1D River Flood Forecasts using Long Short-Term Seq2seq Neural Networks for the City of Kulmbach

MSc. Thesis

At the Department of Civil, Geo and Environmental Engineering of the Technical University of Munich, Chair of Hydrology and River Basin Management (Prof. M. Disse)

Supervised by Dr. phil. Jorge Eduardo Teixeira Leandro

Chair of Hydrology and River Basin Management

Technical University of Munich

Prof. Karsten Arnbjerg-Nielsen

Department of Environmental Engineering

Technical University of Denmark

Submitted by Giampaolo Crotti

03713504

Submitted on October 15th, 2019

Declaration of Authorship

I hereby certify that this MSc. Thesis, under the title “1D River Flood Forecasts using Long-Short Term Seq2seq Neural Network for the City of Kulmbach” has been composed by me and is based on my own work, unless stated otherwise.

The information and statements asserted in this MSc. 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.

Date:_____ **Signature:**_____

Giampaolo Crotti

Acknowledgements

Writing this Thesis was a long journey, and I would have never been able to complete it without the help and support I received.

Firstly, I would like to thank my supervisors, Dr. phil. Jorge Eduardo Teixeira Leandro for its guidance throughout the project and constant support and Prof. Karsten Arnbjerg-Nielsen for supervising me from DTU.

Special thanks go to my family, which has supported me during these years of studies, to all the friends I've met during this adventure, who inspired and helped me, and to Giulia, who no matter how far we were, she always was there for me.

Abstract

Data-driven models are flexible and powerful tools in hydrological forecasting as they don't need any data about the geo-morphology of the catchments. Due to the advancements of Machine Learning techniques and computational power, data-driven models have evolved as well, gaining the potential to model highly non-linear and complex relationships between inputs and outputs.

Artificial Neural Networks built using Long Short-Term Memory cells are the state of the art of data-driven models for rainfall-runoff and flood forecast, but they are limited to the prediction of a single output.

Originally built for natural language processing, Seq2seq Long Short-Term Memory neural networks are now considered as the most effective architecture for sequence processing in a wide range of fields, as they combine the advantage of Long Short-Term Memory cells and the ability to produce a causal sequence of multiple time-steps of forecasts.

In this work, one of the very first applications of Seq2seq Long Short-Term Memory neural networks for one-dimensional river flood forecasts is presented. The study area comprises the catchments upstream of the Bavarian city of Kulmbach, where major inundation events have occurred in the past decades.

The model can forecast 12 hours of river discharge on an hourly basis using data from rain, temperature, air humidity, wind speed, total radiation and previous discharge values. In order to optimize the model, a Genetic Algorithm is used for the tuning of the most important hyperparameters in order to automatise the process and reduce the computational effort. New features, derived from the original inputs, are created in order to increase the model's performance. By performing a sensitivity analysis of the trained model, the derived features are later proved to be useful for the performance and robustness of the model.

The Seq2seq model developed is able to produce robust forecasts of high discharge events for the different catchments, and it can compete in terms of performance with the more traditional conceptual Large Area Runoff Simulation Model (LARSIM) previously used in the area.

Table of Contents

Table of Contents 5

Table of Figures 7

List of abbreviations 10

1. Introduction 12

1.1 Context 12
1.2 Research questions 13
1.3 Objectives 13
2. Literature review 14

2.1 Hydrological and Hydrodynamic models: an overview 14
2.2 FloodEvac Tool and Conceptual Hydrological Bavarian Model LARSIM 15
2.3 River Flood and Runoff Forecasting using ANN 16
3. Study Area and Data 17

3.1 Study area 17
3.2 Data 18
4. Methodology 21

4.1 Tools: Machine Learning Frameworks 22
4.2 Artificial Neural Network Model 23
4.2.1 Artificial Neural Networks 23
4.2.2 Training with Backpropagation 25
4.2.3 Recurrent Neural Network and Long Short-Term Memory 26
4.2.4 LSTM Encoder-Decoder for Seq2seq 29
4.2.5 Avoid overfitting 30
4.2.6 ANN model structure 31
4.3 Data pre-processing 35
4.3.1 Correlation analysis and nan filling 36
4.3.2 Derived features 36
4.3.3 Structuring the data for the ANN 38
4.3.4 Data selection and Training, Validation and Test split 39
4.3.5 Input normalization 41
4.4 Uncertainty analysis and Evaluation Criteria 41
4.5 Hyperparameters Tuning using Genetic Algorithms (GA) 43

4.5.1 GA implementation 43
4.6 Features coefficients calibration 45
4.7 Features Sensitivity Analysis 46
4.8 Evaluation 48
5.Results 51
5.1 Data Pre-processing 51
5.2 Hyperparameters Tuning using Genetic Algorithms (GA) 54
5.3 Features coefficients calibration 56
5.4 Features Sensitivity Analysis 57
5.5 Evaluation 58
6. Discussion 64
6.1 Data Pre-processing 64
6.2 Hyperparameters Tuning using Genetic Algorithms (GA) 66
6.3 Features coefficients calibration 67
6.4 Features Sensitivity Analysis 67
6.4 Evaluation 68
6.5 Advantages and Limitations 69
7. Conclusion 71
8. Outlook 72
References 73

Table of Figures

Figure 1: Large Area Runoff Simulation Model LARSIM 15
Figure 2: Upper Main river basin with the two study areas catchment of Schorgast and Upper Weisser Main. 17
Figure 3:Gauging structure of the Upper Main. The position of the outlet gauges of Kauerndorf (Schorgast catchment) and Ködnitz (UWM catchment) are marked in red and blue respectively. The small sub-catchment of Schorgast ending in the outlet gauge of Untersteinach is marked in green. 18
Figure 4: Position of the Test sub-catchment of Unersteinach, situated in the catchment of Schorgast. The gauging stations used for the ANN models are marked. 20
Figure 5: Study catchments of Schorgast and Upper Weisser Main with the gauging stations used for the ANN models. 20
Figure 6: General scheme of the methodology and the purpose of the different catchments in the contest. 21
Figure 7: TensorFlow framework hierarchic structure 22
Figure 8: Schematic representation of a simple NN with 1 hidden layer and of a node. 24
Figure 9: RNN schematization as loop and unrolled chain. 27
Figure 10: Comparison between simple RNN cells and LSTM cells 28
Figure 11: LSTM Encoder-Decoder architecture for Seq2seq 29
Figure 12: Examples of underfitting, good fit, and overfitting of a model on the data 30
Figure 13: Schematic process of Dropout. On the left fully connected (dense) layers. On the right, the same network is showed where the red neurons represent the fraction of random connections cut off every iteration. 31
Figure 14: Schematic representation of the input/output of the model. The past measurements and future weather forecast are the inputs. The output is represented by the 12h sequence of predicted flow for the target gauge. 32
Figure 15: Encoder-Decoder LSTM structure with output sequence correction layers. 32
Figure 16: Example of the model implementation in Keras with 100 timesteps backwards and 33 features. The components of the encoder, encoded sequence, decoder and correction part of the NN are marked in red, green, blue, and yellow respectively. 33
Figure 17: Data pre-processing main operations 35
Figure 18: Input format required for a recurrent neural network in Keras-TensorFlow 38
Figure 19: Training, Validation, Test set. Splits of the full dataset and function of each. 40
Figure 20: Normalization process for NN inputs 41
Figure 21: Example of uncertainty band and related P-Factor. 42
Figure 22: Schematic process of the Genetic Algorithm developed for hyperparameters tuning. .. 45
Figure 23: Scheme for performing SA on a single feature of the input. The trained model performs different forecasts using each time a different pseudo-feature: a random-normal generated sequence having the same mean as the original feature, and std scaled of factors from 0 to 10. The forecasts are scored, and the std of the scores is divided by the std of the scaling factors of the pseudo-features in order to obtain the variation in std of the output per unit std of the input. .. 47
Figure 24: The two events of the test dataset. The first in December 2012, the second in May 2013. 48
Figure 25: Flow in the gauge of Ködnitz plotted with the average rainfall and snowfall and the temperature of the stations used in the model of UWM during the test event of December 2012. 49
Figure 26:Flow in the gauge of Ködnitz plotted with the average rainfall, snowfall and the temperature of the stations used in the model of UWM during the test event of May 2013. 49

Figure 27: Flow in the gauge of Ködnitz plotted with the average rainfall, snowfall and the temperature of the stations used in the model of UWM during the whole winter and spring 2012-2013. The two test events are marked in blue. 50
Figure 28: Snowpack and related water equivalent measured in Presseck (in Schorgast) during winter and spring 2012-2013. The time windows corresponding to the two test events of December 2012 and May 2013 are marked. 50
Figure 29: Synchronised hourly time series for the runoff gauge of Kauerndorf and the station of Poppenholz in Schorgast for the period 2005-2013. The missing values are marked with red lines. 52
Figure 30: R2 correlation coefficient for the full time-series of Schorgast from 2005 to 2013. 53
Figure 31:Boxplot of the flow of the target gauges of Untersteinach, Kauerndorf (for Schorgast), and Ködnitz (for U. Weisser Main). The outliers are marked in blue, the mean as a red x. 53
Figure 32: Timeseries of the average rainfall and the flow of the target gauges of the models for the three catchments of Untersteinach, Schorgast (Kauerndorf), Upper Weisser Main (Ködnitz). In blue is marked the training set, in yellow the validation, and in green the two test events. 54
Figure 33: Results of the GA hyperparameter tuning of the Ködnitz model. The score and the parameters selected for each individual foe each generation are plotted against each other in the scatters a.1-3 and b.1-3, while the convergence along the generations of the best and average score is represented in the graph c.1. 55
Figure 34: Scatter plot of the snow/rain upper and lower temperature parameters against the score (mean absolute error) obtained for the validation dataset running the trained models on the default parameters. The black dotted lines indicate the default values of upTlim and dwnTlim used in the training of the model. 56
Figure 35: Effect of the snow/rain parameters calibration on the P and R factors of the two test events of December 2012 and May 2013. The effect is measured by the change in % from the values before the calibration. The change in % of the R factor is the same for the two events as it depends only by the change in the width of the uncertainty bands. 57
Figure 36: Result of the SA on the single features of the model for the model of Kauerndorf and Untersteinach carried on the training and validation sets. The sensitivity of the parameters is expressed the standard deviation of the output scores per unit of a measure of the standard deviation of the input. 58
Figure 37: Result of the SA on the single features of the model for the model of Ködnitz carried on the training and validation sets. The sensitivity of the parameters is expressed the standard deviation of the output scores per unit of a measure of the standard deviation of the input. 58
Figure 38: Relative Error Method uncertainty calculation for Untersteinach. The P factor and R factor for the validation set and the uncertainty bands are shown for the values of 90,94 and 96% uncertainty. 59
Figure 39: Relative Error Method uncertainty calculation for Schorgast (Kauerndorf). The P factor and R factor for the validation set and the uncertainty bands are shown for the values of 90,94 and 96% uncertainty. 59
Figure 40: Relative Error Method uncertainty calculation for Upper Weisser Main (Ködnitz). The P factor and R factor for the validation set and the uncertainty bands are shown for the values of 90,94 and 96% uncertainty. 59
Figure 41: 12 h forecasts with uncertainty bands for the catchment of Untersteinach. a.1 shows the event of December 2012 with 90% uncertainty bands, while a.2 shows the event of May 2013 with 90% uncertainty bands. 60
Figure 42: 12 h forecasts with uncertainty bands for the catchment of Schorgast (gauge of Kauerndorf). b.1 shows the event of December 2012 with 90% uncertainty bands, while b.2 shows the event of May 2013 with 90% uncertainty bands 60
Figure 43: 12 h forecasts with uncertainty bands for the catchment of Upper Weisser Main (Ködnitz). c.1 shows the event of December 2012 with 90% uncertainty bands, while c.2 shows the event of May 2013 with 90% uncertainty bands. 60

Figure 44: Comparison of hourly P and R factor between the results in the three catchments with 90, 94, 96% uncertainty bands. The plots a.1-2 refer to the test event of December 2012, b.1-2 to the one of May 2013. 61
Figure 45: Comparison of hourly P and R factor between the results of the hydrological model LARSIM with different uncertainty estimation techniques (Gander 2018) and the LSTM model with 90, 94, 96% uncertainty bands. The plots a.1-2 refer to the test event of December 2012, b.1-2 to the one of May 2013. 62
Figure 46: Flow forecasts for the event of December 2012 at the gauge of Ködnitz of the LSTM model without uncertainty bands (a.1) and the LARSIM Ensemble runs (a.2). 64
Figure 47:Flow forecasts for the event of May 2013 at the gauge of Ködnitz of the LSTM model without uncertainty bands (b.1) and the LARSIM Ensemble runs (b.2). 64

List of abbreviations

AdaGrad: Adaptive Gradient

Adam: Adaptive Momentum Estimation

ANN: Artificial Neural Network

ANFIS: Adaptive Neuro-Fuzzy Inference System

API: Application Programming Interface

ARIMA: AutoRegressing Moving Average

BP: Backpropagation

CNN: Convolutional Neural Network

CPU: Central Processing Unit

DL: Deep Learning

ELU: Exponential Linear Unit

GPU: Graphic Processing Unit

GRU: Gated Recurrent Unit

GA: Genetic Algorithms

H: Relative Air Humidity

LARSIM: Large Area Runoff Simulation Model

LSTM: Long Short-Term Memory

MAE: Mean Absolute Error

ML: Machine Learning

MLP: Multi-Layer Perceptron

MSE: Mean Squared Error

NN: Neural Network

NP-hard: Non-deterministic Polynomial hard problem

ReLU: Rectified Linear Unit

RMSE: Root Mean Squared Error

RMSProp: Root Mean Square Propagation

RNN: Recurrent Neural Network

SA: sensitivity Analysis

Seq2seq: Sequence to Sequence

Std: standard deviation

SVM: Support Vector Machine

T: Temperature

TPU: Tensor Processing Unit

UWM: Upper Weisser Main (Catchment)

W: Wind Speed

Xglob: Total Radiation

1. Introduction

1.1 Context

River floods are widespread and dangerous environmental hazards which affect the lives of millions of people worldwide. Statistically, the damages caused by floods are characterised by a growing trend (Kundzewicz et al. 2010) that will probably increase even more in the future. Several projections suggest that runoff events with return periods over 100 years could become twice as frequent in the next three decades, causing devastating social and economic consequences on the exposed local communities (Alfieri et al. 2015).

Due to the magnitude and diffusion of the phenomena, the prevention and mitigation of casualties and economic losses caused by floods have been considered a matter of primary importance in the European Community. The existing EU Floods Directive incentivizes the use of tools such as hydrological models which are able to consider the characteristics of a river basin or sub-basin in the planning of flood risk management (Bhola, Leandro, and Disse 2018; UNISDR 2014). Due to this reason, for several years researchers and engineers have been extensively working on the development of cost-effective and rapid methods for producing long and short-term reliable, fast and affordable flood forecast tools (Bhatt et al. 2017).

Streamflow prediction has been traditionally performed using physical and conceptual-based models (Adnan 2017). Data-driven models are a particularly useful option in areas where collecting geophysical and topographical information about the catchment is difficult, as even advanced artificial intelligence models generally require more easily available and measurable data to satisfactorily model the complex phenomena in the catchment (Adamowski et al. 2012).

The most common examples of data-driven models for hydrology are the unit hydrograph ~~method~~, linear regression and Auto-regressing Moving Average (ARIMA), while the recent developments in computational intelligence have greatly expanded the capability of data-driven methods with tools such as Support Vector Machine (SVM), Artificial Neural Networks (ANN), Adaptive Neuro-Fuzzy Inference Systems (ANFIS), Genetic Algorithms (GA), Evolutionary Programming and Chaos Theory (Solomatine, See, and Abrahart 2008).

Even though ANNs have been widely used for river flow forecasting in the recent ~~last~~ years (Kratzert et al. 2018), their application for flood forecasting is quite young and in an early stage of development (Mosavi, Ozturk, and Chau 2018). The state of the art regarding neural networks can be found in fields such as image and object recognition and classification (Lee et al. 2017; Zhao et al. 2019), or natural language processing and speech recognition (Young et al. 2018). In those areas, architectures such as Convolutional Neural Network (CNN) or advanced Recurrent Neural

Networks (RNN) which include Long-Short Term Memory (LSTM), Gated Recurrent Units (GRU), and encoder-decoder for sequence-to-sequence (Seq2seq) forecast are commonly used, and the necessity for accuracy and efficiency has pushed the researchers towards a wide range of advanced optimization and fine-tuning solutions (Alom et al. 2019).

In this work, Seq2seq ANN models based on LSTM cells are developed for the catchments upstream of the Bavarian city of Kulmbach in order to forecast a sequence of 12 hourly timesteps of river runoff, with a main focus on high-flow events that may cause floods. The results obtained are compared to that of the Bavarian hydrological conceptual model LARSIM, already used in the area (Beg et al. 2018; Leandro et al. 2019).

Several works already exist on rainfall-runoff modelling using ANNs based on LSTM units (Kratzert et al. 2018; Hu et al. 2018; Le et al. 2019). In those works, one model produces a single forecast step and for different forecast intervals, different ANNs are needed.

The recently published paper of Xu, Luo, and Huang (2019) is the only example in the literature of applying a Seq2seq LSTM model for multi-timesteps hourly discharge forecasting. Still, the innovation of this thesis consists in the use of a fully structured Seq2seq LSTM neural network to model extreme hydrological events explicitly for flood forecast purposes.

1.2 Research questions

This work answers the following questions:

Is it possible to create a NN Seq2seq model which can successfully forecast an hourly sequence of 12 hours of runoff in catchments dominated by both long and short-term dynamics?

Will the quality of the results be comparable to the well-established conceptual hydrological Large Area Runoff Simulation Model (LARSIM)?

Will the ANN model behave similarly to the conceptual model, i.e. the two approaches reach a similar mathematical description of the physical phenomena in the catchments, or will they significantly differ?

1.3 Objectives

This work develops an ML-based model for 1D river flood forecasting in the area of Kulmbach, in northern Bavaria. Specifically, the goal is to produce a 12h forward runoff forecast for high-flow events using a Seq2seq LSTM NN, with quality comparable to the conceptual model LARSIM.

Thereby, the following objectives are pursued:

1. Literature research on ML for multivariate time series forecasting, with a focus on the different ANNs' architectures for sequence processing.

2. Gain knowledge of the programming environment of Python, the efficient open-source ML library TensorFlow, and its high-level API Keras.
3. Collect, analyse and select the data of the catchments of Schorgast and Upper Weisser Main (UWM), upstream of the city of Kulmbach.
4. Test different ANN models on a small test area (Untersteinach).
5. Apply the model for runoff forecasting on the two larger main catchments of Schorgast and UWM.
6. Evaluate the performances of the models on the different catchments and compare the output to the results obtained using the conceptual model LARSIM.

2. Literature review

2.1 Hydrological and Hydrodynamic models: an overview

Hydrological and Hydrodynamical models represent a simplified approximation of the complex natural system, where their inputs and outputs are measurable hydrologic variables connected by a set of equations that try to simulate the natural processes (Singh 2018). Three main typologies of such models exist: hydrodynamics, conceptual, and data-driven (Devia and Ganasri 2015).

Hydrodynamics models are built as a mathematical representation of the real phenomenon and include the principles of the physical processes, where measurable state variables functions of both time and space are used and finite difference equations are the core of the mathematical process (Devia and Ganasri 2015). The drawback of this kind of models is that is necessary to collect a large amount of data about the physical characteristics of the catchment. Examples of hydrodynamical models are MIKE-SHE (Prucha et al. 2016), Hec-RAS (USACE 2016), Storm Water Management Model (SWMM)(Us Epa 2012).

Conceptual models are built on a number of interconnected reservoirs which are filled or emptied by all of the component hydrological processes in a catchment (rainfall, infiltration, runoff etc…) on the basis of Semi-empirical equations (Devia and Ganasri 2015). Examples of conceptual models are the Bavarian Large Area Runoff Simulation Model (LARSIM) (Ludwig and Bremicker 2006) or HEC-HMS (USACE 2000).

Data-driven models, meaning they use only the relationships between the measured data to predict the output variable without any concern about the physical processes (Devia and Ganasri 2015). One of the most known examples is the unit hydrograph (UH) (Pechlivanidis et al. 2011) while more advanced approaches are linear and non-linear multivariate regression, Support Vector Machine (SVM), Artificial Neural Networks (ANN) and Adaptive Neuro-Fuzzy Inference Systems (ANFIS).

2.2 FloodEvac Tool and Conceptual Hydrological Bavarian Model LARSIM

In order to evaluate the quality of the predictions of a data-driven model, a well-developed conceptual hydrological model is the best comparison. The rainfall-runoff modelling interface FloodEvac tool, based on the conceptual hydrological model LARSIM, is the logical choice for this work, as it has been already successfully used for the study area of Upper Weisser Main (Leandro et al. 2019; Beg et al. 2018; Gander 2018).

The FloodEvac tool is built in order to couple in one interface different models such as rainfall uncertainty module, uncertainty and calibration module for the hydrological model, and hydraulic models. The tool was written in MATLAB to facilitate the replacement or adjustment of the different subroutines on which FloodEvac is based on and to offer an effective way to graphically represent the results through graphical function (Leandro et al. 2017).

The hydrological conceptual model used in the FloodEvac tool is LARSIM (Large Area Runoff Simulation Model), built for rainfall-runoff simulations in large catchments. The model is based on three storages, which are able to simulate the dynamics of direct runoff, interflow and baseflow. Those storages are the upper, middle and lower soil, as shown in the image below.

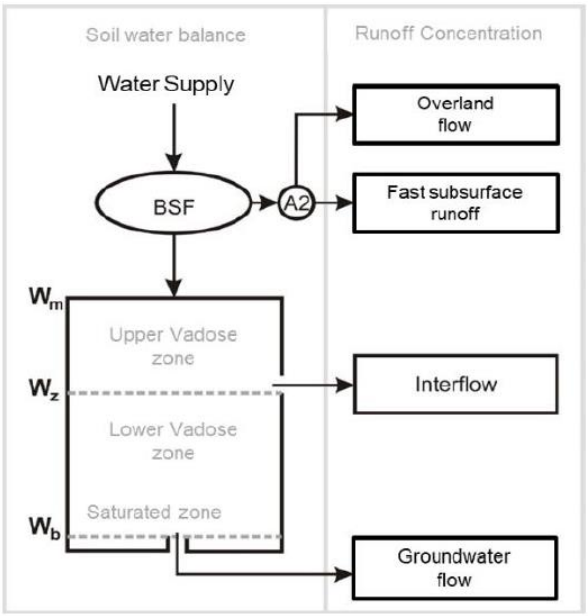

Figure 1: Large Area Runoff Simulation Model LARSIM

LARSIM can take into account different processes using different deterministic models that can simulate the concentration, translation and retention of the water, water storage in the soil, snow accumulation and melting, evapotranspiration, and interception.

The required data are rain, air temperature, air moisture, wind speed, total radiation, and air pressure. Overall the simulations are based on 34 parameters related to different processes such as groundwater flow, direct discharge, and interflow (Disse et al. 2018).

2.3 River Flood and Runoff Forecasting using ANN

Among the different ML techniques, ANNs are well known for the ability to describe highly nonlinear phenomena and are one of the more widespread ML techniques in Hydrology. While the early applications of the method for rainfall-runoff modelling date to the 90s, the use of different ANNs for flood forecasting was already widely researched in the early 2000s (Kim and Barros 2001; Campolo, Soldati, and Andreussi 2003).

While doing time series analysis, the drawback of simple ANN models such as shallow networks and Multi-Layer Perceptrons (MLP) is that the temporal order is totally lost in the structure. The solution to this problem was the application of Recurrent Neural Networks (RNN), able to feed the information of the previous timesteps. The first RNNs, in general, didn't always bring to significant improvements in the quality of the predictions, but in later applications, they managed to outperform the simple ANNs and MLPs (Kratzert et al. 2018).

In the following years several studies for runoff and flood prediction has been carried out with significant improvement for more complex ANN or hybrid models such as Wavelet NNs (Nourani et al. 2014; Adamowski et al. 2012), LSTM (Kratzert et al. 2018; Hu et al. 2018) and GRU NNs (Sit and Demir 2019).

Long Short-Term Memory (LSTM) neural networks are a typology of RNN in which the cells have the ability to store some information in the so-called cell state, a sort of hidden memory, and pass the state to the next timesteps. Therefore, LSTMs and their simplified version Gated Recurrent Units (GRU) are able to learn long term dependencies by mimic the storage effect of traditional hydrological models into the hidden state of the cell.

The limitation of the past applications of LSTM in hydrology, such as in Kratzer et al. (2018), Hu et al. (2018) and Le et al. (2019), is that the NNs used can only forecast a single interval. In order to forecast different discharge timesteps, in Hu et al. 2018 different single output NNs for different timesteps are used. This simple approach has the disadvantage that a NN needs to be trained for every timestep that needs to be forecasted. The approach is computationally intensive and time-consuming, and in addition to that, the information about the flow rate of the previously forecasted timestep is not used for the prediction of the following one.

In order to overcome those disadvantages, it's necessary to use a sequence to sequence (Seq2seq) algorithm. Presented in 2014 by Google as a tool for natural language processing (Sutskever,

Vinyals, and Le 2014), Seq2seq models are based on an Encoder-Decoder structure built on recurrent cells (usually LSTM or GRU). Even though nowadays Seq2seq are widely used in a large number of fields, the only application in hydrology published until now is the work of Xu, Luo, and Huang (2019) which focuses on river discharge forecast for hydropower plants management, while no example of Seq2seq application for extreme hydrological events modelling can be found yet.

3. Study Area and Data

3.1 Study area

The area object of this work is located in the 4244 km^2 Upper Main basin in Northern Bayern, where around 50 flow gauges are located. The study area is situated in the east of the basin and includes the two catchments of Schorgast and Upper Weisser Main (UWM), whose discharge directly cross the downstream city of Kulmbach. An overview of the area can be seen in Figure 2.

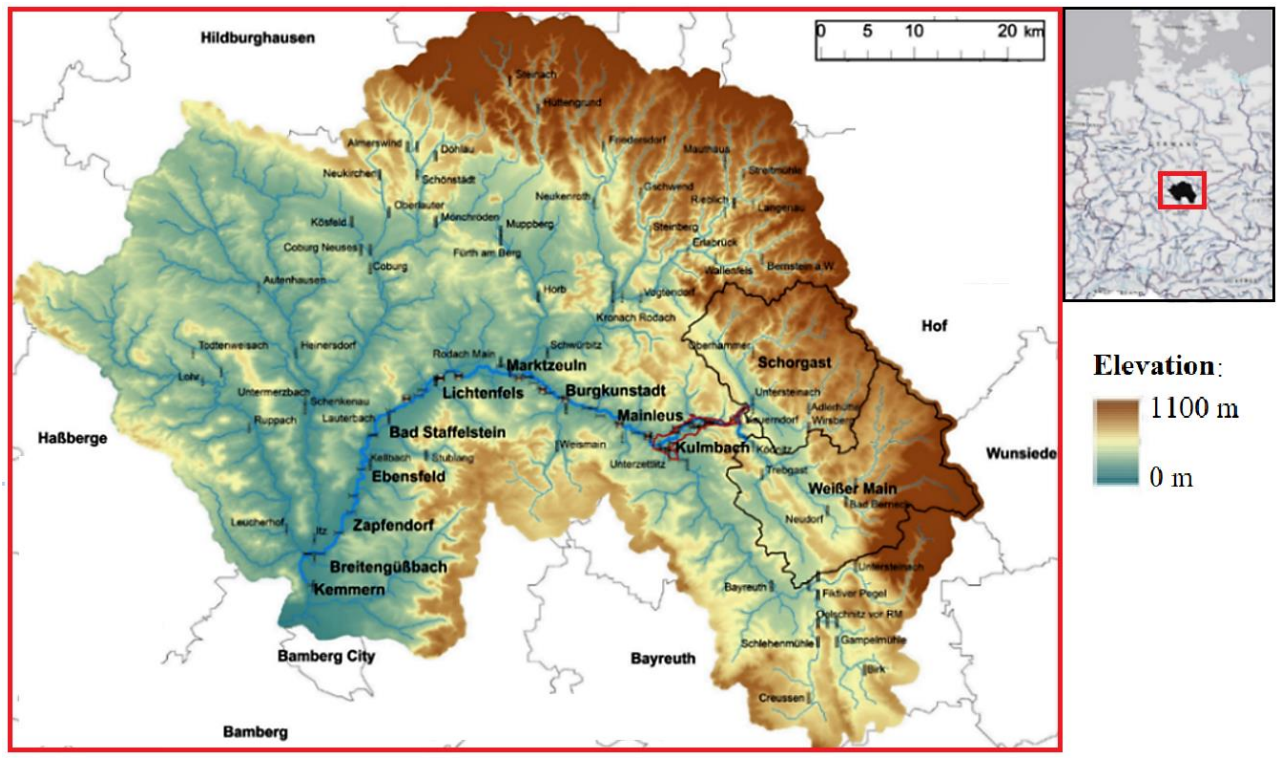

Figure 2: Upper Main river basin with the two study areas catchment of Schorgast and Upper Weisser Main.

The southern catchment Upper Weisser Main converges to the river gauge of Ködnitz, and the northern Schorgast to the gauge of Kauerndorf. In addition to that, a small sub-catchment of Schorgast is used in the preliminary phase to build and test beta model structures that will be later adapted on the two full catchments. The complete network of gauges in the basin is shown in Figure 3, where Kulmbach and the study areas are marked with different colours.

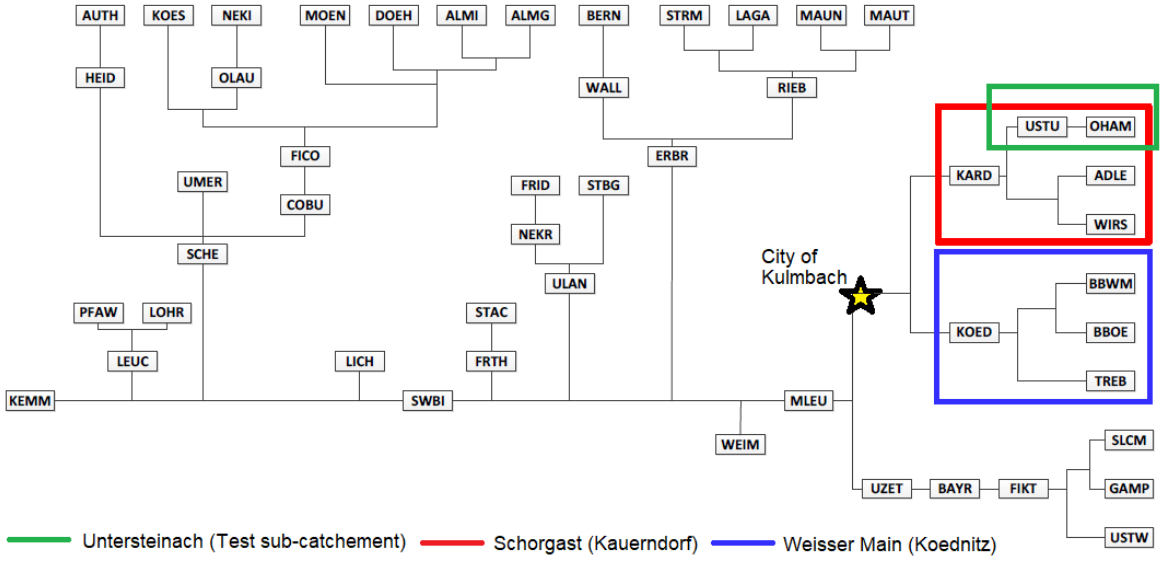

Figure 3:Gauging structure of the Upper Main. The position of the outlet gauges of Kauerndorf (Schorgast catchment) and Ködnitz (UWM catchment) are marked in red and blue respectively. The small sub-catchment of Schorgast ending in the outlet gauge of Untersteinach is marked in green.

The study area is subjected to a central European climate. That means the temperature in Summer can produce relevant evaporation, while in Winter and Spring long-term snow accumulation and melting are to be expected.

The catchments are located between 350 and 600 masl. The hilly topography can have very important influences on the uniformity of parameters such as temperature, wind, rainfall, total radiation etc. Those parameters are in fact very locally dependent on the different exposure and elevation of the ground.

The quite spread agriculture and the presence of small towns in the catchments means that an anthropic influence on the streamflow may be important due to drinking water or irrigation uptakes and cyclic releases from small Waste Water Treatment Plants.

3.2 Data

The data used in this work, available from the Gewässerkundlicher Dienst Bayern (Bavarian Environmental Agency), are the following:

Table 1: Typology of available data from the measurement gauges used in this work with measurements' frequency.

Data:	Flow	Rain	Temperature	Air Humidity	Wind	Global Radiation
Unit:	m^3/s	mm	°C	%	m/s	Wh/m2
Frequency:	15'	15' – 1h	1 h	1 h	1 h	1 h

For the test sub-catchment of Untersteinach, shown in Figure 4, and for the two catchments of Schorgast and UWM, shown in Figure 5, the gauging stations used are shown in Table 2.

Table 2: Gauging stations per each parameter used for the ANN models for each catchment/sub-catchment. The gauging stations can be inside or outside the catchment area. The gauging stations external to the catchment area are marked in bold.

Catchment:	**Untersteinach**	**Schorgast**	**Upper Weisser Main**
Target Runoff Gauge:	Untersteinach	Kauerndorf	Ködnitz
Other Runoff Gauges:	/	1. Untersteinach 2. Oberhammer 3. Wirsberg 4. Adlerhütte	1. Trebgast 2. Bad Berneck_1 3. Bad Berneck_2
Rain:	1. Poppenholz 2. Presseck	1. Poppenholz 2. Presseck 3. Ludwigschorgast **4. Stammbach-Querenb.** **5. Helmbrechts**	**1. Poppenholz** **2. Heinersreuth-Vollhof** **3. Ludwigschorgast** 4. Stammbach-Querenbach **5. Mistelbach** **6. Thurnau-Tannfeld** **7. Markersreuth** **8. Fichtelberg/Oberfr.-Hütt.** **9. Waldstein-Uni Bayreuth** **10. Würnsreuth**
Temperature:	Poppenholz	Poppenholz	**1. Poppenholz** **2. Mistelbach** **3. Markersreuth** **4. Würnsreuth**
Air Humidity:	Poppenholz	Poppenholz	**1. Poppenholz** **2. Mistelbach** **3. Markersreuth**
Wind:	Poppenholz	Poppenholz	**1. Poppenholz** **2. Mistelbach** **3. Markersreuth**
Xglob:	Poppenholz	Poppenholz	**1. Poppenholz** **2. Markersreuth**

Note: If the name of a gauging station is marked in **bold**, it indicates that it is outside the area of the catchment.

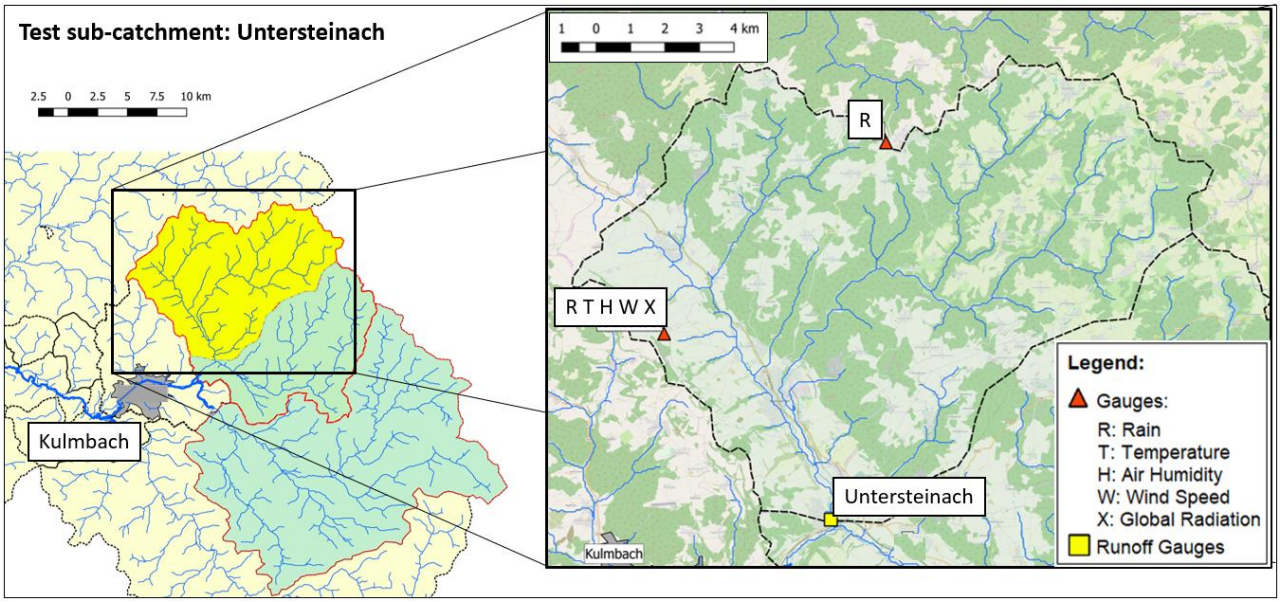

Figure 4: Position of the Test sub-catchment of Unersteinach, situated in the catchment of Schorgast. The gauging stations used for the ANN models are marked.

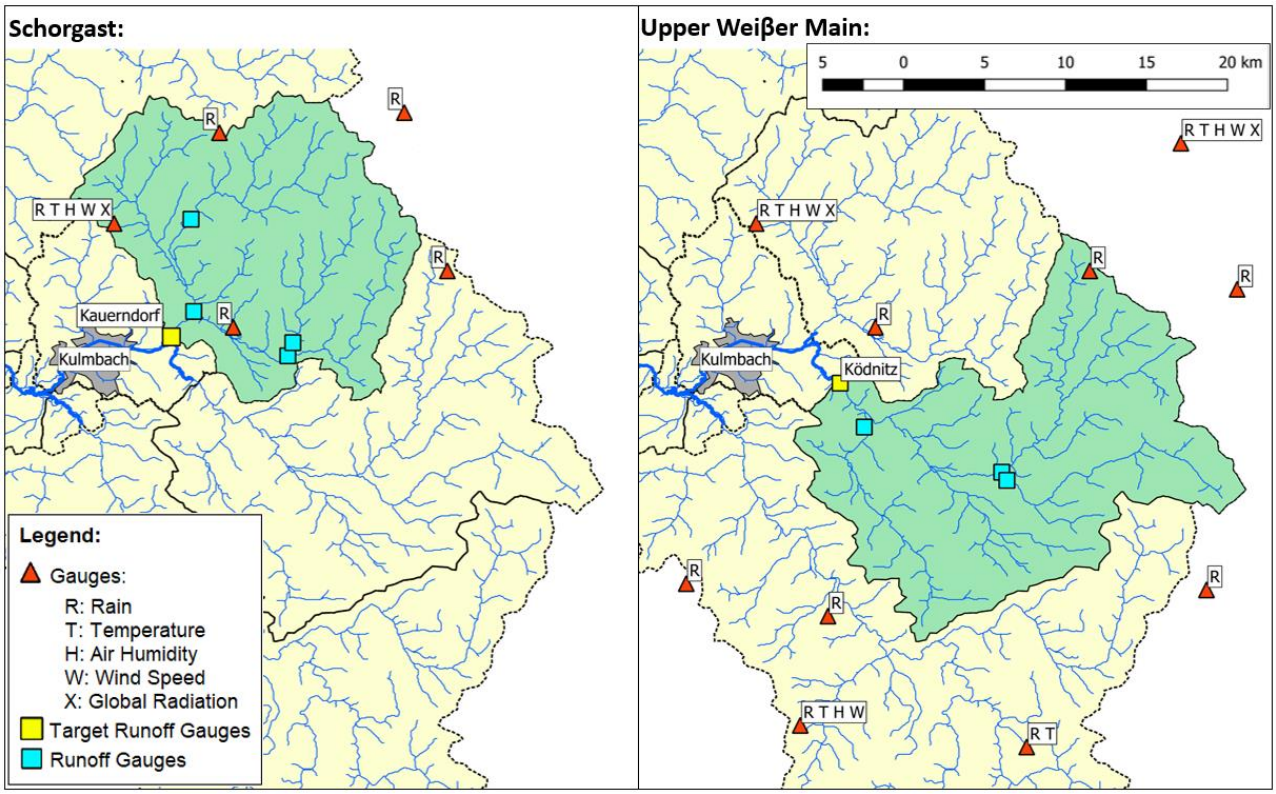

Figure 5: Study catchments of Schorgast and Upper Weisser Main with the gauging stations used for the ANN models.

It's Important to notice how the catchment of Schorgast has the most of the stations inside or very close to the area, while for UWM only one rain gauge is inside the border.

4. Methodology

The process of developing and testing a ML model, as well as any other model, is based on six main points: tools, model structure, data pre-processing, optimization, uncertainty, evaluation.

The tool (platform) used for implementing a ML model doesn't influence in theory the quality of the model, but it's important for the computational speed and the memory requirements (Section 4.1).

Several different NN architectures are well known for a wide range of applications. The right structure applied to the right problem increases the performance of the model (Section 4.2). The basic data pre-processing is based on a series of well-established rules of thumb, that usually ensure a discrete performance (Section 4.3). Regarding the uncertainty analysis and the evaluation well-structured and tested methodologies for hydrology can be applied (Section 4.4 and 4.8).

The optimization of the high-level structural parameters of the model, called hyperparameters tuning, it's a complex non-linear problem that can be managed with the help of some methods such as Genetics Algorithms (Section 4.5). Advanced pre-processing of the input data can evolve in an extremely complicated task also, but simplified methods based on Sensitivity Analysis (SA) of the trained model can bring good results with a fraction of the computational effort (Section 4.6 and 4.7). Figure 6 offers a summary of the methodology and the use of each study area in this Thesis.

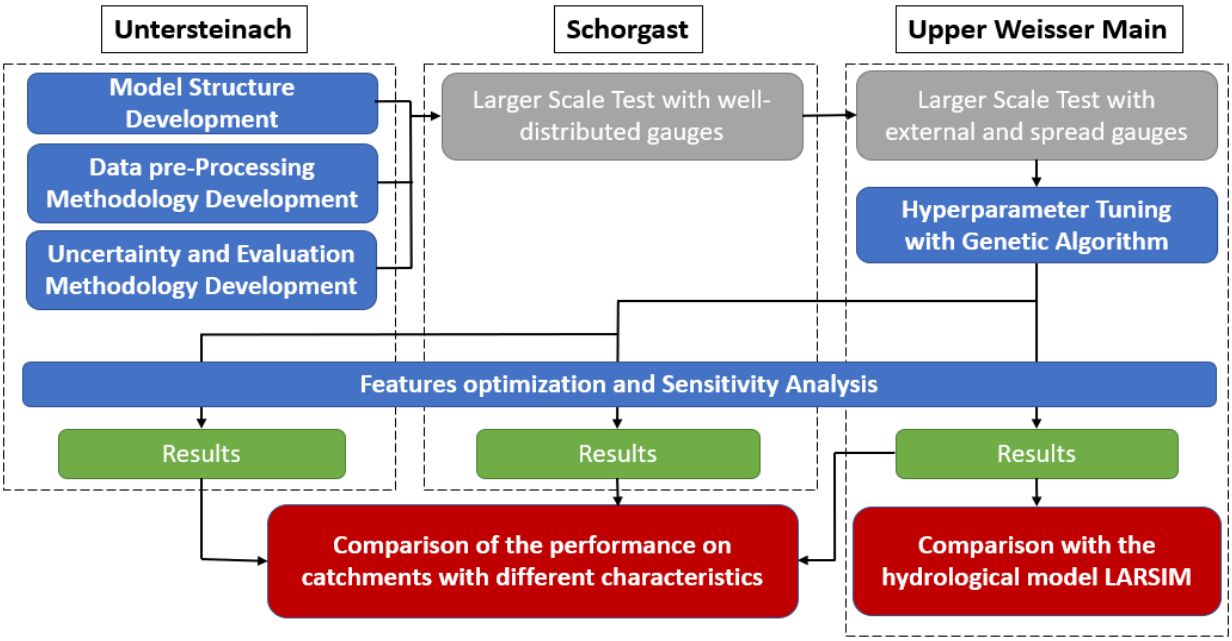

Figure 6: General scheme of the methodology and the purpose of the different catchments in the contest.

The small test sub-catchment of Untersteinach was used to experiment more rapidly the different options for the model structure and the data pre-processing, as well as selecting and consolidating the basic methodologies for uncertainty analysis and evaluation.

The catchment of Schorgast represents the first test on a larger scale. The biggest number of stations used and their favourable locations ensure that the results are not too influenced by the lack of data and offer a good way to evaluate the solutions developed for Untersteinach.

The catchment of Upper Weisser Main (UWM) is the most important, as well-calibrated results of LARSIM exists for the catchment and a direct comparison of the results is possible. In addition to that, almost no meteorological station is inside or very close to the area, so the catchment is a good way to test the ANN approach in sub-optimal conditions.

4.1 Tools: Machine Learning Frameworks

In order to implement and train ANNs models, several toolboxes and libraries in different programming languages are currently available. The most well-known open-source libraries are Caffe (Jia et al. 2014), TensorFlow (Abadi et al. 2016), PyTorch (Paszke et al. 2017), Microsoft Cognitive Toolkit CNTK (Seide and Agarwal 2016), Theano (Al-Rfou et al. 2016), and SciKit-Learn (Pedregosa et al. 2011) which are commonly used in Python but are sometimes compatible with other programming languages such as Java, C++ and R. Among the commercial software MATLAB possess user friendly toolboxes for ML and DL (Moulder et al. 2017).

Developed by Google, TensorFlow it's one of the most used and better-maintained ML libraries worldwide (Unruh 2017). It's efficient and optimised for parallel computation, it can run on both CPU and GPU, and it can interface with both R and Python even though the latter is the most commonly used and stable release.

Another great advantage of TensorFlow is that it can be used indirectly to backend Keras (Chollet 2015), a widely used ML high-level Application Programming Interface (API). That means a fast and user-friendly implementation of the main elements of the model is possible using the default elements in Keras, while any kind of customization can be coded in TensorFlow and then wrapped.

TensorFlow itself is a C++ framework hierarchically structured as can be observed in Figure 7 below.

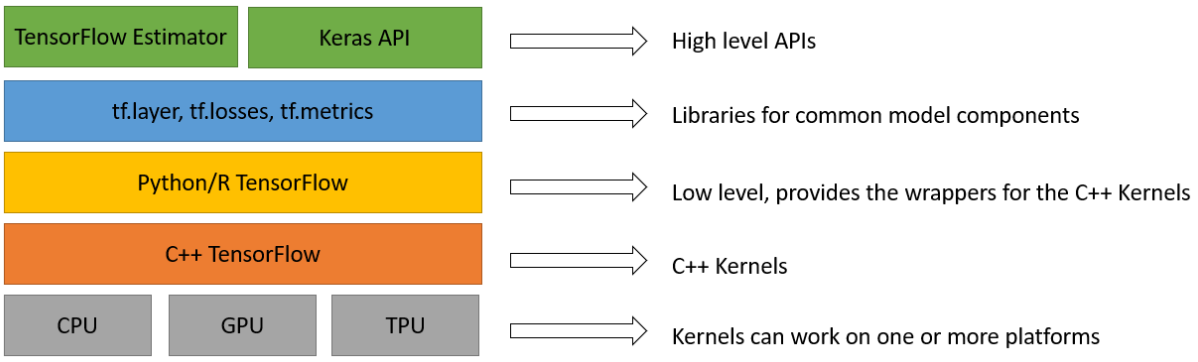

Figure 7: TensorFlow framework hierarchic structure.

Different high-level APIs can be used for high-level programming, while at lower level the base TensorFlow language can be used directly on Python or R. The Python/R implementation is used to wrap the computational kernels (core programs) which are written in C++ and can be run on one or different units at the same time. TensorFlow supports different computational units such as Central Processing Unit (CPU), Graphic Processing Unit (GPU) or even a dedicated Tensor Processing Unit (TPU). This last, designed by Google in order to specifically optimise ML applications on TensorFlow, can be accessed via Cloud (Jouppi et al. 2017).

In this work, TensorFlow version 1.13.1 for GPU is used in combination with the high-level API Keras version 2.2.4 for GPU and implemented in Python 3.6.8 Anaconda distribution (Anaconda Software Distribution 2019). The interface used is the flexible web-based platform Jupyter Notebook (Kluyver et al. 2016). The memory utilization and computational power required for training the models are sufficiently small to be managed by a single PC without having to move to the cloud. The machine utilised is equipped with a CPU Intel® Core™ i7-6700HQ 2.60 GHz and a GPU unit NVIDIA GeForce GTX 960M compatible with CUDA®Toolkit for running TensorFlow on GPU.

In the pre and post-processing phase, the mathematical handling of the data is carried out with the Python libraries Pandas (Mckinney 2010) and Numpy (Van Der Walt, Colbert, and Varoquaux 2011). Plots and graphics are done using Matplotlib (Hunter 2007) and Seaborn (Waskom et al. 2014). Regressions and normalizations of the data are carried out using the Python ML library SciKit-Learn (Pedregosa et al. 2011), simpler and more compact than Keras/TensorFlow for basic regression models and equipped with useful data pre-processing functions.

4.2 Artificial Neural Network Model

4.2.1 Artificial Neural Networks

ANNs are supervised ML algorithms, meaning they need a certain number of labelled examples in order to “learn” how to perform a task. In other words, ANNs need some input-output samples, and “learning” means calibrating the internal parameters in order to minimise the error between the target output value and the actual output of the NN.

The basic element of a NN can be equally called node, neuron or perceptron. Every node performs a simple operation: it receives a certain number of inputs and makes a weighted sum. The result is then transformed by an activation function which adds non-linearity and gives back a single node output as shown in [1].

$$Y = f(w_0 + \sum_1^n w_n . x_n) \qquad [1]$$

Y is the output value of the neuron, x_n are the inputs, the parameters w_n are called weights, while the parameter w_0 is called bias. The most common activation functions for regression problems are

linear [2], logistic (sigmoid) [3], hyperbolic tangent (Tanh) [4] and Rectified Linear Unit (ReLU) [5] with its variations Leaky ReLU [6] and Exponential Linear Unit ELU [7].

$$g(x) = x \quad \text{[2] Linear} \qquad g(x) = \frac{1}{1+e^{-x}} \quad \text{[3] Logistic}$$

$$g(x) = \tanh(x) \quad \text{[4] Tanh} \qquad g(x) = \begin{cases} x & if\ x \geq 0 \\ 0 & if\ x < 0 \end{cases} \quad \text{[5] ReLU}$$

$$g(x) = \begin{cases} x & if\ x \geq 0 \\ 0.1x & if\ x < 0 \end{cases} \quad \text{[6] Leaky ReLU} \qquad g(x) = \begin{cases} x & if\ x \geq 0 \\ \alpha(e^x - 1) & if\ x < 0 \end{cases} \quad \text{[7] ELU}$$

While the classic activation functions for NN are logistic and Tanh, nowadays ReLU and its variations ELU and Leaky ReLU are the most used in deep networks where they prove to be computationally more efficient and performant.

When different nodes work in parallel, they form a layer. The simplest ANN structure is composed of a first input layer which elaborates the input data directly, a central hidden layer, and a final output layer that gives back the final output of the network. When every node of a layer is connected to all nodes of the following layer the layers are called Dense Layers. An example of simple ANN with 3 inputs, 1 output, 1 hidden layer and 3 neurons per layer with a zoom on the neuron structure is shown in Figure 8 below.

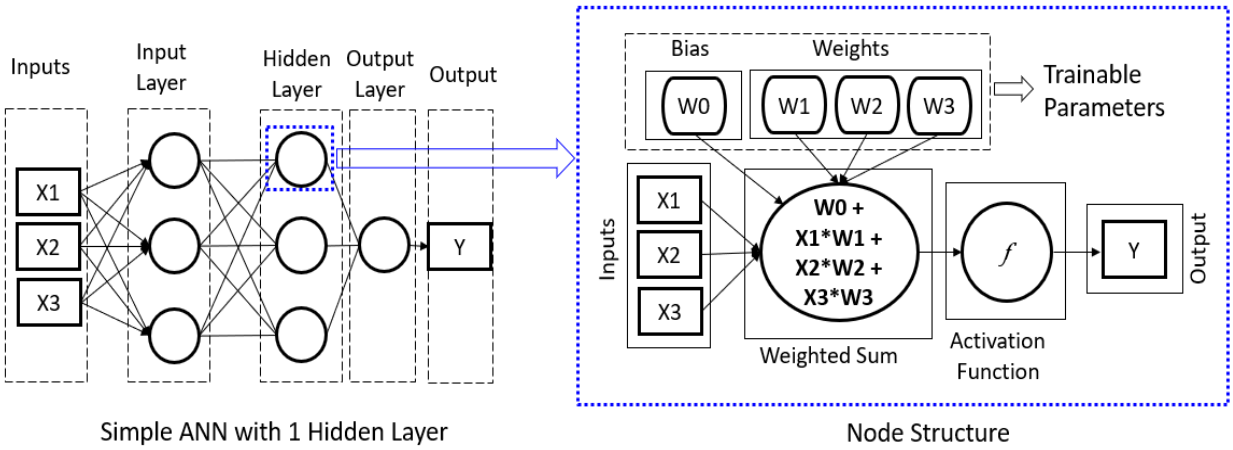

Figure 8: Schematic representation of a simple NN with 1 hidden layer and of a node.

The numbers of nodes for each layer, the activation function, the number of layers and in general every characteristic of the network that can be defined by the user are called hyperparameters. There is no fixed rule for choosing the best value of each hyperparameter, and the tuning can require quite an effort in terms of computation and time (Sections 4.2.6 and 4.5.1).

4.2.2 Training with Backpropagation

The calibration of the weights and biases of the NN is called training. In order to train the model, a method called backpropagation is used. To make the differences between the network outputs and target values as small as possible, the parameters from the output layer to the input are modified using a chain rule. In order to perform backpropagation, it's necessary to establish a loss function for calculating the error. Once fixed the loss function, the partial derivative of the function on each weight is approximated and computed as [8].

$$\frac{\partial L}{\partial w_{ij}} = \frac{\partial L}{\partial O_i} \cdot \frac{\partial y_i}{\partial \sum w_i x_i} \cdot \frac{\partial \sum w_i x_i}{\partial w_{ij}} \quad [8]$$

Where L is the loss function, w_{ij} is the weight from the node i to the node j, y is the output of the model, O the observed data, and $\sum w_i x_i$ represents the weighted sum of the inputs of the node.

The most common loss functions for regression problems are the Mean Absolute Error (MAE) [9], the Mean Squared Error (MSE) [10] and the Root Mean Squared Error (RMSE) [11].

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |Y_i - \hat{Y}_i| \quad [9] \qquad MSE = \frac{1}{N}\sum_{i=1}^{N} (Y_i - \hat{Y}_i)^2 \quad [10] \qquad RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (Y_i - \hat{Y}_i)^2} \quad [11]$$

Where N is the number of samples, Y_i the outputs of the model and $\hat{Y}_i$ the observed values.

When we can calculate the derivative of the weights, the minimization of the loss function can be achieved by an iterative algorithm. The simplest algorithm for backpropagation is called standard gradient descent [12].

$$w_{ij}^{t+1} = w_{ij}^{t} - \lambda \frac{\partial L^t}{\partial w_{ij}^t} \quad [12]$$

In standard gradient descent, the length of the step for each iteration is regulated by the learning rate λ, which is extremely important hyperparameter for the success of the algorithm: if too small the backpropagation would be very slow if too big the algorithm could not converge.

As backpropagation in complex and deep ANN models is usually computationally and memory intensive, different advanced optimization algorithm has been developed in order to adapt dynamically the learning rate during the process, save memory, and in general speed up the computation.

One of the most efficient advanced optimization algorithms is the Adaptive Momentum Estimation (Adam), introduced by Kingma and Ba (2014). Adam stochastic method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the

gradients. By doing so it tries to combine the advantages of two other popular advanced optimisation methods: Adaptive Gradient (AdaGrad), efficient with sparse gradients, and Root Mean Square Propagation (RMSProp), optimal for non-stationary and on-line systems (Kingma and Ba 2014). Adam, for each parameter w_{ij} perform the calculations shown in [13-16]

$$\nu_t = \beta_1 \nu_{t-1} - (1 - \beta_1) * g_t \quad [13]$$

$$s_t = \beta_2 s_{t-1} - (1 - \beta_2) * {g_t}^2 \quad [14]$$

$$\Delta w_t = -\eta \frac{\nu_t}{\sqrt{s_t + \epsilon}} * g_t \quad [15]$$

$$w_{t+1} = w_t + \Delta w_t \quad [16]$$

Where η is the initial learning rate, g_t is the gradient along each parameter w, ν_t is the exponential average of the gradients along the parameter w, s_t is the exponential average of the squares of gradients along the parameter w, and $\beta_1, \beta_1, \epsilon$ are hyperparameters with default values of 0.9, 0.999, and 10^{-8} respectively.

Regardless of the optimization algorithm, the training usually is carried by splitting the dataset into batches, segments of a certain number of randomly selected training samples. The optimization algorithm works on the single batches and then averages the weights obtained for each batch. Every complete iteration on every batch of the training set is called epoch. The batch size is an important hyperparameter, as small batch size helps a better generalization of the results and brings usually to convergence in a smaller number of epochs, but at the same time makes each epoch slower, while a big batch size-speed up the training, but in some situation could cause the model to not converge at all.

4.2.3 Recurrent Neural Network and Long Short-Term Memory

A good option to deal with time-organised data when the output of one timestep is useful for the prediction of the following ones are the Recurrent Neural Network (RNN). An RNN is basically a NN closed in a loop that feeds the obtained output again to the network for a pre-determined number of times. The RNN can also be “unrolled”, starting in an initial timestep t0 and developing like a chain as shown in Figure 9.

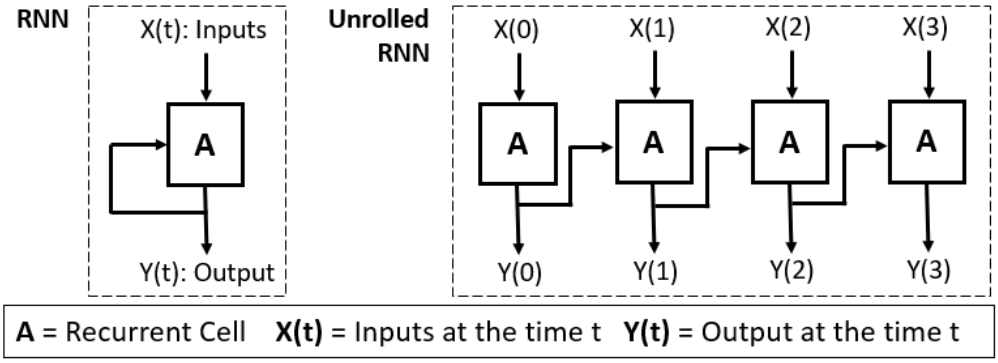

Figure 9: RNN schematization as loop and unrolled chain.

In a simple RNN cell, the output y is computed for every timestep t as [17].

$$Y(t) = g(Wx(t) + Uy(t-1) + b) \quad [17]$$

Where $g()$ is the activation function, W and U are the matrices of the weights for the input x and the previous timestep output y, and b is the bias matrix.

In RNNs the backpropagation works also through time and the gradients are backpropagated till the earliest layer in the sequence. Due to the continuous matrix multiplications, the gradients could either shrink exponentially to very small values or grow exponentially to enormous numbers. The first case is called vanishing gradient problem, which causes serious difficulties in training the network successfully and prevent the RNN in learning long-term dependencies. The second case it's called exploding gradient problem, and it could even bring the model to crash during the training.

Different solutions help to prevent vanishing and exploding gradient such as the use of ReLU or similar activation functions instead of logistic and Tanh, but the most effective solution in RNN architectures is the use of Long Short-Term Memory (LSTM) cells presented in the first version by Hochreiter and Schmidhuber (1997).

Each cell possesses a state, that works as a memory unit. The state and the output are adjusted and regulated by three gates called forget gate, input gate, and output gate which works as a set of filters. Unlike in the normal RNNs, both the output Y and the state C of the cell are passed to the next timestep. The structure of an LSTM cell is more complicated than a simple neuron in an RNN unit as shown in Figure 10.

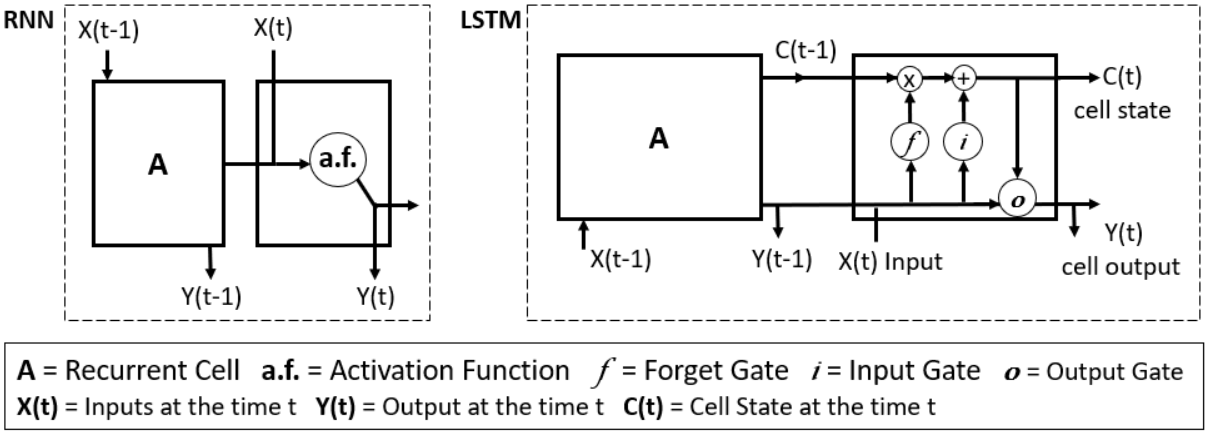

Figure 10: Comparison between simple RNN cells and LSTM cells

Firstly, from the input of the current timestep x(t) and the output of the previous cell y(t-1), a potential state update vector c'(t) is written as [18].

$$c'(t) = tanh(W_{c'}x(t) + U_{c'}y(t-1) + b_{c'}) \quad [18]$$

W_f, U_f are the weights multiplication matrices of the gate for the cell input x(t) and the previous cell output y(t+1), b_f the bias matrix. The $tanh()$ returns a vector of values in the interval [-1,1].

The three gates act as a filter. The first gate, the forget gate, set what and in which degree will be forgotten, hence not passed to the cell state. Mathematically is expressed as [19].

$$f(t) = \sigma(W_f x(t) + U_f y(t-1) + b_f) \quad [19]$$

Where $f(t)$ is the output vector of the gate, W_f, U_f, b_f the weights and bias matrices, and $\sigma()$ is a sigmoid function that gives back a result in the range [0,1], the degree the information would be forgotten in the cell state c.

The input gate then computes which part of c'(t) needs to pass to the cell state as in [20].

$$i(t) = \sigma(W_i x(t) + U_i y(t-1) + b_i) \quad [20]$$

Where $i(t)$ is the output vector in the range [0,1], W_i, U_i, b_i the weights and bias matrices.

The final cell state c(t) is then obtained from [19] and [20] through the pointwise multiplication (indicated as *) between the output of the forget gate and the previous timestep cell state c(t-1) summed to the pointwise multiplication of the output of the input gate i(t) and the cell state vector candidate c'(t) as shown in [21].

$$c(t) = f(t) * c(t-1) + i(t) * c'(t) \quad [21]$$

The interpretation of [21] is the following. The old state c(t-1) is multiplied by the output of the forget gate, a vector between 0 and 1, which in fact delete some of its elements. In the while, the candidate

new state c'(t) is multiplied by the output of the input gate, also in the range [0,1] that determinates which of the elements of c'(t) will be kept and added to the new state c(t).

The last gate is the output gate, which controls what information of the cell state will be passed to the current cell output y(t) is expressed by [22]. The new cell output y(t) is calculated as in [23], by combining [22] and [21].

$$i(t) = \sigma(W_o x(t) + U_o y(t-1) + b_o) \quad [22]$$

$$y(t) = \tanh(c(t)) * o(t) \quad [23]$$

Where * is a pointwise multiplication, W_o, U_o, b_o the trainable parameters of the output gate.

4.2.4 LSTM Encoder-Decoder for Seq2seq

An Encoder-decoder architecture is in general a system where a first NN, the encoder, process the input and output a fixed-length sequence, called encoded sequence. A second NN, the decoder, takes the encoded sequence, and eventually other inputs, and return the final output.

In the case the objective is to process a sequence to obtain another sequence, the encoder and the decoder can be configured as RNN, and in this case, the model is called sequence to sequence, or simply Seq2seq. This structure allows to easily add additional inputs for to decoder. In case of temporal sequence, it's easy to process the past timesteps in the encoder, and then use the decoder to predict multiple timesteps in advance, feeding to each timestep additional inputs and giving to the model a fully developed causal structure.

Even though other options are available, LSTM's ability to store information makes their use in encoder-decoders for Seq2seq a natural choice for sequences with long-term dependencies, especially if they are of a considerable length (Sutskever, Vinyals, and Le 2014). A general example of LSTM encoder-decoder for time series forecasts is shown in Figure 11.

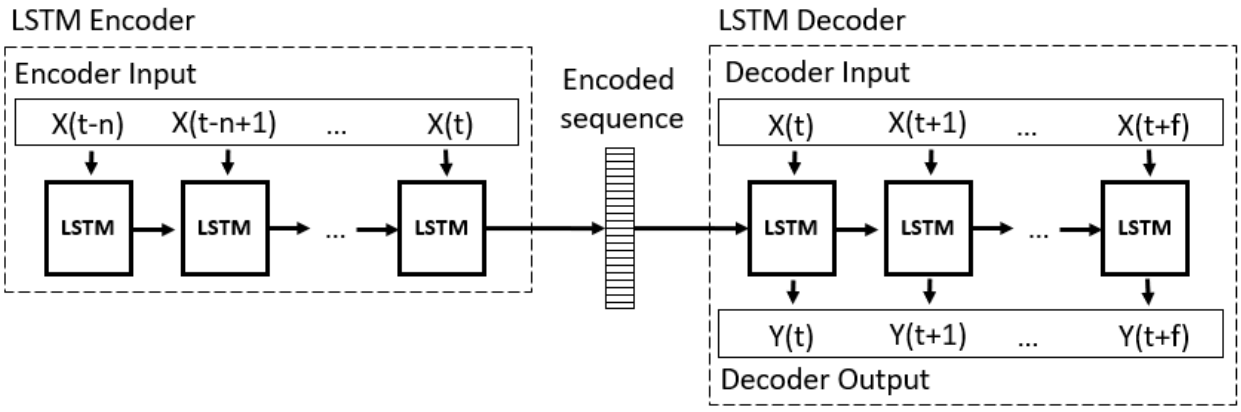

Figure 11: LSTM Encoder-Decoder architecture for Seq2seq

The encoded sequence in an LSTM encoder-decoder can be either produced by the output or the state vector of the LSTM. In the first case, the sequence is usually merged in a matrix with the decoder inputs, in the second the sequence is copied as initial state in the decoder LSTM memory. The first solution is less elegant and more memory intensive, but it gives more freedom for the structure of the different parts of the model, while the second gives constraints about the vectors' dimensionalities between the encoder and the decoder.

4.2.5 Avoid overfitting

One of the main issues in ML is the problem of underfitting and overfitting. A model underfits when it's not able to fit properly the training dataset as the features and parameters are not enough. When a model is instead able to fit perfectly the training data, it overfits. Overfitting is to avoid in order to obtain a model that can generalise the output for future predictions different from the training data. The poor generalization capacity caused by underfitting and overfitting can be seen in Figure 12.

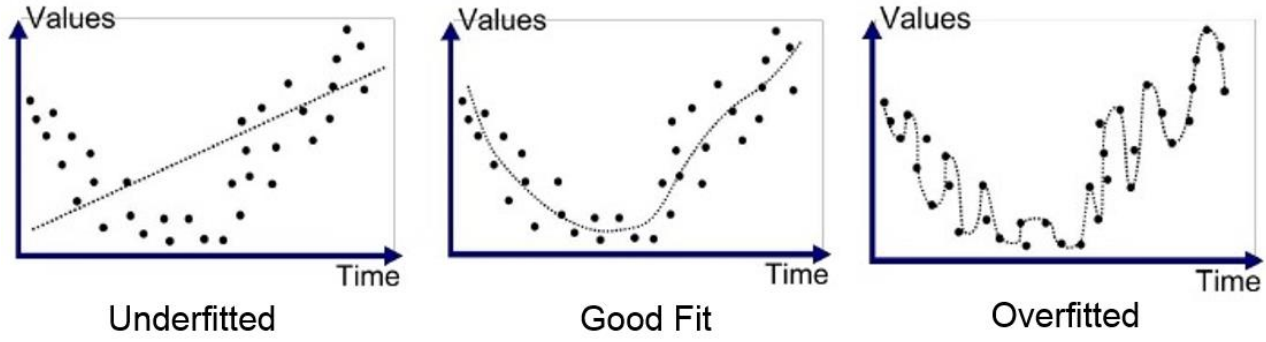

Figure 12: Examples of underfitting, good fit, and overfitting of a model on the data

Models prone to overfit are called high variance model. ANNs, due to the number of parameters and the strong non-linearity are a classic example of high variance ML models. Avoiding overfitting is hence a priority when using this kind of tools.

In order to avoid overfitting different options are available:

- increasing the size of the training set
- decrease the number of features
- decrease the complexity of the model (fewer parameters)
- regularization methods
- early stopping during the training
- batch size

Because the potential and flexibility of NNs are due to their complexity and ability to process several features, the best strategy is to build a complex model with all the features available (high variance

model) and to use many training data, regularization techniques, and later hyperparameters and features optimization in order to avoid overfitting.

The most common methods for regularization are the L1 L2 regularization and dropout. The first method consists in adding a sum term of the parameter in the loss function, to avoid some weights to increase too much and to ensure that the output is not dependent only on few preferential paths. The parameters L1 and L2 control the influence of the sum term on the loss function.

In this work, the Dropout method for regularization is used, as it doesn't influence directly the loss function but is build inside the NN structure. The dropout is a special layer in between two other layers, which randomly disconnect a certain number of connections for every epoch during the training, forcing the NN to not build preferential paths for the predictions but to use instead of a greater number of connections as shown in Figure 13.

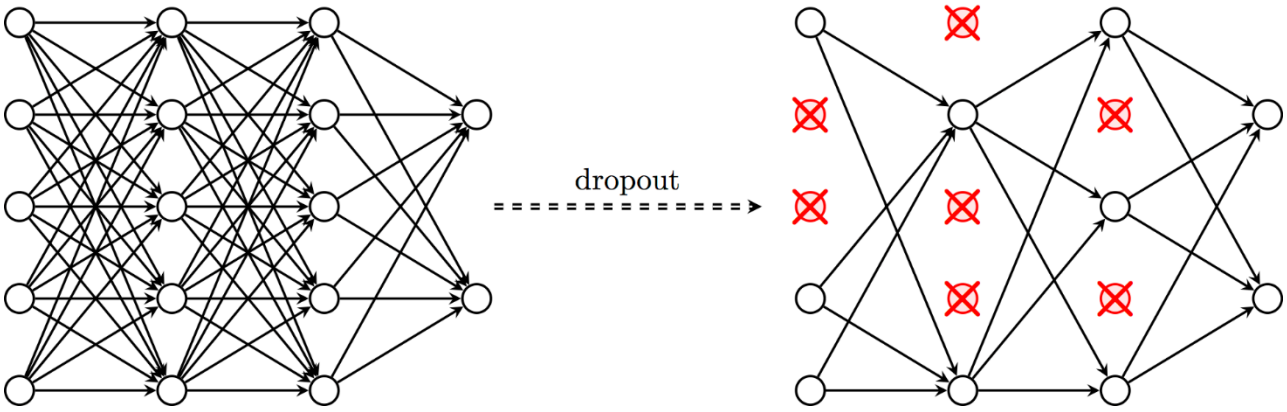

Figure 13: Schematic process of Dropout. On the left fully connected (dense) layers. On the right, the same network is showed where the red neurons represent the fraction of random connections cut off every iteration.

In addition to dropout regularization, an early stopping approach of the is a good method to avoid overfitting without making the regularization methods too invasive and save time during the training. Early stopping simply means that the training stops automatically after a certain number of epochs where the validation error is not decreasing.

4.2.6 ANN model structure

The model is a Seq2seq model that uses past measurements and future weather forecasts in order to predict the flow rate of a single runoff gauge every hour for 12h, as shown in Figure 14 below.

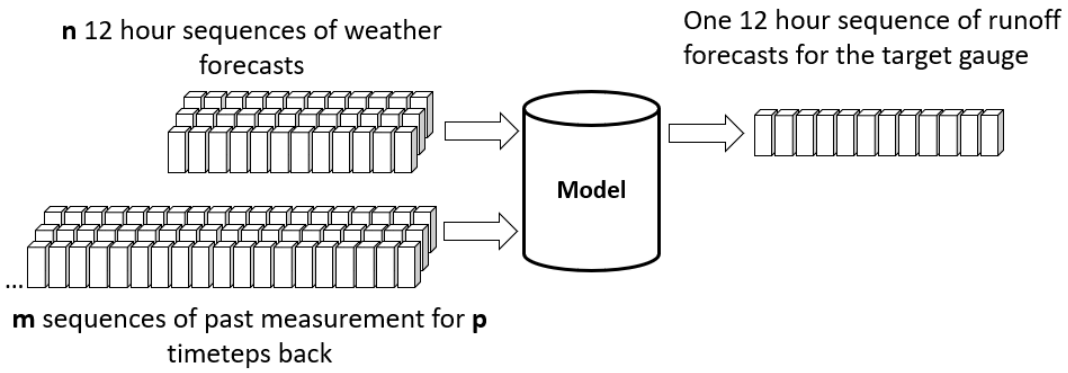

Figure 14: Schematic representation of the input/output of the model. The past measurements and future weather forecast are the inputs. The output is represented by the 12h sequence of predicted flow for the target gauge.

The architecture is an encoder-decoder LSTM with dropout regularization, equipped with an additional group of output sequence correction layers in order to have every prediction to start at t=0. The final custom layers simply receive as input the actual value of the flow at t=0, calculate the difference with the prediction, and use the difference to translate the whole output sequence at the right initial flow. The full NN structure is shown below in Figure 15.

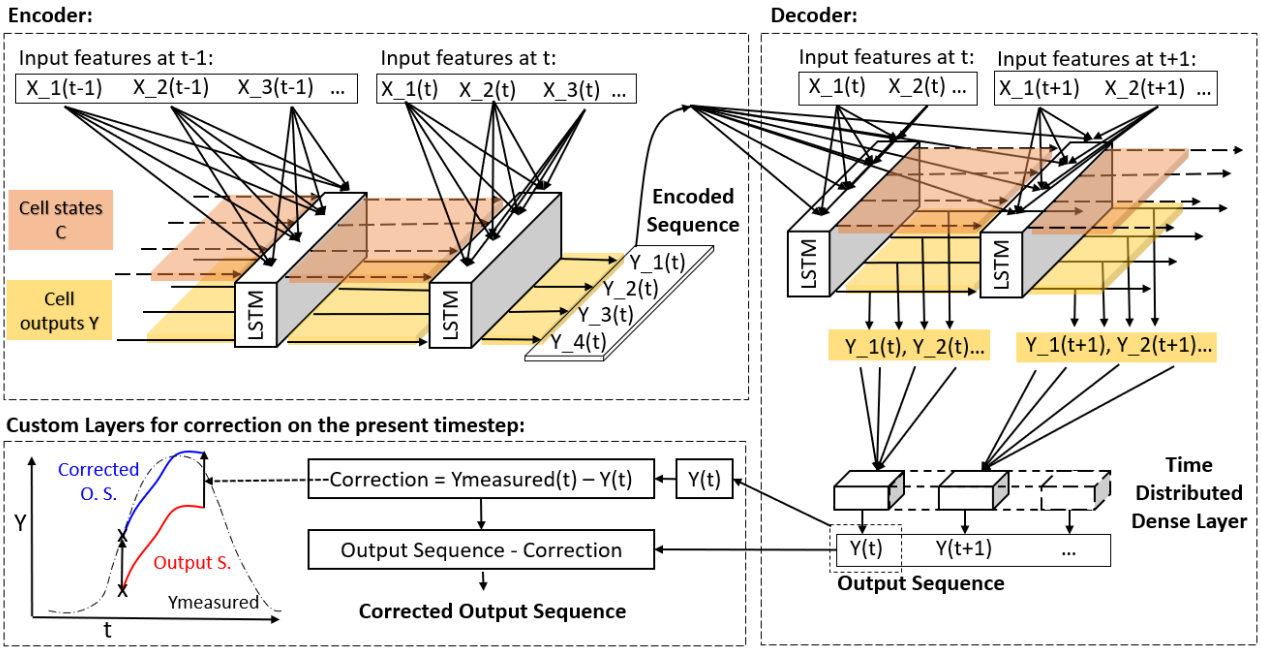

Figure 15: Encoder-Decoder LSTM structure with output sequence correction layers.

More in detail, the different layers used to create the model in Keras are shown in Figure 16.

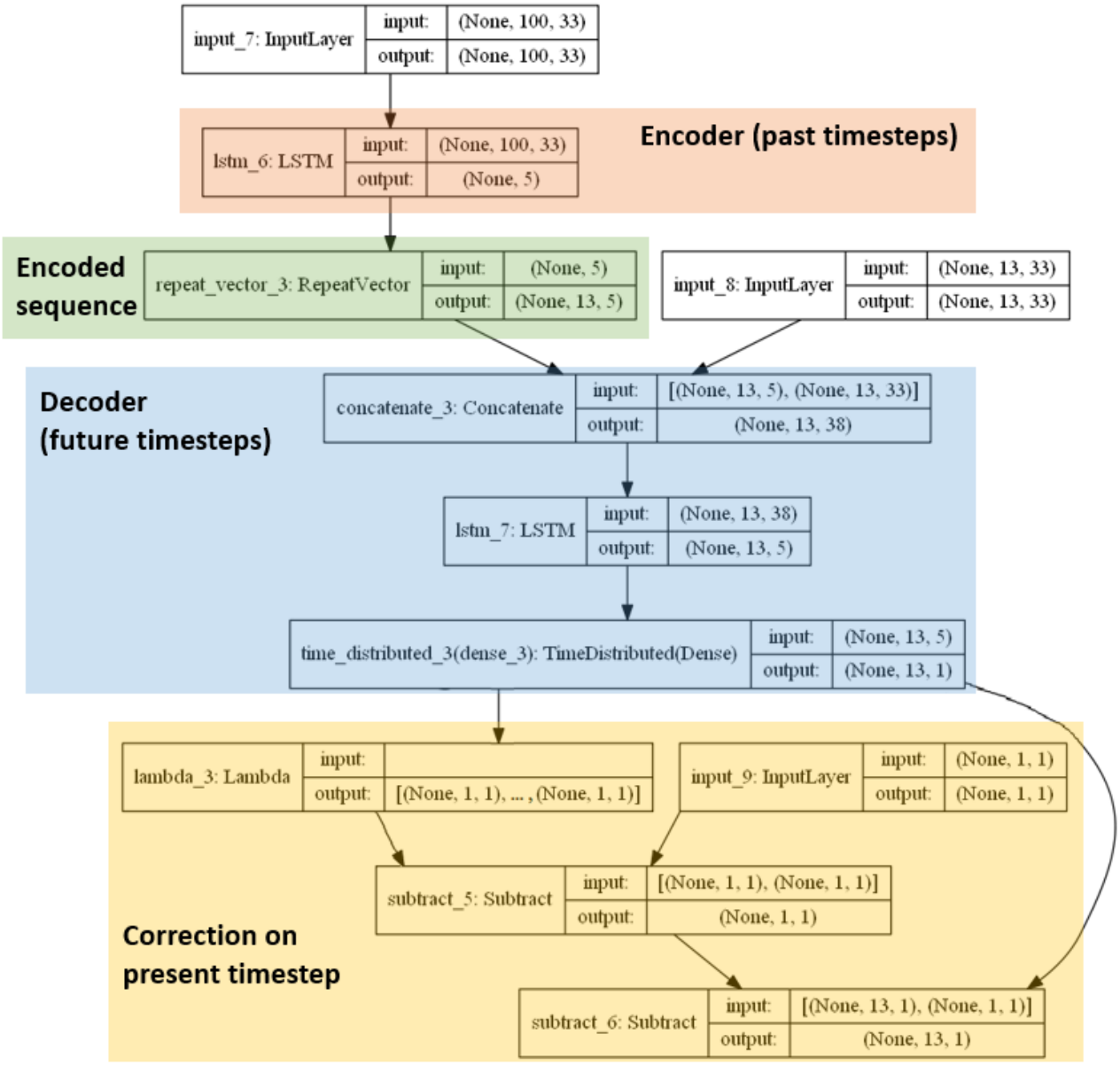

Figure 16: Example of the model implementation in Keras with 100 timesteps backwards and 33 features. The components of the encoder, encoded sequence, decoder and correction part of the NN are marked in red, green, blue, and yellow respectively.

The body of the model is formed by the 2 LSTM layers of the Encoder and the Decoder. The LSTM layer of the encoder produces a single output vector at the last timestep, the encoded sequence. The encoded sequence needs to be adjusted in its dimensions by a repeat_vector layer in order to be stacked with the inputs of the decoder by a concatenation layer.

The new concatenated vector is then processed by the LSTM layer of the decoder, that produces 1 vector result for each time step forecast. The resulting 13 vectors (t= plus 12h forecast) are processed by a time-distributed dense layer, which returns a single value each timestep.

This sequence is the output of the trainable part of the network, but usually, the forecast at t=0 differs from the observed flow, which is an available data. In order to correct the output sequence on the present timestep value, the prediction at t=0 is extracted using a customizable Lambda layer, and feed to a Subtract layer which calculates the error using the input observed flow at t=0. The error is subtracted in the last layer, that returns the final output sequence corrected on the initial time step.

When building the model is necessary to set the hyperparameters. The values of several hyperparameters are set by a mix of manual search and literature advice (number of LSTM stacked layers, optimization algorithm, dropout rate, loss function) others comes as a compromise between ideal values and computational effort (batch size), and some are interdependent (training epochs and Early stopping criteria). At last some hyperparameters can be chosen for their specific characteristics: the ReLU activation function used for the time-distribute layer of the decoder constrains the outputs to be positive, as the flow values to predict are.

Three hyperparameters find no suggestions in literature and a wide range of possible values. Those are the number of timesteps backwards in the encoder, and the vector length of the Encoder and the Decoder. In order to tune these hyperparameters, a simple optimization Genetic Algorithm (GA) is used (Section 4.6.). The main hyperparameters of the model are shown in Table 3.

Table 3: Main hyperparameters for the ANN model

Structural hyperparameters:		**Used value**
Number of LSTM layers in the Encoder	How many stacked layers of LSTMs	1 Layer (non-stacked LSTM)
Number of LSTM layers in the Decoder	How many stacked layers of LSTMs	1 Layer (non-stacked LSTM)
Timesteps backwards	Number of timesteps for the encoder input	Calculated by Hyperparameters Tuning
Encoder vector length	Length of the state and output vectors for the LSTM cells in the encoder. It also determinates the length of the encoded sequence	Calculated by Hyperparameters Tuning
Decoder vector length	Length of the state and output vectors for the LSTM cells in the encoder.	Calculated by Hyperparameters Tuning
Activation function of time-distributed dense layer	Activation function of the time distributed dense layer at the output of the Decoder	ReLU

Training hyperparameters:		
Training epochs	Number of epochs (iterations over all the batches) during training	Determined by Early Stopping
Early Stopping	Number of epochs the train stops after the lowest error for the validation set is reached.	15 epochs
Batch size	Number of random samples in each batch	100
Optimization algorithm	Algorithm to use for backpropagation during training	Adam
Loss function	Type of error the optimization algorithm minimises	MAE
Regularization hyperparameters:		
Dropout rate Encoder	random connections disabled for each iteration in the Encoder	20%
Dropout rate Decoder	random connections disabled for each iteration in the Decoder	20%

It's important to notice that there are several other hyperparameters that were left as Keras default, as there must be a very specific reason for changing them. Some of them are the inner hyperparameters of the Adam algorithm or the activation functions of the LSTM gates.

4.3 Data pre-processing

Preparing the data for the model is extremely important. The main data pre-processing operations are shown in Figure 17.

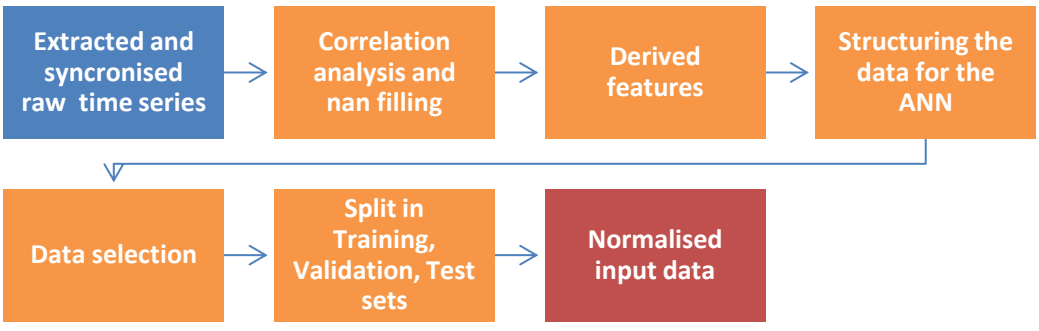

Figure 17: Data pre-processing main operations

Data from the station gauges are firstly extracted, resampled to an hourly timestep where needed and synchronised. Then a suitable time window is selected, where all the stations have the most available data.

In the second step, a correlation analysis is performed in order to find the best way to fill the missing data. Once the series is filled, few new features are derived from the existing ones in order to increase the efficiency of the network.

The data need now to be structured in a proper way in order to be feed to the NN. After this phase, only higher flow periods are selected, while the rest of the data are discarded. At last this dataset is split into the training, validation and test sets, and only the input data (not the observed output) is normalising in order to bring all the values in a common scale.

4.3.1 Correlation analysis and nan filling

The synchronised data needs now to be cleaned by the missing values (nan) that would create problems in the following steps. In order to determinate the best way to fill the nans, an analysis of the correlation between the different parameters is carried out. Based on this analysis, different approaches are applied to different time series.

Regarding the discharge data, the high correlation between the different gauges would, at first sight, suggest a multivariate linear interpolation method. Due to the short periods in which data are missing and the regularity of the pattern, a linear interpolation approach is used instead, as it gives better results.

The rainfall in the different gauges is partially correlated, and the discontinuous nature of the rain series makes a multilinear regression as in [24] using the data of the other gauges the best choice.

$$y = w_0 + \sum_1^n (w_n . x_n) \text{ [24]}$$

Where y is the output value, w_{0-n} the weights, and x_n the values of the inputs.

In the catchment of Schorgast (and Untersteinach), there is only 1 station that gives measurements of temperature, humidity, wind and total radiation. The correlation between the different parameters is very low, but they appear to have a strong daily pattern. Therefore, a linear interpolation based on the values in each hour of the day (h) is applied for long values-missing periods as in [25].

$$for\ h = [1{:}\,24] \quad y^{(h)} = y_1^{(h)} + \left(x^{(h)} - x_1^{(h)} \right) \frac{y_2^{(h)} - y_1^{(h)}}{x_2^{(h)} - x_1^{(h)}} \quad \text{[25]}$$

In the UWM catchment, different gauges give measurements of temperature, humidity, wind and total radiation. A correlation between different stations for the same parameter is present, therefore multivariate regression is used.

4.3.2 Derived features

In order to ease the effort of the NN in deriving new useful features from the input, new features are derived from the existing ones. Even if that's not necessary, it's important to remember that

sometimes the capability of the NN of performing simple operations is much lower than expected. For example, if it's not the temperature itself, but the rise in temperature a useful parameter, the NN could require a considerable effort to extract the difference in temperature from the temperature.

In addition to that, deriving features from the original input representation can be very application-specific and it's an opportunity to incorporate domain knowledge into the data which often brings to increase in performance (Guyon and Elisseeff 2003).

In order to give to the NN some possibly useful feature, new ones are created as shown in Table 4.

Table 4: Derived features.

New feature:	Derived from:	Rule:
Delta Flow (dQ)	Flow (Q)	dQ(t) = Q(t) - Q(t-1)
Delta Temperature (dT)	Temperature(T)	dT(t) = T(t) - T(t-1)
Snow*	Rainfall, Temperature (T)	If T ≥ upTlim: Rain = Rainfall Snow = 0 If dwnTlim > T > upTlim: Rain = Rainfall*(upTlim-T)/(upTlim-dwnTlim) Snow = Rainfall - Rain If T ≤ dwnTlim: Snow = Rainfall Rain = 0
Rain*		

*Rain and Snow take the place of Rainfall, that is not used directly in the NN.

The new features of dQ and dT are easy to obtain. The differentiation of snow and rain is more complex and probably important, as it would be very influential in the winter dynamics. The division is made assuming an inverse linear correlation between snow and rain in between 2 average temperature values defined by upTlim (upper T limit) and dwnTlim (lower T limit). The values those two parameters are initiated are 4°C and -2°C. Those values are based on physical consideration: at an average of 4°C, it's easy to have some snow in the more elevated areas, while temperatures around -2°C ensure that the whole precipitation is snow.

Using additional features is usually a good trade when dealing with NN. The chance that those extra features could have a little impact on the final result increasing at the same time the risk of overfitting is counterbalanced by the fact that well-regularized models with a sufficient amount of training samples can easily learn to ignore features that are not useful. The model doesn't eventually need to synthesize those derived values itself and the learning process could even speed up regardless of the bigger size of the input.

After adding additional features, a feature selection can be carried out to eliminate those that end up to be redundant and to calibrate the hyperparameters directly connected to the features such as the upper and lower boundaries for snow and rain. Several methods are available to perform those operations. In this work the optimal features hyperparameters upTlim and dwnTlim, and the features' impact on the model are calculated through simple methods based on the Sensitivity Analysis (SA) of the trained model as explained in Section 4.6.

4.3.3 Structuring the data for the ANN

The data needs then to be properly structured to be used in the NN. The format required for the temporal recurrent structure of the network in Keras is the number of samples in the first dimension of the vector, the timesteps in the second, and the features in the third as shown in Figure 18.

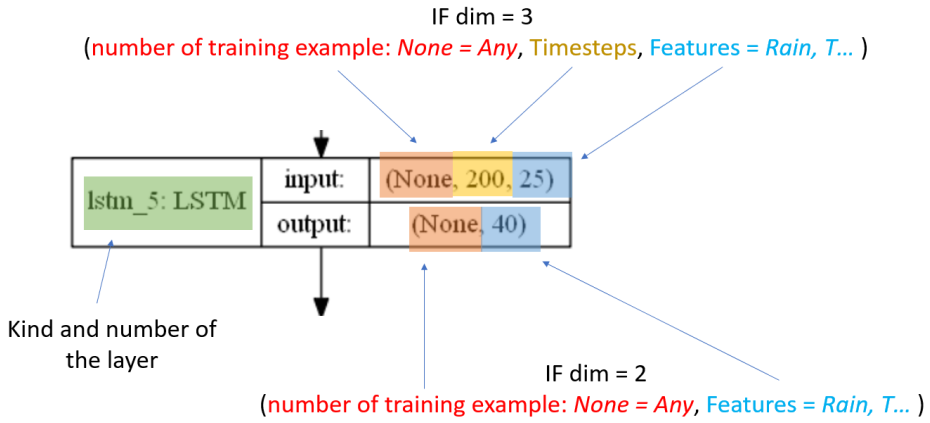

Figure 18: Input format required for a recurrent neural network in Keras-TensorFlow

The data needs first to be divided into the encoder input (X_e), decoder input (X_d) and output (Y). The different structures and the reason for each part are shown below in Table 5.

Table 5: Data structure and meaning for the NN.

	Structure of a single sample:	**Meaning:**
Encoder input (X_e)	n timesteps backwards for each feature, including Q and dQ values.	Past measurements from the stations.
Decoder input (X_d)	13 timesteps for each feature (excluding Q and dQ values) from t to t+12. Q and dQ values only for t=0.	Values at t=0 and weather forecasts.
Output (Y)	13 timestep for the target gauge Q from t to t+12.	12 h forward flow forecast target.

The encoder input includes the timesteps backwards in time for all the features. The decoder input includes the weather forecasts for the 12 hours, plus the flow and dQ at the present time. The output is the flow sequence of the target gauge of 13 timesteps, from t=0 to t=12 h.

4.3.4 Data selection and Training, Validation and Test split

In order to fully evaluate the performance of a ML model, the dataset is divided into 3 parts: training, validation and test. The training dataset is the larger, accounting usually around 60-90% of the data samples. It is used to train the model, so the weights of the NN will be calibrated by minimising the loss function based on the training examples.

The validation dataset usually accounts for the 0-30% of the samples and it is used for various purposes. The main function of the validation is to help to monitor during the training the risk of overfitting or underfitting of the model. If with the increase of iterations during the training the error calculated for the training dataset keeps decreasing, but the error calculated on the validation stagnate, or worst, increases, it means the model is overfitting and therefore is not able to generalise its predictions.

Another important use of the validation is hyperparameters tuning and evaluation of features selection. A different set of hyperparameters are used, and the combination of them that gives the best score for the validation is chosen. The same is valid for the evaluation of the input features of the model.

When performing uncertainty analysis, it should be performed on the validation dataset as well. With the exception of Bayesian NNs, ANNs, in general, can't infer the uncertainty of the inputs and return directly the uncertainty of the output. The simplest way to determinate the uncertainty of the result is then to use the error of the output, and it should be the one of the validation set, as the errors of the training set would be often smaller and hence not realistic.

The dimension of the validation database depends on how much the monitoring of the training, hyperparameters tuning and uncertainty analysis are considered as important. If all those operations are performed thoroughly, it could be up to 30% of the full dataset, while no validation set at all is required in the case none of such operations is performed.

The test database must be used only for the final score of the model and nothing else in order to have a totally unbiased evaluation. It's important not to choose the best set of hyperparameters or features based on the score of the test set, as the hyperparameter tuning and the features selection are, in fact, additional calibrations. For the same reason performing the uncertainty analysis based on the errors of this dataset, it's not suggested, as any other evaluation out of the final score must be performed on the validation set. An overview of the three splits is shown in Figure 19.

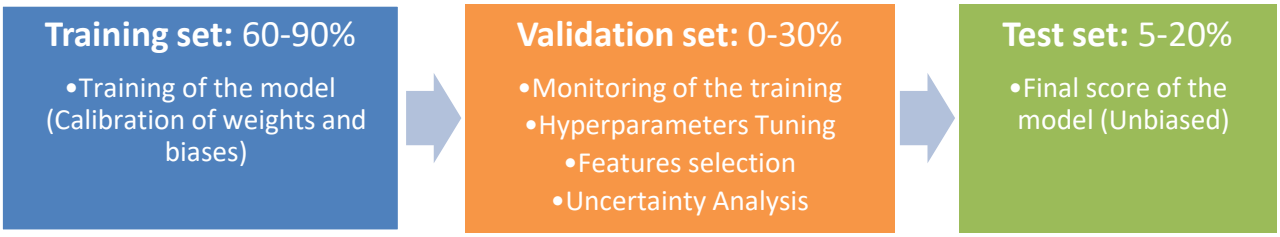

Figure 19: Training, Validation, Test set. Splits of the full dataset and function of each.

In order to produce the three datasets for the models, different steps are taken. In order to create the training set, the structured data are filtered in order to select only interesting periods of high flow and avoid that a big number of low flow data could make difficult the prediction of high runoff peaks. The selection is performed by deleting all the samples not reaching a minimum threshold flow at the target gauge for all the available years till November 2011. The threshold for each gauge is shown in Table 6.

Table 6: Minimum Q threshold for the target gauge used to filter the training dataset.

Gauge	Ködnitz	Kauerndorf	Untersteinach
Q Threshold [m^3/s]	4	3	1

The validation set is selected on the period from December 2011 to March 2012, characterised by high flow rates, while the test set is formed by two single extreme runoff events of December 2012 and May 2013. Once the wanted periods are selected, the database is split in the training, validation, and test set in the following Table 7.

Table 7: Training, Validation, Test split for the dataset.

	Periods	**samples**
Training	11.2005-11.2012 (Schorgast, Untersteinach) or 11.2006-11.2012 (Upper W. Main) only samples where max(target$Q_{(h=1\text{-}12)}$) > Threshold	20683 (Schorgast) 35356 (Untersteinach) 17247 (Upper W. Main)
Validation	From 04.31.2011 to 04.03.2012	1400
Test	Events of 13.12.2012 to 29.12.2012 and 25.05.2013 to 06.06.2013	700

In the specific case, the test and validation sets are small compared to the training. That's reasonable because of the higher-level selection of the data of those two sets, that makes them meaningful besides their relative dimension.

4.3.5 Input normalization

In order to ensure a common range for all the data, is a praxis to apply a normalization for the input. The normality is to normalize the data with std of 1 and mean 0. In this case, due to the possible sensitivity of the temperature to the threshold of 0°C, it's preferred to normalise the data only to std value of 1. In this way, the points where T is equal to 0 are not perturbated by the transformation.

Each input feature of the training set is normalised and the parameters of the normalization are saved. Those parameters are used to apply the same transformation to the test and validation inputs as shown in Figure 20.

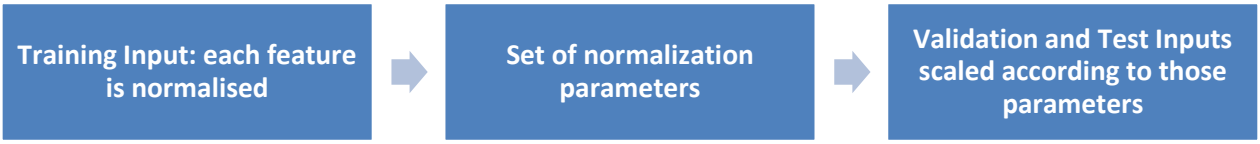

Figure 20: Normalization process for NN inputs

4.4 Uncertainty analysis and Evaluation Criteria

The evaluation of the models is based on uncertainty. As mentioned in section 4.3.7, the uncertainty analysis is performed on the Validation dataset in order to have an unbiased uncertainty estimation based on data were not used for the training.

In order to calculate the uncertainty bands, a simple relative error method is used. For each hour forecast, the error of the predictions is calculated. The width of the uncertainty bands is then calculated as quantiles of the error distribution. Three quantiles are used: 0.95, 0.97, 0.98 for the upper band and the relative 0.05, 0.03, 0.02 for the lower. In order to equilibrate the difference in the flow std and mean between the validation dataset, formed by a general period of higher flow, and the two well defined extreme test events, an additional coefficient is added to the upper and lower bands, corresponding to 0.5 of the value of the bands at t = 1 h.

Each choice of quantiles is translated in uncertainty by subtracting the lower quantile from the higher and expressing the result in percentage as shown in Table 8 below.

Table 8: Upper and lower quantiles, and related uncertainties, used in the Validation set in order to fix the upper and lower uncertainty bands.

Upper quantile [-]	Lower quantile [-]	Uncertainty [%]
0.95	0.05	90
0.97	0.03	94
0.98	0.02	96

Once the uncertainty bands are calculated two evaluation criteria are used in order to evaluate the performance: P and R factor [26] [27].

$$P_{factor(h)} = \frac{\sum_{i=1}^{h} Obs_{(i)} \subset [LB_{(i)}, UB_{(i)}]}{h} \quad [26] \qquad R_{factor(h)} = \frac{UB_{(h)} - LB_{(h)}}{\sigma Obs} \quad [27]$$

Where $Obs_{(h)}$ is the observed flow at the forecast hour h, $LB_{(i)}, UB_{(i)}$ are the low and upper uncertainty boundaries and σObs is the standard deviation of the observed values.

The P factor expresses the percentage of observations falling into the range of the uncertainty bands from the first to the last hour of the forecast, while the R factor expresses the width of the uncertainty bands at every hour forecast weighted on the standard deviation of the observed data. The P factor ranges from 0 to 100, the R factor from 0 to infinity.

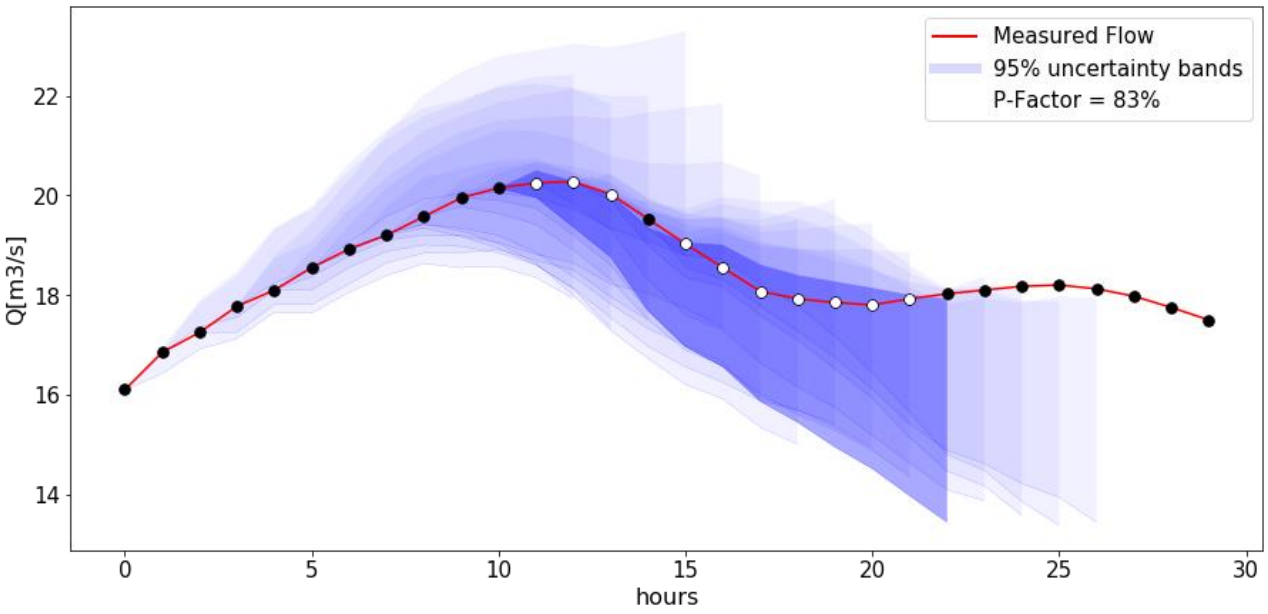

Figure 21: Example of uncertainty band and related P-Factor.

Because in the Validation set the P factor is fixed by the choice of the quantiles, the only variable parameter is the R factor. The evaluation of the model for the validation dataset can be in this way be measured by the only R Factor values as an alternative to the NN loss function. This will be used as criteria for the hyperparameter tuning (Section 4.5), while the features SA will be simply based on the NN loss function MAE (Section 4.6).

For the final evaluation on the test, both R and P Factor values are not fixed, even though are inversely correlated, and the evaluation will be based on both criteria.

4.5 Hyperparameters Tuning using Genetic Algorithms (GA)

Three important hyperparameters of the NN, the timesteps backwards in the encoder, and the LSTM vector length of the encoder and the decoder have a wide range of possible values and are important for the performance of the NN as they set directly the total number of trainable weights in the model and the quantity of past information will be processed.

A proper tuning or calibration of the hyperparameters can be performed with different approaches. In addition to the manual search, the simplest automatic algorithms which can be used are grid search and random search, which can be quite inefficient compared to the computational effort. More advanced algorithms use the information gathered during the optimization process for improving the search of the optimal solution. The state of art of such algorithms is Baesyan Optimization method and Evolutionary Algorithms such as Genetic Algorithms (GA).

GAs are meta-heuristic algorithms that usually can find sub-optimal solutions close to the global minimum for non-deterministic polynomial hard problems (NP) such as the calibration of the best set of hyperparameters of a neural network (Bouktif et al. 2018). A simple GA is used in this work because it usually needs less time-consuming model runs compared to other techniques, it's easy to implement from scratches, and the randomness of its behaviour usually allows to find a reasonable solution even without a full convergence of the parameters.

A GA is a random-based classical evolutionary algorithm inspired on genetics evolutionary concepts. The starting point is the population, composed of different individuals. Each individual is defined by a chromosome, containing different genes.

All the individuals of the population are tested using a fitness function that returns a score. The individuals with the best score are selected for a Mating pool. Those individuals are called parents. The parents will generate offspring by randomly mixing the genes in their chromosomes. This operation is called breeding. In addition to that, some offspring mutate randomly one gene in their chromosome, in order to ensure the born of totally new individuals. The offspring become a new generation, that will be tested with the fitness function repeating the procedure generation after generation. Mathematically, the breeding ensures the convergence towards a minimum of the fitness function, while the randomness of the mutation gives a good way to escape from local minima and an exploration potential.

4.5.1 GA implementation

In this work implementation, the GA is coded in python using only python libraries numpy and matplotlib. The fitness function is the ANN model, and the genes are the three hyperparameters to be tuned: steps backwards, encoder length, decoder length.

The first population (Generation 0) is composed of 10 individuals. Every individual has one chromosome containing 3 genes (the 3 hyperparameters) represented as an integer number. The genes are initialised randomly in the intervals defined by the space of the hyperparameters defined in Table 9.

Table 9: Genes (ANN hyperparameters) space boundaries

Genes:	**Initial searching boundaries**
steps backwards	[50,200]
encoder length	[1,30]
decoder length	[1,30]

The population is then evaluated by the fitness function: the ANN model is trained with each set of hyperparameters given by the genes of each individual. The fitness function's score is a weighted R-factor of the validation with the P factor fixed to 96% expressed by [29].

$$Score = \frac{1}{\sum_{h=1}^{12} h} \cdot \sum_{h=1}^{12} (R^{(h)}_{f(uncertainty=96\%)} \cdot h) \qquad [29]$$

This formulation aims to minimize the R factor, giving more weight to the last hours of the forecast, which are the most critical.

The 4 individuals with the best score form the mating pool and become parents. The mating happens by forming 10 random couples between the individuals, each couple will generate 1 offspring. The genes in a couple are mixed following a simple random-in-range rule as in [30] [31][32].

$$g_i^{offspring} = random\left(\left[\max(1, g_i^{parent1} - \Delta),\ g_i^{parent2} + \Delta\right]\right) \ \ with\ g_i^{parent1} < g_i^{parent2} \qquad [30]$$

$$\Delta = 0.2 * \left(g_i^{parent2} - g_i^{parent1}\right) \qquad if\ g_i^{parent2} - g_i^{parent1} \geq 20 \qquad [31]$$

$$\Delta = 2 \quad if\ g_i^{parent2} - g_i^{parent1} < 20 \qquad [32]$$

Where $random()$ is a function that generates a random integer between a range following a uniform distribution, $g_i^{offspring}$ is the gene i of the offspring $g_i^{parent1,2}$ are the gene i of the two parents, and Δ is an off-range interval that gives the possibility to the new gene to move out of the parent range and eventually from the initial search boundaries extending the space of parameters. The lower boundary is constrained to be > 0 by the term $\max(1, g_i^{parent1} - \Delta)$ to avoid the algorithm to crash by trying to build the model with a null or negative number of steps or vector length.

Once the mating has happened, 2 offspring mutate a random gene. The mutation consists of a simple change of the gene value in a random number in the ranges indicated in Table 9. Now, the

second generation is ready to repeat the process. The total number of generations used for the GA is 4, from Generation 0 to Generation 3. A schematic representation of the GA is shown in Figure 22.

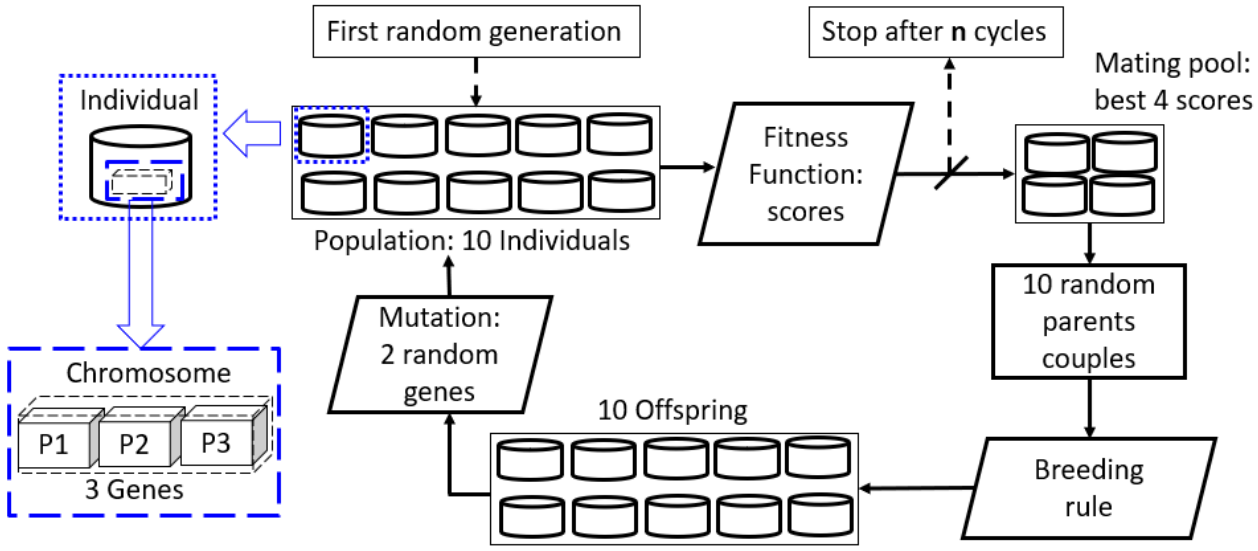

Figure 22: Schematic process of the Genetic Algorithm developed for hyperparameters tuning.

Ideally, the GA should converge towards a well-defined set of genes. Because of the hard non-linearity of the optimization problem and the randomness and variability of the elements acting during a neural network training (initialisation, optimization algorithm, batches), it's possible that a well-defined minimum is not found.

In that case, each hyperparameter is chosen by plotting it against the score and arbitrarily choosing a value where a certain number of better-performing individuals aggregate.

4.6 Features coefficients calibration

Features optimization is usually a complex problem, addressed by a whole branch of ML called Feature Engineering. The derivation of new features has been already treated in Section 4.3.2, while in this section the calibration of the coefficients connected to some of those features (Snow and Rain) is explained.

Calibrating the snow/rain coefficients upTlim and dwnTlim described in Section 4.3.2, which determinates the features of snow and rain for the different stations, can be a way to improve the performance. This is, in fact, a calibration of 2 hyperparameters, not connected to the model directly but to the input series. A time-consuming calibration as performed in Section 4.5 it's no more necessary, as for those hyperparameters the model doesn't need to be rebuilt and retrained every time as for the vector length of the encoder and decoder and the timesteps backwards.

That makes also the use of a GA for minimising the number of sampling superfluous. By using the pre-trained model, a prediction takes between 0.1 and 2 seconds, against the 10-15 minutes of the full NN training, and SA can be easily performed in a single row.

The first objective is obtained by performing a SA on the model previously trained with input data created with values of -2°C and 4°C for dwnTlim and upTlim.

The model is then run for 200 combinations of the two parameters obtained with a random uniform sampling distribution between the values of 0 and -5 for dwnTlim and 0.1 and 6 for upTlim using both the training and the validation set. The score criteria used is the loss function of the NN, the mean absolute error (MAE).

The set of parameters of the best MAE of the validation dataset becomes the new pair dwnTlim and upTlim to use for the predictions on the test events. The MAE of the training set gives a good overview of the sensitivity of the two parameters by showing how the results converge on the two values of upTlim and dwnTlim the model has been trained on. A well-defined convergence is a sign of high sensitivity of one parameter, a diffused and not clear one means the opposite.

The model is not retrained with the new calibrated training inputs as a mean of additionally avoid overfitting and guarantee the best generalization for the predictions. This approach is the one giving the best improvements.

4.7 Features Sensitivity Analysis

In this section, the evaluation of the single features used in the models and the eventual elimination of the redundant ones is carried out.

Feature selection methods essentially divide into wrappers, filters, and embedded (Guyon and Elisseeff 2003). The first methods eliminate all the features that don’t respect certain criteria without taking into account the model. Wrappers use the model as a black box and evaluate its performance using different subsets of variables. Embedding methods are built directly in the model and perform the feature selection directly during the training of the model.

Among the wrapper methods, a very simple approach is used in this work based on the Sensitivity Analysis (SA) of the trained model. Performing SA on complex models can define in an effective way the importance of each feature for the variation of model output by assessing the contribution of each input factor to the performance of the model (Sadeghyan 2018). The evaluation of the features is therefore based once on a single trained model, used only to produce predictions.

In this case, the SA algorithm is based on the std of the input and output features. The first step is to take a feature and replace it with 8 random pseudo-features generated from a normal distribution. The pseudo features have the same mean of the original feature, and the std equal to the std of the

original feature scaled each time for the values of a scaling vector J, ranging between 0 and 10. The std of the original feature is approximately 1, as the input parameters are normalised. That means it's possible to calculate the std of the vector J and use it as a generalised representation of the variation of data range among the inputs pseudo-features. The vector J values and the expression for the variation of the input (std_{input}) are shown in [33] and [34].

$$J = [0 \;\; 0.01 \;\; 0.1 \;\; 0.5 \;\; 1 \;\; 2 \;\; 5 \;\; 10] \qquad [33]$$

$$std_{input} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(std_{pseudofeature(i)} - \overline{std_{pseudofeatures}}\right)^2} \;\cong\; std(\mathrm{J}) \quad [34]$$

The model is run using the pseudo-features and the forecasts are scored using the model loss function MAE. The std of the scores obtained with the different pseudo features represents the variation of the model output for an input variation unit as shown in [35].

$$single\ feature\ sensitivity = \frac{std_{output}}{std_{input}} \qquad [35]$$

The procedure is repeated for every model input feature once per time, obtaining in this way an evaluation of the impact of the single feature on the model. Figure 23 offers a good overview of the algorithm.

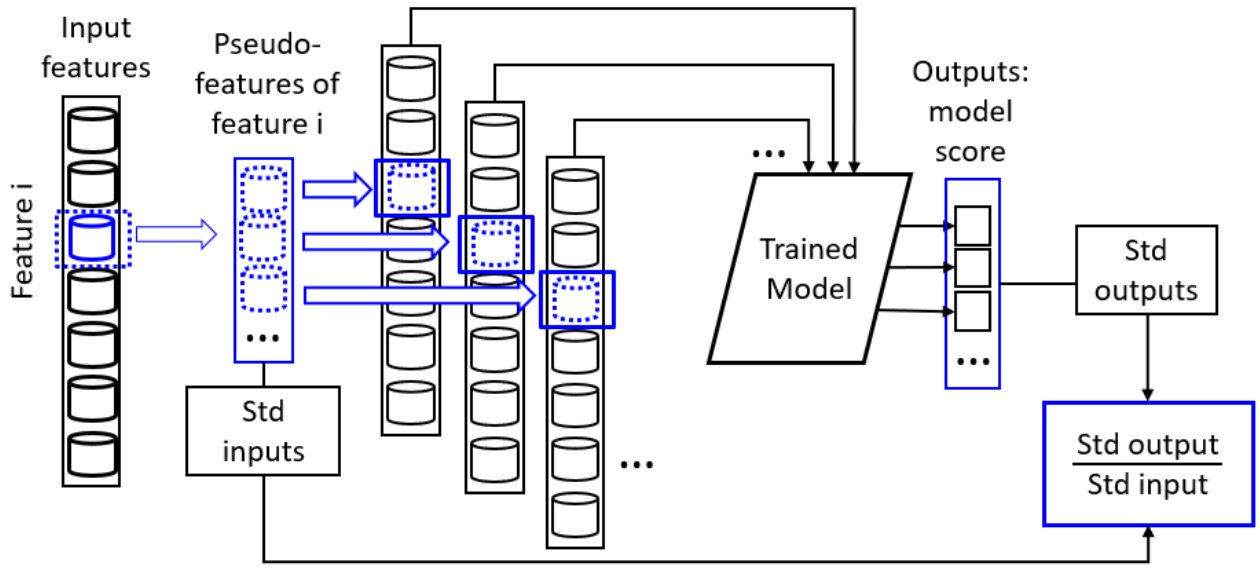

Figure 23: Scheme for performing SA on a single feature of the input. The trained model performs different forecasts using each time a different pseudo-feature: a random-normal generated sequence having the same mean as the original feature, and std scaled of factors from 0 to 10. The forecasts are scored, and the std of the scores is divided by the std of the scaling factors of the pseudo-features in order to obtain the variation in std of the output per unit std of the input.

The procedure is implemented using both the training and the validation set in order to assess both the contribution of the single feature in fitting the data during the training and the contribution in generalising the results during the forecasts.

A feature is considered redundant and remove only in the exceptional case in all the models where it's used (Untersteinach, Schorgast, UWM) obtain a score lower than any other feature. This very restricting criterium is adopted as the validation set may not be big enough to guarantee that some feature would be helpful in the forecasts of different events. Also, the need for reducing the number of inputs is not strong in this model, as the features are not a big number and there are already a series of effective measures to avoid overfitting.

In addition to the practical individuation of redundant features, the SA analysis allows to evaluate the added derived feature in comparison to the original ones and potentially helps the physical interpretation of the features in the model.

4.8 Evaluation

The final evaluation is carried out on the 2 high flow events of December 2012 and May 2013 which compose the test dataset (Figure 24).

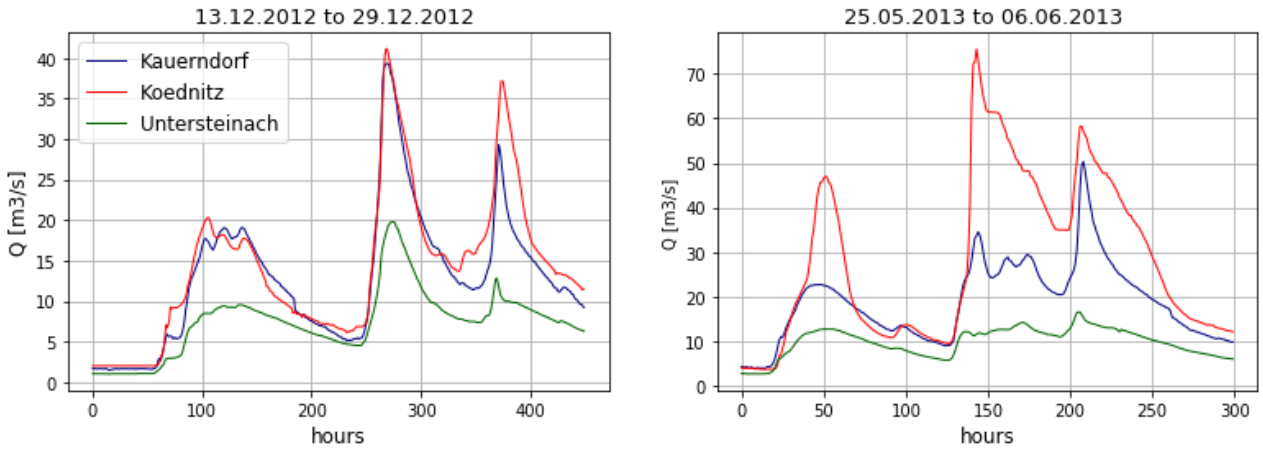

Figure 24: The two events of the test dataset. The first in December 2012, the second in May 2013.

The test set was not used for any operation until now, so the evaluation based on those two events is completely independent of any training, calibration or uncertainty estimation performed in the previous steps. The evaluation parameters are the P and R factors (Section 4.5) for every hour of the 12 h forecast sequence.

The two events of December 2012 and May 2013 which forms the test set are plotted for the gauge of Ködnitz with the average rainfall, snowfall and temperature in Figure 25 and Figure 26 respectively.

The event in 2012 is characterised by marked oscillation in temperature between negative and positive values with snow, rain and mixed precipitation. The event of 2013 happens with positive temperatures with only rain as precipitation.

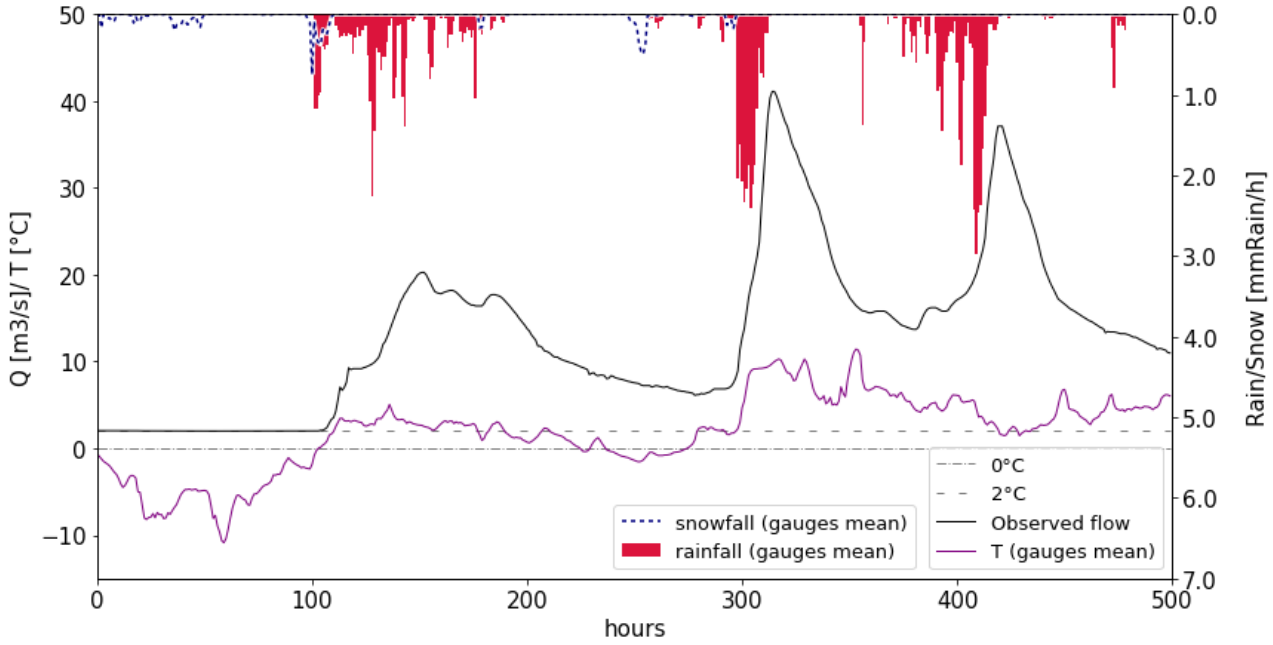

Figure 25: Flow in the gauge of Ködnitz plotted with the average rainfall and snowfall and the temperature of the stations used in the model of UWM during the test event of December 2012.

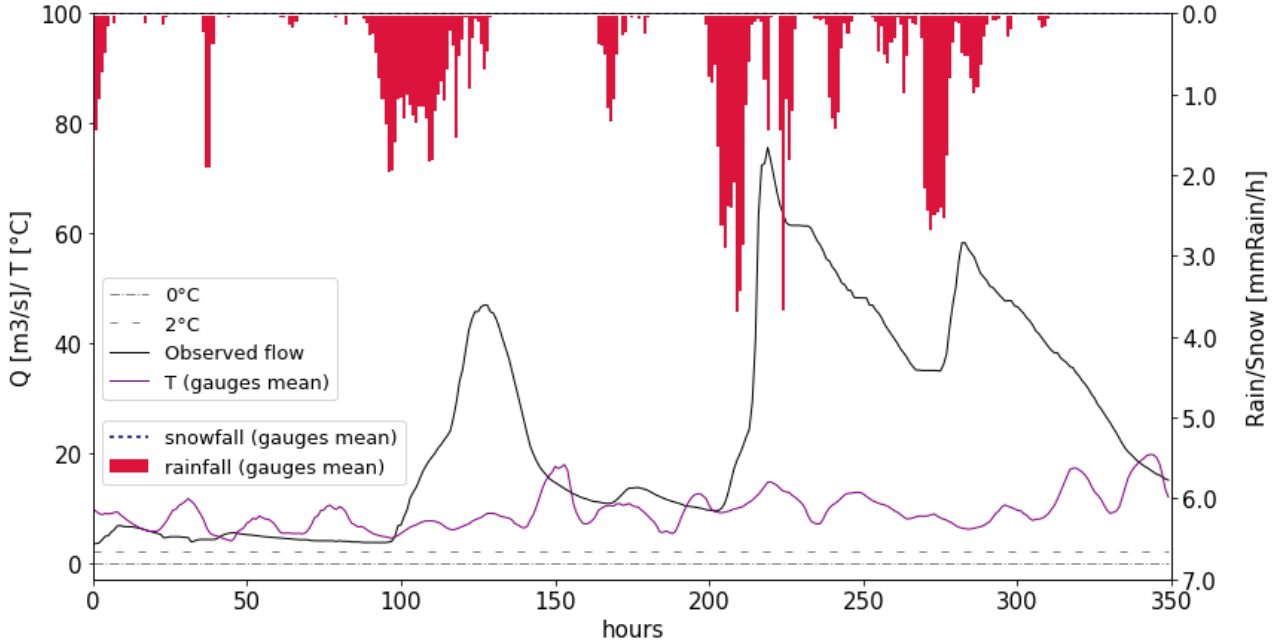

Figure 26:Flow in the gauge of Ködnitz plotted with the average rainfall, snowfall and the temperature of the stations used in the model of UWM during the test event of May 2013.

Figure 27 shows the rain, snow and temperature plotted together with the flow of Ködnitz for the whole winter and spring of 2012-2013 where the test events are located. December 2012 is

preceded by autumn snowfalls and oscillations in temperature above and below 0°C, while May 2013 is far from any influence of snowfalls or low temperature.

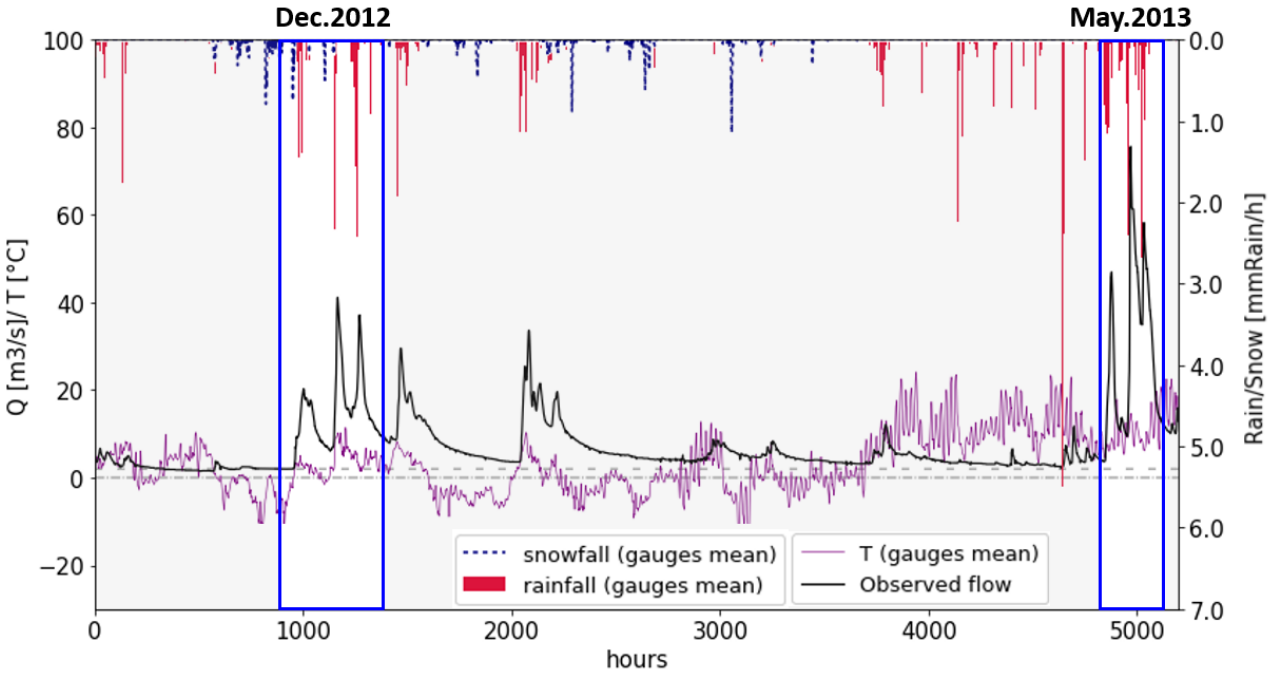

Figure 27: Flow in the gauge of Ködnitz plotted with the average rainfall, snowfall and the temperature of the stations used in the model of UWM during the whole winter and spring 2012-2013. The two test events are marked in blue.

Figure 28 represents the snow accumulation in the measurement station of Presseck (Schorgast) during the winter and spring 2012-2013. It's possible to confirm the important role of the snow dynamics for the forecasts in December 2012, while no snow accumulation remains almost two months before the event of May 2013. That totally excludes that rapid snowmelt of long-term accumulated snow in late May as the cause of the runoff peaks.

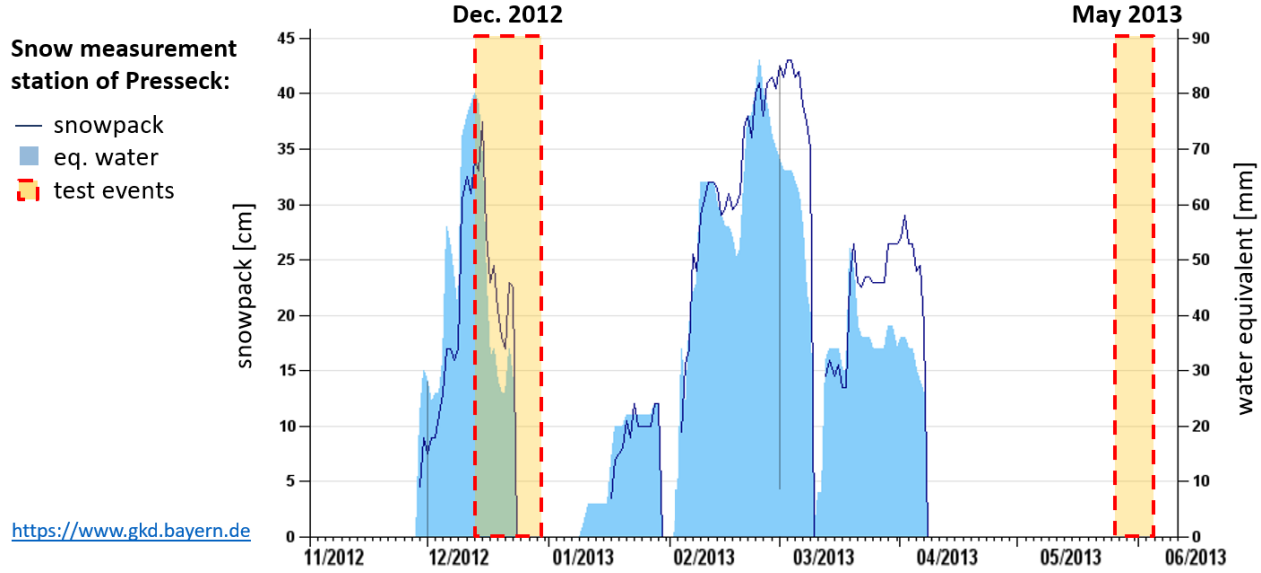

Figure 28: Snowpack and related water equivalent measured in Presseck (in Schorgast) during winter and spring 2012-2013. The time windows corresponding to the two test events of December 2012 and May 2013 are marked.

The event in December 2012 tests the ability of the model to reproduce the complex winter medium and long term dynamics (freezing, snowfall, accumulation and melting). The event in May 2013 test the model in simpler dynamics conditions (only rainfall) but with possibly more irregulars rainfall distributions typical of the later season.

The evaluation is carried out in 2 steps. In the first step, the only results of the ANN models are compared for the different catchments. The aim of this evaluation is to observe if the different availability and distribution of the data stations have a great influence on model performance. Even though the performance is evaluated in different gauges (Untersteinach, Kauerndorf, Ködnitz), the a-dimensional nature of the evaluation criteria (P and R factors) and the similar relative magnitude and correlation of the test events make a meaningful comparison possible.

In the second step, the results of the LSTM models are compared with the results from calibrated results of the hydrological model LARSIM and different methods of uncertainty calculation for the catchment of UWM (Leandro et al. 2019; Gander 2018). The different uncertainty methods used for the LARSIM model are described in Leandro et al. (2019) and more extensively in the BSc. thesis of Gander (2018). They are Ensemble, Relative Error, Discharge intervals, Rising/Receding, and Slope methods. The most direct comparison is the result obtained with the Relative Error method, as is the same methodology used in this work. It's still interesting to see how the other more elaborated approaches compare. This last evaluation is the most important, as it allows to understand if the ANN and the conceptual models are comparable in performance.

5.Results

5.1 Data Pre-processing

Examples of synchronised time series used for the Schorgast catchment are shown in Figure 29 below. The empty values (nan) are marked with red lines for the runoff gauge of Kauerndorf and the station of Poppenholz (Rainfall, Temperature, Air humidity, Wind, Xglob).

In general, the most complete data are the flow rates. The rainfall and the temperature often present some sparse steps of missing data. The humidity, wind and total radiation are the series that can have the longest missing data windows.

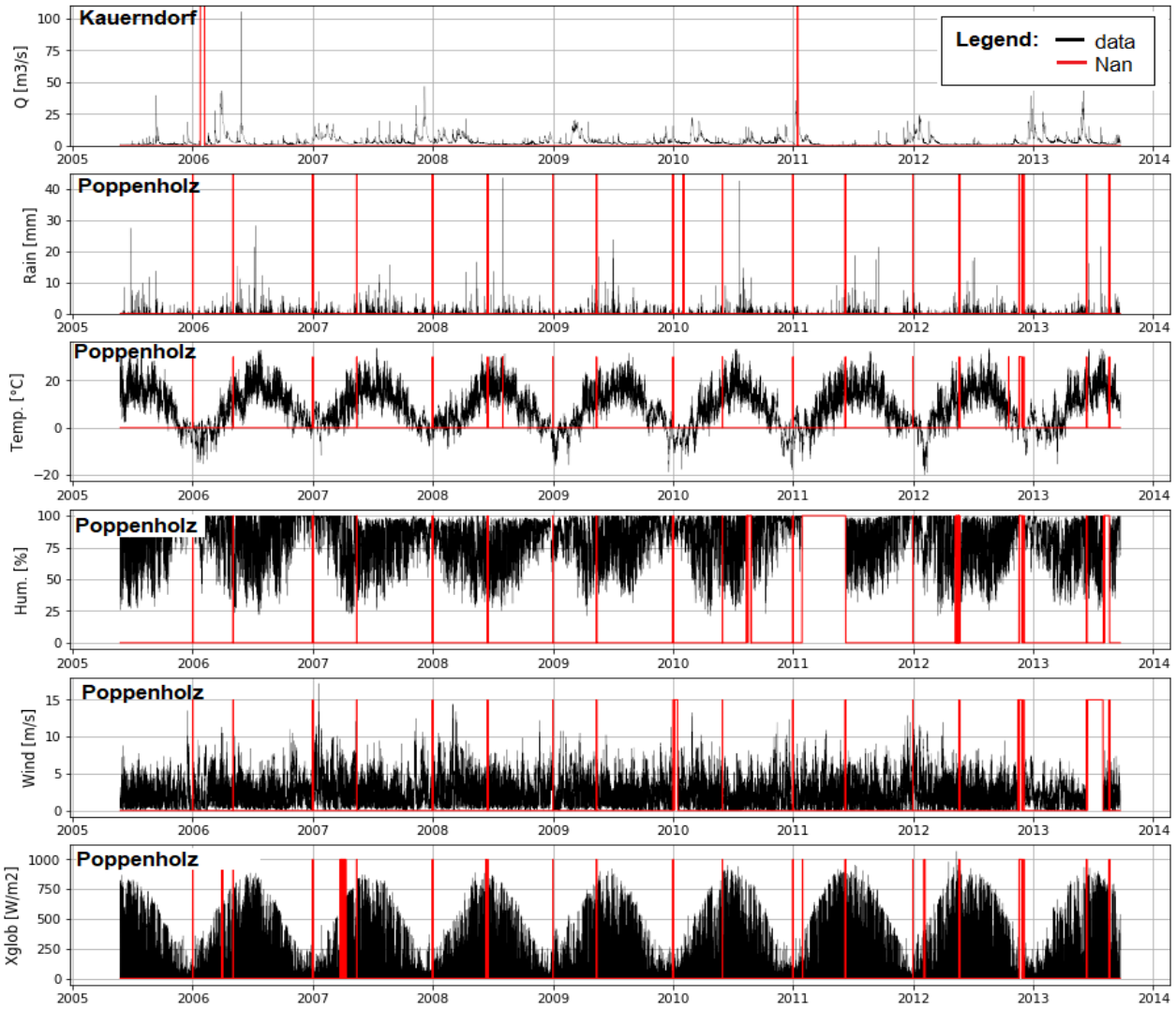

Figure 29: Synchronised hourly time series for the runoff gauge of Kauerndorf and the station of Poppenholz in Schorgast for the period 2005-2013. The missing values are marked with red lines.

The results of the R^2 correlation analysis of the time series used in the models are shown in Figure 30 a. for Schorgast and b. for UWM. For Schorgast the highest correlated inputs are the flow rates, with R^2 values between 0.74 and 0.94, while the rainfalls show only a moderate correlation between 0.14 and 0.48. Some low correlation exists between global radiation, temperature and humidity.

For UWM the same considerations are valid but it is possible to observe a very high correlation between the wind data, the temperatures and the total radiations of different stations. The humidity in the two different gauges has a surprisingly low correlation, while in general the correlation between the rain gauges is lower compared to Schorgast. Also the flow rates present a lower correlation between each other.

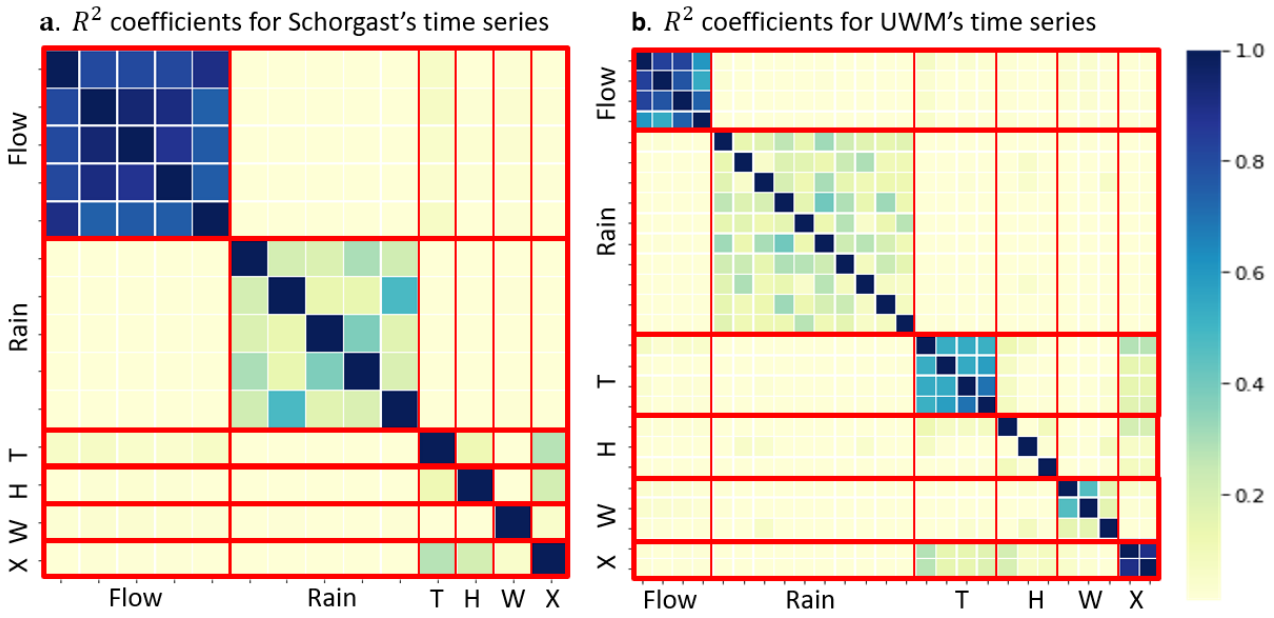

Figure 30: R2 correlation coefficient for the full time-series of Schorgast from 2005 to 2013.

The boxplots in Figure 31 shows the characteristics of the different training, validation, and test splits for the flow at the target gauges of the three study catchments.

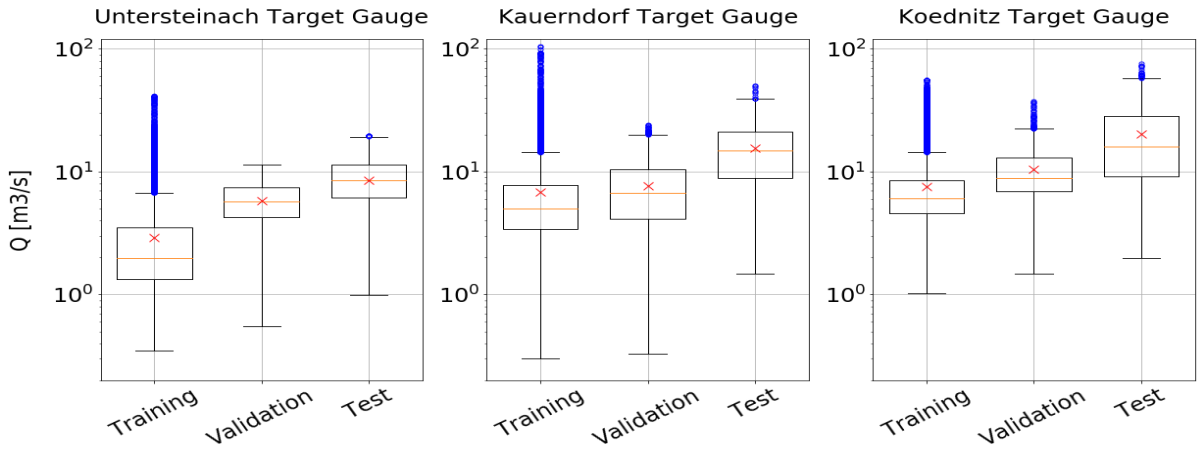

Figure 31:Boxplot of the flow of the target gauges of Untersteinach, Kauerndorf (for Schorgast), and Ködnitz (for U. Weisser Main). The outliers are marked in blue, the mean as a red x.

It can be observed that in all the three catchments the training set is the one with the lower mean, but also with the highest number of high-flow outlayers. The test set is the one having the highest mean, being composed of two specific high flow events, while the validation has intermediate characteristics. The splits in the Training (blue), Validation(yellow), and Test (green) sets are shown for each study catchment in Figure 32 through the rainfall (red) and target gauge flow time-series.

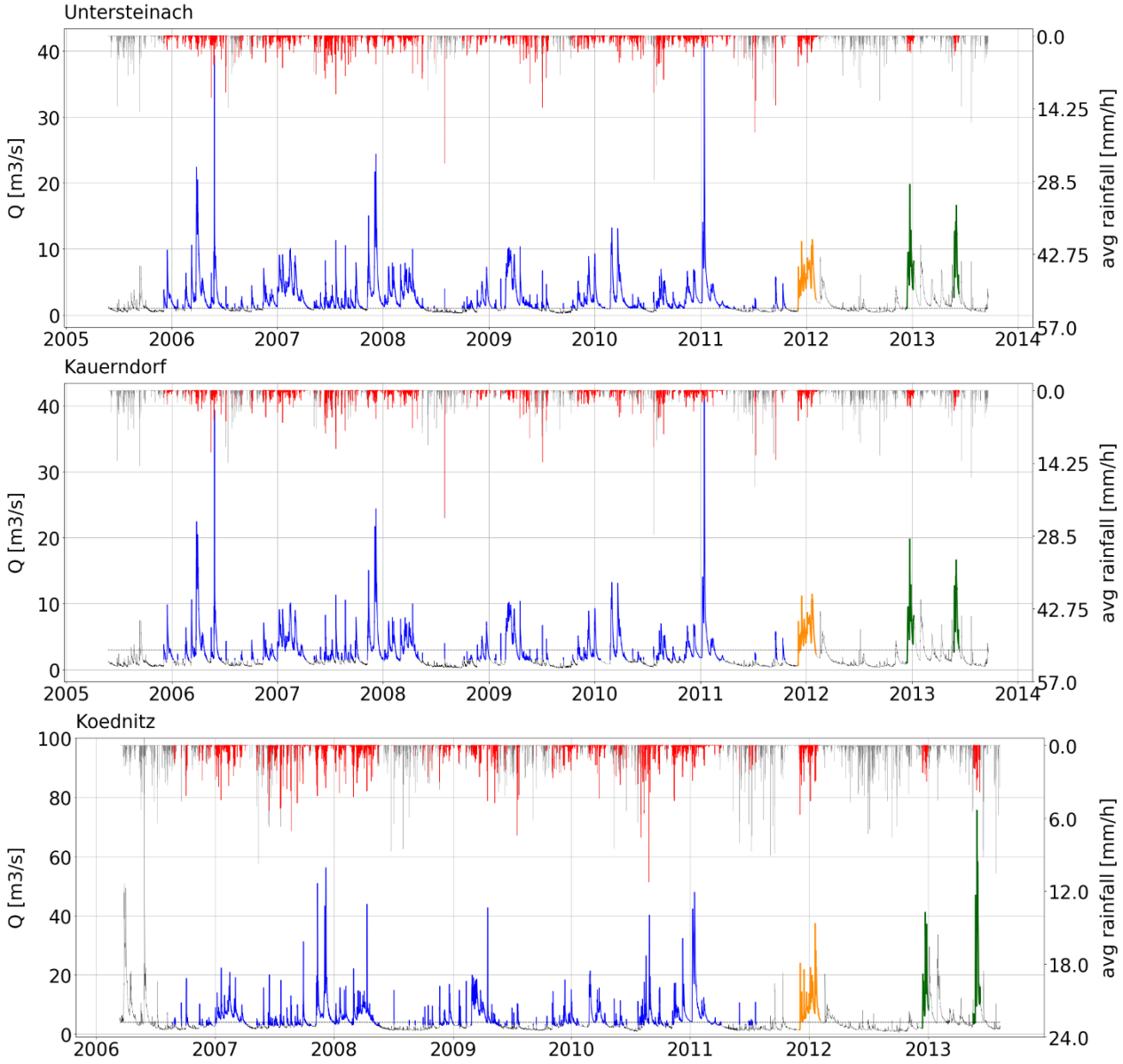

Figure 32: Timeseries of the average rainfall and the flow of the target gauges of the models for the three catchments of Untersteinach, Schorgast (Kauerndorf), Upper Weisser Main (Ködnitz). In blue is marked the training set, in yellow the validation, and in green the two test events.

5.2 Hyperparameters Tuning using Genetic Algorithms (GA)

The GA tuning is performed for 4 generations (0-3) on three hyperparameters of the model of Ködnitz (UWM): timesteps backwards, encoder vector length, decoder vector length using as the fitness function the weighted average of the R-Factor calculated on the validation dataset. The score and the hyperparameters sets are plotted against each other for the different generations in Figure 33, the convergence is shown by plotting the average and the best score for each generation.

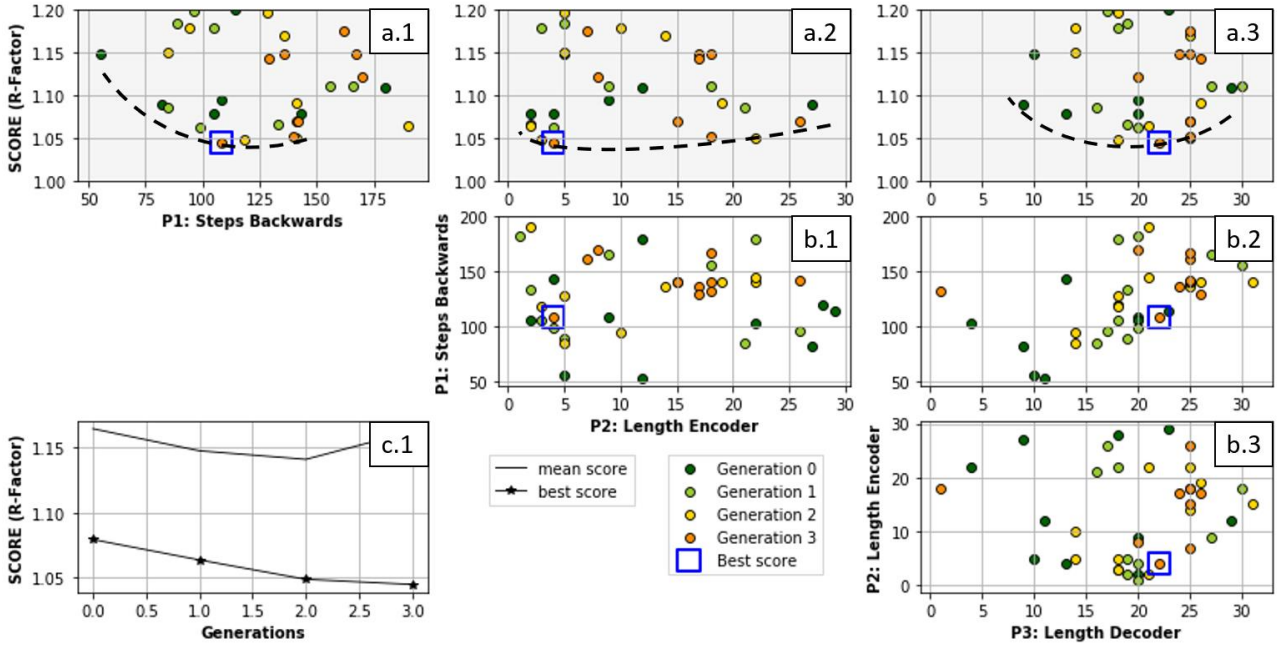

Figure 33: Results of the GA hyperparameter tuning of the Ködnitz model. The score and the parameters selected for each individual foe each generation are plotted against each other in the scatters a.1-3 and b.1-3, while the convergence along the generations of the best and average score is represented in the graph c.1.

In Table 10 the best score overall is noted together with the set of parameters to adopted in the model of Ködnitz after considering the general trend of the results observed in Figure 33.

Table 10: Results of the GA calibration. Best score with the related set of hyperparameters and adopted values.

GA Best Score (weighted average R-Factor 98%)	Hyperparameters	Set of values for the best score	Range of values explored	Set of values adopted
1.044	Timesteps backwards:	108	[50, 200]	110
	Encoder vector length:	4	[1, 30]	7
	Decoder vector length:	22	[1, 30]	20

The adopted set of hyperparameters, found for the model of UWM, is then applied at the other two models. The characteristics of the resulting models are shown in Table 11.

Table 11: Characteristics of the 3 models for Upper Weisser Main, Schorgast, and Untersteinach built with the hyperparameters selected through GA applied to the catchment of UWM.

Model:	Inputs	Trainable parameters			
		LSTM_Encoder	LSTM_Decoder	Time_distributed	TOTAL
Upper W. Main	33	1148	4880	21	6049
Schorgast	25	924	4240	21	5185
Untersteinach	11	532	3120	21	3673

5.3 Features coefficients calibration

The results of the calibration performed by running the trained models of the three catchments on the validation dataset in order to calibrate the two snow/rain parameters (which control the transition between snow and rain precipitation) are shown Figure 34. The two parameters are the upper and lower temperature limits upTlim and dwnTlim, while the score of the model is the MSE of the test and the validation datasets. The default values of the two parameters used to train the models are represented with a black dotted line.

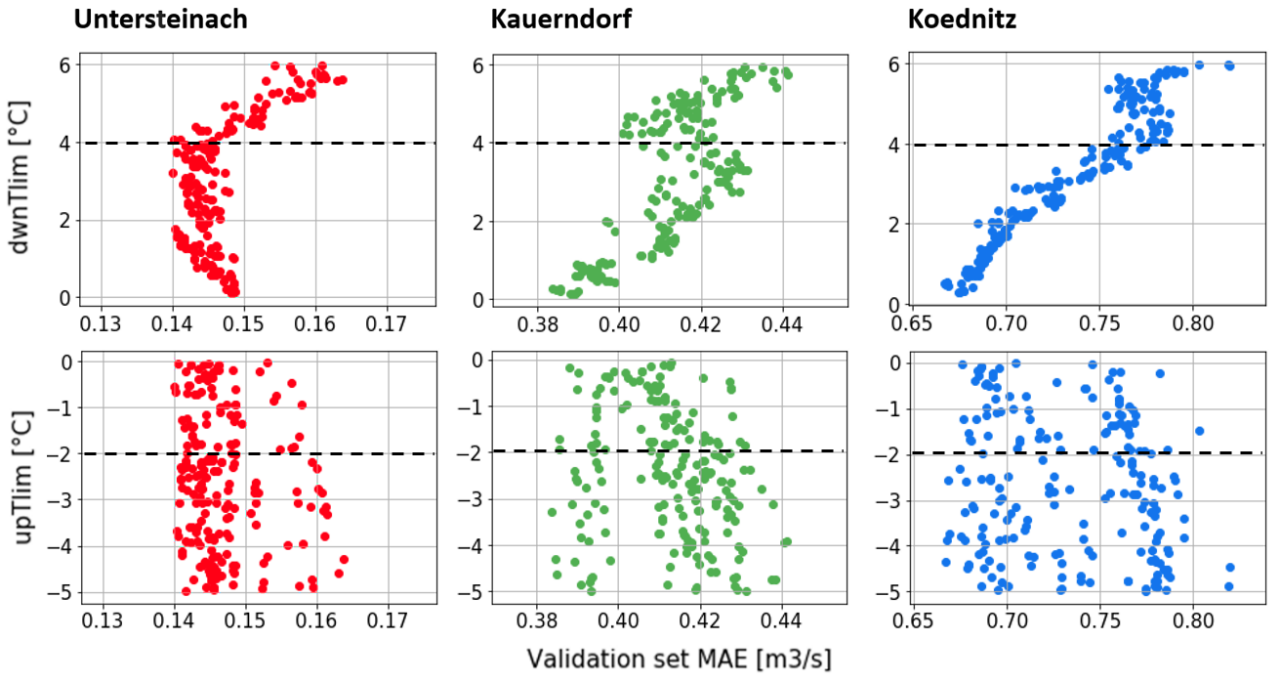

Figure 34: Scatter plot of the snow/rain upper and lower temperature parameters against the score (mean absolute error) obtained for the validation dataset running the trained models on the default parameters. The black dotted lines indicate the default values of upTlim and dwnTlim used in the training of the model.

In all three the models upTlim seems to be a more sensitive threshold, following an accentuated non-linear behaviour, especially regarding Kauerndorf, while dwnTlim hardly show any convergence to an optimal value. The final calibrated values are 3.19, 0.28, and 0.51 °C for upTlim, and -0.54, -3.27, and -4.34°C for dwnTlim for the models of Untersteinach, Kauerndorf and Ködnitz respectively.

Table 12:Default and calibrated values for the snow/rain parameters upTlim and dwnTlim for the three models

Model	upTlim [°C]		dwnTlim [°C]	
	Default	Calibrated	Default	Calibrated
Untersteinach	4	3.19	-2	-0.54
Kauerndorf		0.28		-3.27
Ködnitz		0.51		-4.34

The effect of the two parameters calibration on the three models for the two test events of December 2012 and May 2013 is shown in Figure 35 in % of change of the P and R factors. The graph refers to the 90% uncertainty, but similar results are obtained for 94 and 96%. The calibration brings usually small decrease in P Factor, which are extensively compensated by significant reductions of the R-Factor, meaning smaller uncertainty bands.

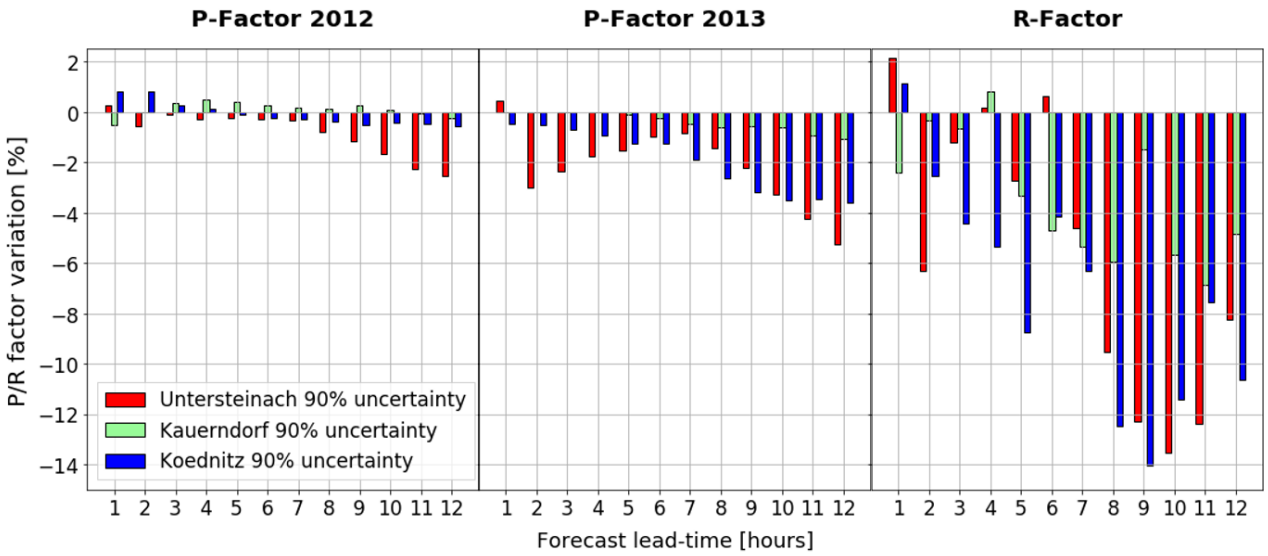

Figure 35: Effect of the snow/rain parameters calibration on the P and R factors of the two test events of December 2012 and May 2013. The effect is measured by the change in % from the values before the calibration. The change in % of the R factor is the same for the two events as it depends only by the change in the width of the uncertainty bands.

5.4 Features Sensitivity Analysis

Figure 36 and Figure 37 shows the results of the individual features sensitivity analysis for the models of Kauerndorf and Untersteinach, and for Ködnitz. The output variation for input variation unit, expressed in term of standard deviations (std Output/ std Inputs), reveals how much each feature is influential in the model.

The results show that almost all the features have an importance in the model. The highest values of response come from dQ, rain, snow, flow and wind, with a score above 0.01 for both validation and test sets for all the model. The less influential feature in Kauerndorf and Ködnitz is the air humidity, which plays an important role in the model of Untersteinach. No feature is judged so uninfluenced to be removed.

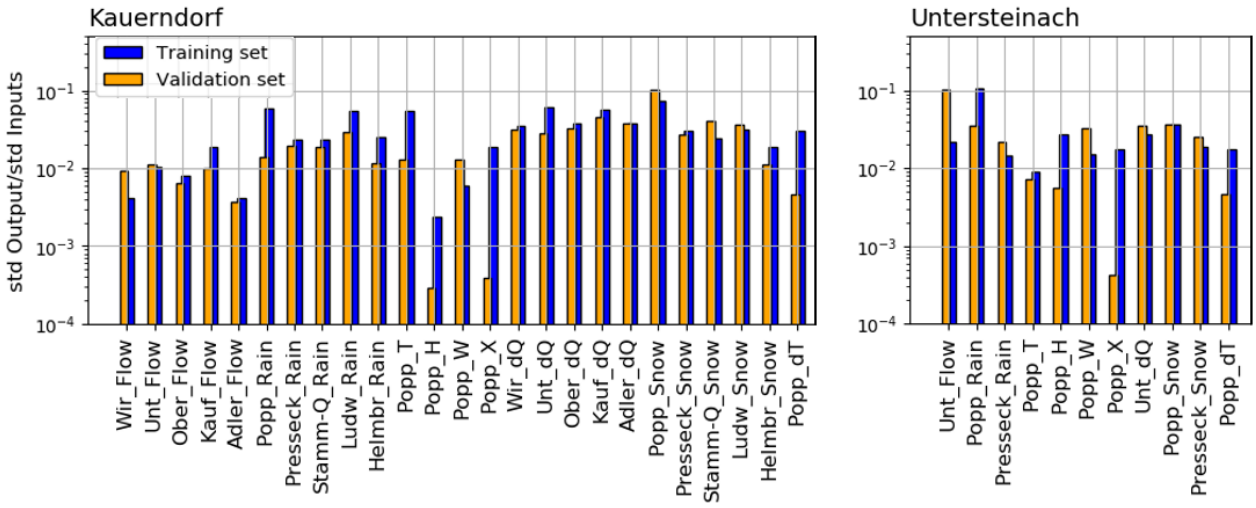

Figure 36: Result of the SA on the single features of the model for the model of Kauerndorf and Untersteinach carried on the training and validation sets. The sensitivity of the parameters is expressed the standard deviation of the output scores per unit of a measure of the standard deviation of the input.

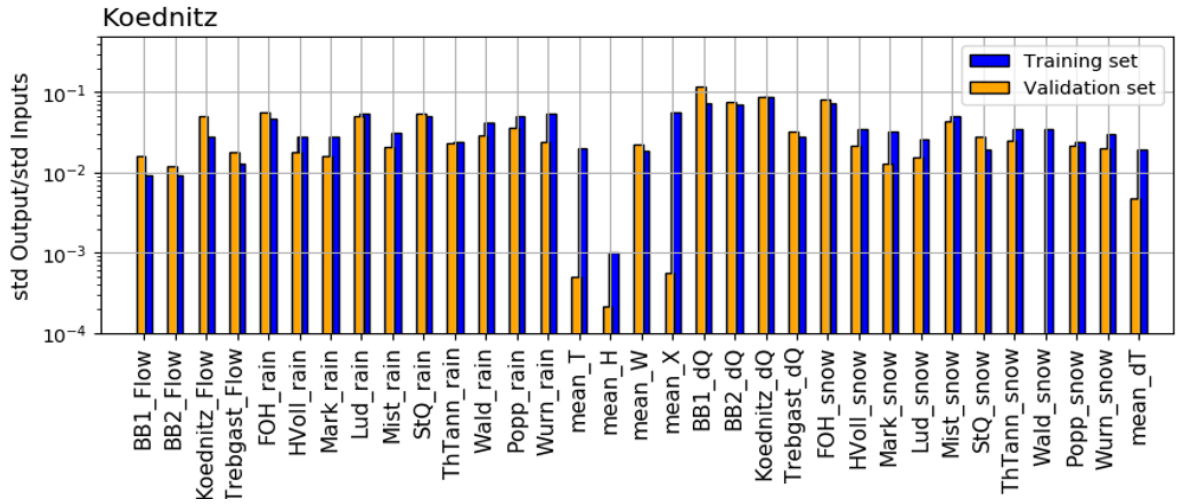

Figure 37: Result of the SA on the single features of the model for the model of Ködnitz carried on the training and validation sets. The sensitivity of the parameters is expressed the standard deviation of the output scores per unit of a measure of the standard deviation of the input.

5.5 Evaluation

Figure 38, Figure 39 and Figure 40 show the results for the uncertainty bands of the three models of Untersteinach, Schorgast, and Upper Weisser Main for the values of 90,94 and 96% equal to the P factor of the validation set.

Untersteinach has the lowest R-factors for the 90%, 94% and 96% of uncertainty (0.5, 0.75, 0.9), while Ködnitz the highest (1.0,1.6,2.2) while Kauerndorf is in between (0.6,0.8,1). The catchment of Untersteinach is the one with the smallest error bands, due to the lower magnitude of the flow at the target gauge, and the difference between the three values of uncertainty is moderate. Ködnitz has

similar widths with a more marked increase in the bands' width for higher values of uncertainty. The bands of Untersteinach and Kauerndorf have a bias for positive errors, especially for the lower uncertainty (90%,94%) while for Ködnitz they are quite symmetric.

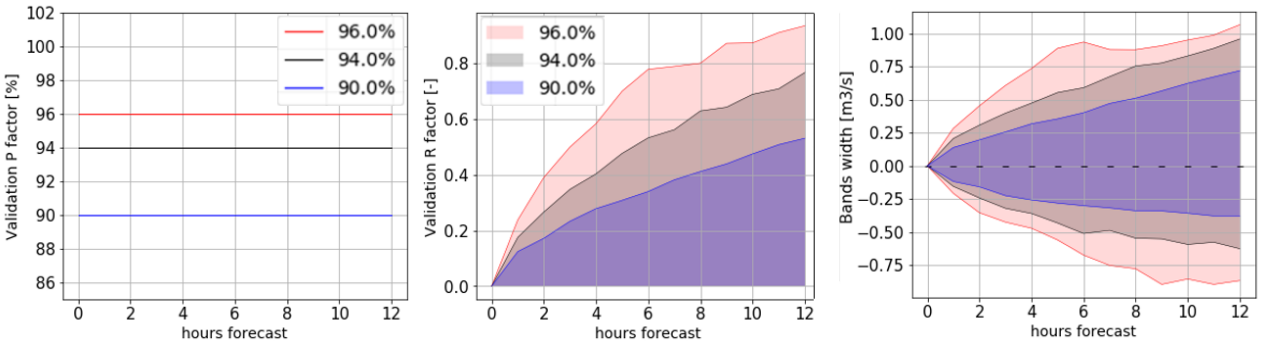

Figure 38: Relative Error Method uncertainty calculation for Untersteinach. The P factor and R factor for the validation set and the uncertainty bands are shown for the values of 90,94 and 96% uncertainty.

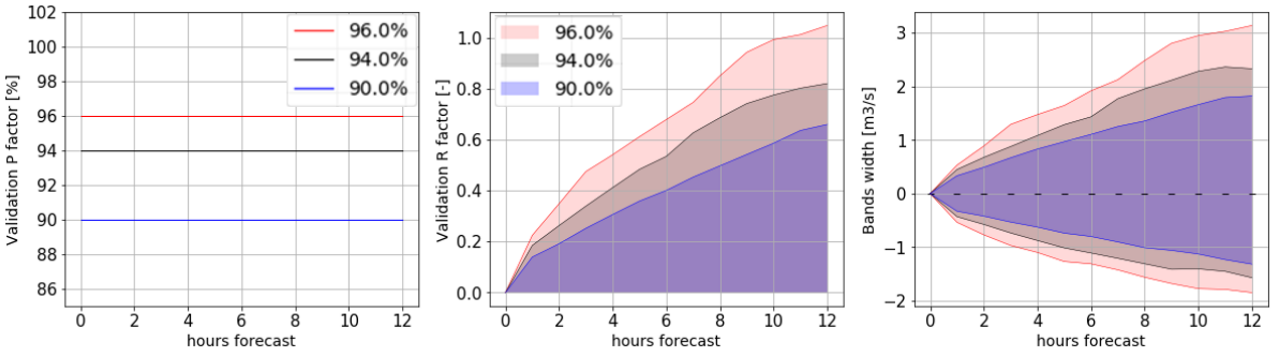

Figure 39: Relative Error Method uncertainty calculation for Schorgast (Kauerndorf). The P factor and R factor for the validation set and the uncertainty bands are shown for the values of 90,94 and 96% uncertainty.

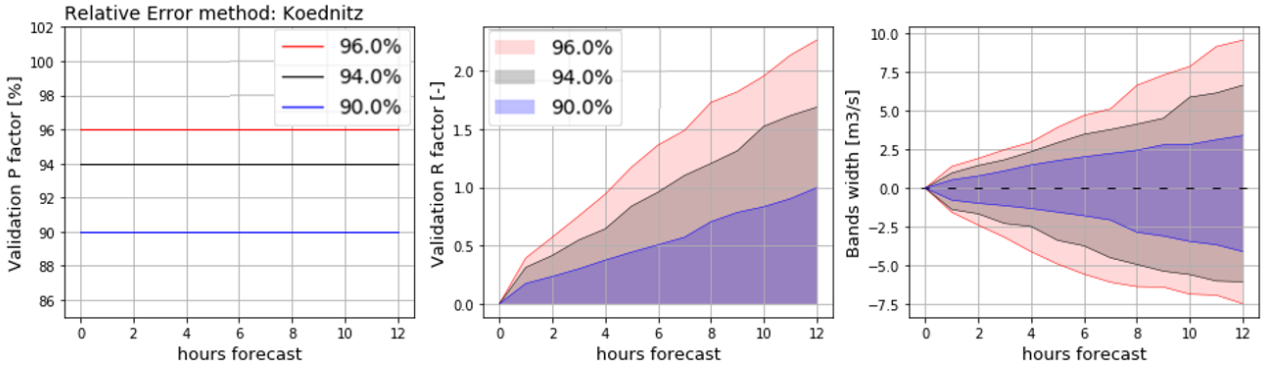

Figure 40: Relative Error Method uncertainty calculation for Upper Weisser Main (Ködnitz). The P factor and R factor for the validation set and the uncertainty bands are shown for the values of 90,94 and 96% uncertainty.

The results of the models for the test dataset is shown in Figure 41, Figure 42 and Figure 43 for the three catchments of Untersteinach (a.1-2), Schorgast (b.1-2) and UWM (c.1-2). The plots show the 90% uncertainty bands. Visually it's possible to say that in all three the catchments the best predictions were obtained for the test event of 2012, while 2013 it's more difficult to predict.

a.1 **a.2**

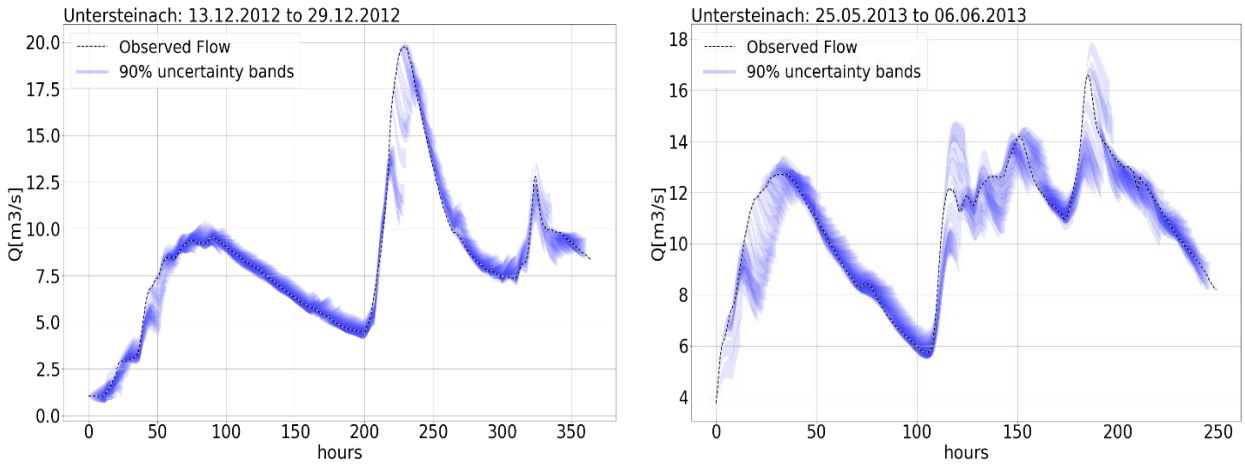

Figure 41: 12 h forecasts with uncertainty bands for the catchment of Untersteinach. a.1 shows the event of December 2012 with 90% uncertainty bands, while a.2 shows the event of May 2013 with 90% uncertainty bands.

b.1 **b.2**

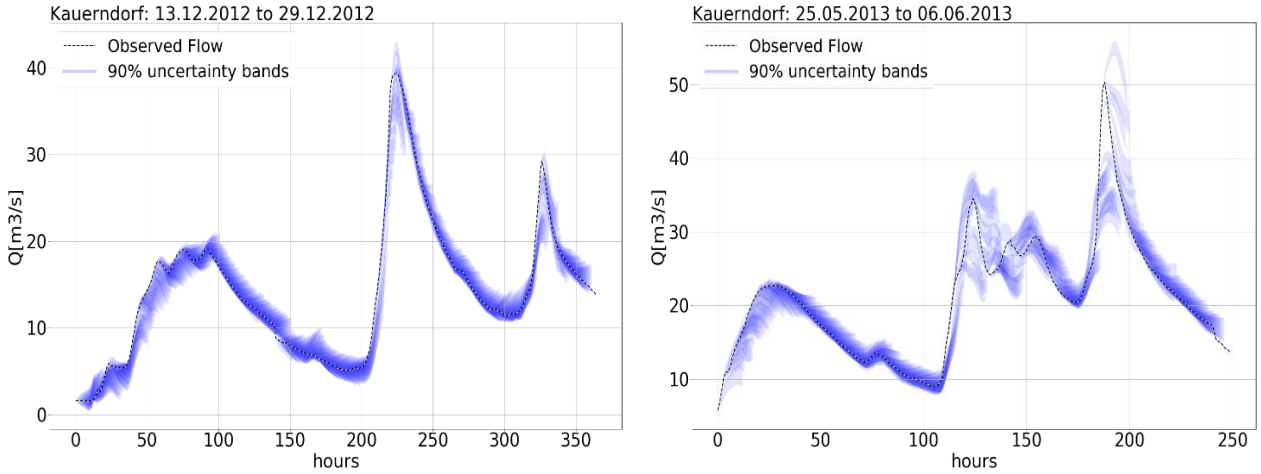

Figure 42: 12 h forecasts with uncertainty bands for the catchment of Schorgast (gauge of Kauerndorf). b.1 shows the event of December 2012 with 90% uncertainty bands, while b.2 shows the event of May 2013 with 90% uncertainty bands

c.1 **c.2**

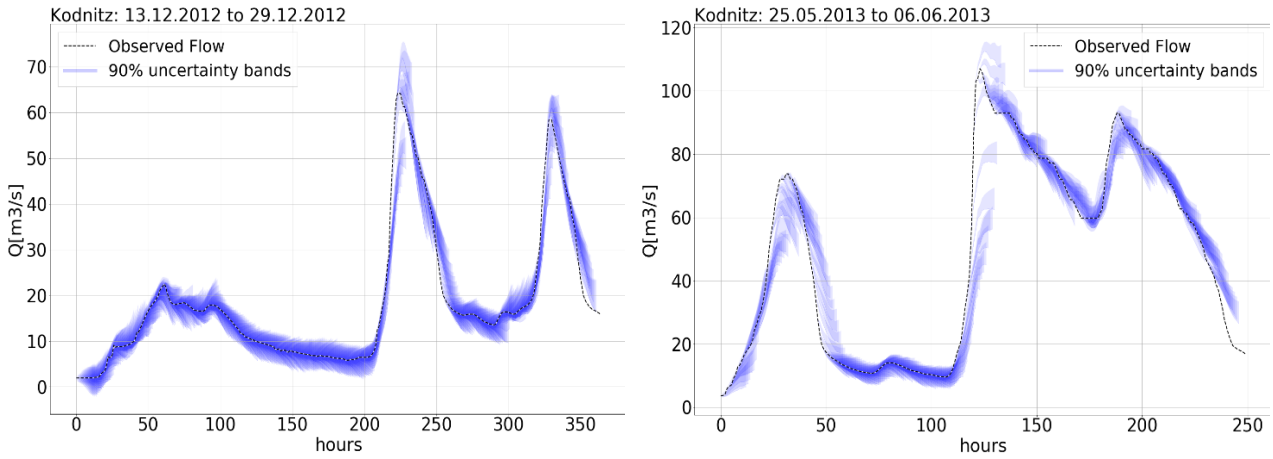

Figure 43: 12 h forecasts with uncertainty bands for the catchment of Upper Weisser Main (Ködnitz). c.1 shows the event of December 2012 with 90% uncertainty bands, while c.2 shows the event of May 2013 with 90% uncertainty bands.

The quantitative evaluation of the models' performances for the different study catchments is shown in Figure 44. The P and R factors calculated for the two test events of December 2012 and May 2013 are shown for 90, 94 and 96% uncertainty.

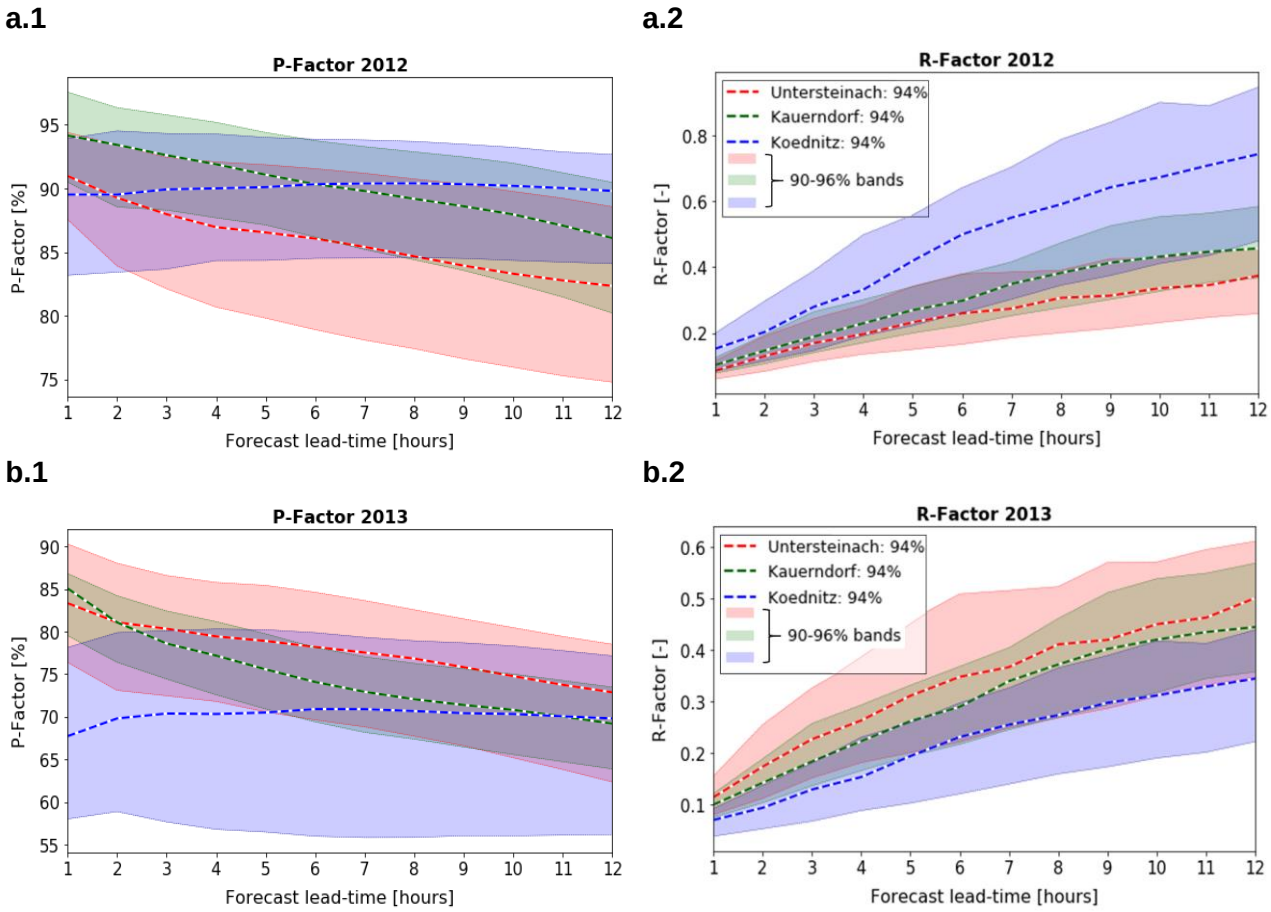

Figure 44: Comparison of hourly P and R factor between the results in the three catchments with 90, 94, 96% uncertainty bands. The plots a.1-2 refer to the test event of December 2012, b.1-2 to the one of May 2013.

In 2012 high P-Factors are obtained for all the catchments, ranging between 75% and 96% and R factors ranging from 0.2 to 0.9 at 12h. In 2013 all the models performed worst, with similar R factors but P-Factors between 60% and 80% at 12h.

In 2012 the models that performed better are Untersteinach and Kauerndorf, with the higher P factors overall and smaller R factors than Ködnitz. For the first 5 hours Kauerndorf performs better with lower R factors for similar P factor. The situation inverts for the last 7 hours when Untersteinach outperforms Kauerndorf.

In 2013 the P-factors for the same uncertainty bands are more similar, while the model of Ködnitz is the one having the smallest R-factor. With the lower R-factor and the similar values of P factors, especially in the last hours of forecasts, Ködnitz seems to outperform the other two models. In the

last 7 hours of the forecast, the second better model is Untersteinach, while the performance is similar to Kauerndorf for the first 5 hours.

In the following Figure 45, the P and R factor obtained from this thesis LSTM Seq2seq model for the gauge of Ködnitz (UWM) are plotted on the results obtained for the same catchment in Gander (2018) and Leandro et al. (2019) using the calibrated LARSIM hydrological model with 5 different uncertainty calculation methods. The more direct comparison is the one with the relative error method (90% uncertainty for the LARSIM). The comparison is on the two test events of December 2012 (a.1-2) and May 2013 (b.1-2).

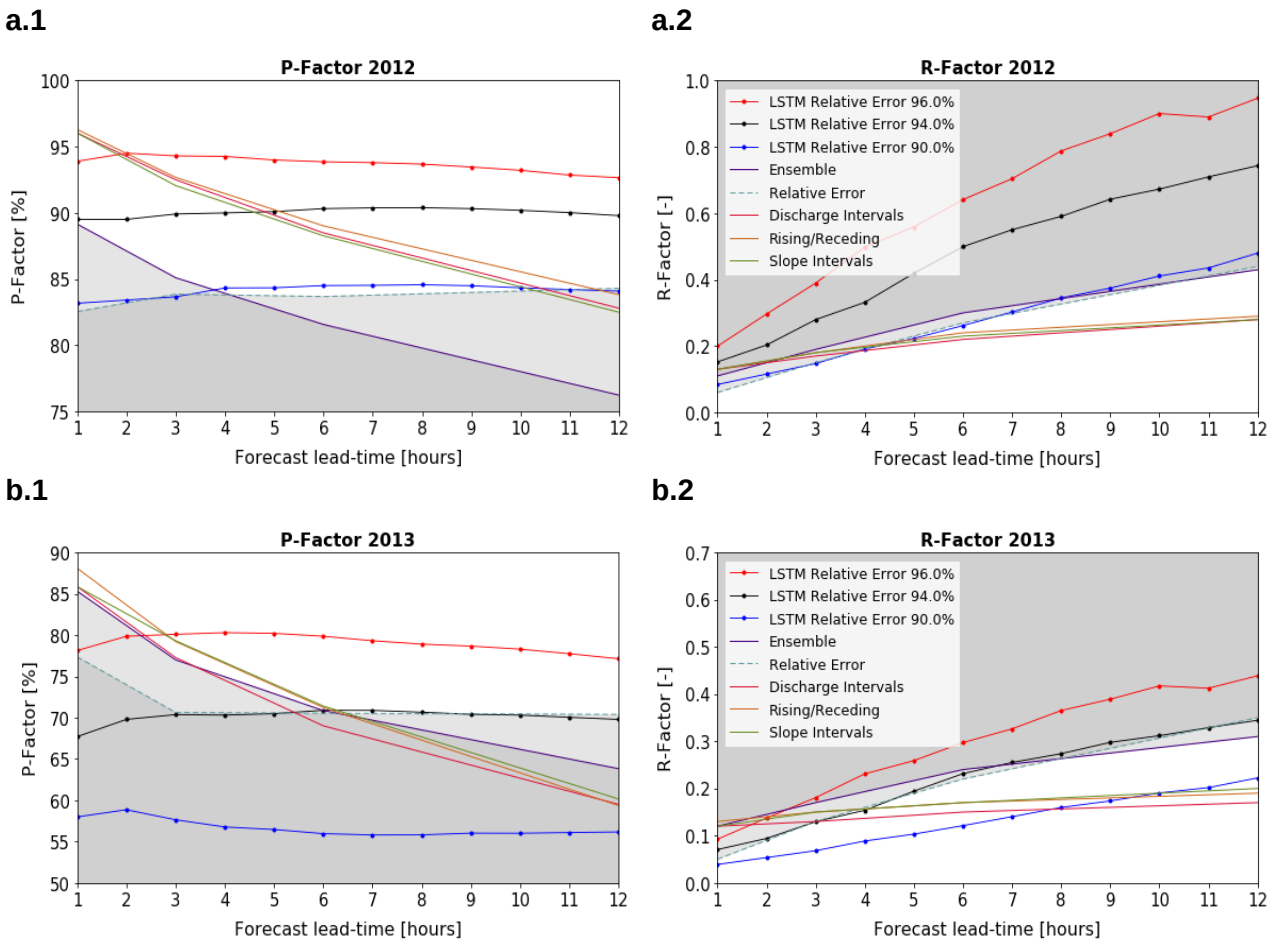

Figure 45: Comparison of hourly P and R factor between the results of the hydrological model LARSIM with different uncertainty estimation techniques (Gander 2018) and the LSTM model with 90, 94, 96% uncertainty bands. The plots a.1-2 refer to the test event of December 2012, b.1-2 to the one of May 2013.

During the event of 2012, the LSTM Seq2seq model using 90% relative error method and the LARSIM relative error have an almost identical P and R factor. The LSTM Seq2seq model has higher P factor from the 4th to the 9th hour of forecasts, while LARSIM has a better R factor with the same P factor for the last 2 hours of the forecast. The discharge interval, rising/receding and slope

intervals methods seem to outperform the LSTM Seq2seq model with the 90% relative error method due to the considerably lower R factor, while for 94 and 96% the comparison it's impossible.

During the event in 2013, the LSTM Seq2seq with the relative error methods for the 94% bands and the LARSIM relative error score almost identical P and R factors for all the 12 hours. The discharge interval, rising/receding and slope intervals methods have a very descending curve for the P factor, but they obtain very low values of R factor.

In general both the LSTM Seq2seq and the LARSIM model, regardless of the uncertainty method used, have better prediction during the event of 2012, with P factors around 85% and R factors between 0.2 and 0.4 after 12 h (excluding 94% and 96% relative error of the LSTM Seq2seq model). In 2013 the R factors are lower, between 0.15 and 0.3, because of the higher flow rate, but the P factors are lower as well, between 73% and 60% after 12 h (excluding 96% relative error of the LSTM Seq2seq model).

The forecast of Ködnitz plotted without uncertainty for the LSTM Seq2seq model and the LARSIM Ensemble run for the test events of 2012 and 2013 are shown in Figure 46 and Figure 47 respectively.

The forecasts in 2012 are in general smoother than in 2013, and it's difficult to individuate clear biases of the two models.

In 2013 both models seem to have greater difficulties. The rise of the first peak is quite difficult to model for LARSIM, while the LSTM Seq2seq model performs better. The following decrease in flow is underestimated in the same way by both. Both models underestimate the rise of the highest peak, even though the LSTM Seq2seq model seems to be better.

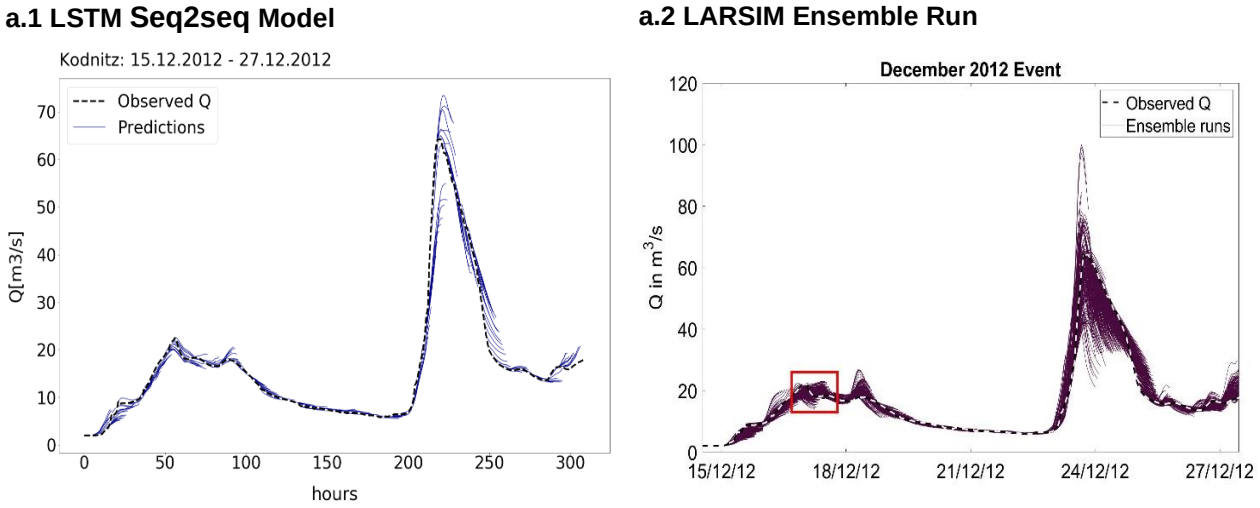

Figure 46: Flow forecasts for the event of December 2012 at the gauge of Ködnitz of the LSTM Seq2seq model without uncertainty bands (a.1) and the LARSIM Ensemble runs (a.2).

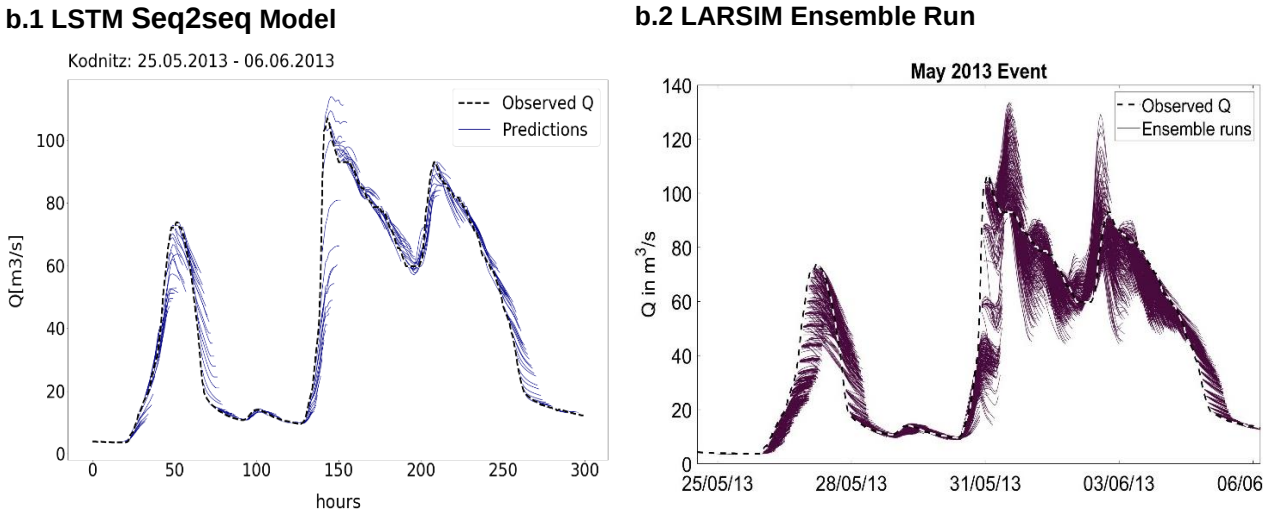

Figure 47:Flow forecasts for the event of May 2013 at the gauge of Ködnitz of the LSTM Seq2seq model without uncertainty bands (b.1) and the LARSIM Ensemble runs (b.2).

6. Discussion

6.1 Data Pre-processing

From Figure 29 it's possible to observe as empty values are quite common in the available data of the area. The example is a good representation of the situation. The flow data are usually the ones having the best quality, while the other time-series are more problematic.

In order to obtain good inputs data and having better performances, in a ML model it's extremely important to deal with the missing data in the best way possible. In Figure 30, the R^2 correlation coefficients between all the inputs (and output) series of Schorgast and UWM are shown. The

existence of some correlation, together with other considerations such as the continuity or irregularity of the data typology, helps to individuate the best way to fill the missing values and gives important information about the most important features in the model (the ones best correlated with the output target gauge) even though also non correlated features may be important when combined with others (Guyon and Elisseeff 2003).

The filling of those empty values can be more or less effective, depending on the predictability of the series, but no matter the method, it will add some noise to the data, so empty values should just be avoided if possible. In this case, due to the long multi-timesteps structure of the samples needed for the LSTM Seq2seq model, it's impossible to exclude all the nan values from the data. Still, the number of filled nans used in the LSTM Seq2seq models is strongly limited by the exclusion of periods where too many data are missing.

The boxplots in Figure 31 shows the distributions of the samples in the three splits of the training, validation, and test datasets. For all three the catchments the same trend is observed. The training database is the one with the lower mean but higher number of extreme values. That comes from the necessity of including as much data as possible that can be useful in the training, including some lower flow values that are no more interesting in the validation or test dataset as their aim is to verify the ability of the NN to model high flow peaks. The test set is always the one with the highest values in general, being formed by two well-selected events. The validation needs to be a tradeoff between containing a certain amount of significant flow events and leaving the more extreme runoff values for quality training or the test.

Even though ideally the balance between the three datasets should be perfect, when dealing with the modelling of extreme events it's necessary to satisfy the primary requirements of the training (number and quality of samples) and the test (significant test events), and the validation set can be seen as an intermediate between the two in terms of number of samples and extreme values.

Figure 32 shows also how the selection based on the flow threshold operated on the training set has shrunk considerably the number of samples. That is done in order to avoid too many low flow data, that could push the model to make "flat" predictions in order to achieve a low error during the training. It's important also to remember that the aim of this work is flood prediction, so the high runoff events are the ones that must weight the most for the model.

It's interesting to notice also that the test and validation sets are small compared to the size of the training. That is justified as in the specific case the quality of the samples is more important than the quantity. As already shown in the methodology in the Evaluation section in Figure 25, Figure 26, Figure 27 and Figure 28, the test database is composed of two well-selected extreme events having

different and important characteristics. The Validation database is also selected during a whole period of high flow, containing several peaks.

6.2 Hyperparameters Tuning using Genetic Algorithms (GA)

The results of the GA tuning for the hyperparameters gives some interesting hints about the model. Firstly, as can be seen in Figure 33, the algorithm has only a partial convergence of the best score of each generation and not the average. The scatter plots a.1, a.2, a.3 shows that there's no direct relation between the score and the single individual hyperparameters, suggesting a strong interdependence between them. The scatters b.1, b.2, b.3 of each hyperparameter against the other show some accumulations of individuals around certain values, that seems to be the ones often performing the best score.

Table 10 shows the set of hyperparameters related to the best score. The final set of parameters adopted is approximated by looking at the scatter to closeby values reaching similar scores. That is done as the dimension of only 4 elements for the encoder LSTM vector length appeared to be too extreme. This choice is balanced by using the value of 20 instead of 22 for the decoder vector length. The number of step backwards is only rounded to 110 from 108.

The convergence of the algorithm may be opposed by the several random elements in the fitness function (the NN) such as the initialization of weights, optimization algorithm, batches creation. Anyway, the advantage of the GA is that it can give reasonable results also without converging completely. The use of the algorithm was overall a success, as with a limited number of runs (40) was possible to gather enough information to decide a set of hyperparameter, even though a better-defined convergence to a local optimum would have been preferred.

The steps backwards found are 110, meaning 4 days and 14 hours. Physically it can be interpreted as the medium-term retention time of the catchment, even though obviously longer-term phenomena exist which could be captured in some "long-memory" features, such as the past flow values, even though it's only a speculation.

The small encoder vector length forces the model to compress all the information in a small encoded sequence. This bottleneck effect may have a regularization effect, helping to avoid overfitting later in the decoder. The relatively bigger decoder vector length could be favourable in order to better fit the inputs and the encoded sequence converging faster.

The application of the same hyperparameters to the different catchments' models is based on the assumption of the similarity of the data and areas. Thought, Table 11 shows how the same set of structural hyperparameters applied to the three different models brings to a difference in the number

of trainable parameters and hence in the dimension of the model. This is due to the number of input features.

6.3 Features coefficients calibration

Figure 34 shows the results of the snow/rain transition coefficients calibration for the validation set. The upTlim parameter follows a well-defined c or s-shaped behaviour. Regarding the dwnTlim, the coefficient struggles to converge, with very spread values.

There are two possible causes. The first is that the samples containing snow values are fewer than the ones containing rain. That would lead to a lower general impact in the prediction, while it could be in reality extremely important for some particular runoff events. The second is that the model may interpret the snow as a feature that doesn't bring significant increases in runoff, especially with lower temperatures. That would lead to minor importance between certain values of the lower limit.

In Table 12 it's possible to observe that for all three the catchments the default valuest for upTlim and dwnTlim of 4 and -2°C changes after the calibration. The upper limit upTlim, in general, tends to decrease. In Kauerndorf and Ködnitz, it becomes less than 1°C, while it stabilises around 3°C in Unternsteinach. The lower limit grows to -0.54°C in Untersteinach, while it decreases to -3/4°C for Ködnitz and Kauerndorf. The differences could be explained by the elevation and exposition of the temperature sensor more than by the topographic differences in the catchments (neighbours and quite similar).

From Figure 35 can be observed that the calibration of the snow/rain transition parameters has a good impact on the model, as the uncertainty of the model decrease a lot compared to the moderate decrease in P factor for the test events. That means the performances of the models are enhanced.

6.4 Features Sensitivity Analysis

The results of the SA on the single features shown in Figure 36 and Figure 37, bring several interesting conclusions. The first one is that the additional parameters derived during the data pre-processing result to be even more influential for the model than the original features: dT and dQ have a higher score than the relatives T and flow, while both the snow and the rain have a similar impact on the model. It's quite surprising the relatively low impact of the temperature, while the wind speed has an unexpected high influence. The air humidity and total radiation have the lowest scores regarding the validation dataset but have a much higher impact on the training dataset. In the smallest model of Untersteinach, also the lowest-impact features appear to be quite influent, as probably the model is forced to exploit that information better due to the low number of inputs. Therefore none of the features is removed in any model.

6.4 Evaluation

The validation R factor and the uncertainty bands for the three models, shown in Figure 38, Figure 39 and Figure 40, show that the error distributions for the models can be quite different. The differences in the width of the different uncertainties (90,94,96%) for the bands and the R factors shows how the errors are distributed. For Kauerndorf the bands have a small expansion when incrementing the uncertainty. That means the error distribution has a low variance. In the case of Untersteinach and Ködnitz, it's easy to see how the bands' width expands faster denoting a higher variance of the errors.

Only Ködnitz has relatively symmetric upper and lower bands, while Untersteinach and Kauerndorf have a wider higher band, meaning the models tends to underestimate more than overestimate as can be visually noticed in the plots in Figure 41 a.1 for the test event of 2012 and a.2 for the test event of 2013 for Untersteinach. In the plots can be seen that the main peak of 2012 and the first round peak of 2013 are systematically underestimated. In Figure 42 it can be noticed how for the model of Kauerndorf only the first peak of 2013 (b.2) is always underestimated, while the 2012 event (b.2) is quite well balanced.

Regarding Ködnitz, Figure 43 b.1 and b.2 show how the model tends to overestimate the highest peaks in the test event of 2012 and underestimate the rise of the highest peak in 2013. Those differences in predictions are quite difficult to explain, and depends on the characteristics of the actual runoff phenomena in the different catchments, on the rainfall distribution, and in the training of the different models.

Among the three catchments, the best results would have been expected for the model of Kauerndorf, as the Schorgast catchment is the one with the best coverture of gauges. Untersteinach, even though is a sub-catchment of Schorgast, have a smaller size, and that could make it more sensitive to very local conditions. Ködnitz should be also critical, as almost no measurement stations, except for the river gauges and 1 rain station, are placed inside the Upper Weisser Main catchment.

The results in Figure 44 a.1 and a.2 for the test event of December 2012 seems to confirm the assumption, with Kauerndorf and its sub-catchment Untersteinach having the best P factor for similar values of R factors. Still, for the event of 2013, shown in Figure 44 b.1 and b.2, the situation is completely twisted, with Ködnitz scoring the best P factors at lower R factors.

Figure 45 shows that, when the results of the LSTM Seq2seq model of Ködnitz are confronted with the results simulated in LARSIM, it appears that generally, the performances are really similar where close values of P or R factors allow a meaningful comparison and the uncertainty is based on the same method (relative error).

It's important to notice that the uncertainty of the LSTM Seq2seq model is based on relative error method only, and the fairest comparison is where the LARSIM uncertainty is also calculated in the same way. When more elaborated uncertainty estimation methods are used for LARSIM (slope, discharge intervals, rising/receding), its scores may outperform the LSTM Seq2seq model based on the simpler relative error. Anyway, that only proves that applying an elaborate method for the uncertainty improves the P and R factors of a model.

It is still surprising that for both models the highest P factors were obtained for the event in December 2012, where temperature, snowfalls, accumulation and melting dynamics were playing a key role, while the event in 2013 was rain-only driven.

One explanation is that the data for the event in 2013 didn't capture the rainfall distribution on the area, while in 2012 they did. That would explain why apparently both the LSTM Seq2seq and the conceptual model can model better "complicated" phenomena.

Figure 46 and Figure 47 shows how the forecasts of the LARSIM and LSTM Seq2seq models behave in the two test events of 2012 and 2013 respectively.

It's interesting to see how the two models mostly show the same biases, such as in underestimating the rise of the highest peak of the 2013 event. That means the relationships between the parameters the neural network built during the training are similar to the equation LARSIM is built on.

Still, the LSTM Seq2seq is not "copying" LARSIM, and it's sometimes modelling aspects of the flow the conceptual model misses. For example, during the rise of the first peak in the event of May 2013, where the neural network forecast can mostly follow the increase of discharge, LARSIM goes "flat" and can't forecast it properly.

Because the performances of the LSTM Seq2seq and LARSIM models for Ködnitz are comparable, it's possible to assume that also the results of Kauerndorf are of the same quality. That means that now two reliable neural network models exist for the two catchments of Schorgast and Upper Weisser Main and are able to forecast extreme runoff events that may cause floods in the downstream city of Kulmbach.

6.5 Advantages and Limitations

Conceptual and hydrodynamics models need to be mathematically built and well-structured in a way to include the description of all the main phenomena which could influence the catchment. Therefore they require both a deep knowledge of the physical phenomena and the geomorphology of the area.

Completely opposite is the situation for a flexible data-driven model such as NNs. The prior knowledge of the dynamics can be helpful, but it's not necessary, and adding additional data is extremely fast and easy due to the unstructured nature of the model. The same Seq2seq model

used in this work to predict the flow rate could be easily retrained using different data to produce 12 timesteps of forecasts of the desired output for any problem where the inputs are someway related to the target output.

The biggest limitation of the NN model is the availability of long time-series for training and validating the model. Long term trends caused by morphologic or anthropic changes to the stream system of the area, or simply a change in the method for the measurement can also be a problem for training a NN, as the relationships between inputs and outputs may suddenly change. On the other hand, ANNs require no data at all about the morphology and structure of the catchments, which can be difficult and expensive to obtain.

Another limitation of the ANN model is its black-box nature, which makes very difficult the interpretation of the internal mechanism. A considerable amount of work has already been done in the machine learning and data mining communities on describing how the black boxes work and explaining the decisions those models take (Guidotti et al. 2018). In the field of hydrology, the gain of an understanding of the internal mechanisms of the data-driven hydrological models is an important element of the development of the model itself which hasn't been sufficiently afforded by researchers and consequently the criticisms associated with "black-box" data-driven models persist and limits the application and spread of those methodologies. (Mount, Dawson, and Abrahart 2013).

7. Conclusion

To date ANN LSTM models are among the most advanced tools for flood forecasting based on ML techniques. Still, all the examples in the literature are only able to produce a single interval forecast per one trained model, while the Seq2seq LSTM architecture used in this work is able to predict a whole sequence of 12 hours of forecasts.

The model, based on an encoder-decoder structure with recurrent LSTM cells, was first built and tested for the small test sub-catchment and later adapted for the two main catchments upstream of the city of Kulmbach, in the Upper Main Basin.

The hourly time-series used for the predictions are between 7 and 8 years long, depending on the catchment. The data are filtered by eliminating low-flow periods and then split into three datasets: training, validation and test. The first one is used for training the weights of the model. The second is for the uncertainty and the tuning of hyperparameters and features-related coefficients. The third set is composed of two historical extreme events in December 2012 and May 2013 and is used only for the final evaluation of the model's performance.

The data used for the predictions are measurements of precipitation, flow, temperature, air humidity, wind speed, and global radiation. From those inputs, additional features are derived. A later calibration and sensitivity analysis reveals how the derived features are as important as the originals in the model.

The dimension of the encoder and decoder and the number of timesteps backwards are the structural hyperparameters which determine the dimension of the model itself. The optimal values are found using a genetic optimization algorithm in order to automatise the process and limit the computational effort.

The final models performed well for all three catchments for the two test events of 2012 and 2013. Thus, they demonstrate the ability to successfully model medium and long-term winter dynamics, as well as short-term rainfall-runoff ones, in areas of different size and with optimal or sub-optimal distribution of the gauging stations.

In the catchment of Upper Weisser Main, the LSTM Seq2seq model performed similarly to the conceptual model LARSIM used in previous studies of the catchment. The scores for both the LSTM Seq2seq model and LARSIM are lower for the test event of May 2013 than for that of December 2012 This is most likely because the data given by the gauges can't properly describe the dynamics of the period, hence the performance of both the models is limited more by the available data than by their structure and mathematical formulation.

While the Seq2seq model doesn't show significant improvements in performance compared to the conceptual model, its strength is that it doesn’t need any prior knowledge about the catchment and it’s easy and fast to set up and use. Therefore, it can be a valid alternative to the conceptual model in areas where collecting geo-morphological data is difficult or expensive.

8. Outlook

This work tried to explore all the main tasks of a well-developed ML application for flood forecasting using neural networks and it represents a possible starting point for a wide series of developments.

For example, the performances could increase by applying advanced features engineering to the inputs in order to further avoid overfitting and to produce a more compact and useful set of features.

Structurally, the NN could be upgraded in different ways. The snow/rain transition coefficients could be directly integrated into the NN as trainable parameters through a special layer. Alternative network architectures for time-series forecasting, such as Causal Convolutional Dilated NN could speed up the training of the model. This would allow for less computationally expensive high-level methods for advance tuning of the hyperparameter or feature analysis, which could result in better forecasts. There’s also a variety of methods for including uncertainty directly in the predictions, such as using a Bayesian neural network.

The flexibility of the ML structures makes them optimal for the use of additional inputs such as satellite data images, used in combination with the traditional gauge measurements, could give a more complete description of the parameters’ distribution. Future developments could even include the use of a hybrid model composed of conceptual and data-driven modules that could combine the strength and advantages of both approaches.

Finally, one of the most interesting future research prospects would be to attempt to “crack” the black-box of the NN model. That would mean shifting the focus of the NN model from its external performance to the internal mechanisms through the development and application of different methodologies (Guidotti et al. 2018), such as brute-force, global and local relative sensitivity analyses (Mount, Dawson, and Abrahart 2013).

References

Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, et al. 2016. “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems,” March. http://arxiv.org/abs/1603.04467.

Adamowski, Jan, Hiu Fung Chan, Shiwu O. Prasher, and Vishwa Nath Sharda. 2012. “(20) (PDF) Comparison of Multivariate Adaptive Regression Splines with Coupled Wavelet Transform Artificial Neural Networks for Runoff Forecasting in Himalayan Micro-Watersheds with Limited Data | Jan Adamowski - Academia.Edu.” *Journal of Hydroinformatics* 14.3: 731–44. https://www.academia.edu/15181471/Comparison_of_multivariate_adaptive_regression_splines_with_coupled_wavelet_transform_artificial_neural_networks_for_runoff_forecasting_in_Himalayan_micro-watersheds_with_limited_data.

Adnan, Rana. 2017. “Streamflow Forecasting Using Artificial Neural Network and Support Vector Machine Models.” *American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS)* 29 (1): 286–94. https://www.academia.edu/35016514/Streamflow_Forecasting_Using_Artificial_Neural_Network_and_Support_Vector_Machine_Models.

Al-Rfou, Rami, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, et al. 2016. “Theano: A Python Framework for Fast Computation of Mathematical Expressions.” https://groups.google.com/group/theano-dev/.

Alfieri, L, P Burek, L Feyen, and G Forzieri. 2015. “Global Warming Increases the Frequency of River Floods in Europe.” *Hydrology and Earth System Sciences* 19 (5): 2247–60. https://doi.org/10.5194/hess-19-2247-2015.

Alom, Md Zahangir, Tarek M. Taha, Chris Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Mahmudul Hasan, Brian C. Van Essen, Abdul A. S. Awwal, and Vijayan K. Asari. 2019. “A State-of-the-Art Survey on Deep Learning Theory and Architectures.” *Electronics* 8 (3): 292. https://doi.org/10.3390/electronics8030292.

Anaconda Software Distribution. 2019. “Anaconda.” https://anaconda.com.

Bavarian Environmental Agency. n.d. “GKD Bayern.” Accessed April 10, 2019. https://www.gkd.bayern.de/en/.

Beg, Md Nazmul Azim, Jorge Leandro, Punit Bhola, Iris Konnerth, Kanwal Amin, Florian Koeck, Rita F. Carvalho, and Markus Disse. 2018. “Flood Forecasting with Uncertainty Using a Fully Automated Flood Model Chain: A Case Study for the City of Kulmbach” 3 (September): 207–196. https://doi.org/10.29007/jb27.

Bhatt, C M, G S Rao, P G Diwakar, and V K Dadhwal. 2017. “Development of Flood Inundation Extent Libraries over a Range of Potential Flood Levels: A Practical Framework for Quick

Flood Response.” *Geomatics, Natural Hazards and Risk* 8 (2): 384–401. https://doi.org/10.1080/19475705.2016.1220025.

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

Bouktif, Salah, Ali Fiaz, Ali Ouni, and Mohamed Adel Serhani. 2018. “Optimal Deep Learning LSTM Model for Electric Load Forecasting Using Feature Selection and Genetic Algorithm: Comparison with Machine Learning Approaches.” *Energies* 11 (7). https://doi.org/10.3390/en11071636.

Campolo, M, A Soldati, and P Andreussi. 2003. “Artificial Neural Network Approach to Flood Forecasting in the River Arno.” *Hydrological Sciences Journal* 48 (3): 381–98. https://doi.org/10.1623/hysj.48.3.381.45286.

Chollet, F. 2015. “GitHub - Keras-Team/Keras: Deep Learning for Humans.” 2015. https://github.com/keras-team/keras.

Devia, Gayathri K., and B.P. Ganasri. 2015. “A Review on Hydrological Models.” *Aquatic Procedia* 4 (January): 1001–7. https://doi.org/10.1016/J.AQPRO.2015.02.126.

Disse, Markus, Iris Konnerth, Punit Kumar Bhola, and Jorge Leandro. 2018. “Unsicherheitsabschätzung Für Die Berechnung von Dynamischen Überschwemmungskarten – Fallstudie Kulmbach.” In *Vorsorgender Und Nachsorgender Hochwasserschutz*, 350–57. Wiesbaden: Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-21839-3_50.

Gander, Armin. 2018. “Entwicklung Und Vergleich von Methoden Zur Hochwasservorhersage Mittels Des.” Technischen Universität München.

Guidotti, Riccardo, Anna Monreale, Salvatore Ruggieri, Franco Turini, Dino Pedreschi, and Fosca Giannotti. 2018. “A Survey Of Methods For Explaining Black Box Models,” February. http://arxiv.org/abs/1802.01933.

Guyon, Isabelle, and André Elisseeff. 2003. “An Introduction to Variable and Feature Selection.” *Journal of Machine Learning Research* 3 (April): 1157–82. https://doi.org/10.1162/153244303322753616.

Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. “Long Short-Term Memory.” *Neural Computation* 9 (8): 1735–80. https://doi.org/10.1162/neco.1997.9.8.1735.

Hu, Caihong, Qiang Wu, Hui Li, Shengqi Jian, Nan Li, and Zhengzheng Lou. 2018. “Deep Learning with a Long Short-Term Memory Networks Approach for Rainfall-Runoff Simulation.” *Water (Switzerland)* 10 (11): 1–16. https://doi.org/10.3390/w10111543.

Hunter, John D. 2007. “Matplotlib: A 2D Graphics Environment.” *Computing in Science &*

Engineering 9 (3): 90–95. https://doi.org/10.1109/MCSE.2007.55.

Jia, Yangqing, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. “Caffe: Convolutional Architecture for Fast Feature Embedding,” June. http://arxiv.org/abs/1408.5093.

Jouppi, Norman P, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, et al. 2017. “In-Datacenter Performance Analysis of a Tensor Processing Unit.” In *Proceedings - International Symposium on Computer Architecture*, Part F1286:1–12. https://doi.org/10.1145/3079856.3080246.

Kim, G, and A P Barros. 2001. “Quantitative Flood Forecasting Using Multisensor Data and Neural Networks.” *Journal of Hydrology* 246 (1–4): 45–62. https://doi.org/10.1016/S0022-1694(01)00353-5.

Kingma, Diederik P., and Jimmy Ba. 2014. “Adam: A Method for Stochastic Optimization,” December. http://arxiv.org/abs/1412.6980.

Kluyver, Thomas, Benjamin Ragan-kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, et al. 2016. “Jupyter Notebooks—a Publishing Format for Reproducible Computational Workflows.” *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, 87–90. https://doi.org/10.3233/978-1-61499-649-1-87.

Kratzert, Frederik, Daniel Klotz, Claire Brenner, Karsten Schulz, and Mathew Herrnegger. 2018. “Rainfall-Runoff Modelling Using Long Short-Term Memory (LSTM) Networks.” *Hydrology and Earth System Sciences* 22 (11): 6005–22. https://doi.org/10.5194/hess-22-6005-2018.

Kundzewicz, Z W, N Lugeri, R Dankers, Y Hirabayashi, P Döll, I Pińskwar, T Dysarz, S Hochrainer, and P Matczak. 2010. “Assessing River Flood Risk and Adaptation in Europe-Review of Projections for the Future.” *Mitigation and Adaptation Strategies for Global Change* 15 (7): 641–56. https://doi.org/10.1007/s11027-010-9213-6.

Le, Xuan Hien, Hung Viet Ho, Giha Lee, and Sungho Jung. 2019. “Application of Long Short-Term Memory (LSTM) Neural Network for Flood Forecasting.” *Water (Switzerland)* 11 (7). https://doi.org/10.3390/w11071387.

Leandro, J.;, I. Konnerth, P. Bhola, K. Amin, F. Köck, and M. Disse. 2017. “Floodevac Interface Zur Hochwassersimulation Mit Integrierten Unsicherheitsabschätzungen.” In *Der Hydrologie, Fachgemeinschaft Hydrologische*, 185–92. Wissenschaften in der DWA Geschäftsstelle: Trier, Germany.

Leandro, J., A. Gander, M. N.A. Beg, P. Bhola, I. Konnerth, W. Willems, R. Carvalho, and M. Disse. 2019. “Forecasting Upper and Lower Uncertainty Bands of River Flood Discharges with High Predictive Skill.” *Journal of Hydrology* 576 (June): 749–63. https://doi.org/10.1016/j.jhydrol.2019.06.052.

Lee, June-Goo, Sanghoon Jun, Young-Won Cho, Hyunna Lee, Guk Bae Kim, Joon Beom Seo, and Namkug Kim. 2017. “Deep Learning in Medical Imaging: General Overview.” *Korean Journal of Radiology* 18 (4): 570. https://doi.org/10.3348/kjr.2017.18.4.570.

Ludwig, K., and M Bremicker. 2006. *The Water Balance Model LARSIM: Design, Content and Applications*. http://www.larsim.de/fileadmin/files/Dokumentation/FSH-Bd22-Bremicker-Ludwig.pdf.

Mckinney, Wes. 2010. “Data Structures for Statistical Computing in Python.” *PROC. OF THE 9th PYTHON IN SCIENCE CONF*. https://conference.scipy.org/proceedings/scipy2010/pdfs/mckinney.pdf.

Mosavi, Amir, Pinar Ozturk, and Kwok-wing Chau. 2018. “Flood Prediction Using Machine Learning Models: Literature Review.” *Water* 10 (11): 1536. https://doi.org/10.3390/w10111536.

Moulder, Stuart, Tish Sheridan, Pietro Cavallo, and Giuseppe Rossini. 2017. “Deep Learning with MATLAB and Multiple GPUs.” https://www.mathworks.com/content/dam/mathworks/tag-team/Objects/d/Deep_Learning_in_Cloud_Whitepaper.pdf.

Mount, N. J., C. W. Dawson, and R. J. Abrahart. 2013. “Legitimising Data-Driven Models: Exemplification of a New Data-Driven Mechanistic Modelling Framework.” *Hydrology and Earth System Sciences* 17 (7): 2827–43. https://doi.org/10.5194/hess-17-2827-2013.

Nourani, Vahid, Aida Hosseini Baghanam, Jan Adamowski, and Ozgur Kisi. 2014. “Applications of Hybrid Wavelet–Artificial Intelligence Models in Hydrology: A Review.” *Journal of Hydrology* 514 (June): 358–77. https://doi.org/10.1016/j.jhydrol.2014.03.057.

Paszke, Adam, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. “Automatic Differentiation in PyTorch.” https://openreview.net/forum?id=BJJsrmfCZ.

Pechlivanidis, I G, B M Jackson, N R Mcintyre, and H S Wheater. 2011. “Catchment Scale Hydrological Modelling: A Review of Model Types, Calibration Approaches and Uncertainty Analysis Methods in the Context of Recent Developments in Technology and Applications.” *Global Nest Journal*. https://pdfs.semanticscholar.org/a419/53b14383e793c87bce4457c4d3f13bee0c51.pdf.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, et al. 2011. “Scikit-Learn: Machine Learning in Python.” *Journal of Machine Learning Research* 12 (Oct): 2825–30. http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html.

Prucha, Bob, Douglas Graham, Marie Watson, Marinda Avenant, Surina Esterhuyse, Alison Joubert, Marthie Kemp, et al. 2016. “MIKE-SHE Integrated Groundwater and Surface Water

Model Used to Simulate Scenario Hydrology for Input to DRIFT-ARID: The Mokolo River Case Study." *Online) = Water SA* 42 (3). https://doi.org/10.4314/wsa.v42i3.03.

Sadeghyan, Saman. 2018. "A New Robust Feature Selection Method Using Variance-Based Sensitivity Analysis." https://arxiv-org.proxy.unimib.it/pdf/1804.05092.pdf%0Ahttp://arxiv.org/abs/1804.05092.

Seide, Frank, and Amit Agarwal. 2016. "CNTK." In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, 2135–2135. New York, New York, USA: ACM Press. https://doi.org/10.1145/2939672.2945397.

Singh, Aparajita. 2018. "A Concise Review on Introduction to Hydrological Models." *GRD Journals-Global Research and Development Journal for Engineering* | 3 (10). www.grdjournals.com.

Sit, Muhammed, and Ibrahim Demir. 2019. "Decentralized Flood Forecasting Using Deep Neural Networks." https://arxiv.org/pdf/1902.02308.pdf.

Solomatine, D., L.M. See, and R.J. Abrahart. 2008. "Data-Driven Modelling: Concepts, Approaches and Experiences." In *Practical Hydroinformatics*, 17–30. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-79881-1_2.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. 2014. "Sequence to Sequence Learning with Neural Networks." In *Advances in Neural Information Processing Systems*, 4:3104–12. https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

UNISDR. 2014. "Stormwater Management Handbook." *International Journal of Engineering and Technology* 2 (479): 44. https://doi.org/10.1596/978-0-8213-8866-2.

Unruh, Amy. 2017. "What Is TensorFlow? | Opensource.Com." 2017. https://opensource.com/article/17/11/intro-tensorflow.

Us Epa. 2012. "Storm Water Management Model (SWMM)." 2012. https://www.epa.gov/water-research/storm-water-management-model-swmm.

USACE. 2000. "HEC-HMS." 2000. https://www.hec.usace.army.mil/software/hec-hms/.

USACE. 2016. "HEC-RAS River Analysis System - Hydraulic Reference Manual, Version 5.0." Davis, CA, USA. https://doi.org/CPD-68.

Walt, Stéfan Van Der, S Chris Colbert, and Gaël Varoquaux. 2011. "The NumPy Array: A Structure for Efficient Numerical Computation." *Computing in Science and Engineering* 13 (2): 22–30. https://doi.org/10.1109/MCSE.2011.37.

Waskom, Michael, Olga Botvinnik, Paul Hobson, John B. Cole, Yaroslav Halchenko, Stephan Hoyer, Alistair Miles, et al. 2014. "Seaborn." https://doi.org/10.5281/ZENODO.12710.

Xu, Jian, Wei Luo, and Ying Huang. 2019. "Dadu River Runoff Forecasting via Seq2seq." In , 494–98. Association for Computing Machinery (ACM). https://doi.org/10.1145/3349341.3349457.

Young, Tom, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. 2018. “Recent Trends in Deep Learning Based Natural Language Processing [Review Article].” *IEEE Computational Intelligence Magazine* 13 (3): 55–75. https://doi.org/10.1109/MCI.2018.2840738.

Zhao, Zhong-Qiu, Peng Zheng, Shou-Tao Xu, and Xindong Wu. 2019. “Object Detection with Deep Learning: A Review.” *ArXiv:1807.05511* v2. https://arxiv.org/pdf/1807.05511.pdf.