge observations at Marienthal station (in total 29 features). Hence, this configuration is used to predict the discharge of different lead times, which are 12, 24 and 36 hours, respectively. In this section the prediction capability is investigated in two ways, as described in section 3.7. On the one hand the LSTM model predicts the discharge in one single shot (all hours at once) without incorporating “forecasted” meteorological data during the prediction period. On the other hand the single- step model with customized prediction loop is used (hereafter referred to as: *customized loop model*), which introduces a method of considering real forecasted meteorological data during the multi-step prediction process. This customized prediction loop is tested for two different prediction scenarios in this analysis. In the first test it is used to predict 12 hours ahead, with an update step of 12 hour. To put it in another way, the model predicts the next 12 steps every other 12 hours. In the second test it is used to predict the next 24 hours, but the model restarts the prediction loop with a frequency of one hour.

After calculating the individual performance metric scores of the different models over the course of future time steps, as described in section 3.7, their progression against the prediction steps can plotted. In Figure 36 and Figure 37, the evolution of model performance against future prediction steps can be observed with respect to NSE and KGNP scores, respectively. A figure showing this evolution in the same manner but with respect to the MAE and RMSE score can be found attached to the Appendix (see Figure 41). In Figure 36, which shows the progression regarding the NSE scores, it can be observed that for all models, except for the *custom loop models*, the NSE score decreases steadily if the lead time increases. As a result the further the model predicts ahead, the more inaccurate the discharge predictions are. In general the progression of NSE scores of all single-shot LSTM models follow the same trend as the scores of the LARSIM model. Around a lead time of 16 hours, all models, except the *custom loop models*, experience a slight stronger decrease of the NSE score indicated by a tiny trend change. The almost perfect NSE score of 1 for the LARSIM simulation at the very first prediction step can be explained by the fact that after each 24 hours the predicted hydrograph is shifted into the measured one (ARIMA correction). Since the LARSIM is feed continuously with forecasted data for the prediction period of 24 hours, it has a clear advantage over the LSTM-model. As a consequence, the NSE scores of the LARSIM model for a lead time of up to 20 hours are above the ones of the 24-step single shot model. At the 20th prediction step ahead, the NSE scores of the 24-step single shot LSTM model and the LARSIM model are almost equal. For further prediction steps the NSE of the LARSIM model drops below the NSE of the single shot models. At a lead time of 24 hours the NSE score of the single shot LSTM models (24-step and 36-step model) are above a NSE of 0.86, which suggests, that their prediction accuracy is reasonably well and comparable to the LARSIM simulation. In contrast the *customized loop model* show for both lead times a nearly constant NSE score and thus a progressively high prediction accuracy. This is due to the nature of the developed double loop function, which basically provides the model with new input data after each time step, as described in section 3.7. *The customized loop model* receives, similar to the LARSIM model, some information about the "future" meteorological conditions. Specifically this new data is composed of true observations meteorological data (see section 3.7) and the predicted discharge values of the previous steps. These advantageous characteristics can explain this continuous good performance of *the customized loop model.*

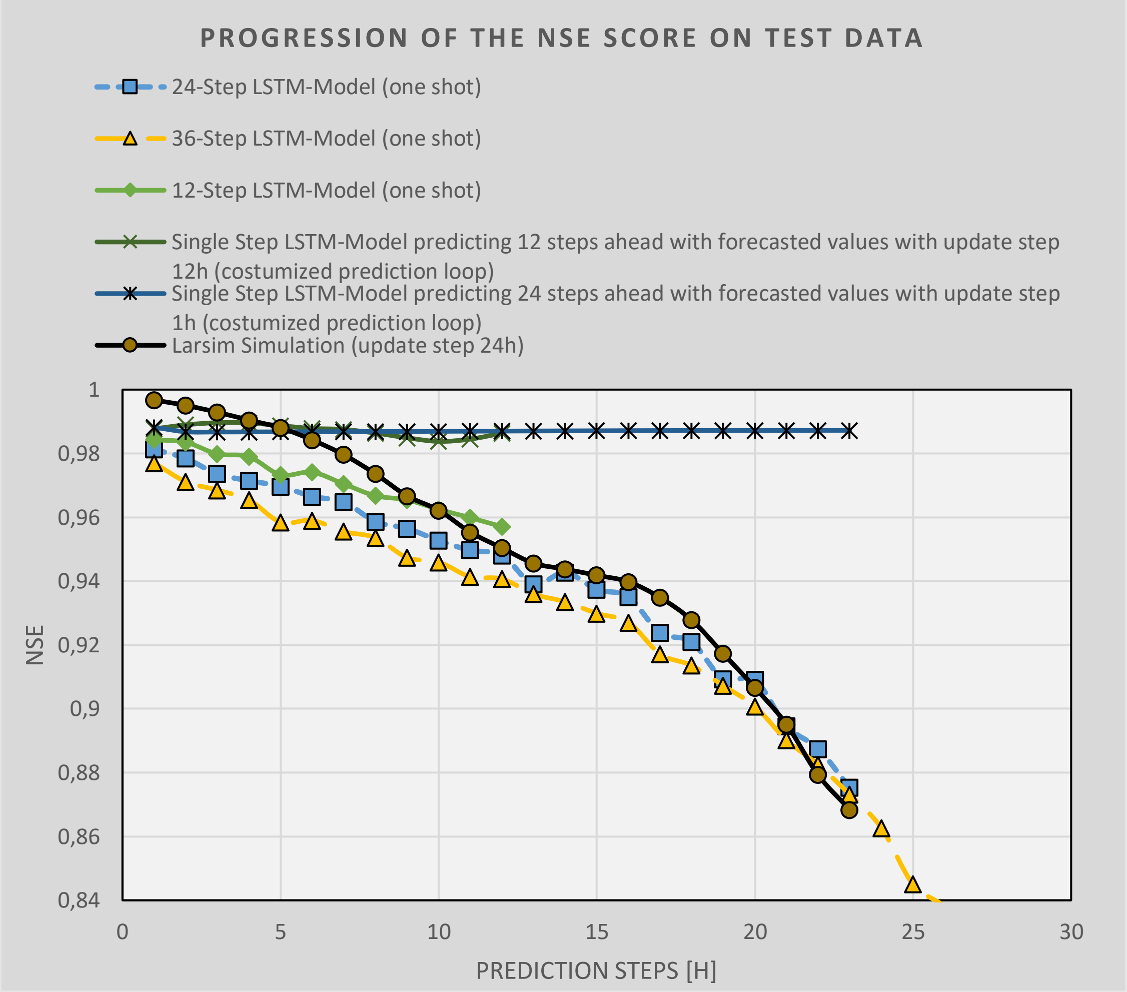


Figure 36: Progression of the NSE score on Test Data over future time steps for all tested models (indicated by different line styles). In order to have more zoomed in view, the graph of the single-shot model with a lead time of 36 steps (yellow triangles), is cut off. For the 36th step this model reached a NSE score of 0.654.

To also have a qualitative measure rather than only a quantitative measure, the decrease of the KGNP score over a prediction time period of up to 30 hours ahead is shown in Figure 37. In principal the aforementioned outcomes and indications are confirmed, when transferred to this figure. Again, the progression of KGNP scores of the single-shot models generally follow the same trend as the scores of the LARSIM model, with the exception of the two model runs based on the customized prediction loop. The KGNP metric, which is essentially a goodness-of-fit measure provides an indication of how well the observed and predicted hydrograph matches with respect to the volume, variability and dynamics of the discharge. The slight decrease of KGNP scores with increasing lead times implies that the individual model predictions deviates more from the observations as the prediction steps are further ahead. Transferred to predicted and measured hydrograph, these deviations are mainly reflected in overestimating peak flows. Furthermore, it seems that, e.g. for the 24th prediction step of a single shot LSTM model, the dynamics of spiking flood peaks are captured with a delay of one or two hours. In contrast, *the customized loop model* is able to provide a constant high accuracy, which eventually reflects in a strong match of predicted and observed hydrograph for the complete lead time of 24 hours.

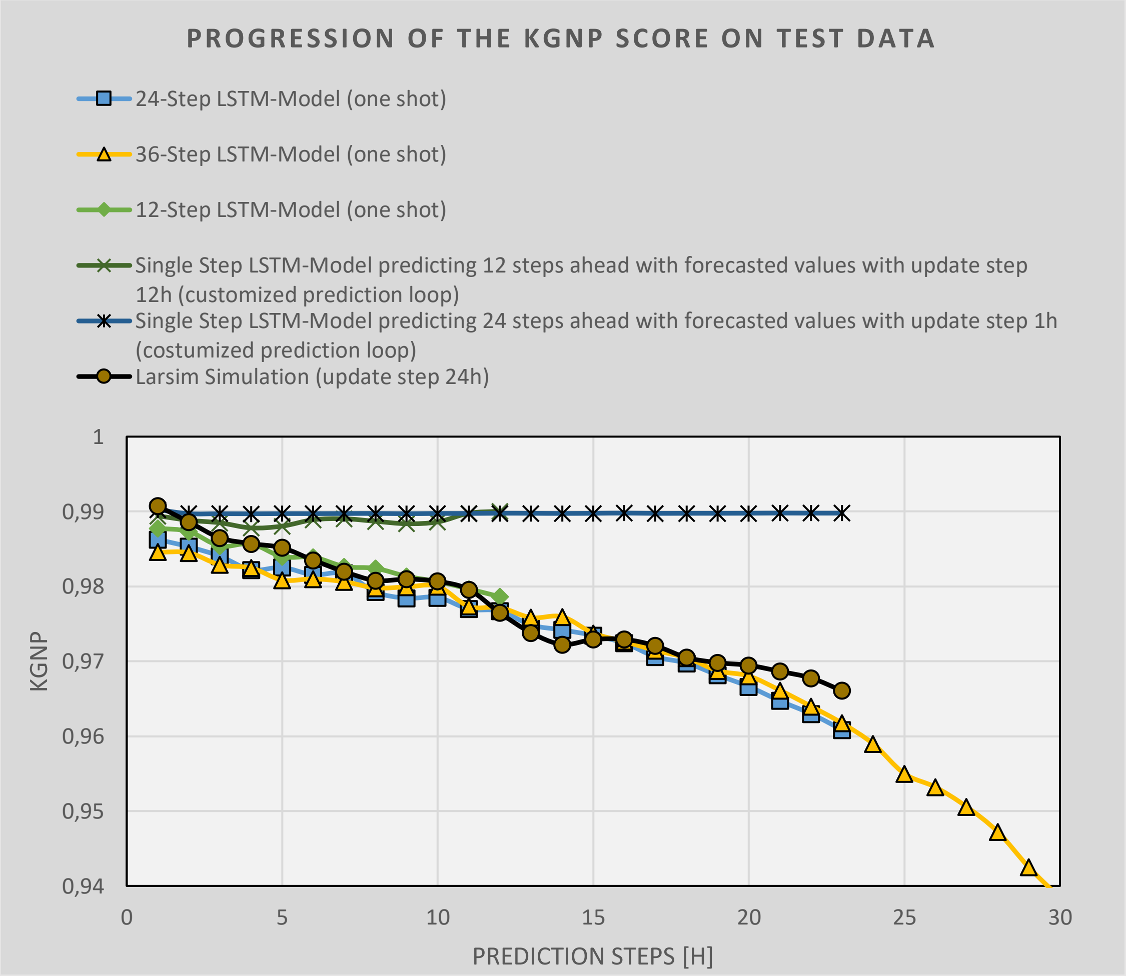


Figure 37: Progression of the KGNP score on Test Data over future time steps for all tested models (indicated by different line styles). In order to have more zoomed in view, the graph of the single-shot model with a lead time of 36 steps (yellow triangles), is cut off. For the 36th step this model reached a KGNP score of 0.912.

It has to be mentioned that the minor deviations in both Figure 36 and Figure 37 between the graphs of the three one shot LSTM models with different lead times has two reasons. On the one hand, each of these three LSTM-models has its own set of trainable parameters, because every particular model is trained on a target window of another size. On the other hand during the training of the models the MAE should be minimized, which is averaged over the lead time (only during training process). Since the lead times is different across the LSTM-models these calculated averages of the MAE differ and though the progression of the metric scores are slightly off. With respect to *averaged* performance metric scores over the lead time of 24 hours, the LARSIM model shows slightly better performance in reference to the corresponding single shot LSTM model (see Table 21 in Appendix). However, obviously the model with customized prediction loop shows superior performance on average (compared to the LSTM model & LARSIM model), as the performance metric scores stay almost constant over the course of 24 hours ahead.

5. Discussion

5.1. Scaling

Regarding the preliminary analysis steps, results show that the LSTM model produces the best performance metric scores on average (10 iterations) if the input data is standardized. However, the discrepancy in performance between the LSTM model and models trained on robustly scaled data is small, especially regarding the NSE. This might be due to the fact that outliers in the measurements are substantially reduced when resampled to daily values (at least for averaged parameters). Thus, robust scaling might not have a considerable influence on the scaling ranges of various parameters in the input data, reducing its leverage effect. Overall, scaling should be considered when training LSTMs, since it generally improves convergence and performance, even though the two described methods do not actually show significant differences in model performance.

5.2. Missing Data

Three different methods for dealing with missing values in the input data have been investigated. It can be argued that *Method 1* and *Method 2* are incorporated in the preprocessing phase, since the input dataset must be modified accordingly. In contrast, for *Method 3*, a specific model structure is required, i.e. the model architecture has to be adjusted in addition. This is actually a disadvantage compared to the other two methods, as the model structure is less complex. *Method 2* is essentially most comparable to the original concept of having a model that is ideally able to learn to identify specific samples/records in the input data as missing. Further, using this approach, the model might be capable of drawing useful information from the missingness patterns within the datasets. As a possible consequence, complex imputation techniques are not required for eliminating data gaps. The LSTM model, which integrates *Method 2* or *Method 3*, showed similar performance, although the performance metric scores of *Method 2* are slightly better.

Therefore, the conclusion can be drawn that, to some degree, knowledge about missing samples is gained by the model without the previous knowledge that a value is signed as missing. *Method 1* performed noticeably worse. This is not surprising, since by applying *Method 1* the amount of input data is substantially reduced, so there are fewer samples that the model could learn from remaining in the input data. However, even if all time stamps (all features) are removed from the input data, the LSTM model is still able to reproduce the observed hydrograph in the original timescale at least with modest accuracy. This suggests that the LSTM model is still able to deal with discontinuous measurement series with irregular measurement intervals.

Method 3 should be applied when the model architect wants to ensure that missing samples in the input are reliably neglected by the model. Nevertheless, this does not necessarily result in a higher model performance. After all, these findings have to be treated with care, because the overall percentage of missing data samples is very low (about 1% or less) compared to the total number of available observations. Hence, the results of the model runs as well as the aforementioned learning effect by the model itself might not be statistically significant. It is hardly distinguishable whether the model shows better performance for *Method 2* or *Method 3* (compared to *Method 1*) either because of the presence of more data, or because the model is actually able to learn from patterns of missing samples. Either way, more extensive research has to be carried out, in which the samples are systematically dropped from the data resulting in a higher percentage of missing observations.

5.3. Imputation of Precipitation

Precipitation is the main streamflow affecting input parameter in a rainfall-runoff simulation, as the name already suggests. Since a continuous rainfall time series is often unavailable, it is a common practice in hydrology to impute missing data samples. Independently of the applied technique, this is always a possible source of uncertainty and could introduce a strong bias to the model, which might not be easy to detect. However, in the framework of this analysis, the application of imputation methods on precipitation data seems to have a positive effect on the prediction results of the LSTM model.

When imputation techniques are applied to the input data, the models show a slight improvement regarding the performance metric scores for a daily and hourly scenario. It appears to be trivial which exact technique is used, as long as the missing rainfall samples are replaced with reasonable values. Additionally, this leads to the conclusion that the applied imputation techniques indeed provide reasonable estimates for missing precipitation data, which is reflected in a higher rainfall-runoff prediction accuracy. Notwithstanding, the percentage of imputed values is very low (only up to approx. 0.4%). Thus, even more sophisticated methods, e.g. multiple linear regression, would probably not have improved the accuracy of the discharge prediction. As mentioned in the preceding paragraph, the level of common statistical significance for the different model tests between various imputed datasets might not be given. However, this hypothesis is not proved and should be one part of future investigations.

5.4. Sensitivity analysis

One of the objectives was to identify parameters that are most responsive to the LSTM model performance. The sensitivity analysis was carried out with the intention of finding suitable input feature sets rather than identifying individual parameters or features that might be most relevant for the model to predict discharge accurately. Within the first part of the sensitivity analysis, in which especially the influence of meteorological features was investigated, the LSTM model showed similar behaviors in both temporal resolutions. Reducing the number of meteorological input features by considering stations only within the watershed boundaries resulted in a slight increase of performance.

These contrasting relations support the intuitive assumption from a hydrological perspective that the influence of stations – regardless of their observation parameters – which are located outside a watershed, might not be decisive for the model performance. This further corroborates that including features of all available stations does not only fail to add any valuable information to the model but also threatens to obscure the model with redundant information. Regarding the scenario analyzed in this thesis, having up to three quarters fewer input features means a less complex input features space with fewer non-linear correlations between meteorological features. The results suggest that this potentially leads to a better convergence to a global optimal point in the search space.

Nevertheless, one advantage of having access to a spatially well distributed measuring network is that it provides a valuable source for imputation and interpolation of missing precipitation data. As expected, it turns out that precipitation is the most crucial meteorological parameter for accurate predictions. However, the model seems to be slightly sensitive to other parameters apart from precipitation as well, even though their effect on the performance seems to be almost negligible. This is confirmed by the fact that the LSTM model produced accurate predictions with an NSE over 0.75, regardless of the temporal resolution – if the input feature set includes the complete variety of available meteorological parameters besides precipitation.

With respect to the applied performance metrices, the LSTM model trained on hourly data shows significantly higher scores compared to the one trained on daily input features. Moreover, only an hourly model outperforms the conceptual benchmark model (LARSIM) with respect to all applied performance metric scores. However, this high accuracy could be explained by taking into account that past discharge observations at the target stations have also been incorporated in the input, suggesting that the model does predominantly learn from the feature it actually tries to predict. To examine this hypothesis, another objective was to exclusively investigate the influence of additionally considering discharge measurements as part of the model input.

In the second part of the sensitivity analysis, the LSTM model fails to continue showing consistent results when comparing daily and hourly predications. In a daily scenario, adding the upstream located discharge gauges leads to an overall performance increase of the model. Before including additional discharge observations in the input, it was difficult for a daily model to capture peak flows. There are two possible reasons for this. On the one hand, using the MAE as loss function gives less weight to high values and on the other hand, the temporal information about periods with high rain intensities is lost during the resampling process. Though, these high intensity rain events might lead to fast surging discharge peaks in this catchment. However, as soon as data from discharge gauges is considered as part of the input, a daily model is able to capture the peaks in the hydrograph more accurately, whereas baseflow conditions were slightly overestimated. This holds true even if the discharge at the target station is excluded from the input. Apparently, the information of the hydrograph at *Marienthal* station is partially composited of the discharge rates at all other upstream discharge gauges, although the discharge gauges are widely distributed across the catchment area. In other words, all the runoff generated in the catchment is routed through *Marienthal* station to a certain extent, reflecting the dynamics of rainfall-runoff response more transparently for the model.

In an hourly setting, once additional discharge data is incorporated, the model performance drops, which is the opposite behavior to the one observed in the daily scenario. This discrepancy could be attributed to low correlations among hourly discharge observations between all gauges compared to daily resolution, making it harder for the hourly model to learn valuable dependencies. This is mainly reflected in a worse prediction accuracy regarding the peak flows, but especially the baseflow periods are systematically overestimated. Hence, the LSTM model based on hourly resolution performance drops below the one of the benchmark model, when considering discharge measurements in the input data at other gauges apart from the target station.

Furthermore, a third objective was to test how well the LSTM can model the rainfall-runoff relation within the catchment without any knowledge on discharge, i.e. solely using meteorological data. Neglecting any discharge stations and training the model exclusively on meteorological data resulted in the worst LSTM performance with respect to both temporal resolutions. In both scenario, the LSTM model does not provide satisfactory performance metric scores. This is demonstrated by examining the predicted hydrographs visually for the models, which both yield an overall poorly fit compared to the observed discharge. However, for a daily data basis, the discharge hydrograph indeed captures temporal dynamics of the discharge to a certain extent, which is consistent to the general hydrological understanding. This suggests that the model can, at least to some degree, approximate hydrological rainfall-runoff relations within the catchment when trained on a coarser resolution.

In contrast, the LSTM model trained on data with hourly resolution does not appear to be able to learn the complex dependencies in rainfall-runoff processes in this catchment at all. This is slightly surprising, since for instance short and highly intensive rain events ordinarily have an impact on streamflow on a sub-daily timescale. Hence, the conception of this causal relation should be better reflected in a model that receives hourly measurement data. One possible explanation is that high peak flows might not mainly be driven by these types of rain events within this watershed. From a computational perspective, another potential explanation is that the hyperparameters might not have been properly chosen. For example, there might not be enough trainable parameters in the current model configuration to enable it to converge to a more optimal solution. Moreover, these findings corroborate with the previously mentioned hypothesis that the model trained on hourly data, in which the target discharge is included in the input, does predominantly learn from the feature it actually tries to predict, i.e. the discharge measurements at *Marienthal* station. Overall, the prediction capability of a model exclusively trained on meteorological input data, independent of the temporal units of the data, is unreliable and cannot provide satisfying accuracy.

In order to be able to identify individual parameters or stations (rather than sets of features) that have the most or the least influence on model performance, a more detailed analysis should be undertaken. This further analysis should put the main focus on a structural search of particular parameters or features that are the most decisive ones for model response. Besides this, a more advanced method exists that could be utilized in further research to identify the most dominant input features. This method is called integrated gradients, first introduced by Sundararajan et al. (2017), and follows a more mathematical perspective. These integrated gradients can quantify how the LSTM model combines meteorological features over time and space to predict the current streamflow (Kratzert et al. 2020). Additionally, it would be interesting to investigate the effect of the spatial distribution of measuring stations within a catchment.

5.5. Hyperparameter Tuning

One of the main objectives of the hyperparameter tuning was to investigate whether the preliminary chosen lookback windows, regarding both temporal resolutions, were appropriately chosen considering an unknown catchment response time regarding precipitation events. Furthermore, it should be examined whether the model capacity, reflected in the number of LSTM-units, is sufficient for particular problem domains and whether the model performance could be enhanced. The grid-search procedure focused specifically on input feature sets, which showed a poor performance in the sensitivity analysis. Moreover, the influence of changing the objective function on model prediction accuracy should be explored.

Due to the mathematical formulation of the MSE and NSE, models trained with these functions resulted in the expected model behavior on daily data basis. The MSE and NSE put more weight on larger errors than on smaller ones. This is reflected in an increased prediction accuracy for flood peaks considering exclusively meteorological input data. However, changing the objective function the model is trained with does generally not provide an increase in performance as big as previously assumed.

In contrast, as soon as other discharge measurements are incorporated, changing the objective function does not show any significant differences anymore. In an hourly scenario, switching the objective functions even resulted in a minor performance drop and decreased accuracy in all tested cases, respectively. All these findings suggest that switching the objective function does not have a significant impact and that the MAE, which provides a generic and even measurement of how well the LSTM model is performing, is proven to be a decent choice for hydrological streamflow forecasting.

Modifying the LSTM-units and the lookback window size showed very diverse results, which suggests that there is an irregular interaction between these two hyperparameters, which is in turn dependent on the input features. For example, once additional discharge stations are part of the input, the hyperparameter tuning showed less effect; this was indicated by either a slight decrease or a neglectable increase of performance. Both hyperparameters seem to have a higher impact on model performance than changing the objective function. Regarding models predicting on a daily timescale, reducing the window size to 60 days while simultaneously increasing the LSTM units to 256 lead to the overall best performance. This suggests that hydrological catchment processes that are relevant for the rainfall-runoff relation within the study area are no longer delayed than 60 days. In other words, the additional 60 precedent days (the period of 60 to 120 days in the past compared to current prediction time ***t***) might not be influential on the discharge prediction.

In combination with an increased model capacity, i.e. increased dimensionality of each hidden unit, the model can potentially learn to remember more distinctive attributes that are hidden within the input features. Thus, more of these hidden characteristics can be kept in the memory of the LSTM for the previous 60 days, which might provide additional information for the model to predict the discharge with higher precision. In contrast, LSTM models that predict on hourly timescale are configured by setting the window size to 120 hours to focus on processes in the catchment with short response times. The choice of 120 hours as a lookback window seems reasonable for the catchment size and proved to be appropriate as soon as any discharge observation was included in the input. Just increasing the model capacity while training an hourly model exclusively on meteorological data does not seem to be sufficient enough for the model to learn the non-linear relations in the rainfall-runoff processes for this catchment. Hence, there is abundant space for further progress in analyzing LSTM models exclusively trained on meteorological data and how their prediction accuracy can be enhanced. This could be done, on the one hand, from a hydrological perspective and, on the other hand, from a model-architectural perspective, in which for instance the hyperparameter optimization is carried out in a wider search space.

5.6. Multi-step Forecasting

For all tested models, the performance metric scores decrease over the course of prediction steps, which is expected, as it gets more difficult for the model to accurately predict multiple steps with increasing lead time. This is specifically true for the single-shot LSTM models, which do not receive any data and thus no information of future meteorological conditions at all. Nevertheless, their performance can well compete with the prediction accuracy of the conceptual benchmark model at least for a lead time of up to 24 hours. The benchmark model starts off with a slightly higher NSE score for short lead times. However, with further prediction steps, the performance metric slowly converges to the scores of the single shot LSTM models and eventually drops below them. This is to some extent surprising, since the LARSIM model incorporates “forecasted” data within the lead time window during the model prediction process. Further, the result is even more unexpected considering that these “forecasted” observations, in this case, are actually real recorded measurements. This suggests that the LSTM models is able to remember not only characteristic patterns within the past discharge observations over a certain period but also the sequence of these patterns. As a consequence, the LSTM model seems capable of mapping the correlations in the input features to a single output step (discharge value) roughly as well as to at least a multiple of up to 24 time steps. This is confirmed by comparing the performance of the single step model with the averaged performance of the multi-step model, as they both show reasonably good performance metric scores. Regarding the averaged performance, the LARSIM model shows slightly better performance metric scores for a lead time of 24 hours compared to the corresponding single shot LSTM model. This is also reflected in the comparison of predicted hydrographs, in which the temporal dynamics, especially the timing of the flow peaks and their rising limps, seem to be more accurately captured by the benchmark model.

In contrast, the introduced method takes the advantage of a customized prediction loop, which partly circumvents the drawbacks of the single shot multi-step LSTM model by regularly updating the lookback window the predictions are based on. This procedure offers the possibility to feed the model with forecasted meteorological forcing data for each future prediction step. However, in this thesis, this feature is used to provide the LSTM model with true measurements and the predicted discharge values of the past steps, which the LARSIM model receives as well. Despite this affinity, there is one difference between the benchmark and the *customized loop model*. The latter one is not able to update its internal parameters or to apply any other kind of correction procedure during the prediction process. Nonetheless, due to the aforementioned advantages, the *customized loop model* shows superior performance compared to all other tested models. After all, these results have to be treated with care, since the customized prediction loop actually utilizes a trained single step LSTM model but within a loop to predict multiple steps ahead. However, to employ true forecasted meteorological data, it would require a potential link to a weather forecasting model. Theoretically, the analysis forecasting models could be used to predict other meteorological or hydrological features in addition to streamflow. This has already been achieved by operational conceptual hydrological models. There are architectural configurations, in which the LSTM model can be trained in an autoregressive loop, but this requires the model to predict *all* used meteorological input features simultaneously (in addition to discharge), before feeding the forecasted values back as input for the next time step. Hence, multi-objective optimization of LSTM models is an open branch of future research.

Apart from all the above mention limitations, there are another two substantial limitations of the research conducted in this thesis. The first one is the fact that all presented LSTM models were only tested on discharge events, which are (almost twice) below the annual peak discharge (Q = 304 m³/s) and not very close to generally observed extreme events (highest peak in test data: Q ≈ 170 m³/s; highest peak ever observed: Q = 720 m³/s). Hence, there is no evidence of the generalization ability of the LSTM models in predicting extreme events of this magnitude (e.g. HQ_{100}). The main reason for this is that the available test period does not include such high extreme events. However, to compensate these limitation, further studies could explore the LSTMs on new available data, which contain a particularly high number of extreme events. Furthermore, more sophisticated data splitting methods could be applied for creating training, validation and test sets, respectively.

The second main limitation is the missing estimation of predictive uncertainty, as already stated previously. However, the prediction uncertainty is important for assessing in how far to trust the forecast produced by the models. Hence, another part of further research should incorporate methods, such as the "relative error method", "Discharge Intervals" or "Slope interval methods" introduced in Leandro et al. (2019). These methods could be applied to quantify the predictive uncertainty of discharge forecasts, by providing upper and lower uncertainty bounds (Leandro et al. 2019). There are also alternative approaches, which can take model uncertainties, inherent noise, and model misspecifications into account from a Bayesian perspective. For example, this could be achieved by using a Bayesian NN, which is a basic framework to provide uncertainty estimations of DL models (Zhu and Laptev 2017).

6. Conclusion and Outlook

This thesis utilizes a state-of-the art deep learning method to test its potential applicability for rainfall-runoff modeling and streamflow forecasting in the watershed of the river *Regen* located in Germany. That deep learning method is called long short-term memory network and has the powerful ability to extract information from a large and diverse dataset containing any possible hydrological and meteorological variables. This thesis demonstrated that a developed data-driven model has the capability to learn the complete "hydrological model" purely from the available data. The data basis in this study did not consist of, as commonly used in other papers, highly preprocessed meteorological forcing products, but raw measurement records of stations located within the catchment. Further, this study also identified the most valuable input feature combination among all available stations. What is more, a method is proposed to handle missing data gaps within the input features simultaneously enhancing the prediction accuracy of the LSTM. The results indicate how well LSTM networks can predict cases they were not trained for and principally not only in a single step manner but also in a multi-step scenario with lead times of up to 24 hours. Moreover, to approach a real-world operational flood forecasting scenario, a possible method is provided to theoretically consider forecasted meteorological data in the input of a LSTM during the prediction period.

The research in this study was conducted in a continuously progressive way. It started off with a preliminary assessment, which provided the foundation for the preprocessing steps and helped to create a proper data basis the LSTM model could handle. Furthermore, this assessment explored suitable methods of dealing with missing samples in data. A sensitivity analysis was carried out to investigate the influence of particular combinations of features in the input data for the LSTM model. Further, the sensitivity analysis explored the effect of considering multiple other discharge gauges besides the discharge measurements at the target station as part of the input. Moreover, the solitary impact of including observations at the target station in the input data on the model prediction accuracy was investigated. In the next step, hyperparameters were adapted in a grid-search manner to find more appropriate model settings from a hydrological perspective with respect to the problem domain. Additionally, the ability of LSTMs to forecast multiple time steps ahead regarding different lead times was examined. Finally, the customized prediction loop was tested in order to explore the possibility to feed potentially forecasted meteorological data to the LSTM during the forecasting procedure. The performance of the different LSTM models was continuously compared to the conceptual and physical-based LARSIM model throughout the thesis.

The preliminary analysis showed that for the underlying data basis in general, the standard scaling procedure in combination with applying missing values *Method 2* are the most appropriate settings regardless of the temporal resolution of the data. Further, it was demonstrated that imputing missing precipitation samples by using appropriate techniques generally improved the model performance. Despite the small number of samples, imputing these data gaps with unphysical values proved to be a valid technique to deal with remaining missing samples and resulted in an enhanced prediction performance.

The sensitivity analysis demonstrated that the combination of input features out of a mixture of various meteorological parameters has a significant impact on the model performance. Investigation showed that the model trained on hourly data is more sensitive to the presence of discharge measurements at the target station in the input data than the model trained on daily basis. This result suggests – although this could not be proven within the scope of this thesis – that the model is able to remember the sequence of characteristic patterns in the data, especially in the hydrograph observed at the target station. Moreover, the results revealed that precipitation is the most crucial meteorological parameter for accurate discharge prediction, although concurrently not the stand-alone feature either. What is more, other meteorological parameters proved to have at least a minor impact on the performance of a rainfall-runoff LSTM model. With the appropriate combination of input features and suitable hyperparameter settings, the LSTM model could outperform a conceptual physic-based model, which calibrated for area of investigation. On hourly basis, the generalization ability of the single step LSTM model proved to replicate the discharge hydrograph almost perfectly on unseen data. In contrast, results from the sensitivity analysis also revealed that within the framework of this thesis none of the configurations of the LSTM model is able to reproduce the rainfall-runoff processes in the catchment exclusively from meteorological input data. Overall, it can be concluded that more data does not necessarily lead to a better prediction model, which is a common misconception regarding DL models. It is rather the right combination of input features and the correct type of data that leads to the most accurate predictions.

The hyperparameter tuning revealed that changing the objective function the model is trained on does not have as much of an impact on model performance as changing the window size and the number of LSTM-unit. Adapting these specific hyperparameters showed especially an enhanced accuracy in capturing flow peaks and base flow periods. It was proved that adapting the window size and the model’s capacity, a quite significant increase in performance could be achieved. It was found that this impact on performance is again dependent on the type and amount of input features, supporting the conclusion that the configuration and model settings of each LSTM model has to be accurately adapted regarding the desired prediction task and the available data foundation.

The research on multi-step short-range streamflow forecasting demonstrated that LSTM models can compete in terms of prediction accuracy with the LARSIM model regarding a lead time of up to 24 hours. Moreover, it was proven that the LSTM models were able to reach the performance of the benchmark model in an operational setting, in which the LSTM had no access to any forecasted meteorological data for the prediction period, in contrast to the LARSIM model. Furthermore, the investigation revealed that with increasing lead time, the prediction accuracy of the LSTM models decreases in terms of peak-timing. The introduced method of applying a customized prediction loop demonstrated one valid procedure of potentially incorporating forecasted meteorological data in the prediction process. By using a single step model within this customized prediction loop, the multi-step ahead prediction showed a constant high performance for all future prediction steps and outperformed every other model tested in this thesis. The aforementioned outcomes highlight the flexibility of machine learning models and demonstrate the ability of LSTM networks to learn abstract patterns of complex input-to-output relationships. Besides the ideas mentioned in the discussion section, there still remain a number of steps to be taken before these models can be truly operational.

A first step would be to investigate the performance of the presented LSTM model in more detail, with respect to correlation errors between hydrologic signatures of observed and predicted discharge. With more advanced performance metrices, a better interpretation, comparability and a stronger indicator of good prediction accuracy could be provided to compare different LSTM models. In turn, this could help to optimize hyperparameter settings and to find the most favorable input feature combinations for LSTMs with respect to the problem domain. A second step would be to apply different techniques to obtain the whole spectrum of sources for uncertainties within the LSTM model and the prediction process. For reliable flood forecasting, it is important to understand where these sources are, how to accurately estimate uncertainties, and most importantly how to reduce them. This is especially important when LSTM models should be able to forecast extreme flood events with peak flows of a return period of 100 years or above. A third step would be to find a suitable way to link an appropriate weather forecast model to a LSTM model. The former one could provide the necessary forecasted meteorological input data for the prediction period, whereas the latter one would conduct the streamflow predictions. In conclusion, LSTM models could help to obtain a more general understanding of the rainfall–runoff processes and may play a key role in the design of environmental forecasting systems in the near future.

Appendix

A. Tables

Table 12: Complete list of available measurement stations. "Parameters" refers to observed parameters at corresponding station. "Parameters_considered" refers to the parameters actually considered in the input dataset. Some parameters had to be discarded due to a large amount of missing data.

Station	Stationsnummer	Stationskennung	X-Koordinate	Y-Koordinate	Höhe	Koordinatensystem	Parameters	Parameters_considered
Aholfing	46	AHOL	4534250	5423180	330	31468	['n']	['n']
Hagelstadt	103	HAGE	4516000	5417680	365	31468	['n']	['n']
Altendorf	118	ALTE	4520820	5474280	391	31468	['n']	['n']
Beratzhausen	359	BERA	4485190	5440570	463	31468	['n']	['n']
Bogen-Pfelling	589	BOGE	4554760	5416630	345	31468	['n']	['n']
Kelheim	2550	KELH	4491410	5420780	350	31468	['n']	['n']
Lindberg-Buchenau	3012	LIND	4597170	5433560	740	31468	['n']	['n']
Metten	3271	METT	4567500	5413420	313	31468	['n', 'rflu', 'tlu', 'zsos']	['n']
Nabburg	3429	NABB	4512680	5480320	366	31468	['n']	['n']
Neukirchen bei Heiligen Blut	3525	NEUH	4570020	5458380	460	31468	['n']	['n']
Nittenau-Harting	3617	NITT	4520450	5447660	430	31468	['n']	['n']
Oberviechtach	3739	OBEV	4531758	5479545	596	31468	['n', 'rflu', 'tlu']	['n', 'rflu', 'tlu']
Roding-Neubäu	4224	RODI	4530140	5456100	388	31468	['n']	['n']
Saldenburg-Entschenreuth	4354	SALE	4596705	5405909	456	31468	['n', 'rflu', 'tlu', 'zsos']	[]
Sankt Englmar	4387	SANK	4560410	5429840	810	31468	['n']	['n']
Schmidgaden	4493	SCHM	4505620	5474990	416	31468	['n']	['n']
Schmidmühlen	4494	SMID	4493900	5460340	453	31468	['n']	['n']
Schorndorf-Knöbling	4559	SCHK	4545140	5447650	399	31468	['n', 'rflu', 'tlu']	['n', 'tlu']

Schwandorf	4592	SCHO	4506440	5465649	356	31468	['n', 'rflu', 'tlu', 'zsos']	[]
Stallwang	4816	STAL	4546870	5435720	390	31468	['n']	['n']
Teisnach	5002	TEIS	4572814	5434399	400	31468	['n']	['n']
Teublitz	5013	TEUB	4506830	5453700	350	31468	['n']	['n']
Treffelstein-Witzelsmühle	5036	TIEF	4544106	5474255	470	31468	['n']	['n']
Weiding, Kreis Cham-Dalking	5401	WEID	4554330	5459700	425	31468	['n']	['n']
Wiesenfelden-Utzenzell	5548	WIES	4540360	5432900	680	31468	['n']	['n']
Osterhofen-Thundorf	6161	OSTH	4574600	5402720	310	31468	['n']	['n']
Kirchberg/Niederbayern - Zell	6191	KIBE	4583960	5418950	705	31468	['n']	['n']
Lam-Lambach	6215	LAML	4577790	5453680	692	31468	['n']	['n']
Neunburg vorm Wald-Eixendorf	6281	NENB	4532630	5469430	420	31468	['n']	['n']
Moos, Kr. Deggendorf-Maxmühle	6296	MOOS	4570510	5404650	320	31468	['n']	['n']
Prackenbach-Neuhäusel	7350	PRNE	4560206	5441601	588	31468	['n', 'rflu', 'tlu']	[]
Regensburg WST	10776	REGE	4507283	5434477	366	31468	['n', 'rflu', 'tlu', 'ttau', 'xglob', 'xludr', 'xwind', 'zsos']	['n', 'rflu', 'tlu', 'ttau', 'xludr', 'xwind', 'zsos']
Waldmünchen	10782	WALU	4549742	5472914	499	31468	['n', 'rflu', 'tlu', 'ttau', 'xglob', 'xludr', 'xwind', 'zsos']	[]
Grosser Arber	10791	GROA	4582449	5442870	1446	31468	['n', 'rflu', 'tlu', 'ttau', 'xludr', 'xwind', 'zsos']	[]
Zwiesel	10796	ZWIE	4589909	5432976	612	31468	['n', 'rflu', 'tlu', 'ttau', 'xludr', 'xwind', 'zsos']	['n']
Eging am See-Rohrbachholz	14301	EGIG	4593837	5399947	415	31468	['n']	[]
Regensburg-Uniklinik	107740	REUN	4507319	5427778	406	31468	['n', 'rflu', 'tlu', 'xwind']	[]
Nabburg	107780	NABU	4514502	5479693	385	31468	['n', 'rflu', 'tlu', 'xwind']	[]
Eschlkam	107890	ESCH	4566669	5463395	500	31468	['n', 'rflu', 'tlu', 'xwind']	[]
Lalling	107900	LALL	4583176	5413565	430	31468	['n', 'rflu', 'tlu', 'xwind']	[]
Klingenbrunn	108900	KLIN	4598912	5423092	790	31468	['n', 'rflu', 'tlu', 'xwind']	[]

Großer Falkenstein	108961	GROE	4593750	5439894	1308	31468	['n', 'tlu', 'xwind']	[]
Uttenkofen	200003	UTTE	4562132	5408813	323	31468	['n', 'rflu', 'tlu', 'xglob', 'xwind']	['n', 'rflu', 'tlu', 'xglob', 'xwind']
Köfering	200017	KOEF	4514285	5422075	350	31468	['n', 'rflu', 'tlu', 'xglob', 'xwind']	['n', 'tlu']
Sarching	200018	SARC	4517028	5430644	330	31468	['n', 'rflu', 'tlu', 'xglob', 'xwind']	['n', 'rflu', 'tlu', 'xglob', 'xwind']
Kitzenried	200033	KITE	4530300	5463450	470	31468	['n', 'rflu', 'tlu', 'xglob', 'xwind']	['n', 'rflu', 'tlu', 'xglob', 'xwind']
Steinach	200042	STEC	4544810	5426840	350	31468	['n', 'rflu', 'tlu', 'xglob', 'xwind']	['n', 'rflu', 'tlu', 'xglob', 'xwind']
Wullnhof	200066	WULL	4543681	5472182	510	31468	['n', 'rflu', 'tlu', 'xglob', 'xwind']	['n', 'rflu', 'tlu', 'xglob', 'xwind']
Sitzenhof	200107	SITZ	4504440	5465982	370	31468	['n', 'rflu', 'tlu', 'xglob', 'xwind']	['n', 'rflu', 'tlu', 'xwind']
Eiersdorf	200119	EIER	4487698	5435171	537	31468	['n', 'rflu', 'tlu', 'xglob', 'xwind']	[]
Pösing	200120	POES	4540800	5455050	380	31468	['n', 'rflu', 'tlu', 'xglob', 'xwind']	['n', 'rflu', 'tlu', 'xglob']
Allmannsdorf	200127	ALMD	4559056	5443944	557	31468	['n', 'rflu', 'tlu', 'xglob', 'xwind']	[]
Eiserszell	200132	EISZ	4544195	5435272	617	31468	['n', 'rflu', 'tlu', 'xglob', 'xwind']	[]
Kirchberg	200135	KIRG	4587321	5418452	624	31468	['n', 'rflu', 'tlu', 'xglob', 'xwind']	[]
Eschlkam	200146	ESCL	4565700	5463845	444	31468	['n', 'rflu', 'tlu', 'xglob', 'xwind']	[]

Table 13: Pseudo Code of the customized prediction loop (left) and corresponding array dimensions (right)

1. **Input**: *test_x*	Array: [#windows, window size, #features]
2. **Define parameters**: *update_step (u), prediction_steps(n)*	Scalar
3. **Initialize**: *test_y_predict_multi*	Array:[]
4. **For** *i ….. #windows[test_x] – n* **step**: *u* **do:**	
Initialize: *window_prediction_steps*	Array:[]
Copy: *test_x[i : i + n][:,:]* → *test_x_C*	Array:[n, window size ,features]
For *s ….. n* **do**:	
Expand array dimension: *test_x_C[s]*	Array: [window size, features] → Array: [1, window size, features]
Predict: *y_s*	Array:[1, 1]
Convert to scalar: *y_s* → *y_s*	Scalar
Append: *y_s* → *window_prediction_steps*	Array:[1, …, s]
If *s* < *prediction_steps* - 1:	
test_x_C[s + 1][-1,-1] → y_s	
Append: *window_prediction_steps* → *test_y_predict_multi*	
5. **Reshape:** *test_y_predict_multi*	Array: [(windows – n)/ u, n]

Table 14: Resulting performance metric scores calculated for Training, Validation and Test Period, respectively. Scores highlighted in red indicate best score per metric within one input feature set.

Loss Functions vs. Hourly Data		Input feature set 3			Input feature set 5		
		Training	Validation	Test	Training	Validation	Test
MAE	Max Error	71.54691	62.99847	65.48376	134.43320	157.24356	168.27906
	MAE	1.41036	2.78776	5.53163	5.77063	12.82795	9.56902
	MSE	6.83703	23.18207	60.27196	96.67410	448.47428	294.30330
	RMSE	2.61477	4.81478	7.76350	9.83230	21.17721	17.15527
	MeAE	0.87801	1.86386	3.81733	3.25380	8.00427	5.49932
	REV	0.20512	-0.65835	4.50727	-4.60044	24.86953	12.64088
	NSE	0.99246	0.94787	0.85610	0.89341	-0.00845	0.29733
	KGNP	0.98774	0.94254	0.71620	0.84249	0.48211	0.70729
MSE	Max Error	23.34827	69.41605	82.74979	63.10980	169.88589	147.66061
	MAE	1.50714	2.93569	6.85119	6.59831	13.90105	11.18246
	MSE	4.97889	26.65503	92.90125	84.43324	500.44116	326.86888
	RMSE	2.23134	5.16285	9.63853	9.18876	22.37054	18.07952
	MeAE	1.06070	1.92925	4.48065	4.88747	8.99265	6.74408
	REV	0.15255	0.24336	5.66633	-0.44768	36.08189	20.90230
	NSE	0.99451	0.94006	0.77819	0.90691	-0.12531	0.21958
	KGNP	0.98676	0.92841	0.64515	0.80862	0.43561	0.65726
NSE	Max Error	22.83310	68.62089	82.40802	71.51025	169.32424	167.50821
	MAE	1.48860	2.93261	6.64309	6.54957	14.27290	11.15339
	MSE	4.87928	26.11210	87.99399	83.95735	528.54998	347.21338
	RMSE	2.20891	5.11000	9.38051	9.16282	22.99021	18.63366
	MeAE	1.05234	1.95192	4.41806	4.81946	9.05518	6.57384
	REV	0.09439	0.16406	5.40550	-0.62398	36.97774	21.69642
	NSE	0.99462	0.94128	0.78991	0.90744	-0.18851	0.17101
	KGNP	0.98705	0.93150	0.65244	0.80945	0.42608	0.65888

Table 15:Resulting performance metric scores calculated for Training, Validation and Test Period, respectively. Scores highlighted in red indicate best score per metric within the complete table.

		Input feature set 5					
Lookback vs. LSTM-Units Hourly Data		Units: 120			Units: 256		
		Training	Validation	Test	Training	Validation	Test
Window size: 120	Max Error	134.43320	157.24356	168.27906	67.85744	201.81418	77.28967
	MAE	5.77063	12.82795	9.56902	4.07998	12.48574	9.78383
	MSE	96.67410	448.47428	294.30330	48.57171	421.32800	201.22729
	RMSE	9.83230	21.17721	17.15527	6.96934	20.52628	14.18546
	MeAE	3.25380	8.00427	5.49932	2.44656	7.66227	6.36482
	REV	-4.60044	24.86953	12.64088	-1.81391	23.54498	7.34105
	NSE	0.89341	-0.00845	0.29733	0.94645	0.05259	0.51956
	KGNP	0.84249	0.48211	0.70729	0.89969	0.47015	0.66390

Table 16: Resulting performance metric scores calculated for Training, Validation and Test Period, respectively. Scores highlighted in red indicate best score per metric within the complete table.

		Input feature set 3					
Lookback vs. LSTM-Units Hourly Data		Units: 120			Units: 256		
		Training	Validation	Test	Training	Validation	Test
Window size: 120	Max Error	71.54691	62.99847	65.48376	29.58212	44.99391	52.68394
	MAE	1.41036	2.78776	5.53163	1.11393	2.75309	4.72466
	MSE	6.83703	23.18207	60.27196	3.35386	18.88645	46.48347
	RMSE	2.61477	4.81478	7.76350	1.83136	4.34585	6.81788
	MeAE	0.87801	1.86386	3.81733	0.72577	1.86360	3.35900
	REV	0.20512	-0.65835	4.50727	0.24226	-2.22571	-5.35895
	NSE	0.99246	0.94787	0.85610	0.99630	0.95753	0.88902
	KGNP	0.98774	0.94254	0.71620	0.99008	0.93383	0.86405

Table 17: Resulting performance metric scores calculated for Training, Validation and Test Period, respectively. Scores highlighted in red indicate best score per metric within the complete table.

		Input feature set 1					
Lookback vs. LSTM-Units Hourly Data		Units: 120			Units: 256		
		Training	Validation	Test	Training	Validation	Test
Window size: 120	Max Error	58.49919	57.78577	31.88927	49.63545	58.03492	36.29649
	MAE	1.11405	1.41415	1.29055	0.91396	1.22933	1.16665
	MSE	4.39644	7.72361	5.00020	2.66143	5.97628	4.72741
	RMSE	2.09677	2.77914	2.23611	1.63139	2.44464	2.17426
	MeAE	0.60758	0.79544	0.75433	0.52032	0.67076	0.68197
	REV	-0.24937	1.23141	-0.10534	-0.16919	0.82414	-0.01669
	NSE	0.99515	0.98263	0.98806	0.99707	0.98656	0.98871
	KGNP	0.99328	0.97993	0.98836	0.99451	0.98471	0.99027

Table 18: Resulting performance metric scores calculated for Training, Validation and Test Period, respectively. Scores highlighted in red indicate best score per metric within one input feature set.

Loss functions vs. Daily Data		Input feature set 3			Input feature set 5		
		Training	Validation	Test	Training	Validation	Test
MAE	Max Error	166.48567	61.24812	54.73794	213.86329	142.73939	121.84778
	MAE	3.01417	3.62553	4.95731	4.87026	6.34230	6.47695
	MSE	54.83620	41.50689	60.40066	147.32681	120.47554	176.35427
	RMSE	7.40515	6.44258	7.77179	12.13783	10.97613	13.27984
	MeAE	1.31925	2.34818	3.49870	1.93344	4.27259	3.26866
	REV	-2.45806	-1.43021	-0.34832	-5.05781	8.82931	-5.59361
	NSE	0.93414	0.90690	0.85968	0.82306	0.72977	0.59030
	KGNP	0.96322	0.93654	0.82007	0.92622	0.87023	0.83190
MSE	Max Error	83.51384	63.52064	57.31191	119.86014	132.78917	96.06190
	MAE	3.41231	4.42887	4.74316	5.60231	7.51185	7.04087
	MSE	32.45247	47.80842	57.60874	87.44698	137.26683	134.96428
	RMSE	5.69671	6.91436	7.59004	9.35131	11.71609	11.61741
	MeAE	2.05856	3.26372	3.05917	3.40603	5.26221	4.85161
	REV	0.57259	3.55432	-3.21988	-0.59915	13.12392	6.80926
	NSE	0.96102	0.89277	0.86616	0.89497	0.69211	0.68645
	KGNP	0.95683	0.89571	0.84760	0.89621	0.80277	0.82030
NSE	Max Error	168.73225	54.78302	58.70370	212.31050	141.30550	111.96499
	MAE	3.21596	3.86554	4.70676	5.56273	7.42264	6.52151
	MSE	50.33928	42.36220	62.94731	149.42563	142.79438	149.90292
	RMSE	7.09502	6.50863	7.93393	12.22398	11.94966	12.24348
	MeAE	1.80856	2.60728	2.89322	3.03413	5.30825	4.26812
	REV	-0.97073	1.79079	-4.27555	-3.29526	11.91166	0.16780
	NSE	0.93954	0.90498	0.85376	0.82053	0.67971	0.65175
	KGNP	0.96359	0.92641	0.82220	0.91342	0.82149	0.83727

Table 19: Resulting performance metric scores calculated for Training, Validation and Test Period, respectively. Scores highlighted in red indicate best score per metric within the complete table.

Lookback vs. LSTM-Units Daily Data: Input feature set 5		Units: 120			Units: 256			Units: 512		
		Training	Validation	Test	Training	Validation	Test	Training	Validation	Test
Window size: 30	Max Error	242.508	127.867	84.995	189.468	100.770	65.482	116.082	82.751	62.856
	MAE	6.435	8.307	6.096	4.755	8.760	6.884	3.438	9.496	7.350
	MSE	199.789	138.699	124.386	101.317	141.772	142.319	46.633	175.133	127.749
	RMSE	14.135	11.777	11.153	10.066	11.907	11.930	6.829	13.234	11.303
	MeAE	2.890	6.548	3.555	2.324	6.817	3.897	1.764	6.891	4.736
	REV	-5.845	17.239	4.354	-3.464	20.678	9.418	-1.322	23.838	14.316
	NSE	0.772	0.677	0.703	0.884	0.670	0.660	0.947	0.592	0.695
	KGNP	0.882	0.741	0.883	0.925	0.704	0.849	0.959	0.664	0.822
Window size: 60	Max Error	203.354	148.287	103.775	140.766	132.144	85.256	86.151	119.624	83.141
	MAE	5.002	7.372	5.641	3.712	6.830	5.696	2.902	6.846	5.718
	MSE	143.300	140.544	113.625	67.662	114.716	91.093	36.626	104.280	91.558
	RMSE	11.971	11.855	10.660	8.226	10.711	9.544	6.052	10.212	9.569
	MeAE	2.084	5.379	3.399	1.648	5.211	3.639	1.296	5.592	3.544
	REV	-4.242	13.052	-0.333	-3.363	13.015	0.545	-1.939	12.221	0.634
	NSE	0.830	0.681	0.715	0.920	0.740	0.771	0.957	0.764	0.770
	KGNP	0.926	0.820	0.867	0.948	0.813	0.861	0.966	0.809	0.844
Window size: 120	Max Error	213.863	142.739	121.848	143.496	125.343	121.102	85.628	115.657	100.658
	MAE	4.870	6.342	6.477	3.703	5.644	6.775	2.787	6.159	6.050
	MSE	147.327	120.476	176.354	70.714	89.737	182.955	34.250	92.925	122.328
	RMSE	12.138	10.976	13.280	8.409	9.473	13.526	5.852	9.640	11.060
	MeAE	1.933	4.273	3.269	1.539	4.159	3.546	1.172	4.551	3.647
	REV	-5.058	8.829	-5.594	-4.594	10.120	-8.723	-1.959	11.496	0.566
	NSE	0.823	0.730	0.590	0.915	0.799	0.575	0.959	0.792	0.716
	KGNP	0.926	0.870	0.832	0.940	0.871	0.834	0.966	0.847	0.856

Table 20: Resulting performance metric scores calculated for Training, Validation and Test Period, respectively. Scores highlighted in red indicate best score per metric within the complete table

Lookback vs. LSTM-Units Daily Data: Input feature set 3		Units: 120			Units: 256			Units: 512		
		Training	Validation	Test	Training	Validation	Test	Training	Validation	Test
Window size: 30	Max Error	168.240	61.730	69.567	91.432	32.844	57.622	43.477	33.013	58.234
	MAE	3.168	3.543	5.580	2.350	3.015	5.151	1.977	3.145	5.105
	MSE	63.706	38.986	75.123	23.449	22.224	68.934	15.194	25.932	90.246
	RMSE	7.982	6.244	8.667	4.842	4.714	8.303	3.898	5.092	9.500
	MeAE	1.404	2.267	4.504	1.152	2.097	3.721	0.965	2.120	2.428
	REV	-2.601	-1.723	1.319	-0.917	-1.088	-3.595	-0.302	-2.046	-8.556
	NSE	0.927	0.909	0.820	0.973	0.948	0.835	0.983	0.940	0.784
	KGNP	0.962	0.932	0.803	0.978	0.950	0.823	0.984	0.935	0.805
Window size: 60	Max Error	162.369	57.916	54.260	96.652	36.952	51.109	45.466	43.054	47.967
	MAE	3.009	3.696	5.083	2.326	3.093	5.133	1.901	3.196	5.140
	MSE	53.422	43.036	61.435	23.598	24.211	65.598	13.982	27.544	81.606
	RMSE	7.309	6.560	7.838	4.858	4.920	8.099	3.739	5.248	9.034
	MeAE	1.357	2.331	3.763	1.104	2.186	3.331	0.891	2.185	2.881
	REV	-1.877	-1.414	-0.400	-0.993	-1.811	-6.614	-0.564	-3.007	-8.635
	NSE	0.937	0.902	0.846	0.972	0.945	0.835	0.983	0.938	0.795
	KGNP	0.967	0.927	0.819	0.978	0.946	0.813	0.984	0.934	0.803
Window size: 120	Max Error	166.486	61.248	54.738	91.642	39.292	52.210	45.245	43.977	45.368
	MAE	3.014	3.626	4.957	2.345	3.021	5.123	1.937	3.196	5.121
	MSE	54.836	41.507	60.401	23.331	24.295	67.037	14.056	29.915	82.567
	RMSE	7.405	6.443	7.772	4.830	4.929	8.188	3.749	5.469	9.087
	MeAE	1.319	2.348	3.499	1.114	2.104	2.881	0.928	2.019	2.658
	REV	-2.458	-1.430	-0.348	-1.272	-1.874	-9.289	-0.526	-3.036	-10.832
	NSE	0.934	0.907	0.860	0.972	0.946	0.844	0.983	0.933	0.808
	KGNP	0.963	0.937	0.820	0.976	0.952	0.803	0.983	0.932	0.797

Table 21: Resulting performance metric scores calculated for Training, Validation and Test Period, respectively. Scores are calculated by averaging over the respective lead time. 12 Steps, 24 Steps and 36 Steps refer to multi-step one shot LSTM models. The customized prediction loop was only applied to the Test period.

Average Scores multi-step predic-tions	12 Steps			24 Steps		
	Training	Validation	Test	Training	Validation	Test
Max Error	128.384	65.961	45.137	146.374	82.211	69.147
MAE	1.385	1.986	1.808	1.724	2.481	2.375
MSE	13.833	18.854	11.925	22.656	33.547	25.696
RMSE	3.710	4.293	3.409	4.703	5.675	4.884
MeAE	0.729	0.993	0.981	0.848	1.163	1.175
REV	-0.610	1.575	0.178	-1.076	1.610	0.225
NSE	0.985	0.958	0.971	0.975	0.925	0.938
KGNP	0.989	0.974	0.983	0.983	0.969	0.975
	LARSIM Simulation (24 Steps)			36 Steps		
	Training	Validation	Test	Training	Validation	Test
Max Error	103.620	80.570	44.117	160.839	92.888	87.292
MAE	2.137	1.800	1.879	2.318	3.057	3.038
MSE	28.395	18.951	21.067	47.177	61.896	51.792
RMSE	5.329	4.353	4.236	6.449	7.483	6.712
MeAE	0.786	0.773	0.934	0.991	1.329	1.353
REV	0.325	0.367	0.605	-1.991	0.529	-0.752
NSE	0.969	0.957	0.948	0.948	0.861	0.874
KGNP	0.984	0.983	0.977	0.970	0.961	0.962
	12 Steps (customized prediction loop)			24 Steps (customized prediction loop)		
	Training	Validation	Test	Training	Validation	Test
Max Error	not calculated	not calculated	27.454	not calculated	not calculated	38.262
MAE	not calculated	not calculated	1.216	not calculated	not calculated	1.214
MSE	not calculated	not calculated	5.318	not calculated	not calculated	5.300
RMSE	not calculated	not calculated	2.299	not calculated	not calculated	2.302
MeAE	not calculated	not calculated	0.704	not calculated	not calculated	0.701
REV	not calculated	not calculated	-0.027	not calculated	not calculated	-0.027
NSE	not calculated	not calculated	0.987	not calculated	not calculated	0.987
KGNP	not calculated	not calculated	0.989	not calculated	not calculated	0.990

A. Figures

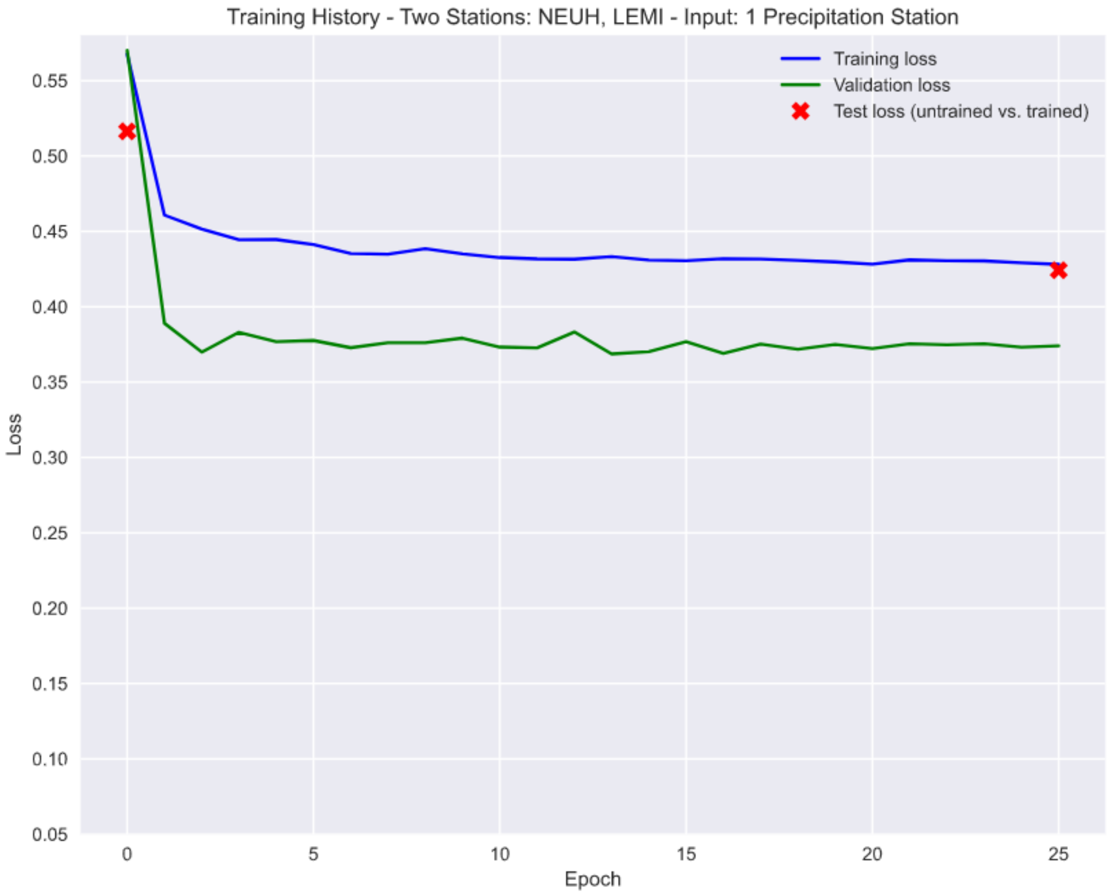


Figure 38: Training history of the trained LSTM model based on one precipitation station as input feature.

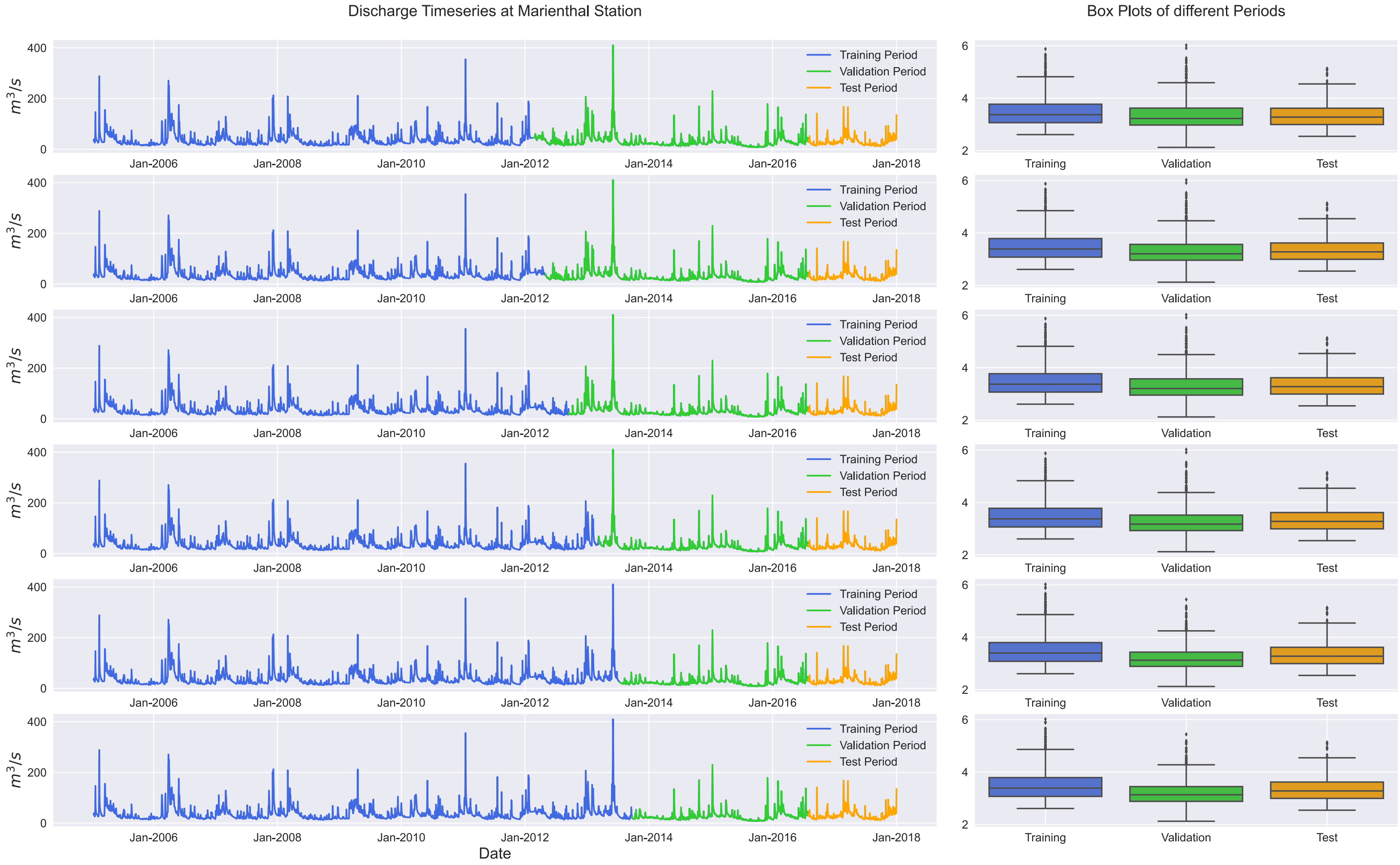


Figure 39: Rep-Holdout Method visualized on the discharge hydrograph at Marienthal station (left) and corresponding Boxplots (right) based on the underlying data distribution (log-transformed) of the three different subsets. Here, as example, 6 model iterations are shown, each having a slightly different split point between Training and Validation set.

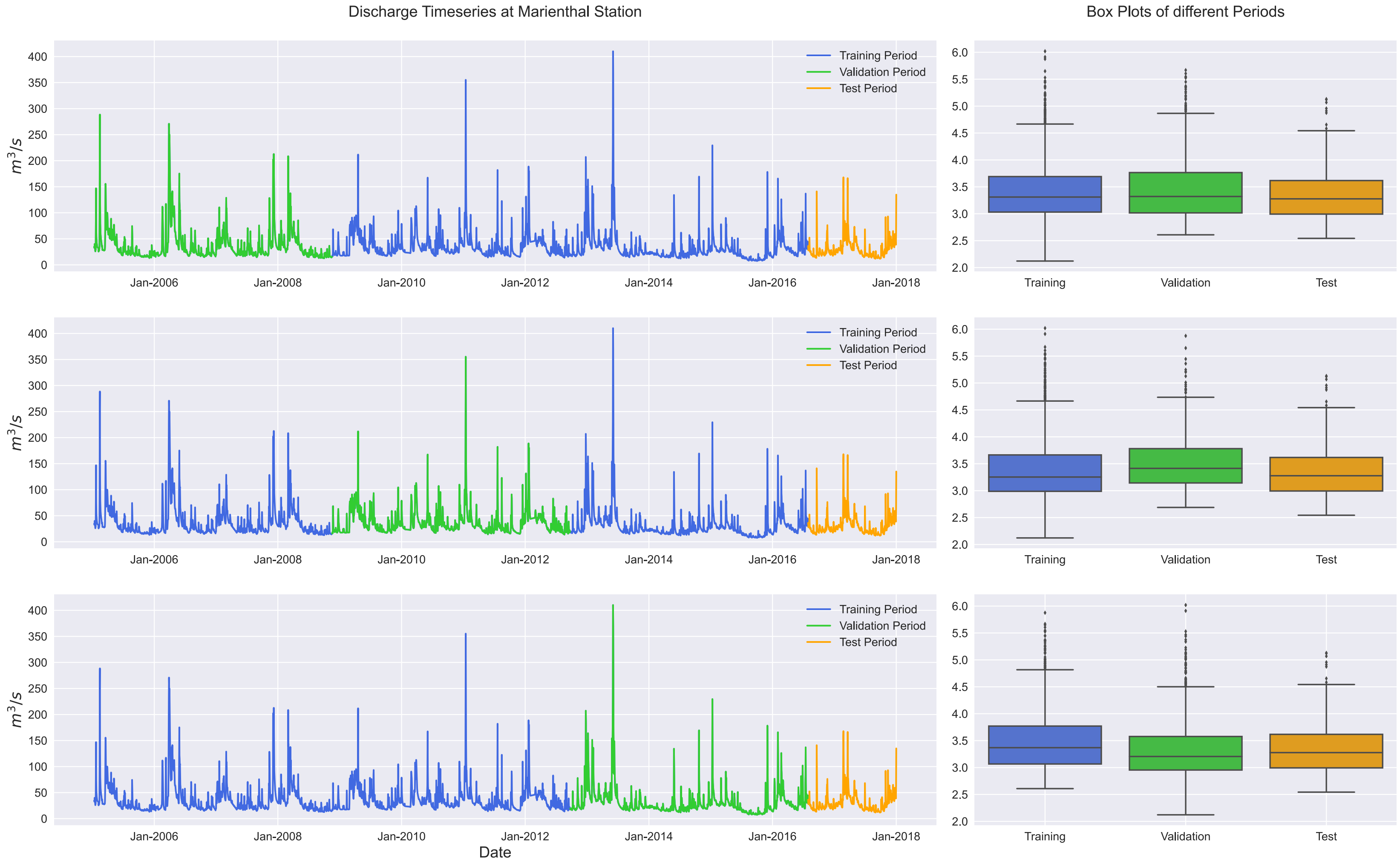


Figure 40: K-fold cross validation visualized on the discharge hydrograph at Marienthal station (left) and corresponding Boxplots (right) based on the underlying data distribution (log-transformed) of the three different subsets. Here, as example, 3 model iterations are shown, where each model is trained on different training subsections within the measurement series.

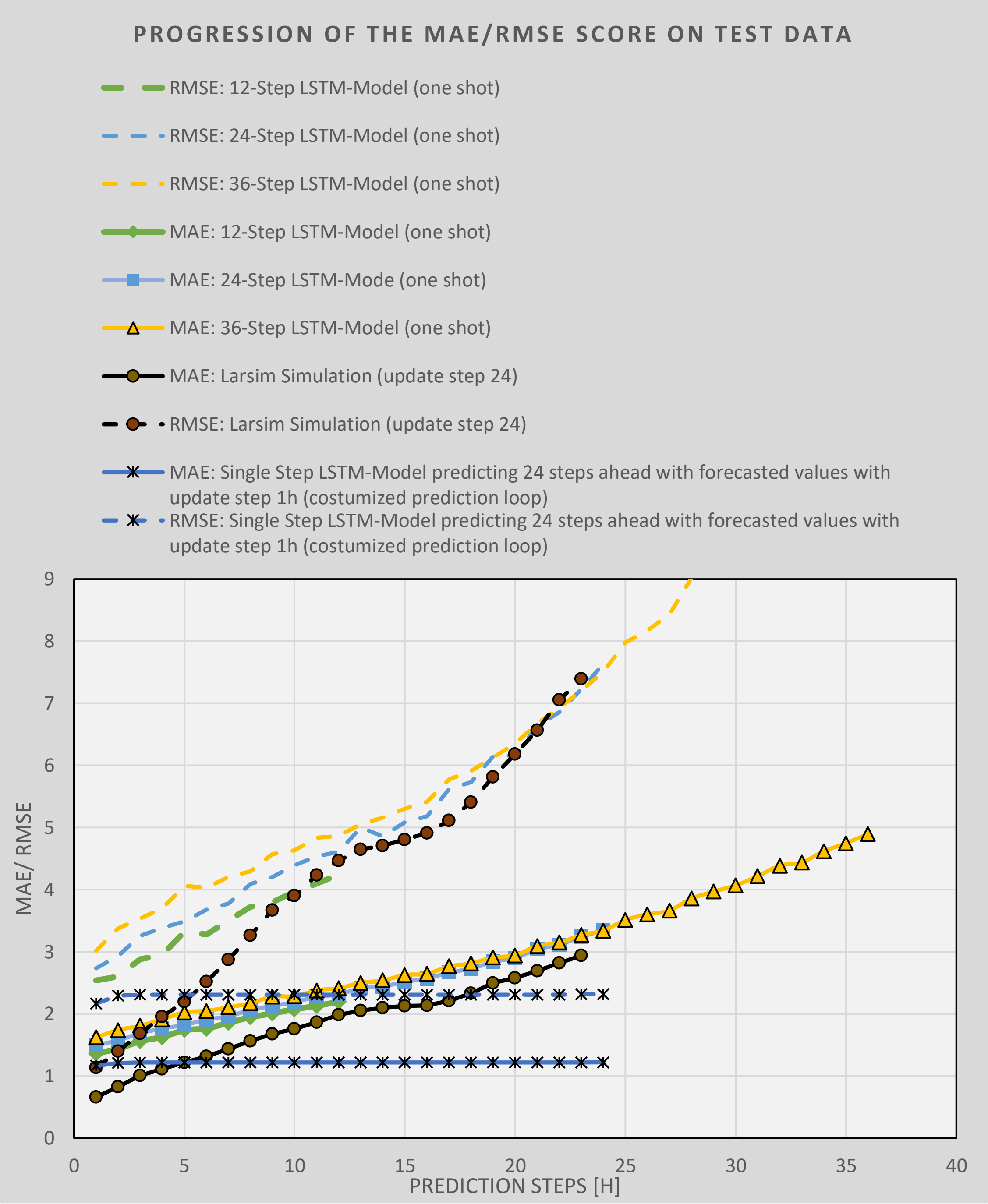


Figure 41: Progression of the MAE/ RSME score on Test Data over future time steps for all tested models (indicated by different line styles).

Publication bibliography

Caldera, H.P.G.M.; Piyathisse, V.R.P.C.; Nandalal, K.D.W. (2016): A Comparison of Methods of Estimating Missing Daily Rainfall Data. In *ENGINEER - Vol. XLIX* (04), pp. 1–8.

Campozano, L.; Sanchez, E.; Aviles, A.; Samaniego, E. (2014): Evaluation of infilling methods for time series of daily precipitation and temperature: The case of the Ecuadorian Andes. In *MASKANA* (Vol. 5, No. 1).

Cerqueira, Vitor; Torgo, Luis; Mozeti, Igor (2019): Evaluating time series forecasting models. An empirical study on performance estimation methods. Available online at https://arxiv.org/pdf/1905.11744.pdf.

Che, Zhengping; Purushotham, Sanjay; Cho, Kyunghyun; Sontag, David; Liu, Yan (2018): Recurrent Neural Networks for Multivariate Time Series with Missing Values. In *Scientific reports* 8 (1), p. 6085. DOI: 10.1038/s41598-018-24271-9.

Chollet, François (2018): Deep learning with Python. Shelter Island, NY: Manning (Safari Tech Books Online). Available online at http://proquest.safaribooksonline.com/9781617294433.

Doycheva, Kristina; Horn, Gordon; Koch, Christian; Schumann, Andreas; König, Markus (2017): Assessment and weighting of meteorological ensemble forecast members based on supervised machine learning with application to runoff simulations and flood warning. In *Advanced Engineering Informatics* 33, pp. 427–439. DOI: 10.1016/j.aei.2016.11.001.

Fotovatikhah, Farnaz; Herrera, Manuel; Shamshirband, Shahaboddin; Chau, Kwok-wing; Faizollahzadeh Ardabili, Sina; Piran, Md. Jalil (2018): Survey of computational intelligence as basis to big flood management: challenges, research directions and future work. In *Engineering Applications of Computational Fluid Mechanics* 12 (1), pp. 411–437. DOI: 10.1080/19942060.2018.1448896.

Freeze, R. Allan; Harlan, R. L. (1969): Blueprint for a physical-based digitally-simulated hydrologic response model. In *Journal of Hydrology* (9), pp. 237–258.

Gal, Yarin; Ghahramani, Zoubin (2015): Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. Available online at http://arxiv.org/pdf/1506.02142v6.

Gauch, Martin; Kratzert, Frederik; Klotz, Daniel; Nearing, Grey; Lin, Jimmy; Hochreiter, Sepp (2020): Rainfall-Runoff Prediction at Multiple Timescales with a Single Long Short-Term Memory Network. Available online at http://arxiv.org/pdf/2010.07921v1.

Géron, Aurélien (2019): Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. Concepts, Tools, and Techniues to Build Intelligent Systems. Second Edition: O'Reilly Media.

Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016): Deep Learning. Available online at www.deeplearningbook.org.

Gupta, Hoshin V.; Kling, Harald; Yilmaz, Koray K.; Martinez, Guillermo F. (2009): Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling. In *Journal of Hydrology* 377 (1-2), pp. 80–91. DOI: 10.1016/j.jhydrol.2009.08.003.

Hochreiter, Sepp; Schmidhuber, Jürgen (1997): Long Short-Term Memory.

Hu, Caihong; Wu, Qiang; Li, Hui; Jian, Shengqi; Li, Nan; Lou, Zhengzheng (2018): Deep Learning with a Long Short-Term Memory Networks Approach for Rainfall-Runoff Simulation. In *Water* 10 (11), p. 1543. DOI: 10.3390/w10111543.

Jabbari, Aida; Bae, Deg-Hyo (2018): Application of Artificial Neural Networks for Accuracy Enhancements of Real-Time Flood Forecasting in the Imjin Basin. In *Water* 10 (11). DOI: 10.3390/w10111626.

Jason Brownlee (2018): Deep Learning for Time Series Forecasting. - Predict the Future with MLPs, CNNs and LSTMs in Python. Edition: v1.4.

Karpatne, Anuj; Watkins, William; Read, Jordan; Kumar, Vipin (2018): Physics-guided Neural Networks (PGNN): An Application in Lake Temperature Modeling. Available online at http://arxiv.org/pdf/1710.11431v2.

Knoben, Wouter J. M.; Freer, Jim E.; Woods, Ross A. (2019): Technical note: Inherent benchmark or not? Comparing Nash–Sutcliffe and Kling–Gupta efficiency scores. In *Hydrol. Earth Syst. Sci.* 23 (10), pp. 4323–4331. DOI: 10.5194/hess-23-4323-2019.

Kratzert, Frederik; Herrnegger, Mathew; Klotz, Daniel; Hochreiter, Sepp; Klambauer, Günter (2019a): NeuralHydrology - Interpreting LSTMs in Hydrology. In *0302-9743* 11700 (7), pp. 347–362. DOI: 10.1007/978-3-030-28954-6_19.

Kratzert, Frederik; Klotz, Daniel; Brenner, Claire; Schulz, Karsten; Herrnegger, Mathew (2018): Rainfall–runoff modelling using Long Short-Term Memory (LSTM) networks. In *Hydrol. Earth Syst. Sci.* 22 (11), pp. 6005–6022. DOI: 10.5194/hess-22-6005-2018.

Kratzert, Frederik; Klotz, Daniel; Herrnegger, Mathew; Sampson, Alden K.; Hochreiter, Sepp; Nearing, Grey S. (2019b): Toward Improved Predictions in Ungauged Basins: Exploiting the Power of Machine Learning. In *Water Resour. Res.* 55 (12), pp. 11344–11354. DOI: 10.1029/2019WR026065.

Kratzert, Frederik; Klotz, Daniel; Hochreiter, Sepp; Nearing, Grey S. (2020): A note on leveraging synergy in multiple meteorological datasets with deep learning for rainfall-runoff modeling. In *Hydrol. Earth Syst. Sci.* DOI: 10.5194/hess-2020-221.

Kratzert, Frederik; Klotz, Daniel; Shalev, Guy; Klambauer, Günter; Hochreiter, Sepp; Nearing, Grey (2019c): Benchmarking a Catchment-Aware Long Short-Term Memory Network (LSTM) for Large-Scale Hydrological Modeling. In *Hydrol. Earth Syst. Sci.* DOI: 10.5194/hess-2019-368.

La Fuente, Alberto de; Meruane, Viviana; Meruane, Carolina (2019): Hydrological Early Warning System Based on a Deep Learning Runoff Model Coupled with a Meteorological Forecast. In *Water* 11 (9), p. 1808. DOI: 10.3390/w11091808.

Leandro, J.; Gander, A.; Beg, M.N.A.; Bhola, P.; Konnerth, I.; Willems, W. et al. (2019): Forecasting upper and lower uncertainty bands of river flood discharges with high predictive skill. In *Journal of Hydrology* 576, pp. 749–763. DOI: 10.1016/j.jhydrol.2019.06.052.

Leibundgut, Christian; Demuth, Siegfried; Lange, Jens (Eds.) (2006): The Water Balance Model Larsim. - Design, Content and Applications. Universität Freiburg: Institut für Hydrologie (Freiburger Schriten zur Hydrologie, Band 22).

Mosavi, Amir; Ozturk, Pinar; Chau, Kwok-wing (2018): Flood Prediction Using Machine Learning Models: Literature Review. In *Water* 10 (11), p. 1536. DOI: 10.3390/w10111536.

Ogunmolu, Olalekan; Gu, Xuejun; Jiang, Steve; Gans, Nicholas (2016): Nonlinear Systems Identification Using Deep Dynamic Neural Networks. Available online at http://arxiv.org/pdf/1610.01439v1.

Pool, Sandra; Vis, Marc; Seibert, Jan (2018): Evaluating model performance: towards a non-parametric variant of the Kling-Gupta efficiency. In *Hydrological Sciences Journal* 63 (13-14), pp. 1941–1953. DOI: 10.1080/02626667.2018.1552002.

Reichstein, Markus; Camps-Valls, Gustau; Stevens, Bjorn; Jung, Martin; Denzler, Joachim; Carvalhais, Nuno; Prabhat (2019): Deep learning and process understanding for data-driven Earth system science. In *Nature* 566 (7743), pp. 195–204. DOI: 10.1038/s41586-019-0912-1.

Seo, Youngmin; Kim, Sungwon; Singh, Vijay (2018): Machine Learning Models Coupled with Variational Mode Decomposition: A New Approach for Modeling Daily Rainfall-Runoff. In *Atmosphere* 9 (7), p. 251. DOI: 10.3390/atmos9070251.

Shen, Chaopeng (2018): A Transdisciplinary Review of Deep Learning Research and Its Relevance for Water Resources Scientists. In *Water Resour. Res.* 54 (11), pp. 8558–8593. DOI: 10.1029/2018WR022643.

Sundararajan, Mukund; Taly, Ankur; Yan, Qiqi (2017): Axiomatic Attribution for Deep Networks. Available online at http://arxiv.org/pdf/1703.01365v2.

Tian, Ye; Xu, Yue-Ping; Yang, Zongliang; Wang, Guoqing; Zhu, Qian (2018): Integration of a Parsimonious Hydrological Model with Recurrent Neural Networks for Improved Streamflow Forecasting. In *Water* 10 (11), p. 1655. DOI: 10.3390/w10111655.

Unnikrishnan, Vyshakh (2019): Implementation of a deep learning based model for rainfall-runoff modelling. Master's Thesis. Technische Universität München, München.

Viglione, Alberto; Merz, Ralf; Salinas, José Luis; Blöschl, Günter (2013): Flood frequency hydrology: 3. A Bayesian analysis. In *Water Resour. Res.* 49 (2), pp. 675–692. DOI: 10.1029/2011WR010782.

Wang, Wen (2006): Stochasticity, nonlinearity and forecasting of streamflow processes. Amsterdam, Fairfax VA: IOS Press.

Zhu, Lingxue; Laptev, Nikolay (2017): Deep and Confident Prediction for Time Series at Uber, pp. 103–110. DOI: 10.1109/ICDMW.2017.19.