Performance of Different Machine Learning Algorithms in Rainfall Runoff Modelling Compared to the Conceptual Model PREVAH in Selected Swiss Catchments

Master Thesis

Benjamin Meyer

Institute of Geography

University of Bern

Supervisors

Prof. Dr. Bettina Schaefli

Dr. Pascal Horton

# Abstract

Table of Contents

[Abstract 1](#__RefHeading___Toc1127_620400557)

[Introduction 6](#__RefHeading___Toc1224_620400557)

[Research Questions 6](#__RefHeading___Toc1226_620400557)

[Theoretical Background 7](#__RefHeading___Toc1323_620400557)

[The History of Machine Learning 7](#__RefHeading___Toc1325_620400557)

[What is Machine Learning 7](#__RefHeading___Toc1325_620400557_Kopie_1)

[Advantages and Disadvantages of Machine Learning 7](#__RefHeading___Toc1012_2975168869)

[Hyperparameter Optimization 8](#__RefHeading___Toc1022_2975168869)

[Manual Experience Based Optimization 8](#__RefHeading___Toc1024_2975168869)

[Grid Search Optimization 9](#__RefHeading___Toc1026_2975168869)

[Random Search Optimization 9](#__RefHeading___Toc1028_2975168869)

[Gradient Based Optimization 9](#__RefHeading___Toc1030_2975168869)

[Bayesian Parameter Optimization 9](#__RefHeading___Toc1141_2523509422)

[Multi Fidelity Optimization 9](#__RefHeading___Toc1143_2523509422)

[Meta Heuristic Optimization 10](#__RefHeading___Toc1145_2523509422)

[Machine Learning Models 11](#__RefHeading___Toc1147_2523509422)

[XGBoost 11](#__RefHeading___Toc1331_620400557)

[LightGBM 11](#__RefHeading___Toc1333_620400557)

[Support Vector Regression (SVR) 12](#__RefHeading___Toc1341_620400557_Kopie_1)

[Multilayer Perceptron (MLP) 12](#__RefHeading___Toc1335_620400557)

[Long short-term memory (LSTM) 13](#__RefHeading___Toc1337_620400557)

[Gated Recurrent Units (GRU) 15](#__RefHeading___Toc1339_620400557)

[Activation Functions 15](#__RefHeading___Toc1341_620400557)

[Rectified Linear Unit (ReLU) 15](#__RefHeading___Toc1507_2136232950)

[Soft Plus 16](#__RefHeading___Toc1509_2136232950)

[Sigmoid 16](#__RefHeading___Toc1511_2136232950)

[Tan-h 16](#__RefHeading___Toc1513_2136232950)

[Softmax 16](#__RefHeading___Toc1515_2136232950)

[Methods and Data 17](#__RefHeading___Toc1343_620400557)

[Research area (Switzerland) 17](#__RefHeading___Toc1345_620400557)

[Data 17](#__RefHeading___Toc1347_620400557)

[Data Preparation 18](#__RefHeading___Toc1349_620400557)

[Methods 19](#__RefHeading___Toc1351_620400557)

[PREVAH 19](#__RefHeading___Toc1353_620400557)

[XGBoost 20](#__RefHeading___Toc1355_620400557)

[LightGBM 21](#__RefHeading___Toc1355_620400557_Kopie_1)

[Multilayer Perceptron (MLP) 22](#__RefHeading___Toc1355_620400557_Kopie_1_Kopie_1)

[Long short-term memory (LSTM) 23](#__RefHeading___Toc1357_620400557)

[Gated Recurrent Units (GRU) 25](#__RefHeading___Toc1355_620400557_Kopie_1_Kopie_2)

[Calibration and Validation 25](#__RefHeading___Toc1359_620400557)

[Results 27](#__RefHeading___Toc1361_620400557)

[Discussion 28](#__RefHeading___Toc1363_620400557)

[Conclusion 29](#__RefHeading___Toc1365_620400557)

[Literature 30](#__RefHeading___Toc1367_620400557)

[Apendix 34](#__RefHeading___Toc1369_620400557)

List of Figures

[Abbildung 1: Bucket cascade of the conceptual model PREVAH 15](#Abbildung!0|sequence)

List of Tables

Tabelle 1: Catchments used in this study 13

# Introduction

Our physical and economic well-being depends on the availability of water. More than half of the SDGs are, in one way or another, related to water resources. Sustainable management of water resources is a necessity to ensure everyone's access to water in sufficient quality and quantity, without harming the environment. But as important as water is, it is also a potential source of danger. Flooding is just one example of how water can cause damage, either directly or indirectly. The basis for effective resource management is good data. Knowing where, when and how much water is available is fundamental. One of the key environmental variables for water resource management is runoff. While measurement is the foundation, it can only provide current and historical data. To gain insight into future conditions and make predictions, some form of modelling is required. That's where Rainfall Runoff Models (RRM) come in.   
Countless models have been developed over the past decades for different purposes, each with trade-offs in terms of degree of realism, data requirements, computational power needed and ease of use (Beven, 2011). RRMs can be classified into three types: physics-based models that attempt to model the physical processes, conceptual models that are strongly simplified representations of the real world, and finally data-driven models such as standard statistical models and, more recently, machine learning models (Sitterson et al., 2019). In this study, six machine learning algorithms (XGBoost, LightGBM, SVM, MLP, LSTM, GRU) are compared with the process-oriented hydrological model "Precipitation-Runoff-Evapotranspiration HRU Model PREVAH" (Viviroli et al., 2009) and their potential in modelling runoff in Swiss catchments.   
Catchments in Switzerland cover a wide range of main runoff generating characteristics, such as glaciated high mountain catchments, snow dominated alpine catchments, highly karstified catchments and lowland catchments. This wide range of dominant factors poses a challenge to models, resulting in different model performance depending on the dominant process.

While it has been shown that LSTMs can outperform conceptual models in RRM (Lees et al., 2021; Li et al., 2021; Kratzert et a., 2018), no study has yet been published on Swiss catchments.

# Research Questions

As mentioned above, there is a research gap on the performance of RRM machine learning models in Swiss catchments. This study thus aims to answer the following questions:

How good is the performance of the six machine learning models XGBoost, LightGBM, SVM, MLP, LSTM, GRU compared to the conceptual model PREVAH using data from 1985 – 2016 (Muelchi et al. 2020).

How do different catchment characteristics influence the performance of the machine learning models in Swiss catchments?

Does an extended data period (1981 – 2020) influence the performance of the machine learning models?

(How do the simulation results using the CH2018 data look compared to the result of Mülchi et al. (2020).)

# Theoretical Background

## The History of Machine Learning

Machine learning (ML), an umbrella term for a wide range of computational methods, has its roots in the mid-20th century. In its modern sense, it was first introduced by the psychologist Frank Rosenblatt in 1957. He and his group built a machine called "perceptron" to recognise letters, based on the idea of how nervous systems work (Fradkov 2020). A big step towards modern algorithms was the invention of back-propagation in the mid-1980s. Using the chain rule, the derivative of the loss function for all internal parameters is calculated. With a method called gradient decent, the internal parameter are then updated to minimize the loss function (Fradkov 2020). The next major breakthrough came in 1995, when Cortes & Vapnik introduced a version of the support vector machine algorithm for the general non-separable case (Cortes & Vapnik 1995).

Fradkov (2020) identified three synergistic trends that have led to the success story of ML in the 21st century. The first is big data. The amount of data became so large that new ways of analysing it became necessary. The second was the cost reduction of parallel computing. And finally, the third trend was the proliferation of new algorithms. As a result, ML has found applications in a wide range of fields, such as healthcare, natural language processing, advertising or finance, to name just a few. Machine learning is increasingly becoming an important tool for businesses and researchers (Çelik 2018).

## What is Machine Learning

Let's take a cooking recipe as a simple illustrative example. Traditional algorithms take an input and perform a defined set of actions to produce an output. The traditional algorithm takes the ingredients (inputs), washes and chops them and combines them in a predefined way (actions) and as a result we get a meal (output). In many cases this is sufficient. But suppose you don't like the recipe because it's too spicy. You would have to change the algorithm itself to change the result. A machine learning algorithm, on the other hand, does not have a predefined recipe. It can mix and match the ingredients freely. Each time the machine learning algorithm cooks for you, you can evaluate the result. The algorithm then adjusts the recipe according to your rating, and eventually you get the meal you like without having to adjust the algorithm itself.

Machine learning algorithms adjust internal parameters to achieve the desired result. They improve on their own errors and try to minimise the number of errors they make (Alzubi et al. 2018). More specifically, they build mathematical models based on so-called training data to make predictions or decisions without being specifically programmed to do so (Bishop, 2006 in Khanzode & Sarode, 2020).

Alternativer Vorschlag (so als Basis, aknn immernoch angepasst und ergänzt werden):

Machine Learning (ML) is a new kind of computational algorithms that are designed to "learn" from the data. Traditional algorithms take an input and perform a defined set of actions to produce an output. In opposition to a traditional algorithm a ML algorithm performs a computational process without having every step of the process literally coded beforehand (i.e. “hard coded”). These algorithms are coded in a way so they can automatically adapt through repetition to produce the best possible output. This process of repetition and adaptation is called training and is basically an automated calibration. For the training a set of input data (e.g. rainfall, temperature and catchment size) as well as the desired output (the effectively measured runoff) need to be provided. Based on this input data the algorithm optimizes itself to predict the runoff from data which the model was not trained on (El Naqa and Murphy 2015).

Machine learning algorithms adjust internal parameters to achieve the desired result. They improve on their own errors and try to minimise the number of errors they make (Alzubi et al. 2018). More specifically, they build mathematical models based on so-called training data to make predictions or decisions without being specifically programmed to do so (Bishop, 2006 in Khanzode & Sarode, 2020).

## Advantages and Disadvantages of Machine Learning

One advantage of machine learning algorithms is their ability to handle massive amounts of data. They can analyse datasets that are too large for humans to easily make meaningful predictions (Alzubi et al. 2018). To apply machine learning, it is not necessary to understand how the predictor relates to the target. Khanzode & Sarode (2020) also mention the often good performance in highly repetitive tasks, as well as the fact that their decision is not biased by emotions.

While it's true that the algorithm itself is not biased by emotion, this leads us to the problem of having biased data to train it on. It is often difficult to create unbiased models because a model is only as good as the data it is trained on. Often, prejudices or, for historical reasons, unbalanced datasets can lead to unintentional biases that may discriminate certain groups (Barocas & Selbst 2016). It is therefore important to establish best practice rules to avoid the reproduction of historically intentional discrimination when working with personal data (Veale & Binns, 2017). Machine learning algorithms often require large amounts of data to make accurate predictions. Much data is not openly available. This creates a very uneven playing field. This is why big tech companies are leading in this area (Alzubi et al. 2018). Another drawback is the high demand for computing power and thus energy consumption. So far, the development of machine learning algorithms has been based on accuracy rather than efficiency. Although there are some developments in this direction (e.g., the Low Power Image Recognition Challenge (LPIRC)), it is not yet a main goal (García-Martín et al. 2019).

## Hyperparameter Optimization

In machine learning models two types of parameters exist. One are the “model parameters” that can be initialized and updated through the data learning process. The other type of paramaters are the “hyperparameters” (HPs). Hyperparameters are parameters that are not learned from the data, but are set before training a ML model. HPs define the model architecture and control various aspects of the learning process, such as the complexity of the model or the speed at which it learns (Yang and Shami 2020). There are different types of HPs: categorical, discrete, and continuous HPs (Decastro-García et al. 2019).

The process of finding the model architecture, best suited for the problem, with the optimal hyper-parameter configuration is called hyperparameter optimisation (HPO), also known as hyper-parameter tuning (Yang and Shami 2020). Due to the important role of HPs the process of hyperparameter optimisation is considered one of the most important steps in machine learning. Some argue that it is even more important than the type of model used (Szczepanek 2022). This is because, when applied to the same model and data, different sets of hyperparameters can produce very different results. Therefore, finding the optimal set of hyperparameters can significantly improve the performance of a machine learning model (Probst et al. 2019; Claesen & De Moor, 2015).

Many different types of optimisation techniques have emerged over the years, with different strengths and drawbacks. In the following the most important ones are briefly explained.

### Manual Experience Based Optimization

The most low-tech version of HP optimisation is manual optimisation, based on experience. For this method a profound understanding of the used ML algorithms and their hyper-parameter value settings is needed (Yang and Shami 2020). While it can be used for models with very few HPs it becomes almost impossible within a reasonable time for more complex models. We believe it is better practice to use experience to constrain the search space of the hyperparameters using one of the techniques mentioned below.

### Grid Search Optimization

Grid search is still a commonly used optimization method. It is based on the permutation of a defined set of parameter values (Yang & Shami 2020). This spans a finite n-dimensional search space, where n is the number of hyperparameters. This method can get quite resource-intensive because the number of combinations grows exponentially as the number of parameters increases. One way of optimising would be to first evaluate over a coarse selection of parameter values and then perform a finer optimisation around the best combinations in a second step. In a case with few well-constrained hyperparameters, grid search may still be a viable option (Yang & Shami 2020).

### Random Search Optimization

Random search works similar to grid search. The difference is that instead of a predefined set of parameter values random sets are tested. Bergstra & Bengio (2012) have shown that random search leads to an equally good or better parameter set in less time than grid search. While it is more efficient than grid search, there is no way to directly exploit the previously well-performing regions.

### Gradient Based Optimization

Gradient descent takes a randomly chosen parameter set and calculates the gradient to choose the direction to move to a better parameter set until a local optimum is reached (Bengio, 2000). This is of course not unproblematic, as only for convex functions is the local optimum also the global optimum (Yang & Shami 2020). In addition, it suffers from the same problems of complexity as the optimisation of the grid search.

### Bayesian Parameter Optimization

Like the gradient based optimization, the Bayesian optimization is an iterative algorithm. This allows the selection of a new parameter set based on previous results (Snoek et al., 2012). Bayesian optimisation is made up of two main components. A probabilistic surrogate model that models the performance of different parameter combinations, and an acquisition function that selects the next parameter set to test based on the surrogate model (Yang & Shami 2020). The acquisition function manages a trade-off between exploring new areas and continuing to investigate promising areas.   
There are a number of different surrogate models. The most commonly used include Gaussian process (Seeger, 2004), random forest (also called SMAC) (Hutter et al. 2010), and tree-structured Parzen estimator (Bergstra et al. 2011).   
It has been shown that near-optimal parameter sets can be found within a few steps (Snoek et al., 2012).

### Multi Fidelity Optimization

Training a model can take a considerable amount of time. Training times of minutes can be considered fast. Complex models with large datasets can take several hours to days for a single training run (Claesen & De Moor, 2015). This means that each optimisation iteration is costly. The multi-fidelity approach first optimises on a subset of the training data to explore promising regions. In a second step, the entire training data is used to optimise the entire model (Bottou, 2010). In this way, poorly performing parameter sets can be discarded before costly training with the full dataset (Yang & Shami 2020).

### Meta Heuristic Optimization

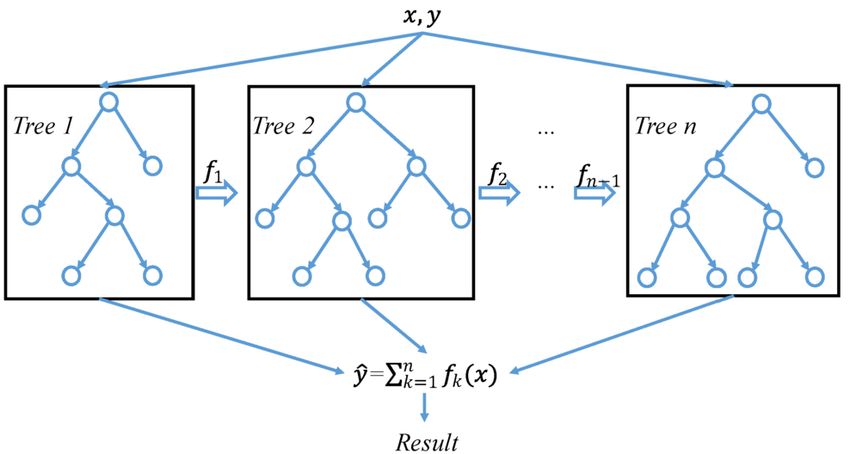
Yang & Shami (2020) describe the meta heuristic as follows: “*Metaheuristic algorithms are a set of algorithms mainly inspired by biological theories and widely used for optimization problems. Unlike many traditional optimization methods, metaheuristics have the capacity to solve non-convex, non-continuous, and non-smooth optimization problems.”*

## Machine Learning Models

In this section the machine learning algorithms used in this study are briefly explained and their main benefits and drawbacks outlined.

### **XGBoost**

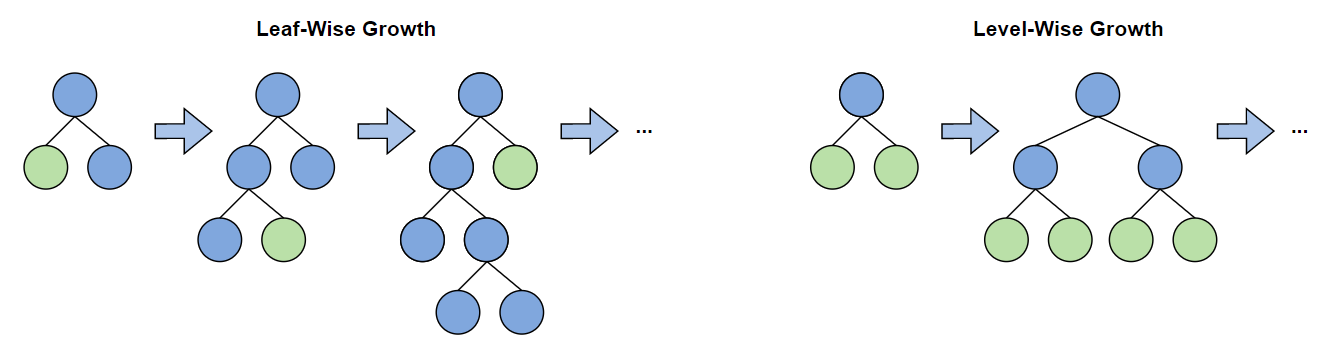
The open source software library originated from a research project at the University of Washington (Chen & Guestrin, 2016) and is developed by an active group of community members (<https://github.com/dmlc/xgboost>). It is available in 6 programming languages and is compatible with all major platforms.   
XGBoost stands for “extreme gradient boosting” and is a regression forest based gradient boosting algorithm that can handle large datasets very efficiently. It uses gradient boost on the loss function to determine the best data split during tree building (Chen & Guestrin, 2016). The bias after building and applying a tree is the basis for building the next tree, until the previously set maximum number of trees is reached. In simple terms, each tree corrects the error of the previous tree. It can handle sparse (incomplete) data and has several features that make it very efficient with large amounts of data (Chen & Guestrin, 2016). These features include those that optimise on the statistical side, such as the approximate greedy algorithm, parallel learning, and weighted quantile sketch, as well as implementations that affect hardware usage, such as cache-aware access and blocks for out-of-core computation (Chen & Guestrin, 2016).



The main advantages of XGBoost are its ease of use and its resource efficiency. The low entry barrier opens it up to a wider range of users than more complicated neural network based models, and its resource efficiency makes hyperparameter optimisation fast, even on large datasets. Another major advantage is the ability to interpret tree-based models. Approaches from game theory can be used to calculate the contribution of predictors to the model output (Lundberg & Lee 2017; Wang et al. 2022). The main drawback of tree-based algorithms is the limited range of values predicted, which is restricted to the range of values in the training set. This is a major limitation when predicting extreme events with low probability of occurrence, especially if there is a long-term trend.   
It has been used to some extent in RRM (Szczepanek, 2022; Wang et al. 2022, Kumar et al. 2021), but is not as popular as other machine learning models.

### LightGBM

LightGBM, short for “light gradient-boosting machine”, was developed by a Microsoft research group and is available as an open source library (https://github.com/microsoft/LightGBM). It is available in several programming languages and runs on all major platforms.   
Like XGBoost, LightGBM is a regression forest based gradient boosting algorithm (Ke et al. 2017). They are closely related but have some distinct differences. Most algorithms grow trees level-wise and prune them in one way or another to remove low information gain splits. LightGBM grows trees leaf by leaf, always evaluating where the next best split is. This method gives better results for the same number of leaves (Shi, 2007).

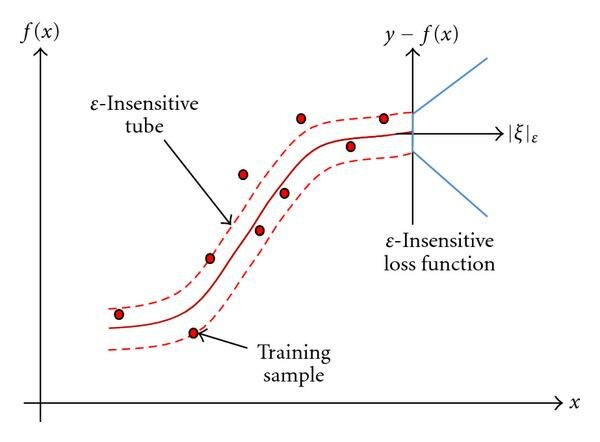


Two new features were implemented to be even more efficient with large datasets, gradient-based one-sided sampling (GOSS) and exclusive feature bundling (EFB) (Ke et al. 2017). When assessing the bias to grow new trees, LightGBM groups the data into already well-performing data points and poorly performing data points. To grow a new tree, only the bias of bad performing data points and a random sample of good performing data points are used (Ke et al. 2017). GOSS thus saves computational time with minimal loss of model performance. With EFB LightGBM can combine mutually exclusive features (e.g. gender) into a new feature. This reduces the total number of features and with it, computation time (Ke et al. 2017).

Being closely related, it shares the advantages and disadvantages with XGBoost. Despite the good performance it has seen considerably less use in RRM than XGBoost (Szczepanek, 2022; Cui et al. 2021; Bian et al. 2023).

### **Support Vector Regression (SVR)**

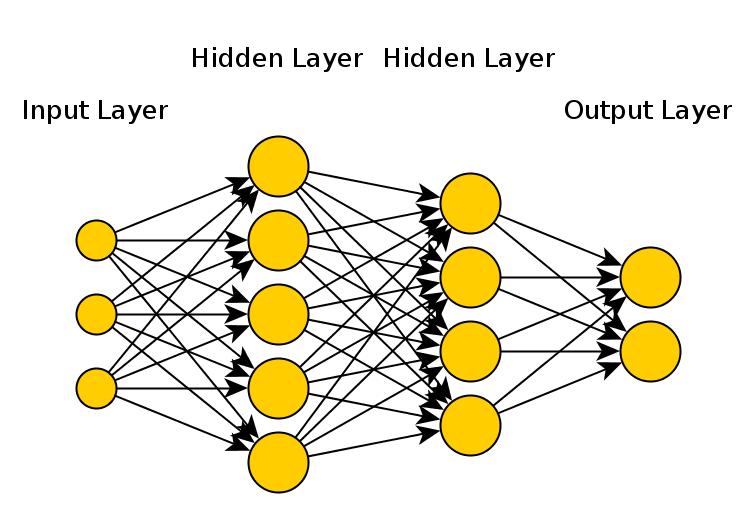
Support vector regression (SVR) is a generalised form of support vector machine (SVM) (Awad et al. 2015). SVMs are designed for multivariate binary classification (Cortes & Vapnik, 1995). They formulate classification as a convex optimisation problem (Awad et al. 2015). The optimisation finds the maximum margins of a hyperplane that splits the data as well as possible while still allowing for misclassification. In SVMs, the margins are expressed as support vectors (Awad et al. 2015). SVR performs regression by finding the simplest non-linear tube with r = ε, where ε is a defined tolerance level, that contains as many data points as possible while minimizing the distance from the tube to the points outside the tolerance level (Vapnik 1998, 55ff).



Advantages of SVR is the lower amount of data required to yield good model results and the high generalization capabilities (Liu et al. 2020). A disadvantage is that performance loss is noticeable with larger datasets.

### Multilayer Perceptron (MLP)

Multilayer perceptrons (MLPs) are the most basic and widely used artificial neural networks (ANN) (Popescu et al. 2009). ANNs are inspired by neurons in the brain. Neurons have multiple inputs, some of which are stimulating and some of which are inhibitory. When a certain threshold is reached, the cell fires a signal to all other connected cells. Learning occurs by strengthening or weakening certain input connections (Drew & Monson, 2000). ANNs consist of three basic components: nodes, layers and connections. They typically consist of an input layer where the data is fed, one or more so-called hidden layers where the learning takes place, and an output layer that gives us the values we are looking for. Each layer consists of one or more nodes (neurons). Each node in a layer is then connected to the next layer in some way. In most cases the layers are densely connected, i.e. every node in layer n is connected to every node in layer n+1 (Figure ???). However, linearly connected and other variants also exist (Zou et al. 2009). The number of layers and nodes in each layer, also called topology, strongly affects model accuracy.



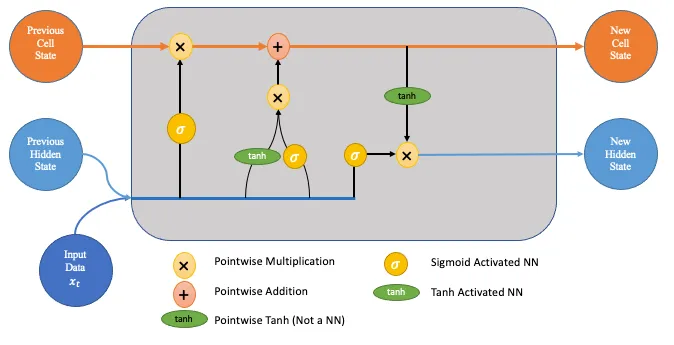
The type of ANN defines what the nodes of the hidden layers look like. In MLPs, the nodes consist of a single activation function (Popescu et al. 2009). Activation functions are usually non-linear functions used to transform the input value of a node. An overview of the most commonly used activation functions is given in the corresponding chapter. The links between nodes consist of weights and biases. These weights and biases can flip and stretch the output of a node. This is the part where the "learning" takes place. The weights control how much information is passed along the connections (Popescu et al. 2009) and are adjusted during training. The fitting is done by a process called back-propagation. To initialise the model, all weights are set to a small random number and biased to zero. Then the training data is fed into the model and the loss is calculated. A commonly used loss function is the mean squared error (mse). Now the derivative of the loss function with respect to each weight and bias is formed. Gradient descent is then used to update the weights and biases step wise to minimise the loss (Krogh, 2008; Drew & Monson, 2000). Several implementations with small additions to the classical gradient descent have been implemented.

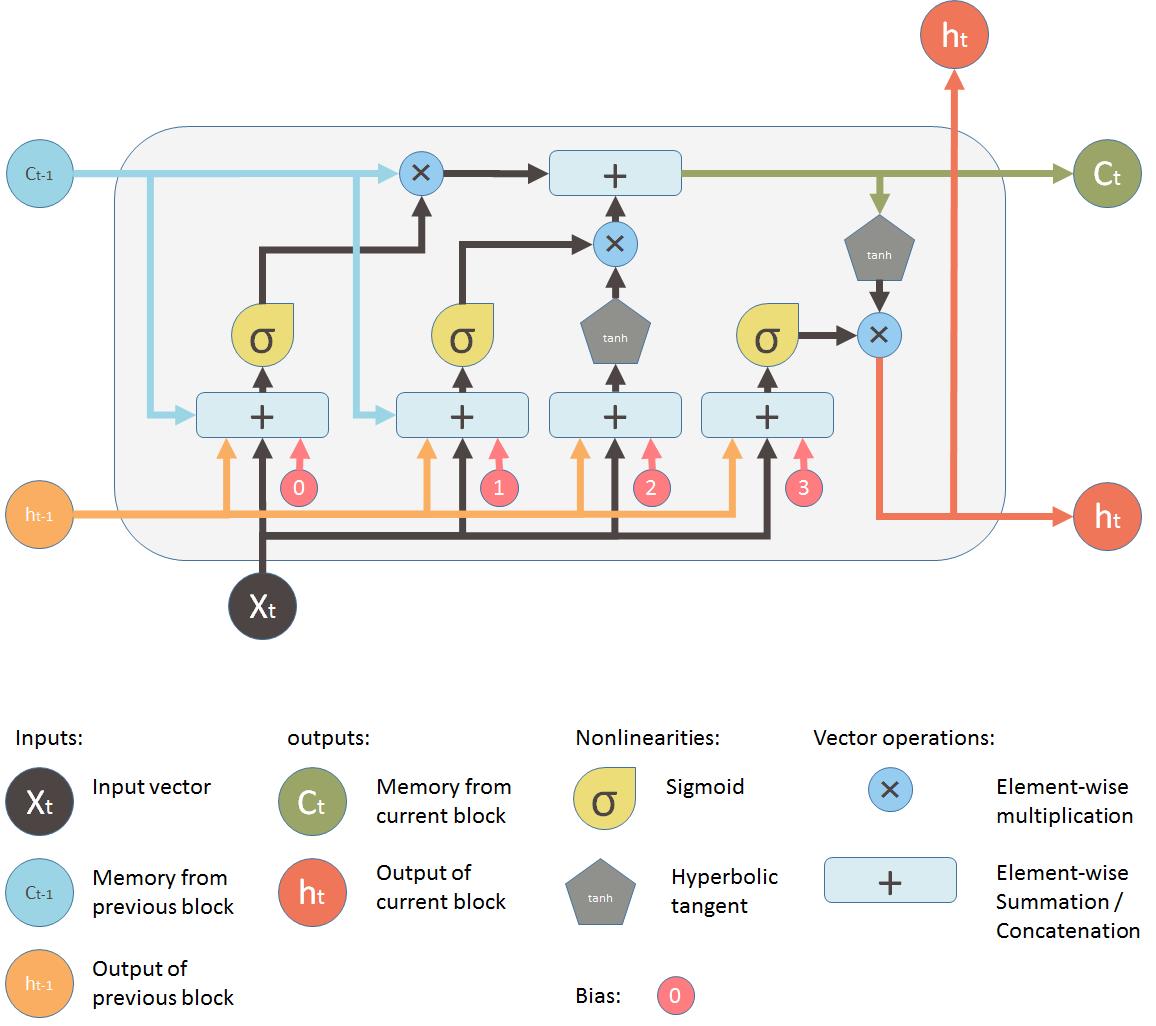
The main advantages of neural networks are the ability for good generalisation and the ability to capture non-linear trends. However, it is difficult to analyse the model itself to see how the result is generated. Additionally, the training of a MLP model is computationally much more expensive than the previous methods. Thus, hyperparameter optimization can be time consuming (and costly) which requires the use of an efficient optimization method.

### **Long short-term memory (LSTM)**

The LSTM is a recurrent neural network (RNN) that solves the exploding/vanishing gradient problem. When using long sequences, traditional RNNs face the problem that the long-term gradient tends to zero or infinity.

LSTMs are designed to classify, process and predict data based on time series and have been used in speech recognition, translation, robot control and time series prediction. As it was designed to process time series type data, it has naturally been used extensively in connection with RRM (ca. 7300 results on google scholar) and has been shown to give good results. The disadvantage of LSTMs is the computational power required to train the model, which makes hyper-parameter optimisation more difficult.





### Gated Recurrent Units (GRU)

## Activation Functions

In this chapter the most commonly used activation functions are explained.

### Rectified Linear Unit (ReLU)

### Soft Plus

### Sigmoid

### Tan-h

### Softmax

# Methods and Data

## Research area (Switzerland)

The six representative catchments defined by Muelchi et al. 2020 (Rosegbach, Kander, Plessur, Emme, Venoge, Verzasca). They cover a wide range of catchment characteristics and are unregulated. All catchments were used by Mülchi et al. (2020) These 6 catchments had good model quality with the PREVAH model. In a second step, we also compare catchments where the PREVAH model performed poorly. This comparison should provide insight into how the ML models perform and where their strengths and weaknesses lie.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Catchment | MQ | Station elevation | Catchment area | Mean elevation | Glaciation |
| Rosegbach | 2.8 | 1770 | 66 | 2704 | 21.7 |
| Kander | 20 | 650 | 491 | 1854 | 5.1 |
| Plessur | 8.0 | 565 | 264 | 1868 | 0.0 |
| Emme |  |  |  |  | 0.0 |
| Venoge | 4.1 | 384 | 228 | 686 | 0.0 |
| Verzasca | 11 | 495 | 185 | 1651 | 0.0 |

Tabelle 1: Catchments used in this study

## Data

The following data are used in this study:

Shape-files of catchments

Discharge (BAFU)

Gridded mean temperature (TabsD; Frei, 2014; MeteoSwiss, 2019a)

Gridded daily precipitation sum (RhiresD; Frei and Schär, 1998; MeteoSwiss, 2019b)

(Gridded temperature and precipitation data of the dataset CH2018)

Gridded snow cover?

Gridded min/max Temperature

Gridded relative humidity (availability?)

## Data Preparation

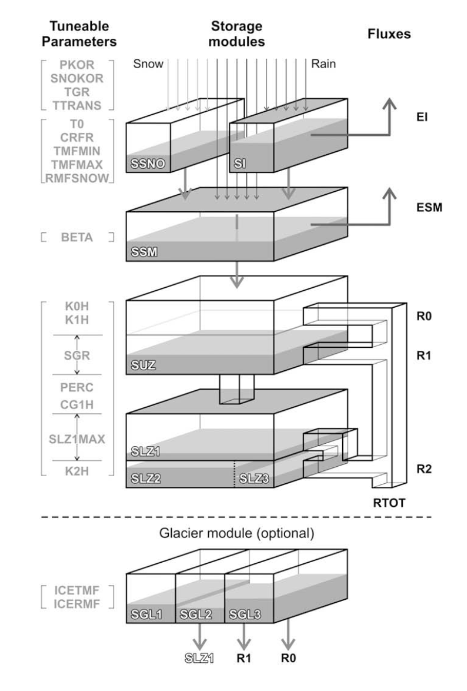
Daily mean temperature and total daily precipitation were provided as gridded data. Using the catchment shape files, the gridded data were aggregated to daily means. In order to provide the models with information about past conditions, several lagged, rolling mean and summed time series were derived from their originals. Additionally for all neural network based models all data was standard normalized to ensure that all input features are on the same scale. This leads to a faster and more accurate convergence.

## Methods

Three different machine learning algorithms are considered and compared with PREVAH. The comparison is based on the Nash-Sutcliffe Efficiency (NSE) and the Kling-Gupta Efficiency (KGE). The above models are explained below. (For an in-depth video explanation visit: https://www.youtube.com/@statquest)

### PREVAH

The model PREVAH which is used as a benchmark benchmark is a spacially explicit, process oriented model designed to model catchments with complex topography. The catchment is split into so called hydrological response units (HRU), surfaces with similar response. For each HRU a storage cascade is simulated and then combined to an area mean value. Figure ?? shows a storage cascade used in the model.

Abbildung 1: Bucket cascade of the conceptual model PREVAH

### XGBoost

XGBoost is a regression forest based gradient boosting algorithm that can handle large datasets very efficiently. It has seen some use in RRM, but is not as popular as other machine learning models.

The open source software library originated from a research project at the University of Washington and is developed by an active group of community members (https://github.com/dmlc/xgboost).

It's available in 6 programming languages and runs on all major platforms. The main advantages of XGBoost are its ease of use and it's resource efficiency. The low entry barrier opens it up to a wider range of users than more complicated deep learning models, and it's resource efficiency makes hyperparameter optimisation fast, even on large datasets. The main drawback of the algorithm is the limited value range of the predictions, which is restricted to the value range of the training set. This is a major limitation when predicting extreme events with low probability of occurrence, especially those that have not yet been observed.

XGBoost has many hyperparameters that can be adjusted. The models here have been optimised for the following hyperparameters:

|  |  |  |
| --- | --- | --- |
| Hyperparameter | Description | Optimization range |
| eta | Eta is the learning rate and controls how much is learned per round. High values lead to fast overfitting | 0.001 – 0.25 |
| max depth | Maximum depth of a tree. Larger tree depths allow to learn more complex data structures and more different output values, but lead to overfitting. | 2 – 12 |
| Min child weight | Controls how much weight a branch must have to be further partitioned. Higher numbers lead to more conservative trees. | 1-50 |
| alpha | L1 regularisation on weights | 1 – 12 |
| lambda | L2 regularisation on weights | 1 – 12 |
| nrounds: | maximum number of boosting iterations | 1 – 250 |

The optimization was done using a Bayesian optimization process to find the optimal parameter set.

### LightGBM

|  |  |  |
| --- | --- | --- |
| Hyperparameter | Description | Optimization range |
| eta | Eta is the learning rate and controls how much is learned per round. High values lead to fast overfitting | 0.001 – 0.25 |
| max depth | Maximum depth of a tree. Larger tree depths allow to learn more complex data structures and more different output values, but lead to overfitting. | 2 – 12 |
| num\_leaves | Controls the max number of leaves per tree | 1 – 120 |
| nrounds: | maximum number of boosting iterations | 1 – 200 |

The optimization was done using a Bayesian optimization process to find the optimal parameter set.

### Multilayer Perceptron (MLP)

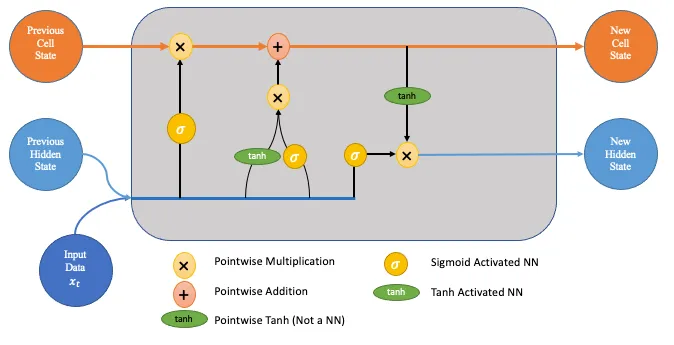
|  |  |  |
| --- | --- | --- |
| Hyperparameter | Description | Optimization range |
| nlayers | Number of layers in the neural network | 1 – 5 |
| nnodes | Number of nodes in each layer | 5 – 150 |
| Dropout rate |  | 0 – 0.4 |
| batch size | Step size after which the weights are updated within an epoch | 5 – 100 |
| epoche | Number of times the dataset is used to train the neural network | 1 – 100 |

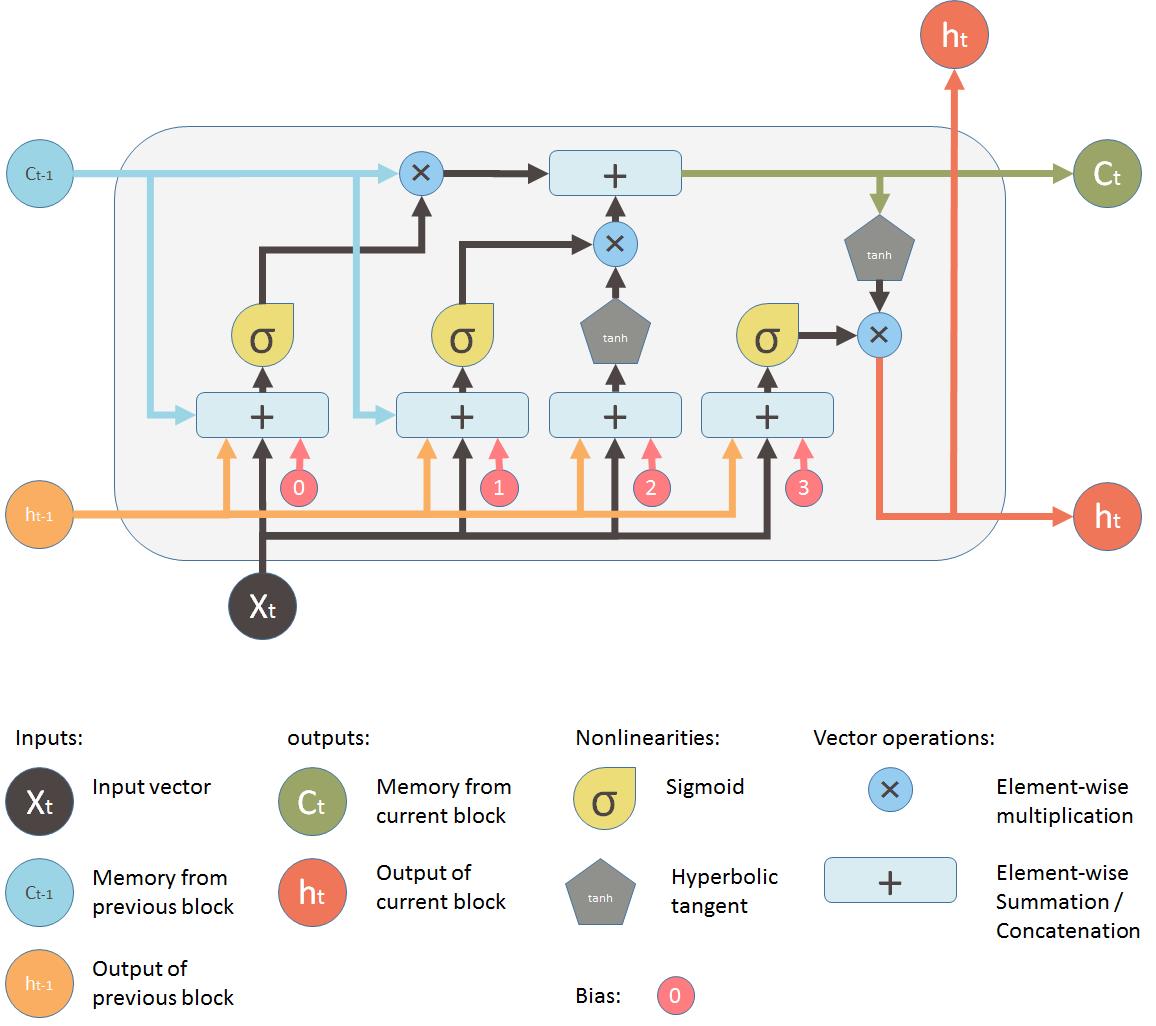
The optimization was done using a Bayesian optimization process to find the optimal parameter set. In addition the loss function “mse” and optimizer “adam” were used. The activation function used was “softplus”.

### Long short-term memory (LSTM)

The LSTM is a recurrent neural network (RNN) that solves the exploding/vanishing gradient problem. When using long sequences, traditional RNNs face the problem that the long term gradient tends to zero or infinity.

LSTMs are designed to classify, process and predict data based on time series and have been used in speech recognition, translation, robot control and time series prediction. As it was designed to process time series type data, it has naturally been used extensively in connection with RRM (ca. 7300 results on google scholar) and has been shown to give good results. The disadvantage of LSTMs is the computational power required to train the model, which makes hyper-parameter optimisation more difficult.





LSTMs have many hyperparameters that can be adjusted. The models here have been optimised for the following hyperparameters:

|  |  |  |
| --- | --- | --- |
| Hyperparameter | Description | Optimization range |
| nlayers | Number of layers in the neural network | 1 – 5 |
| nnodes | Number of nodes in each layer | 5 – 150 |
| Dropout rate |  | 0 – 0.4 |
| batch size | Step size after which the weights are updated within an epoch | 5 – 100 |
| epoche | Number of times the dataset is used to train the neural network | 1 – 100 |
| step size | Number of time steps of the input data | 5 – 100 |

The optimization was done using a Bayesian optimization process to find the optimal parameter set. In addition the loss function “mse” and optimizer “adam” were used.

### Gated Recurrent Units (GRU)

|  |  |  |
| --- | --- | --- |
| Hyperparameter | Description | Optimization range |
| nlayers | Number of layers in the neural network | 1 – 5 |
| nnodes | Number of nodes in each layer | 5 – 150 |
| Dropout rate |  | 0 – 0.4 |
| batch size | Step size after which the weights are updated within an epoch | 5 – 100 |
| epoche | Number of times the dataset is used to train the neural network | 1 – 100 |
| step size | Number of time steps of the input data | 5 – 100 |

The optimization was done using a Bayesian optimization process to find the optimal parameter set. In addition the loss function “mse” and optimizer “adam” were used.

## Calibration and Validation

Where available, the models were calibrated and validated with data from 1985 to 2016. This period was used in the Muelchi et al. 2020 study. In addition, the models were calibrated and validated over a longer time period (1981 – 2020) to see if the models improved with more data. Two out of every three years were used to calibrate the models. Every third year was used for validation. This pattern minimised the influence of random and non-random trends on model calibration which is particularly noticeable in catchments with high mean elevation. Muelchi et al. (2020) used Nash-Sutcliffe efficiency and Kling-Gupta efficiency to analyse model performance. Both performance indicators measure model bias, where 1 describes a perfect model. The Kling-Gupta efficiency focuses more on the correct variability and correlation of the runoff.

# Results

# Discussion

# Conclusion

# **Literature**

Peel, M. C. and McMahon, T. A.: Historical development of rainfall-runoff modeling, Wiley Interdisciplinary Reviews: Water, 7, e1471, <https://doi.org/10.1002/wat2.1471>, 2020. [a](https://hess.copernicus.org/articles/25/5517/2021/" \l "xref_paren.20)

Addor, N. and Melsen, L.: Legacy, rather than adequacy, drives the selection of hydrological models, Water Resour. Res., 55, 378–390, 2019. [a](https://hess.copernicus.org/articles/25/5517/2021/" \l "xref_paren.29)

Hoedt, P.-J., Kratzert, F., Klotz, D., Halmich, C., Holzleitner, M., Nearing, G. S., Hochreiter, S., and Klambauer, G.: MC-LSTM: MassConserving LSTM, in: Proceedings of the 38th International Conference on Machine Learning, vol. 139 of Proceedings of Machine Learning Research, edited by: Meila, M. and Zhang, T., 4275–4286, PMLR, available at: <http://proceedings.mlr.press/v139/hoedt21a.html>(last access: 1 October 2021), 2021. [a](https://hess.copernicus.org/articles/25/5517/2021/" \l "xref_paren.24)

[Aktuelle Daten der Nationalen Daueruntersuchung der Fliessgewässer (NADUF) (admin.ch)](https://www.bafu.admin.ch/bafu/de/home/themen/wasser/zustand/daten/aktuelle-messwerte-der-gewaesser/aktuelle-daten-der-nationalen-daueruntersuchung-der-fliessgewaes.html)

[Application of a conceptual runoff model in different physiographic regions of Switzerland: Hydrological Sciences Journal: Vol 37, No 3 (tandfonline.com)](https://www.tandfonline.com/doi/abs/10.1080/02626669209492583)

[IHACRES, GR4J and MISD-based multi conceptual-machine learning approach for rainfall-runoff modeling | Scientific Reports (nature.com)](https://www.nature.com/articles/s41598-022-16215-1)

https://link.springer.com/article/10.1007/s00506-021-00767-z

https://tubaf.qucosa.de/landing-page/?tx\_dlf[id]=https%3A%2F%2Ftubaf.qucosa.de%2Fapi%2Fqucosa%253A22570%2Fmets

<https://en.wikipedia.org/wiki/Timeline_of_machine_learning>

https://xgboost.ai/about

Jafar Alzubi, Anand Nayyar, Akshi Kumar (2018): Machine Learning from Theory to Algorithms: An Overview. Journal of Physics: Conference Series, Volume 1142, Second National Conference on Computational Intelligence (NCCI 2018) 5 December 2018, Bangalore, India, doi: 10.1088/1742-6596/1142/1/012012

Awad, M., Khanna, R., Awad, M., & Khanna, R. (2015). Support vector regression. *Efficient learning machines: Theories, concepts, and applications for engineers and system designers*, 67-80.

Barocas, S., & Selbst, A. D. (2016). Big Data’s Disparate Impact. *California Law Review*, vol. 104, no. 3, 2016, pp. 671–732. http://www.jstor.org/stable/24758720

Bengio Y.: Gradient-Based Optimization of Hyperparameters. *Neural Comput* 2000; 12 (8): 1889–1900. doi: https://doi.org/10.1162/089976600300015187

Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. Advances in neural information processing systems, 24.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of machine learning research*, *13*(2).

Beven, K. J.: Rainfall-runoff modelling: the primer, John Wiley & Sons, 2011. [a](https://hess.copernicus.org/articles/25/5517/2021/" \l "xref_paren.7), [b](https://hess.copernicus.org/articles/25/5517/2021/" \l "xref_paren.8), [c](https://hess.copernicus.org/articles/25/5517/2021/" \l "xref_paren.26)

Bian, L., Qin, X., Zhang, C., Guo, P., & Wu, H. (2023). Application, interpretability and prediction of machine learning method combined with LSTM and LightGBM-a case study for runoff simulation in an arid area. *Journal of Hydrology*, *625*, 130091.

Bishop, C. M. (2006), Pattern Recognition and Machine Learning, Springer, ISBN 978-0- 387-31073-2.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers (pp. 177-186). Physica-Verlag HD.

Çelik, Ö. (2018). A Research on Machine Learning Methods and Its Applications. Journal of Educational Technology and Online Learning, 1 (3), 25-40. DOI: 10.31681/jetol.457046

Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794). https://dl.acm.org/doi/pdf/10.1145/2939672.2939785

Chhaya A. Khanzode, Ravindra D. Sarode (2020): ADVANTAGES AND DISADVANTAGES OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING: A LITERATURE REVIEW. International Journal of Library & Information Science (IJLIS) Volume 9, Issue 1, January-April 2020, pp. 30-36

http://www.iaeme.com/IJLIS/issues.asp?JType=IJLIS&VType=9&IType=1

Claesen, M., & De Moor, B. (2015). Hyperparameter search in machine learning. arXiv preprint arXiv:1502.02127. https://doi.org/10.48550/arXiv.1502.02127

Cortes, C., Vapnik, V. Support-vector networks. *Mach Learn* 20, 273–297 (1995). https://doi.org/10.1007/BF00994018

Cui, Z., Qing, X., Chai, H., Yang, S., Zhu, Y., & Wang, F. (2021). Real-time rainfall-runoff prediction using light gradient boosting machine coupled with singular spectrum analysis. *Journal of Hydrology*, *603*, 127124.

Drew, P. J., & Monson, J. R. (2000). Artificial neural networks. *Surgery*, *127*(1), 3-11.

DeCastro-García, N., Muñoz Castañeda, A., Escudero García, D., Carriegos, M. (2019): Effect of the Sampling of a Dataset in the Hyperparameter Optimization Phase over the Efficiency of a Machine Learning Algorithm, *Complexity*, vol. 2019, Article ID 6278908, 16 pages, 2019. https://doi.org/10.1155/2019/6278908

El Naqa, I., Murphy, M.J. (2015). What Is Machine Learning?. In: El Naqa, I., Li, R., Murphy, M. (eds) Machine Learning in Radiation Oncology. Springer, Cham. https://doi.org/10.1007/978-3-319-18305-3\_1

Fradkov A. L. Fradkov: Early History of Machine Learning, IFAC-PapersOnLine, Volume 53, Issue 2, 2020, Pages 1385-1390, ISSN 2405-8963

https://doi.org/10.1016/j.ifacol.2020.12.1888

Eva García-Martín, Crefeda Faviola Rodrigues, Graham Riley, Håkan Grahn: Estimation of energy consumption in machine learning, Journal of Parallel and Distributed Computing, Volume 134,

2019, Pages 75-88, ISSN 0743-7315, https://doi.org/10.1016/j.jpdc.2019.07.007.

Gupta, H. V., Kling, H., Yilmaz, K. K., and Martinez, G. F. (2009): Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling, J. Hydrol., 377, 80–91, https://doi.org/10.1016/j.jhydrol.2009.08.003

Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers 5 (pp. 507-523). Springer Berlin Heidelberg.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T. Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, *30*.

Kratzert, F., Klotz, D., Brenner, C., Schulz, K., and Herrnegger, M.: Rainfall–runoff modelling using Long Short-Term Memory (LSTM) networks, Hydrol. Earth Syst. Sci., 22, 6005–6022, https://doi.org/10.5194/hess-22-6005-2018, 2018.

Krogh, A. (2008): What are artificial neural networks?. *Nat Biotechnol* 26, 195–197. https://doi.org/10.1038/nbt1386

Kumar, D., Roshni, T., Singh, A., Himayoun, D., & Samui, P. (2021). A simplified approach for rainfall-runoff modeling using advanced soft-computing methods. *Jordan Journal of Civil Engineering*, *15*(3).

Lees, T., Buechel, M., Anderson, B., Slater, L., Reece, S., Coxon, G., and Dadson, S. J.: Benchmarking data-driven rainfall–runoff models in Great Britain: a comparison of long short-term memory (LSTM)-based models with four lumped conceptual models, Hydrol. Earth Syst. Sci., 25, 5517–5534, https://doi.org/10.5194/hess-25-5517-2021, 2021.

Li, W., Kiaghadi, A. & Dawson, C. High temporal resolution rainfall–runoff modeling using long-short-term-memory (LSTM) networks. *Neural Comput & Applic* 33, 1261–1278 (2021). https://doi.org/10.1007/s00521-020-05010-6

Liu, Y., Chen, H., Zhang, L., Wu, X., & Wang, X. J. (2020). Energy consumption prediction and diagnosis of public buildings based on support vector machine learning: A case study in China. *Journal of Cleaner Production*, *272*, 122542.

Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. Advances in neural information processing systems, 30.

Muelchi R., Roessler O., Schwanbeck J., Weingartner R., Martius O. 2020. Neue  
hydrologische Szenarien für die Schweiz. Im Auftrag des Bandesamts für Umwelt  
(BAFU), Bern, Schweiz, 51 S., doi: 10.7892/boris.148715.

Nash, J. E. and Sutcliffe, J. V.: River flow forecasting through conceptual models part I – A discussion of principles, J. Hydrol., 10, 282–290, https://doi.org/10.1016/0022-1694(70)90255-6, 1970.

Popescu, M. C., Balas, V. E., Perescu-Popescu, L., & Mastorakis, N. (2009). Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, *8*(7), 579-588.

Probst P., Boulesteix A., Bischl B. 2019: Tunability: Importance of Hyperparameters of Machine Learning Algorithms. Journal of Machine Learning Research 20 (2019) 1-32

https://www.jmlr.org/papers/volume20/18-444/18-444.pdf

Seeger, M. (2004). Gaussian processes for machine learning. *International journal of neural systems*, *14*(02), 69-106. https://doi.org/10.1142/S0129065704001899

Shi, H. (2007). Best-first Decision Tree Learning (Thesis, Master of Science (MSc)). The University of Waikato, Hamilton, New Zealand. https://hdl.handle.net/10289/2317

Sitterson, Jan; Knightes, Chris; Parmar, Rajbir; Wolfe, Kurt; Avant, Brian; and Muche, Muluken, "An Overview of Rainfall-Runoff Model Types" (2018). International Congress on Environmental Modelling and Software. 41. https://scholarsarchive.byu.edu/iemssconference/2018/Stream-C/41

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. Advances in neural information processing systems, 25.

Szczepanek, R. 2022. "Daily Streamflow Forecasting in Mountainous Catchment Using XGBoost, LightGBM and CatBoost" Hydrology 9, no. 12: 226. https://doi.org/10.3390/hydrology9120226

Vapnik, Vladimir N. 1998: *Statistical Learning Theory*. New York. Wiley

Veale, M., & Binns, R. (2017). Fairer machine learning in the real world: Mitigating discrimination without collecting sensitive data. Big Data & Society, 4(2). https://doi.org/10.1177/2053951717743530

Wang, Yuanchao & Pan, Z. & Zheng, J. & Qian, L. & Mingtao, Li. (2019): A hybrid ensemble method for pulsar candidate classification. Astrophysics and Space Science. 364. 10.1007/s10509-019-3602-4.

Yang Li, Shami Abdallah 2020: On hyperparameter optimization of machine learning algorithms: Theory and practice. Neurocomputing, Volume 415, 2020, Pages 295-316, ISSN 0925-2312,

https://doi.org/10.1016/j.neucom.2020.07.061

Zou, J., Han, Y., & So, S. S. (2009). Overview of artificial neural networks. *Artificial neural networks: methods and applications*, 14-22.

# Apendix