Fakultät Informatik

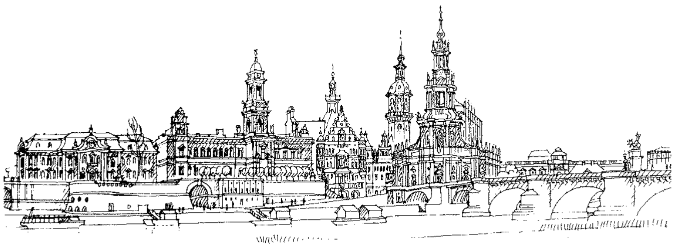
Technische Berichte
Technical Reports
ISSN 1430-211X

TUD-FI07-06 September 2007

Thomas Krauße

Institut für Künstliche Intelligenz

Development of a Class Framework for Flood Forecasting



Technische Universität Dresden
Fakultät Informatik
D−01062 Dresden
Germany
URL: http://www.inf.tu−dresden.de/

Der vorliegende Technische Report basiert auf der Diplomarbeit / The present technical report bases on the diploma thesis

Thomas Krauße

"Development of a Class Framework for Flood Forecasting"

Supervisors:

Doz. Dr.-Ing. habil. Uwe Petersohn
(Faculty of Computer Science)

Prof. Dr.-Ing. habil. Gerd H. Schmitz
(Faculty of Forest, Geo and Hydro Sciences)

Contents

List of Symbols and Abbreviations

The dimension of the units is indicated by the following abbreviations: [M] for mass, [T] for time, [L] for length and [Θ] for temperature.

	Symbols	
γ	Weight density	$[M\,L^{-3}]$
A	Streamed cross sectional area	$[L^2]$
e	Relative air humidity	$[-]$
g	Acceleration of gravity	$[L\,T^{-2}]$
I_E	Local gradient of the energy line	$[-]$
I_R	Local gradient of the river bed	$[-]$
K_h	Hydraulic conductivity	$[L\,T^{-1}]$
P	Precipitation	$[L]$
Q	Runoff	$[L^3\,T^{-1}]$
R	Global radiation	$[M\,T^3]$
T	Temperature	$[\Theta]$
u	Wind speed	$[L\,T^{-1}]$
V	Degree of vegetation cover	$[-]$
V	Volume	$[L^3]$
v	Flow velocity	$[L\,T^{-1}]$
w_i	Weight vectors of a neural network indexed by i	$[-]$

	Abbreviations
ANN	Artificial Neural Network
API	Application Programming Interface
DLL	Dynamic Link Library
EBNF	Extended BackusŰNaur Form
GMDH	Group Method of Data Handling
HEC-RAS	Hydrologic Engineering Centers River Analysis System
LSTM	Long Short-Term Memory neural network
MLFN	Multi Layer Feed-forward Network
NSE	Nash-Sutcliffe Efficiency Coefficient
PAI-OFF	Process Modelling and Artificial Intelligence for Online Flood Forecasting
PoNN	Sigma-pi network / Polynomial Neural Network
RNN	Recurrent neural network
SSR	Stepwise Serial Regression
WaSiM-ETH	Water Balance Simulation Model, developed at the ETH Zurich

	Glossary of Terms
Conceptual model	A hydrological model defined in the form of mathematical equations which describe the runoff process as a whole, but not urgently the running physical subprocesses.
Ensemble forecast	An ensemble forecast is simply a collection of two or more forecasts verifying at the same time.
Hydrograph	A chart that display the change of a hydrologic variable over time. Mostly and in this work exclusively, this means the chart of the stream discharge at a specific location.

Hyetograph	A map or chart displaying the temporal or areal distribution of precipitation.
Lead time	The required time for a forecast ahead of the current time in real-time forecasting
Linear store	A model component in which the output is directly proportional to the current storage value. It is often used in *conceptual models.*
Unit hydrograph	The hydrograph of surface runoff resulting from a relatively short, intense rain with defined quantity, called a *unit storm*

1 Introduction

The calculation and prediction of river flow is a very old problem. Especially extremely high values of the runoff can cause enormous economic damage. A system which precisely predicts the runoff and warns in case of a flood event can prevent a high amount of the damages.

On the basis of a good flood forecast, one can take action by preventive methods and warnings. An efficient constructional flood retention can reduce the effects of a flood event enormously. With a precise runoff prediction with longer lead times (>48h), the dam administration is enabled to give order to their gatekeepers to empty dams and reservoirs very fast, following a smart strategy. With a good timing, that enables the dams later to store and retain the peak of the flood and to reduce all effects of damage in the downstream. A warning of people in possible flooded areas with greater lead time, enables them to evacuate not fixed things like cars, computers, important documents and so on. Additionally it is possible to use the underlying rainfall-runoff model to perform runoff simulations to find out which areas are threatened at which precipitation events and associated runoff in the river. Altogether these methods can avoid a huge amount of economic damage.

To build up a flood forecasting system, the first and most important task is to develop a confident rainfall-runoff model which computes the runoff for a given rainfall and a event-history in the catchment. There exist many models to predict the rainfall-runoff relation of different types of catchments, but especially for *fast responding catchments* a precise and confident prediction of the runoff is a really hard task. Insofar this is a problem, that possible flood events, so called *flash floods* [Cul06], occur rapidly and can cause a high amount of damage (figure 1.1). In fast responding catchments not only existing gauge measurements in the upper reaches, but also a good quantitative rainfall prediction plays a very important role. Therefore models, resolving this problem, have to deal with a high amount of uncertainty and have to present this uncertainty in the right way. This work presents the core of a new system for this task, named FloodNet, which bases upon the new PAI-OFF approach presented in [SCG+05]. FloodNet can furthermore be driven

Figure 1.1: Flash flood event of the Müglitz river in the Ore Mountains[a]

[a]Source: Archiv Harald Weber (`http://www.harald-weber.info`)
Copyright: GNU Free Documentation License (`http://www.gnu.org/licenses/fdl.txt`)

by meteorologic forecast ensembles as input data, which enables a better estimation of a confidence belt of the computed runoff curve. The system was implemented by a modular, high-performance and platform independent class framework. The modularity enables another main advantage, to use the system also with arbitrary new prediction methods without problems and and to use the system in a flexible manner as the core of an online based flood forecast system or a desktop system.

The following work is organized in three parts. The first part gives a brief introduction into the domain of rainfall-runoff modelling and flood forecasting. The basic rainfall-runoff processes and their physical descriptions are briefly introduced. Subsequently, there follows a presentation of some approaches to represent these processes with process based models. This class of models is state-of-the-art and includes physically based and hydrodynamic models. The whole rainfall-runoff process can also be portrayed by time series predicting neural networks. There follows a discussion of current approaches to use artificial neural networks in rainfall-runoff modelling and a presentation of different neural network architectures which can be used to fulfill this purpose. The second part consists of a presentation of the new flood forecasting framework FloodNet and the underlying approach. After a listing of the requirements and boundary conditions for the design and implementation of the system, there follows an overview of the approach, the system bases upon. This is in principal the new PAI-OFF approach for quickly responding catchments, presented in [SCG$^+$05] and [Cul06]. It uses artificial neural networks which have been trained by time series of meteorological data and associated runoff. The used time series are a combination of historical measurements in nature and synthetically generated extreme flood events and associated calculated output of a process based model. The approach was extended by some further ideas and small improvements. After this, there follows the presentation of the the design and implementation of the new flood forecasting class framework FloodNet. To round off the work, the third part gives a discussion of achieved results and the current state of the implementation. That includes explicitly a view into the future with further ideas to improve the reliability, to extend the possible field of applications, e.g. snow models to provide predictions of the winter term, and to enable the system once to perform real-time flood forecasting in operational mode.

II

Process based Rainfall-Runoff Modelling

2.1 Basics of runoff processes

The whole rainfall-runoff process beginning with the formation of the precipitation in the atmosphere by a complex condensation process until to the runoff in a stream at the outlet of a catchment can be divided in three main stages. First, the fallen precipitation interacts with the vegetation, land surface and soil at each point in the catchment area and forms an amount of water which is available for the further runoff process. This stage is named **formation of the runoff** and is a result of a precipitation event. After this, the formed runoff is concentrated in an on-site preflooder, which is called **runoff concentration**. This is followed by the **runoff transformation**. This means that flowing water is influenced by friction, backwater effects, etc. according to universal hydrodynamic flow laws. In general, this causes a change of the hydrograph, and peaks of the hydrograph in the upper flows are attenuated and prolonged in the lower flows. In this section there is given a presentation of these three stages of the rainfall-runoff process and the underlying physical processes. figure 2.1 gives a schematical overview for a better understanding. A precipitation event in the mountainous region at points 1 and 2 leads to the local formation of the runoff. The runoff is concentrated in a preflooder in the sub-mountainous region as in point 3 and the existing flood wave is transformed by hydrodynamic processes until the outlet of the catchment at point 4 is reached.

2.1.1 Formation of runoff

The rainfall-runoff process begins with the stage of the formation of the runoff. This is a local process at each point of the catchment, which is influenced by falling precipitation. The falling water interacts with the vegetation, the land surface and the soil. An amount of the precipitation is retained by the vegetation. Solid precipitation can form a snow cover, which changes its properties in the course of time, influenced by the meteorological

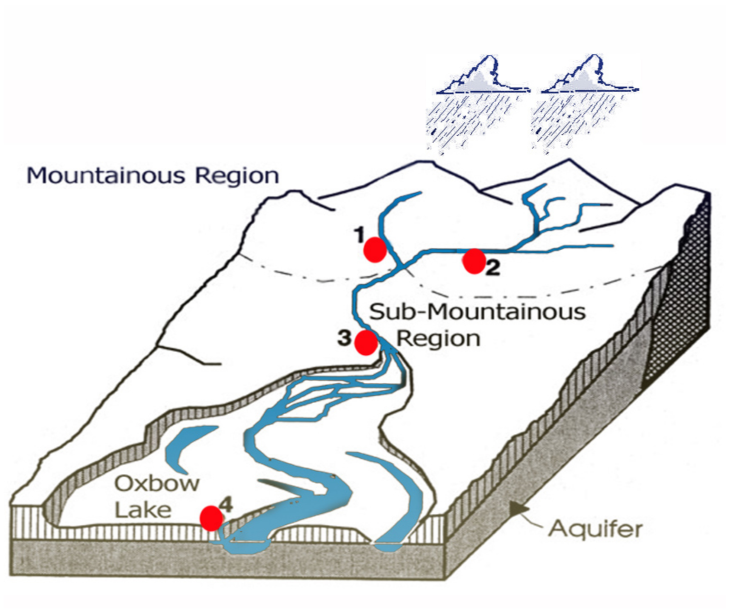


Figure 2.1: Rainfall-runoff processes in a mountainous catchment

situation. At temperatures above the freezing point it is subjected to melt processes. According to the state and type of the soil, an amount of the fallen precipitation infiltrates into the soil. Also a storage for a certain time on the land surface is possible. The infiltrated water in the soil can be stored there for a certain time. It can be subjected to evapotranspiration, it can be available for interflow or it percolates until the groundwater table. All these processes yield as their output the locally formed runoff. Within these processes different climatic, geomorphological and man-made factors play an important role. A presentation more in detail is given in the following.

Interception

At the beginning of a precipitation event the falling rain drops do not reach the ground, but wet the vegetation, when present. This process is called *interception* and causes the so called *interception loss*. It depends on the season and the type of the vegetation. Possible losses are caused for example by wetting the leaves or leaf canopy of the trees in a forest or the grass in the open space. The interception loss can be modeled by a conceptual linear storage unit model or a regression analysis. Usually the following equation is established via regression analysis [Din94].

$$Y = M_Y R + B_Y \tag{2.1}$$

Thereby R is the *gross rainfall* which means the rainfall measured above the vegetative canopy and Y is one of the components of the interception loss. This can be for instance the water that evaporates from the canopy and the water that evaporates from near-ground plants. The coefficients M_Y and B_Y are determined via regression analysis with existing measurements for a specific type of vegetation.

Evapotranspiration

Before reaching the land surface, another process plays an important role in the events of the runoff formation, the *evapotranspiration*. This is a collective term for all the processes by which water water in the liquid or solid phase becomes atmospheric vapor. Some authors [Din94] also subordinate the interception under this term. This processes bases upon many complex physical processes mainly dependent on the global radiation, temperature and the vegetation. In practice, for modelling the rainfall-runoff process, the estimation of the *potential evapotranspiration* and the *actual evapotranspiration* is sufficient.

The *potential evaporation* is defined as the rate at which evatranspiration would occur from a large area completely and uniformly covered with growing vegetation which has access to an unlimited supply of soil water and without advection or heat-storage effects. Of course this definition is theoretical. In practice all the processes influence each other. But nevertheless the potential evapotranspiration is a kind of index of the "drying power". This process is influenced by temperature, radiation and wind. In practice there exist several methods which include all or at least a subset of these values, to estimate the potential evaporation as precisely as possible.

The *actual evapotranspiration* can theoretically be estimated by the *water-balance approach*, given in equation 2.2 according to [Din94]). Thereby ET is the actual evapotranspiration, SW is the in- and outflowing surface water respectively, GW is the in- and outflowing groundwater respectively and ΔV is the change in the amount of water stored in the vegetation, soil and groundwater of this area.

$$ET = W + SW_{in} + GW_{in} - SW_{out} - GW_{out} - \Delta V \tag{2.2}$$

In practice the data requirements of this equation cannot be fulfilled by measurements for a greater area. Therefore the the actual evapotranspiration is often measured indirectly, for instance with the *soil-moisture balance*. Following this approach, it is possible to estimate the actual evapotranspiration in a time period Δt by monitoring the amount of precipitation and the *soil-water content* (equation 2.6) throughout the root zone:

$$ET = W - Q_d + \int_0^{z_{rz}} \theta_1(z)\,\mathrm{d}z - \int_0^{z_{rz}} \theta_2(z)\,\mathrm{d}z \tag{2.3}$$

where W is total water input into the area, Q_d is downward drainage in time Δt, z is depth; and z_{zr} is depth of the root zone. $\theta_1(z)$ and $\theta_2(z)$ are water-content profiles at the beginning and end of Δt respectively. However note that this approach is not suggested for populated and sealed areas. A deeper insight into these processes and other possible approaches are discussed in [Din94], [DP95] and [BL96].

Snow and Snowmelt

Especially in mountainous regions the accumulation of snow and the snowmelt plays an important role in runoff processes. At the top of typical low mountain ranges in Central Europe the solid fraction of the annual precipitation can amount approximately up to 40% [Her01]. The processes in the snow cover and its interaction with the vegetation and soil are subjected to high spatial and temporal dynamics and differences. There exist some reliable physically based snow models, using global and lateral radiation and the current temperature as input for the *energy-balance approach*. This bases upon the energy balance of one snow element defined as

$$(\Delta t)S = \Delta Q \tag{2.4}$$

where S is the net rate of energy exchanges into the element by all processes over a time period Δt, and ΔQ is the change in heat energy absorbed by the snowpack in Δt. A good comparison of different models is given in [EMB$^+$01] [1].

Due to the hard-to-fulfil data requirements of many sophisticated physically based snow models, many hydrological models represent the snowmelt by the *temperature-index approach*. It estimates the snowmelt, w for each day as a linear function of average air temperature T_a with a *melt coefficient*, B.

$$\begin{aligned} \Delta w &= B(T_a - T_m), \quad && T_a \geq T_m \\ \Delta w &= 0, && T_a < T_m \end{aligned} \tag{2.5}$$

The melt coefficient varies with latitude, elevation, slope inclination and vegetation. It can only estimated approximately via measurements in test areas. Nevertheless this approach is often very unprecise and especially in catchments in mountain ranges where a significant portion of the precipitation falls as snow, a good snowmelt modelling is a very difficult task.

Infiltration

As the precipitation water reaches the land surface it begins to infiltrate into the ground as long as the soil is not completely saturated. This process is determined by the capillary power and the gravitation. An important parameter for this processes is the *volumetric water content*, θ which is the ratio of water volume to soil volume:

$$\theta = \frac{V_W}{V_S} \tag{2.6}$$

The infiltration process itself is an unsaturated flow in a porous media the soil. This can be described by DARCY'S LAW:

$$V_x = -K_h \frac{\partial(z + \frac{p}{\gamma_w})}{\partial x} \tag{2.7}$$

where V_x is the volumetric flow rate in the x direction per unit cross-sectional area of medium, z is the elevation above an arbitrary reference level, often the groundwater table, p is the water pressure, γ_w is the weight density of water, and K_h is the *hydraulic conductivity* of the medium. For hydrologic problems dealing with rainfall-runoff modelling, the weight density of the

[1] `http://www.cnrm.meteo.fr/snowmip/`

water γ_w can be assumed as effectively constant. Applying the definition for the pressure head ψ, given in equation 2.8 to Darcy's law (equation 2.7), we obtain equation 2.9 for unsaturated flows.

$$\psi = \frac{p}{\gamma_w} \tag{2.8}$$

$$V_x = -K_h(\theta)\frac{\partial(z+\psi(\theta))}{\partial x} \tag{2.9}$$

It is important to consider that both the pressure head and the hydraulic conductivity depend strongly on the volumetric water content, θ and the type of the soil. With equation 2.9 and the mass conservation law you can deduce the RICHARD'S EQUATION (equation 2.10 for vertical percolation in the soil, according to the American soil physicist A.L. Richards, who first derived it.

$$-\frac{\partial K_h(\theta)}{\partial z'} + \frac{\partial}{\partial z'}\left[K_h(\theta)\frac{\partial\psi(\theta)}{\partial z'}\right] = \frac{\partial\theta}{\partial t} \tag{2.10}$$

The Richard's Equation has high data requirements in reference to pedologic measurements in the catchment, to estimate the relation of hydraulic conductivity K_h against the degree of saturation or volumetric water content θ. This relation differs for different types of soil. Additionally there exists no closed analytical solution for this partial differential equation and the computational effort of a numerical solution is comparatively high.

To avoid these high demands, it is also possible to portray the infiltration process by more simplistic conceptual models. A commonly used example is the TOPMODEL approach, developed by *Beven* and *Kirkby* in 1979. It calculates the soil moisture as a function of the hill slopes. The underlying simple approach is, that plane areas for instance at the bottom of valleys are saturated faster than steep hillsides, due to the proceeding runoff processes. The *infiltration* is one of the most important processes in the formation of the runoff. The infiltrated water can be stored for a certain time and influences the different components of the runoff, as presented in the following paragraph.

Runoff components

Although the processes in the rainfall-runoff process influence each other and cannot be divided strictly, you can usually divide the total runoff into three components, according to figure 2.2, *direct runoff*, *interflow* and *baseflow*.

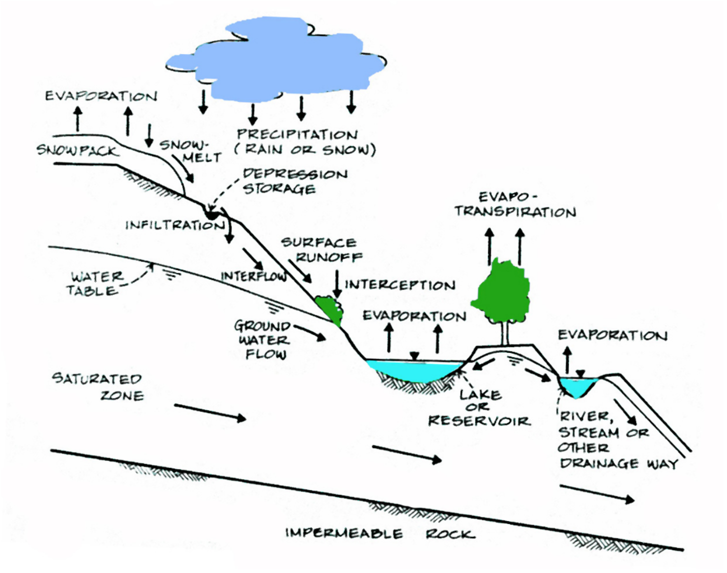


Figure 2.2: Schmematical overview of different runoff components

If the precipitation rate exceeds the maximum infiltration rate of the soil, a little amount of the superfluous water can be stored in stores on the land surface as for instance swales or basins. If the amount of water on the land surface additionally exceeds the capacity of these storage, *direct runoff* is formed. Rarely this means a real overland flow but more often a so called hypodermic flow in the soil but very close to the land surface. The infiltrated water percolates through the soil and is stored there for a certain time. If it is not subjected by evapotranspiration, it can form a downslope flow above the groundwater table, which is called the *interflow*. Only a part of the infiltrated water percolates completely through the soil and reaches the groundwater table and forms the *baseflow*. The storage effects in the soil and groundwater reservoirs are often modeled by cascades of linear stores.

2.1.2 Runoff concentration

The runoff concentration includes the whole flow process, beginning with the formed runoff at each point in the catchment until the concentration in on-site preflooders. Without any measurements the runoff concentration can only be computed by geomorphologic models. Within these models the direction and the velocity of the flow is computed on the basis of the hill slopes and type of the soil. If measurements in form of time series of the runoff and meteorological data for a specific preflooder are available, conceptual or empirical models give in general better results. In conceptual models the flow of the water in the runoff concentration is usually represented by a *cascade of linear stores*. An old and simple empirical approach, which is commonly used, is the *unit hydrograph*. The underlying method is the following. The catchment is divided into *areas of similar hydrologic response (ASR)*. That means that a defined impulse, the unit precipitation, causes at each point of such an area approximately the same runoff response at the catchment outlet, the *unit hydrograph*. An example is presented in figure 2.3. The actual hydrograph for a specific precipitation event, is estimated by convolution of the actual mean precipitation of each ASR with its unit hydrograph. The total runoff is then computed by superposition of the runoff responses of all ASRs in the catchment.

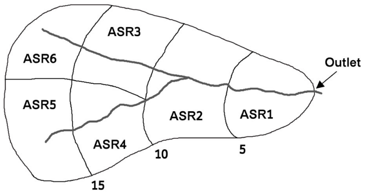


Figure 2.3: Areas of similar hydrologic response (ASR) according to [Cul06]

2.1.3 Runoff transformation

It is a questions of interpretation when the runoff concentration ends and the runoff transformation starts. A common definition is, that the runoff concentration continues until the preflooder exceeds a specified size. This size depends on the properties of the examined catchment area. The original hydrograph at the preflooder fixes the outer boundary condition for the process of runoff transformation at the lower flows. In terms of hydrodynamics this process is an *open channel flow*. In general the hydrograph is attenuated and prolonged by the process of runoff transformation. This means that peaks of the hydrograph in the upper flows become wider and are flattened in the lower reaches as shown in figure 2.4.

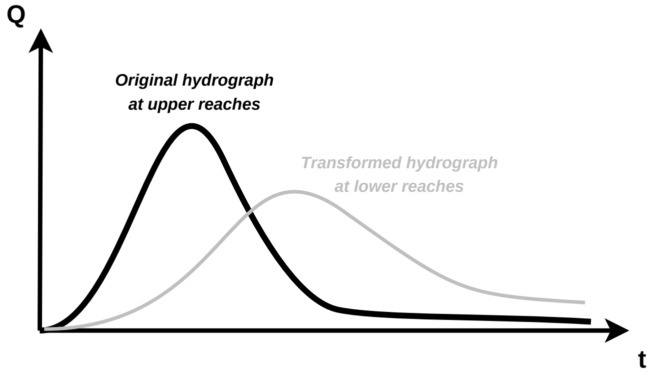

Q
Original hydrograph at upper reaches
Transformed hydrograph at lower reaches
t

Figure 2.4: Attenuation of the hydrograph for a given arbitrary flood event

The flow process in rivers are in general *instationary*. That means that the flow velocity at one specific location changes with the time and therefore equation 2.11 holds.

$$\frac{\partial v}{\partial t} \neq 0 \tag{2.11}$$

The process of instationary open channel flow can be described by the SAINT-VENANT EQUATIONS given in 2.12 and 2.13. They are derived from the law of conservation of mass and the law of conservation of energy. In the domain of flow dynamics the law of conservation of energy can also be expressed by the law of momentum balance. The Saint-Venant equations were enunciated first by Jean Claude Barré de Saint-Venant in 1843. For a stationary flow these equations are also valid but can be substituted by simpler ones. The streamflow in a river can be described as an open channel flow of an incompressible Newtonian and Eulerian fluid which holds the following equations:

$$v \cdot \frac{\partial A}{\partial s} + A \cdot \frac{\partial v}{\partial s} + \frac{\partial A}{\partial t} = q \tag{2.12}$$

$$\frac{\partial v}{\partial t} + v \cdot \frac{\partial v}{\partial s} + g \cdot \frac{\partial h}{\partial s} - g \cdot (I_R - I_E) + \frac{q}{A} \cdot (v - v_q \cdot \cos\alpha) = 0 \tag{2.13}$$

where A is the streamed cross sectional area, v is the medium flow velocity ($\frac{Q}{A}$), q is a lateral inflow ($q > 0$) or outflow ($q < 0$), h is the medium water depth, I_R is the local gradient of the river bed, I_E is the local gradient of the energy line, v_q is the medium flow velocity of the tributary stream and α is the angle, the tributary stream flows into the main stream. Further informations about the basics of hydrodynamics and open channel flow can be found in [PB00].

The Saint-Venant Equations (equation 2.12 and 2.13) are partial differential equations which cannot be solved analytically for the general case, but only by numerical methods. Possible methods for a numerical solution are the finite difference method (FDM), the method of characteristics or the finite element method (FEM). For all methods an implicit formulation is more preferable than the explicit one, because this allows greater distances in spatial and temporal discretization. A good overview over existing numerical solutions and the presentation of an efficient method of characteristics is presented in [Sch81]. It is important to know that also the most efficient numerical solution methods require a high computational effort, but with the increasing performance of computers this disadvantage becomes smaller and smaller.

Therefore in reality it is often possible to apply less complex approximations. In general you can say, that as long as the gradient of the river bed is steep enough and backwater effects do not play a role, it is sufficient to leave out the accelerating members of equation 2.13. More simplistic approaches are for instance the *difussive wave approximation* and the *kinematic wave approximation* which use instead of the energy equation of the Saint-Venant Equations, equation 2.14 or 2.15 respectively.

$$g \cdot \frac{\partial h}{\partial s} - g \cdot (I_R - I_E) + \frac{q}{A} \cdot (v - v_q \cdot \cos\alpha) = 0 \tag{2.14}$$

$$-g \cdot (I_R - I_E) + \frac{q}{A} \cdot (v - v_q \cdot \cos\alpha) = 0 \tag{2.15}$$

Another solution of the runoff transformation in steeper reaches is also the use of conceptual models or empirical black-box approaches. A deeper insight into the required outer and inner boundary conditions follows with the example of *HEC-RAS* in the next section.

2.2 Physically based rainfall-runoff and hydrodynamic river models

The processes involved in the rainfall-runoff process can be represented by different types of models. In general there are three classes of approaches, the empirical *black-box* approach, *conceptual* approaches derived from system theory which use especially cascades of linear stores and *physically based* approaches which try to represent the processes by appropriate physical equations. In practice models are often using a mix of the different types of approaches for the different subordinated processes.

A special class of models are *hydrodynamic* river models which can only be used to represent the runoff transformation in the downstream. They implement and represent the hydrodynamic flow laws (section 2.1.3), derived from fundamental physical laws.

This section gives a presentation of two typical examples, a rainfall-runoff model and a hydrodynamic river model. The first one, WaSiM-ETH is is a state-of-the-art physically based rainfall-runoff model. That means that it implements mainly physical based approaches. Especially for the formation of the runoff WaSiM-ETH uses exclusively physically based approaches. Only the runoff transformation in the downstream is represented by more simplistic approaches. The second example is the hydrodynamic river model HEC-RAS. It is used to represent the runoff transformation in the lower sections of the downstream, where the gradient of the river bed is too flat to be represented by simplistic approaches and backwater effects can occur.

2.2.1 WaSiM-ETH - used as a modern physically based rainfall-runoff model

The model WASIM-ETH was developed by *Jörg Schulla* and *Karsten Jasper* as a Water balance Simulation Model at the Swiss Federal Institute of Technology Zurich (ETH). Because of its well-founded physically based approaches it can nevertheless also be used as a rainfall-runoff model for flood forecasting. The lower limit of the temporal resolution of the process descriptions in WASIM-ETH is approximately one hour. There are two versions of WASIM-ETH. Version 1 uses the conceptual TOPMODEL approach for the water flow in the unsaturated soil, whereas Version 2 uses the physically based RICHARD'S EQUATION, that was presented in section 2.1. WASIM-ETH is driven by time series of distributed meteorological data of the catchment areas as input and computes the total discharge at the catchment outlet. Distribution means that the meteorological data is available in form of a matrix, representing a grid layer over the catchment. Considered meteorological input values are temperature, precipitation, humidity, global radiation and wind speed. Both the meteorological data of the history and the forecast are taken into account. A good overview over the structure of WASIM-ETH is given in figure 2.5. The model also needs information about the terrain of the catchment in form of a matrix. That are mainly digital elevation data, a map of preflooders and rivers for discharge rooting and information of land use and soil. WASIM-ETH can be extended by modules, for instance there exists a module for the hydrological modelling of glaciers.

The data flow in WASIM-ETH can be described, as follows. First, the meteorological data undergoes some corrections to get a better accuracy. Especially measurements of the precipitation can be influenced considerably by elevation or wind. After this correction, the input data is interpolated and processed by different submodules, representing the different processes of the runoff formation. Then the estimated components of the formed runoff are rooted by a conceptual discharge model which can be variably parameterized for direct runoff, interflow and base flow. The transformation of the runoff is estimated by a linear translation-diffusion approach and estimates the total runoff at the outlet of the catchment. A detailed documentation of WASIM-ETH is given in [SJ06].

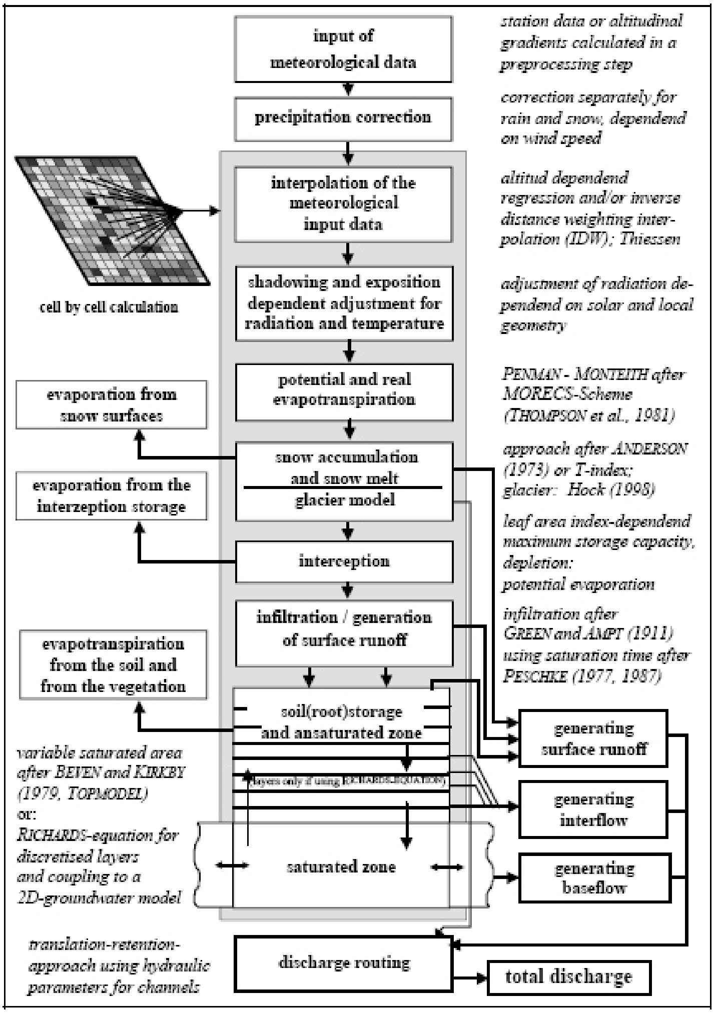

input of meteorological data
station data or altitudinal gradients calculated in a preprocessing step
precipitation correction
correction separately for rain and snow, dependend on wind speed
interpolation of the meteorological input data
altitud dependend regression and/or inverse distance weighting interpolation (IDW); Thiessen
cell by cell calculation
shadowing and exposition dependent adjustment for radiation and temperature
adjustment of radiation dependend on solar and local geometry
potential and real evapotranspiration
PENMAN - MONTEITH after MORECS-Scheme (THOMPSON et al., 1981)
evaporation from snow surfaces
snow accumulation and snow melt
glacier model
approach after ANDERSON (1973) or T-index; glacier: Hock (1998)
evaporation from the interzeption storage
interception
leaf area index-dependend maximum storage capacity, depletion: potential evaporation
infiltration / generation of surface runoff
infiltration after GREEN and AMPT (1911) using saturation time after PESCHKE (1977, 1987)
evapotranspiration from the soil and from the vegetation
soil(root)storage and ansaturated zone
layers only if using RICHARDS-EQUATION
variable saturated area after BEVEN and KIRKBY (1979, TOPMODEL) or: RICHARDS-equation for discretised layers and coupling to a 2D-groundwater model
saturated zone
generating surface runoff
generating interflow
generating baseflow
translation-retention-approach using hydraulic parameters for channels
discharge routing
total discharge

Figure 2.5: Model structure of WaSiM-ETH from [SJ06]

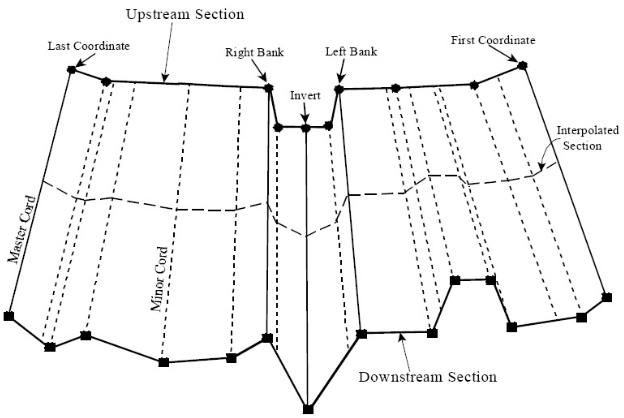


Figure 2.6: Example of an intersection profile in HEC-RAS from [Bru02]

2.2.2 HEC-RAS - a hydrodynamic river model

As soon as backwater effects play a roll in the process of runoff transformation the use of a hydrodynamic river model is inevitable. The HEC-RAS model is a well-engineered system to perform hydrodynamic calculations for river reaches. It considers the hydrograph of the upper reaches as input to estimate the discharge in the downstream.

This is done by a numerical solution of the Saint-Venant equations 2.12 and 2.13. For a correct estimation of the gradient of the energy I_E, and a correct water level to discharge relation the model needs additionally information about river intersection profiles as given in figure 2.6. Furthermore a correct estimation of friction loss requires information about the channel roughness. They can be estimated by water level fixation measurements but there are also some reference tables for this values according to *Manning* [Bru02] [PB00]. Local losses, for instance for narrow bridges and weirs, can be estimated by guess and empirical validation or by a physical models. That is a scaled model of the original which can be used for simulations. Measurements on the physical model can be transfered to the original according to the physical laws of similitude. Further information about the whole field of hydrodynamic modelling and calculation is given in [PB00].

The algorithm of HEC-RAS is approximately the following. The considered river reach is described by a set of intersection profiles (figure 2.6). Thereby two adjacent intersection profiles have to be within a specific maximal distance. The hydrograph of the upper reaches and of other feeder rivers fix the upper boundary conditions of a numerical solution of the Saint-Venant equations. Then the Saint-Venant equations for the whole considered river reach are solved and the hydrograph at the downstream intersections is calculated. The initial conditions, for example the discharge rates at each intersection can be obtained by measurements or by longer runs of the model with a default initial state of the model, for example the normal runoff. More detailed information on the model is given in the official reference manual [Bru02].

III

Portraying Rainfall-Runoff Processes with Neural Networks

The physically based and hydrodynamic models presented in the previous chapter try to simulate the rainfall-runoff process or parts of it by well-founded physical equations for the involved processes and are currently state-of-the-art in this domain. In spite of this there rests a not negligible amount of uncertainty in the model. The conceptual submodules of the model can be adjusted by a couple of free model parameters. These parameters have to be calibrated for each considered catchment, to estimate an optimal approximation of the model to this catchment. Unfortunately there does not exist a global set of optimal parameters for one catchment, but the optimal parameter set can change for different scenarios in one and the same catchment [Cul06]. This model and parameter uncertainty cannot be removed, because the whole dynamic of the runoff processes is only described approximately by the model approaches and the processes in the ground and their parameters can only partially sensed by measurements. For instance the properties of the soil which strongly influence the infiltration process can change dramatically even for small distances and it is almost impossible and too expensive to undertake measurements with a raster distance of some meters [1]. Additionally physically based models demand a high computational effort, due to the numerical solution of multiple partial differential equations. Especially for the operation of a rainfall-runoff model with meteorological forecast ensembles, this is a troublesome disadvantage, because there have to be executed a new model-run for each ensemble member which multiplies the high computational effort to a not acceptable amount.

These two main disadvantages of physically based models can be removed by a data-driven approach with an *artificial neural network* (ANN). This class of machine learning systems is suitable to represent a uncertain re-

[1]Consider that besides the uncertainty of the model, the parameters and the soil, there exist also an amount of uncertainty in the meteorological input data.

lation which cannot be described by deterministic algorithms. Although their training is computationally intensive, they can perform a runoff prediction in a very low operation time. To use neural networks in rainfall-runoff modelling you have to pre-process a huge set of meteorological data and use the result and an according runoff to train a neural network. Afterward the neural network can be used for discharge predictions of the considered catchment. In this chapter, there is presented the general challenge to portray rainfall-runoff processes with ANNs. This is followed by to some different state-of-the-art approaches, including the new PAI-OFF approach [SCG$^+$05] [Cul06]. The chapter is rounded off by an overview of specific ANN architectures for time series prediction, due to the fact that a runoff prediction is in the end a specific type of time series prediction.

3.1 The Challenge in General

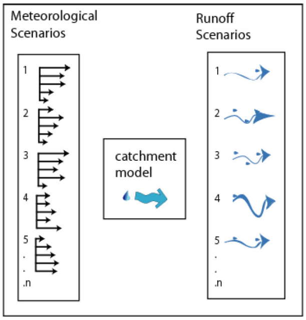


Figure 3.1: Generating an artificial training base with a catchment model after [Cul06]

This section gives an overview of the challenge of portraying rainfall-runoff processes with neural networks in general (figure 3.2). The starting point is a *training database*. This is a set of time series of meteorological data and associated runoff for the considered catchment. Often the source of the time series are longtime historical measurements from meteorological and gauging stations, but it is also possible to use synthetically generated meteorological scenarios and assigned runoff values, computed by a process based catchment model (figure 3.1). Another possibility is the use of distributed meteorological data instead of station measurements. This can be estimated by interpolation methods from meteorological stations, for example by the THIESSEN polygon approach, KRIGING, or by spatial measurements with remote sensing, e.g. precipitation measurements by ground based radar stations.

For each *reference point* which is covered by the training base, a specific *training algorithm* takes a specific interval, the *history* of recent runoff and

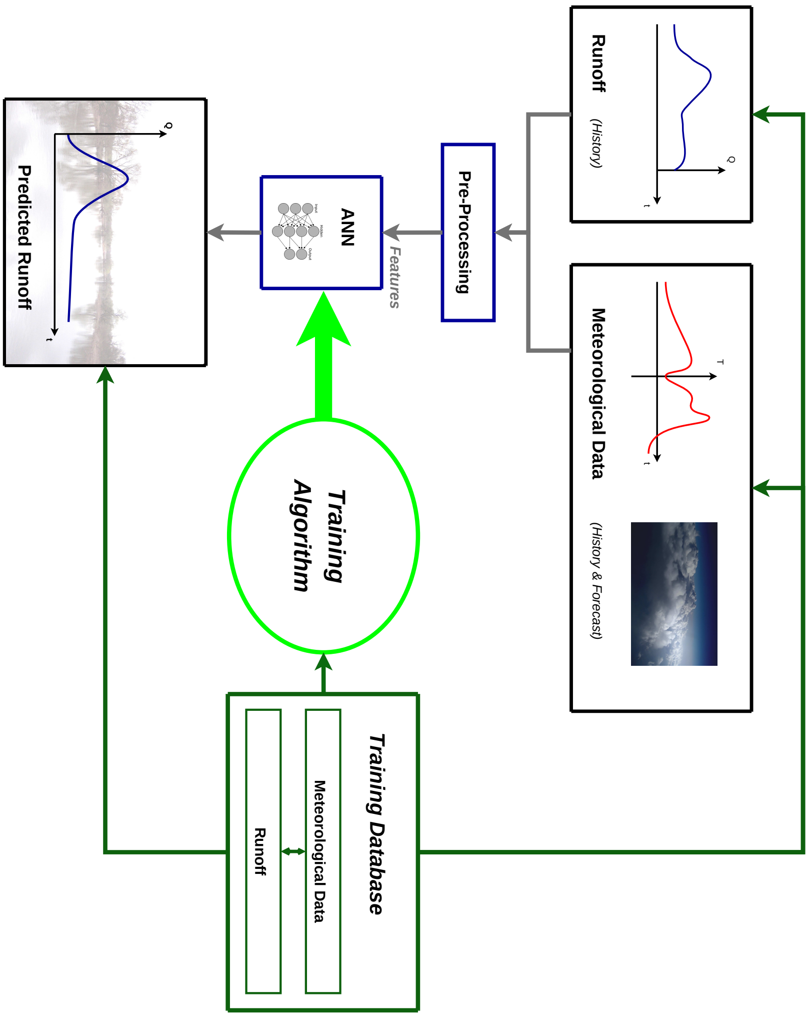


Figure 3.2: General scheme of portraying rainfall-runoff processes with neural networks

meteorological data. The input values are pre-processed and summarized to a smaller number of features that form the input of an *ANN*. In general the type of architecture of the ANN is arbitrary, but it is strongly advised, to use an ANN architecture which shows a good performance for time series prediction, to obtain a sufficiently low prediction error in the validation. After this the ANN is trained with the preprocessed input data and the assigned runoff values which represent the desired output of the ANN. This is done by a specific *training algorithm* according to the chosen ANN architecture. It is possible to train different ANNs for different lead times or to use one ANN for all lead times, by shifting the reference point.

3.2 State-of-the-art Approaches

This section presents some concrete approaches of portraying rainfall-runoff processes with neural networks that are published in literature. All of these approaches are state-of-the-art and can theoretically be used in operational flood forecasting systems. The presentation is concluded by pointing out the main disadvantage of these systems which caused the development of a new generation flood forecasting system.

All existing approaches use a training database which consists of historical time series of measured meteorological data and observed runoff. They differ in the type of the used network architecture, in the preprocessing of the used input data and in the use of underlying catchment models.

- In [HHA93] a three layer MLFN (section 3.3) was developed, to portray recorded hydrographs at Bellevue, Washington in the USA. The used MLFN has five nodes in the hidden layer. The neural network was driven by observed rainfall hyetographs as input. Altogether five storm events were considered. On a rotation basis, data from the hyetographs[2] of four storm events were used for training the ANN, while data from the fifth storm were used for testing the network performance. The results were promising but not satisfying.

- A very simple but intuitive approach is presented in [DW98]. It uses different moving averages from a few precipitation gaging stations in the catchment and the current streamflow at the catchment outlet as input of a neural network of type MLFN. Each input has a lead

[2]A map or chart displaying the temporal or areal distribution of precipitation.

time to simulate a delay in the runoff concentration and flow in the catchment. The neural network was trained with time series of historical measurements of the stations. The results are in spite of this simple approach astonishingly good, but this has its reasons also in the very simple structure of the examined catchment. Additionally the approach was not tested for major flood events. A further development of this approach [DW01] stands exemplary for using a simple feed-forward network, trained with historical time series from gaging stations in the catchment.

- For a better representation of the temporal retention-characteristics of the catchment, it is advisable to use neural networks. This can be done by connecting delay elements between the input and the ANN, or even better by using ANN architectures with internal states. A feed-forward network can only represent a non-linear static transfer function, but it does not have any internal state, whereas a *recurrent neural network (RNN)* provides this feature. A RNN is a neural network, which feeds back the information from neurons of one layer to neurons of a previous layer (section 3.3). This technique enables a RNN to conserve information from previous runs. This internal state, representing the previous operations, can be used for the computation of coming predictions. In the domain of rainfall-runoff modelling the internal state can represent the state of the catchment, which is influenced by previous events. An example of an application of a RNN in the domain of rainfall-runoff modelling is given in [KRS04]. As in the previously presented approaches historical time series from gaging stations where used to train both a MLFN and a RNN. The trained networks were applied to a validation set and the computed runoff outputs were compared with the desired runoff. As expected the RNN gave significantly better results than the MLFN.

Although the previously presented approaches show the potential of neural networks and give promising results, they have one main disadvantage. If a neural network is trained with (preprocessed) historical time series, it cannot interpolate its knowledge to predict extreme flood events reliably. Extreme floods occur rarely and do not need to be comprised in historical time series, which are mostly shorter than sixty years. At least historical time series do not contain a sufficient amount of extreme flood events to train a neural network sufficiently good. To improve the trustworthiness of neural network

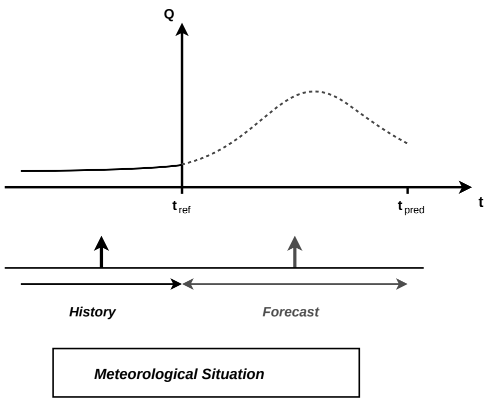


Figure 3.3: Runoff prediction as a kind of time series prediction

based flood forecasting models, it is inevitable to eliminate this handicap. Exactly this problem is resolved with the PAI-OFF approach [SCG+05] [Cul06]. This approach is the base for the design and implementation of the flood forecasting system FloodNet, which is presented in the next part of this work. Thereby the presentation of FloodNet is preceded by a presentation of the PAI-OFF approach (chapter 5).

3.3 Architectures of neural networks for time series prediction

This section concludes the chapter with an overview and discussion of different architectures of neural networks in the domain of time series prediction. The runoff prediction at a catchment outlet can also be interpreted as a time series influenced by previous input. The previous input is in our case the previous meteorological situation, especially the amount and distribution of precipitation in the catchment. For predicting the runoff for greater lead times, also the forecasted meteorological situation between the reference point t_{ref} and the prediction point t_{pred} have to be taken in account, because the forecasted meteorological situation also influences and changes the state of the catchment between t_{ref} and t_{pred} (figure 3.3).

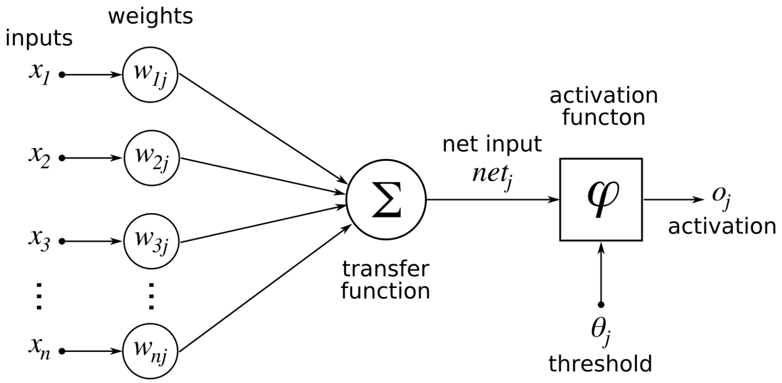


Figure 3.4: Model of an artificial neuron

3.3.1 Multi-Layer Feed-forward neural Network (MLFN)

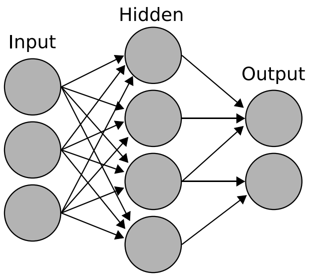


Figure 3.5: Design of a MLFN

The most classical architecture of an ANN is the *multi-layer feed forward neural network (MLFN)*. It is a standard type of a neural network usable for time series prediction [Thi98]. Feed-forward means that there exist no backfeed from posterior to previous layers (figure 3.5) and multi layer means that the network contains of different layers of neurons. It is common to use three or four layers, one for input, one or two hidden layers and one output layer. Each variable of an input vector is given to a neuron in the input layer. The input of the neurons in the input layer is now propagated to the hidden layers and then to the output layer. Thereby in general the values of all neurons of one layer are given to all neurons in the next layer. Only architectures for very specific problems break with this rule.

The neurons of a MLFN can be described by the following model (figure 3.4). All input values of the neuron x_i are weighted by specific weights w_i and added by the *transfer function*. Then an activation function φ is applied to the computed net input net_j and a threshold or bias value θ_j. The weights and the threshold value are adapted during the training process. A trained MLFN with a sufficient amount of neurons in the hidden layer can theoretically approximate an arbitrary nonlinear function. A common training algorithm for a MLFN is the backpropagation algorithm [Hay94] [Thi98]. One main disadvantage of this architecture in time series prediction is that it cannot hold any internal state. The only solution is to pre-process and summarize the previous events for each new prediction step and pass it as an input to the network. Besides the computational effort of this operation, the preprocessing and compression of information is often very subjective and fault-prone. For example in the domain of rainfall-runoff modelling, it is very hard to estimate universal features which express the state of the catchment sufficiently good.

3.3.2 Sigma-Pi Polynomial Neural Network (PoNN)

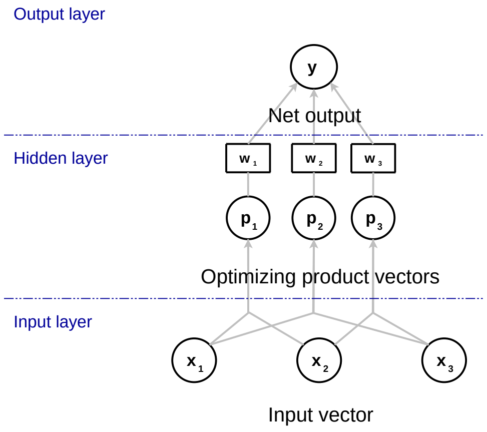


Figure 3.6: Design of a PoNN

The *sigma-pi polynomial neural network (PoNN)* is a feed-forward neural network with a single hidden layer. The components x_i of the input vector $\vec{x}$ are multiplied which form a set of *optimizing polynomials*. That means that the input vector is classified by a set of polynomials. In general all permutations of the inputs are possible, but it is also allowed to define an arbitrary subset of all theoretically possible polynomials, so that not all permutations of the input are considered. The computed results of the optimizing polynomials are multiplied by specific weights, added and then yield the output y of the neuron [DR90]. Formally this is defined as

$$y = \prod_{i=1}^{k} p_i(\vec{x}) \tag{3.1}$$

In [Fok99] it is shown that a PoNN can be used successfully in time series prediction. One advantage is that polynomial outputs near to zero, block the according neuron. The PAI-OFF approach (chapter 5), implemented in my system, also bases upon a rainfall-runoff portraying with a PoNN.

3.3.3 Long Short-Term Memory neural network (LSTM)

To resolve some problems with the architecture of feed-forward networks, it is possible to introduce bi-directional data flow within layers or between posterior and previous layers by so called *backfeeds*. Each ANN architecture with such a bi-directional data flow is classified into the class of *recurrent neural networks (RNN)*. With the backfeeds the ANN is enabled, to keep information about previous input in an inner state. This resolves the main disadvantage of feed-forward networks in time series prediction [MJ99] [Hay94]. Unfortunately, in a regular RNN the input of each new operation step influences the complete internal state immediately. That makes it difficult to keep information about previous input for a longer time in the internal state. This problem can be resolved by a new architecture of a RNN, the *Long Short-Term Memory neural network (LSTM)* [HS97].

In a LSTM neurons are structured in a unit, called *memory cell*. Each memory cell has an inner *central linear unit* which holds the internal state. This unit has a fixed self connection to hold the old state in a procession step. To prevent the internal state from being changed by irrelevant inputs as in a regular RNN architecture, the central linear unit is protected by a prefixed multiplicative unit called *input gate*. Likewise the output is influenced by a postfixed multiplicative unit, the *output gate*. It prevents other memory cells from possible perturbation. An overview is given in figure 3.7. With that technique the memory cell can hold its content as a long term memory, but with an "open" input or output gate also short term informations can be

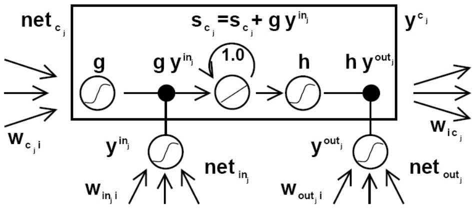


Figure 3.7: Architecture of a memory cell in a LSTM with its gate units after [HS97]

stored. In the following, there is given an outline of the internal processing mechanism of a memory cell in a LSTM.

Consider an arbitrary memory cell in the net, indexed by j, c_j. At time t it gets input from the input gate $net_{in_j}(t)$, the output gate $net_{out_j}(t)$ and its normal activation $net_{c_j}(t)$. The memory cell's output $y^{c_j}(t)$ is computed by multiplying its scaled internal state and the input at the output gate as

$$y^{c_j}(t) = y^{out_j}(t)h(s_{c_j}(t)) \tag{3.2}$$

The change of c_j's internal state $s_{c_j}(t)$ is computed by the old internal state and a multiplication of the input at the input gate and the normal activation as

$$\begin{array}{rcll} s_{c_j}(0) & = & 0, & t = 0 \\ s_{c_j}(t) & = & s_{c_j}(t-1) + y^{in_j}(t)g(net_{c_j}(t)), & t > 0 \end{array} \tag{3.3}$$

Consider that the normal activation is squashed by the differentiable function g. A possible topology of a LSTM with eight inputs and four outputs is given in 3.8. Consider that the memory cells of one block hold the same input and output gates. A detailed presentation is given in [HS97].

3.3.4 Group Method of Data Handling (GMDH)

A special self-organizing method for time series prediction, the *Group Method of Data Handling (GMDH)*, is proposed in [Fok99]. There are very good results for the prediction of stock prices as a time series, but nevertheless this approach could also be successful in time series prediction in hydrology. This method produces a set of models for a complex system by handling

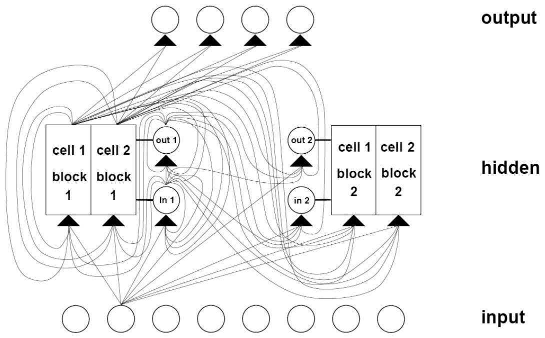


Figure 3.8: Possible topology of a LSTM after [HS97]; Cell1/block1's architecture is identical to the one in figure 3.7; for reasons of simplicity outgoing weights for only one type of unit are shown.

data-samples of observations. Within an iteration cycle the models become increasingly complex whereas only the best models, according to an external criterion of quality, are prototypes for more complex models. Most GMDH algorithms use the *Kolmogorov-Gabor Theorem* to construct the models. It proves the fact that every arbitrary function $y = f(\vec{x})$ can be represented as given in equation 3.4. Other reference functions but polynomial as difference or harmonic can also be used. A GMDH algorithm constructs at first only a very simple model as for instance $y = a_0 + a_1 x_i$ and increases its complexity with each iteration step.

$$y = a_0 + \sum_i a_i x_i + \sum_i \sum_j a_{ij} x_i x_j + \sum_i \sum_j \sum_k a_{ijk} x_i x_j x_k + ... \qquad (3.4)$$

The models are sorted-out by quality. This can be estimated by an external error criterion *CR*. As given in equation 3.5, a GMDH algorithm solves the argument $\tilde{g}$ to get the fittest models according to the external criterion. Thereby G is a set of candidate models and g is the considered member. A detailed description of this architecture and the scheme of some genetic GMDH algorithms can be found in [Fok99].

$$\tilde{g} = \arg\min_{g \subset G} CR(g) \qquad (3.5)$$

3.3.5 Modular networks

A special type of an ANN qualified for time series prediction is the *modular network* [Hay94]. A modular network consists of different neural networks, so called *expert networks*. Each expert network is a neural network of arbitrary type which was trained with a smaller subset of the training set. This gives in general a better approximation for the members of the trained subset. The generalization is instantiated by a *gating network* which weights the output of each expert network depending on the current input. A design of the layout is given in figure 3.9.

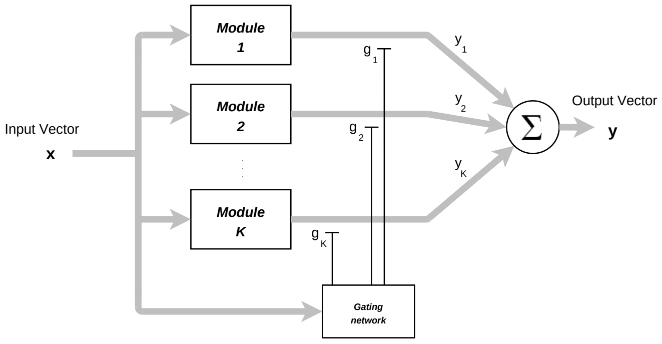


Figure 3.9: Diagram of a modular network; the outputs of the modules are weighted by a gating network

The training is done by a stochastic gradient algorithm which is presented in [Hay94]. In domains with a huge set of sub-classifications even different modular networks can be clustered recursively. For that purpose the developer has to carry out an exact analysis of the domain to construct an accurate network architecture.

IV

Requirements specification

At the beginning of each software development process stands an analysis of the requirements and demands on the planned software product. This is the most important task in creating a well-engineered software product, because it is the basis on which a clearly structured and smart design and implementation relies on. The goal of my work was to develop a **flood forecasting class framework**, FloodNet, embedded in a library as the core of a flood forecasting system using the methods of the aforementioned PAI-OFF approach. The specification of this framework was formed during a process of intensive discussions between hydrologists with a lot of domain-specific knowledge and experience and computer scientists with good skills in artificial intelligence and software development. Base of my work is a monolithic prototype implemented in *Visual Basic*[1] [SCG+05]. For the design and implementation of the framework the following requirements were specified:

- The library has to provide functions to **import and load time series of meteorological data and runoff** in the ASCII format specified by WaSiM-ETH, **subsume and pre-process the data** and use it to **operate a PoNN to predict the runoff** at the catchment outlet according to the PAI-OFF approach.

- Information about the catchment, e.g. the geomorphology, orography or land use, and parameters for the data preprocessing and the embedded neural prediction methods, e.g. the weights of an ANN, can be accessed by SQL[2] from an existing **relational database**.

- As an extension of the PAI-OFF approach, the system has to provide methods to enable the prediction of the runoff for different weather **ensemble forecasts** and a suitable conversion of the computed runoff data.

[1]`http://www.vb6.us`
[2]`http://sqlzoo.net/`

- The whole design and implementation of the system has to follow the principles of **modularity** and **platform independence**. That means that the design of the class framework is clearly laid out and new functions, e.g. prediction methods using new neural architectures, can be integrated and implemented without greater adaption problems. Due to the diversity of the architectures of modern server and desktop computer systems, the goal of platform independence is an important condition to use the framework in a flexible way as the core of a real-time online flood forecasting system or a desktop system.

- The prediction mechanism of the library has to be implemented in the fast, portable and object-oriented high-level programming language C++ [34]. This enables the library to operate with **high-performance**. To facilitate this goal especially computationally intensive parts of the prediction mechanism have to be implemented with fast algorithms.

- The class framework will be mainly **used by developers** to do further research and to integrate it in future flood forecasting systems with graphical user interface. An operational application without graphical user interface is absolutely not recommended. Additionally the documentation has to present an idea of a graphical visualization of the output data, especially with respect to uncertainty.

Before reading the next chapters with the presentation of the system, it has to be considered that the whole system was mainly build under the aspect, to use and further develop it in coming research projects up to the capability to use it in operational mode. Therefore it was attached importance to the aspect of **extensibility** and a precise **documentation** of the library interface. Additionally also the complete internal application programming interface (API) of the library is documented automatically to support the integration of further developers in the future.

[3]http://www.research.att.com/~bs/homepage.html
[4]http://www.cplusplus.com/

V

The PAI-OFF approach as the base of the system

The FloodNet flood forecasting framework presented in this work, is based upon the PAI-OFF approach [SCG+05] [Cul06]). PAI-OFF stands for Process Modelling and Artificial Intelligence for Online Flood Forecasting. As already outlined (section 3.2), the problem of neural approaches in the domain of rainfall-runoff modelling is, that historical time series do not contain extreme flood events in a sufficient quantity, to be trained by an ANN. This problem can be resolved by the PAI-OFF approach. The principal philosophy of PAI-OFF can be described, as follows. An as realistic as possible weather generator

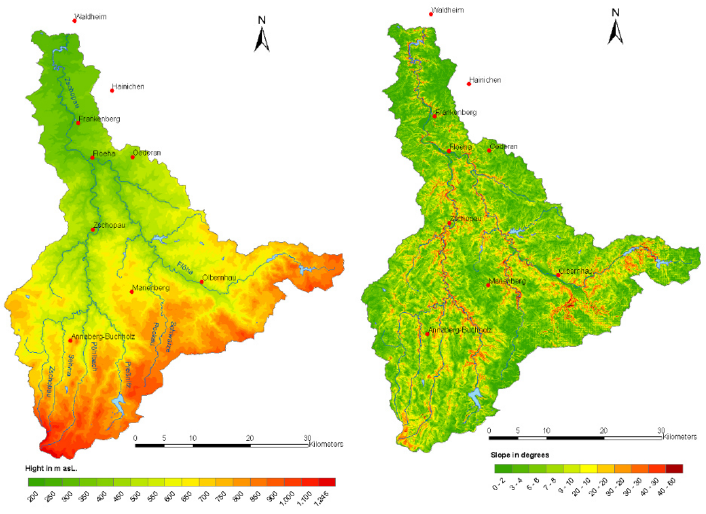


Figure 5.1: Digital elevation model (left) and the slope distribution (right) of the test catchment

Figure 5.2: Upper reaches of the Freiberger Mulde (Schwarze Pockau)[a]

Figure 5.3: Lower reaches of the Freiberger Mulde near Westewitz[a]

[a]Source: Wikimedia Commons (`http://commons.wikimedia.org`)
Copyright: GNU Free Documentation License (`http://www.gnu.org/licenses/fdl.txt`)

generates a big set of synthetical meteorological time series, causing extreme flood events. The assigned runoff is computed by a process based model, especially calibrated for extreme flood events. Together with a set of measured historical time series they form a training database for a neural network. The meteorological input data is preprocessed and summarizes in an adequate way and after a specific training process the neural network is able to perform flood forecasting for the considered catchment.

The whole approach was developed and tested on the example of the Freiberger Mulde catchment area till the gauge at Erlln (figure 5.1). The rough and steep upper reaches (figure 5.2) were modeled by the rainfall-runoff model WaSiM-ETH, because they are almost exclusively dominated by the processes of runoff formation and concentration. However the lower reaches (figure 5.3) are dominated by hydrodynamically influenced processes as backwater effects and river junctions. Thus the lower reaches were portrayed by the hydrodynamic river model HEC-RAS, which provides good abilities for flood routing. The training database of synthetical weather events causing extreme floods was generated by overlaying the historical precipitation time series with the output of a rainfall generator, which was extra designed for the considered catchment. For the architecture of the used ANN, a PoNN (section 3.3) was preferred to a MLFN, because it gave significantly better results [Cul06].

Table 5.1: Used input data

Input Data Type	Symbol	scalar	distributed
Meteorological Input Data			
Global Radiation	R	✗	✓
Precipitation	P	✗	✓
Relative Air Humidity	e	✗	✓
Temperature	T	✗	✓
Vegetation	V	✓	(✗)
Wind Speed	u	✗	✓
Hydrological Input Data			
Runoff at reference gauges	Q	✓	✗

This chapter delivers a more detailed insight into the PAI-OFF approach. First, there is given a presentation of the preprocessing of the input data. This includes the spatial subsummation of distributed meteorological input data by zones and a further temporal preprocessing of the estimated time series, to gain a significant and sufficient set of *features* which form the input of an ANN. Subsequently, there follows an overview of the used ANN architecture in the context of PAI-OFF, including a specific training algorithm for this domain. The chapter is rounded of by a summary of the PAI-OFF approach, as a form of an intelligent system.

5.1 Pre-Processing of the Input Data

Before operating an ANN for runoff prediction there has to be a pre-processing mechanism which summarizes the huge amount of input data to reduce the number of input variables for the ANN used for the runoff prediction. This process has to be done with domain-specific knowledge to gain as much significant information as possible, as kind of filter of the essential information. PAI-OFF deals with different types of meteorological and hydrological input data. According to table 5.1 there are distributed and scalar input values. Note that in the current version the values of global radiation, relative air humidity and wind speed are still ignored. A special input value is the vegetation. It estimates the vegetation coverage degree, which is a crop and catchment specific function of date as a annual course.

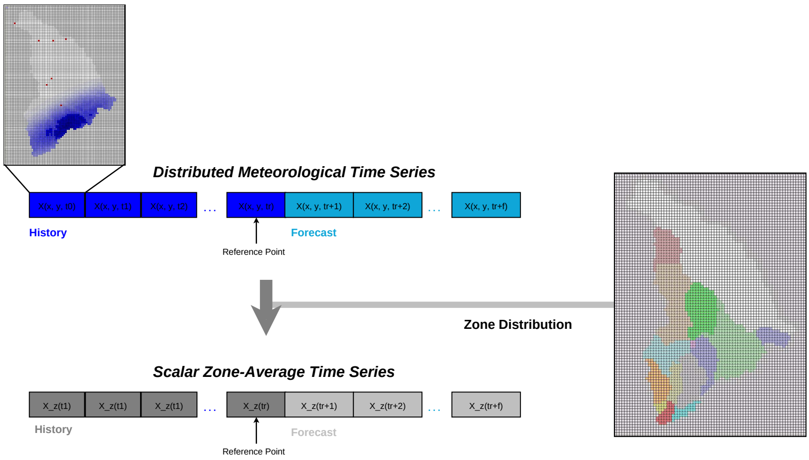


Figure 5.4: Scheme of spatial pre-processing

This input value is important to represent the influences of interception and evapotranspiration. Note, that the ANN in PAI-OFF takes a scalar time series with intermediate value of the vegetation of the annual course as input, whereas the process based model WaSiM-ETH is driven by crop and catchment specific values, which are additionally adapted by an orographic adjustment factor.

5.1.1 Spatial subsummation of meteorological data

As a first pre-processing step time series of distributed meteorological input data in raster form $X(x,y,t)$ is subsumed according to specific spatial zones which are subareas of the main catchment. This process is called *spatial subsummation*. It is a simple computation of the average of all grid cells, a particular zone consists of, and yields time series of scalar zone-average data $X_{zone}(t)$. The zones for a specific catchment are determined by the geomorphologic structure and domain-specific knowledge. Because of this first each considered catchment structure has to be observed and classified by persons with expert knowledge in the domain of rainfall-runoff modelling. According to this survey, a set of zones is defined which is stored in a suitable database. Thus the catchment is divided into different zones which help to summarize the distributed meteorological input data to scalar zone-average data. An illustration of this process is given in figure 5.4.

The underlying idea of this approach is the following. The pre-processing of the meteorological input data has to extract several features which express several significant aspects of the current state of the catchment. An aspect can be for example the monthly or average temperature in the top mountain region, or the average relative air humidity in a specific subcatchment. Correspondingly to the significant aspects, one spatial zone is defined as an area which is similar in respect of at least one or even more aspects. Unfortunately the spatial distribution according to different aspects is not every time the same in the catchment. Because of this, PAI-OFF defines different classes of zones, where one class comprises all spatial zones that were formed according to a specific aspect. After some considerations with hydrological and meteorological background knowledge it is reasonable to define the following three classes of zones:

- **Orographic Zones (OROZ)**
- **Subcatchments (SCMT)**
- **Travel-Time Zones (TTZ)**[1]

In the following there is given a short explanation of the meaning and definition of the different zone classes. The first class of zones is the **orographic zone (OROZ)** which represent the global situation in different orographic regions with a similar altitude. The definition of this class is reasonable due to the fact, that the weather situation in neighboring areas with similar altitude is approximately the same considering advective events, which are the predominant events, causing extreme flood events. The second zone-class is the **subcatchment (SCMT)**. It represents the situation in a specific subcatchment of the whole catchment area. This areas form the runoff of particular preflooders which play an essential role for the whole catchment. The class of **travel-time zones (TTZ)** summarizes zones of similar *hydrological response*. That means that a unit impulse in form of short intensive precipitation event causes approximately the same response at the catchment outlet for each point within this zone (chapter 2.1.2). This implicates, that the runoff from one TTZ can be considered as approximately

[1]only for precipitation data

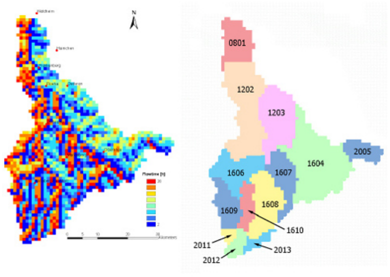


Figure 5.5: Travel-times (left) and according zones (right) at the Freiberger Mulde according to [Cul06]

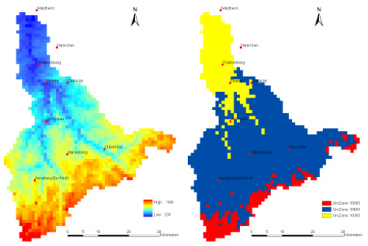


Figure 5.6: Digital elevation model (left) and according orographic zones (right) at the Freiberger Mulde according to [Cul06]

equal. The unit response of a TTZ is called *unit hydrograph*. A TTZ can only be defined for precipitation input. Within later pre-processing steps the zone-average precipitation of a TTZ is convolved with the unit hydrograph of the zone. Further details are given in the next section. An example of a distribution of the orographic and travel-time zones in the test catchment is given in the figures 5.5 and 5.6. Concluding, within the first pre-processing step the distributed spatial information of the meteorological input data is summarized into scalar information. This is done by the help of characterizing subordinated zones of the catchment.

5.1.2 Temporal compression of time series

The spatial subsummation is followed by a *temporal compression* of the time series of computed zone-averages or regular scalar input. This process yields the features which form the input of the postfixed PoNN. The compression in time is performed by a set of operators and follows these two fundamental hydrological laws:

1. One of the best factors to describe the state of the catchment is the current runoff.

2. The predicted runoff strongly depends from the current state of the catchment and the hydrologic responses of subordinated areas of the

catchment with approximately the same travel time to the catchment outlet.

Table 5.2: List of defined state operators

Operator	Type of Input Data							
	Q	R	P		e	T	V	u
			OROZ & SCMT	TTZ				
D	✔	✔	✔	✔	✔	✔	✔	✔
M	✔	✔	✔	✔	✔	✔	✔	✔
W	✔	✔	✔	✗	✔	✔	✔	✔
G	✔	✗	✗	✗	✗	✗	✗	✗
H	✔[a]	✗	✔[a]	✗	✗	✗	✗	✗
J	✗	✗	✔	✗	✗	✗	✗	✗
K	✗	✗	✔	✗	✗	✗	✗	✗
N	✔[a]	✗	✔[a]	✗	✗	✗	✗	✗
R	✗	✗	✔	✗	✗	✗	✗	✗
V	(✔)[b]	✗	(✔)[b]	✗	✗	✗	✗	✗

[a]Operator has different definitions according to input type
[b]Operator uses a combination of inputs

Following these laws, PAI-OFF defines two classes of operators to estimate the features of the PoNN. The first class of operators, the *state operators*, are responsible to estimate the state of the catchment, expressed by the **state features**. PAI-OFF defines a whole set of state operators. Consider that not any operator can be applied to every type of input value and that the definition of some operators can differ with respect to the type of input data. An overview over the defined state operators and their valid domains is given in table 5.2. It is also possible to apply the state operators to time series which include history and forecast, as a kind of trend. A state operator θ_{state} is applied to a defined interval of a scalar time series $\vec{X}$ and computes a scalar Y as output variable as given in equation 5.1.

The second class of operators, the *hydrologic response operators*, is defined as follows. An hydrologic response operator θ_{hr} takes the same type of input as a state operator, but computes again a scalar time series $\vec{Y}$ as given in equation 5.2. Currently PAI-OFF defines only one operator of that type, the partial flow operator P. This operator is responsible to compute an estimated actual hydrologic response of subordinated regions of the catchment with approximately the same hydrologic response. Thus, the computed features are called **hydrologic response features**. P convolves the normalized hydrologic response of a particular TTZ, the unit hydrograph, with its average precipitation time series, including history and forecast. Thereby you create a possibility to feed the predicting PoNN with physically based knowledge about the runoff behavior of the catchment.

$$Y = \theta_{state}(\vec{X}) \qquad (5.1) \qquad\qquad \vec{Y} = \theta_{hr}(\vec{X}) \qquad (5.2)$$

It is important to mention, that the output of each operator is scaled with a scaling factor s_f and a scaling constant s_c to the interval [0,1] (equation 5.3). This yields normalized features F which are then used as input of the predicting neural network.

$$F = s_f \cdot Y + s_c \qquad (5.3)$$

An overview of the temporal compression and pre-processing is presented in figure 5.7. Keep in mind that each operator can be applied several times to each input value. First, there exist different zone-average time series in respect to the spatial subsummation by zones. Additionally it is possible to apply one operator to one and the same time series several times, because the applications can differ in the length and type (history and/or forecast) of the included interval. Of course, not all potentially possible applications of operators to the input are taken into account to compute the input features for a specific catchment. The goal of the pre-processing is to reduce the number of input variables and therefore the most significant features are estimated by statistical methods associated with domain-specific knowledge about rainfall-runoff processes with special attention to the specific catchment. After the significant features of a specific catchment were identified, their definitions are stored in a database. Information more in detail is presented in [Cul06] and [Gör07].

By now, there is given a detailed definition of the different state operators with a short description. For all the following definitions we presume, that the operator θ is applied to a history interval $[L - t_0, t_0]$ of defined length L

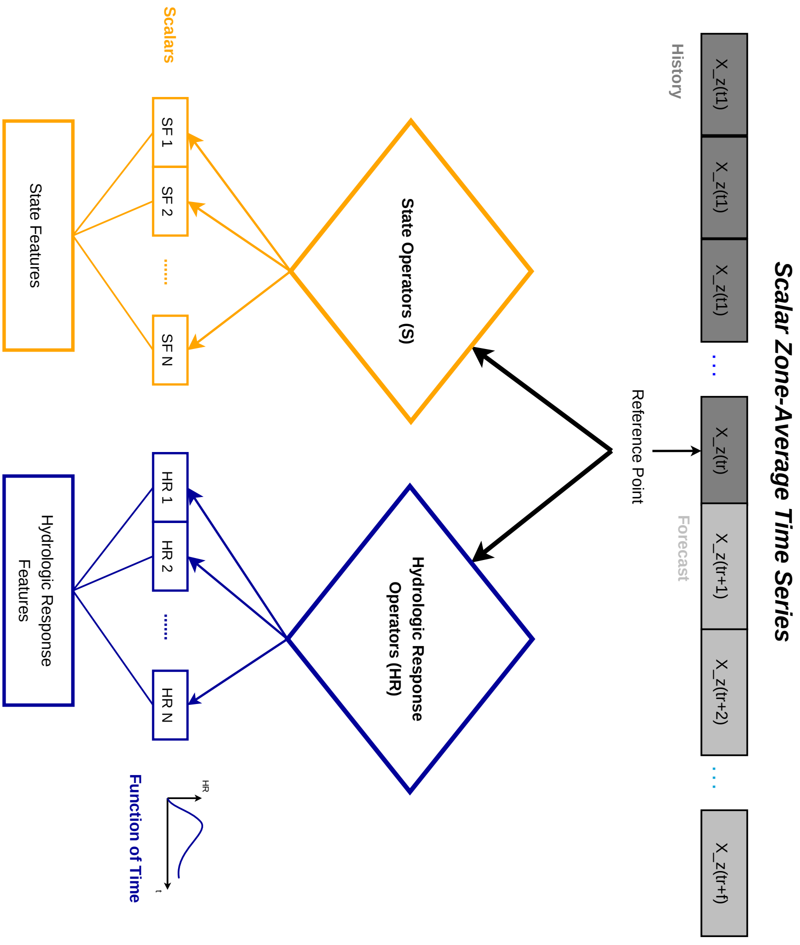


Figure 5.7: Scheme of temporal pre-processing

previous to a reference point t_0. The first set of the three state operators D, M

and W are the **global operators**. They are valid for each type of input $\vec{X}$, and are always defined in the denoted way. An exception is formed by operator W which is not defined for time series subsumed by travel-time zones. Time series of that type are always convolved with operator P. Additionally you are supposed to keep in mind that the application of the operator D is in general only recommendable for zone average time series of larger areas or time series of original scalar input values as for example the runoff.

- **D**: Direct value
$$Y(t_0) = X(t_0) \tag{5.4}$$

- **M**: Medium value in the history interval
$$Y(t_0) = \frac{1}{L}\sum_{l=0}^{L-1} X(t_0 - l) \tag{5.5}$$

- **W**: Convolution of the history interval with a definable exponential core
$$Y(t_0) = \frac{\sum_{l=0}^{L-1} X(t_0 - l)\, e^{-\frac{l}{\tau}}}{\sum_{l=0}^{L-1} e^{-\frac{l}{\tau}}} \tag{5.6}$$

The next set of state operator definitions were especially defined for time series of **precipitation** values $\vec{P}$. An exception are again time series summarized by a travel time zone. Consider that operator H has another definition for runoff values.

- **H**: Maximum value of a smoothed curve of the precipitation P_s in the history interval with an exponential weighting over its lag

$$P_s = \frac{1}{5} \sum_{l=-2}^{2} P(t+l)$$
$$Y(t_0) = \max \left[P_s(t_0 - l)\, \mathrm{e}^{-\frac{l}{\tau}} \right]_{L=0}^{L} \tag{5.7}$$

- **J**: Number of samples with a precipitation higher than a defined minimal value Θ

$$Y(t_0) = \sum_{l=0}^{L-1} IIf^{a} \left(P(t_0 - l) > \Theta, 1, 0 \right) \tag{5.8}$$

[a]IIf is an abbreviation for the immediate if function, which is defined as $IIf(expr, truepart, falsepart)$.

- **K**: Duration of the longest period without any precipitation in the history interval with an exponential weighting over its lag
  d_k: duration of the k-th rainless period
  L_k: lag of the end of the k-th rainless period

$$Y(t_0) = \frac{1}{L} \max \left[d_k\, \mathrm{e}^{-\frac{l_k}{\tau}} \right]_k \tag{5.9}$$

- **R**: Relation of peak and medium value within the history interval

$$Y(t_0) = \frac{\max \left[P(t_0 - l) \right]_{l=0}^{L}}{\frac{1}{L} \sum_{l=0}^{L-1} P(t_0 - l)} \tag{5.10}$$

Below, there follow the definitions of state operators applicable to time series of the **runoff**, $\vec{Q}$, at reference gauge measurements.

- **G**: Gradient of the last five time steps

$$Y(t_0) = \frac{1}{60}(-137Q(t_0) + 300Q(t_0-1) - 300Q(t_0-2) + 200Q(t_0-3) - 75Q(t_0-4) + 12Q(t_0-5) \tag{5.11}$$

- **H**: Low water within the history interval with an exponential weighting over its lag

$$Y(t_0) = \max\left[Q(t_0) + [\overline{Q(t_0-l)} - Q(t_0)]\,\mathrm{e}^{-\frac{l}{\tau}}\right]_{l=0}^{L} \tag{5.12}$$

- **N**: High water within the history interval with an exponential weighting over its lag

$$Y(t_0) = \min\left[Q(t_0) + [\overline{Q(t_0-l)} - Q(t_0)]\,\mathrm{e}^{-\frac{l}{\tau}}\right]_{l=0}^{L} \tag{5.13}$$

The last definition of a state operator, the pre-moisture V according to Schwarze/Ulrich [US04], is an especialness. It applies two different state operators N and M to their input domain and multiplies the results. Thus it takes a **combination of two inputs**, namely from the monthly average of the precipitation in the catchment $M(\vec{P})$, and monthly low water $N(\vec{Q})$ and computes the output.

- **V**: Pre-moisture according to Schwarze/Ulrich

$$Y(t_0) = N(\vec{Q}) * M(\vec{P}), L = 720 \tag{5.14}$$

The **partial flow operator** operator P is a hydrologic response operator. It is only defined for the application to time series of subsumed medium values of travel-time zones. These time series have to contain both history and forecast data and, as already declared in section 5.1.1, a subsummation by travel-time zones is only allowed for precipitation input data $\vec{P}$. The operator P convolves the history interval with a core $\vec{K}$, the *unit hydrograph*, according to the linear storage unit model. The parameters of the hydrologic response c, K_1, K_2, N, L_0 are identified from the training base. L_0 is the flowtime from the outlet of the reference zone to the reference gauge. This operator computes again a time series while it is applied to all samples between the reference point for the prediction t_r and the end of the forecast.

- **P**: Partialflow

$$\begin{aligned} K(l) &= c \cdot \left(\frac{l - L_0}{K_2} \right)^{N-1} \mathrm{e}^{-\frac{l-L_0}{K_1}} \\ Y(t_0) &= \sum_{l=0}^{L-1} P(t_0 - l) K(l) \end{aligned} \tag{5.15}$$

5.2 Operating and training the PoNN

After the preprocessing of the input data there follows the actual prediction of the runoff with a *sigma-pi polynomial neural network (PoNN)* (section 3.3). More precisely the prediction is done by a set of PoNNs. Thereby each PoNN takes a set of the detected features from the preprocessing and computes the runoff for a specific lead time (figure 5.8). The runoff curve between the computed sampling points is computed by linear interpolation. In a further development of the first approach of PAI-OFF, the sampling points are interpolated by ***natural cubic splines*** [Kno00]. It turned out in practice, that a step width of two hours between the lead times of the different neural networks is a good compromise between computational effort and feasibility of the system with respect to accuracy of the prediction.

5.2.1 Design and functionality

The fundamental design of the used PoNN appears simple but is absolutely specific to the feature classification in the preprocessing. The features are summarized by **optimizing product vectors** or short *polynomials* which form the output as described by equation 5.16 and 5.17.

The absolute runoff Q or a relative difference to the current runoff ΔQ is computed by adding a bias weight w_0 and the weighted product vectors $w_i P_i$. The weights are determined in the training. In general it is better to use the difference approach (equation 5.17) for the prediction of short lead

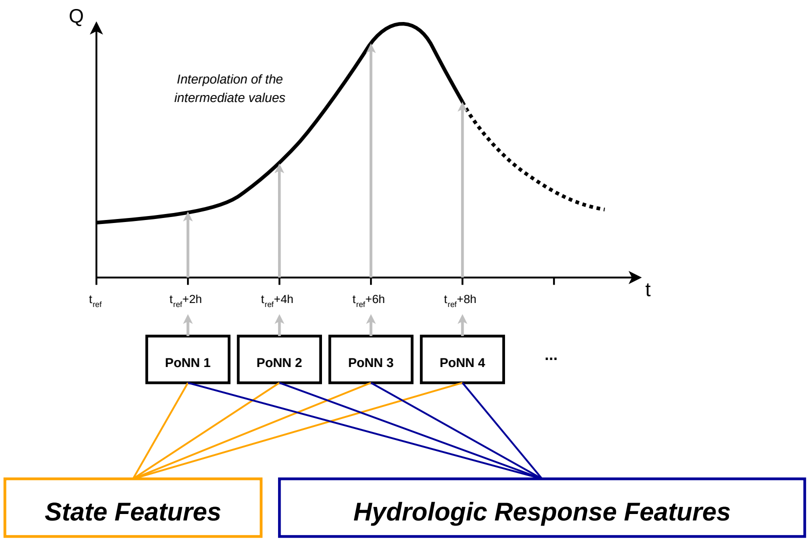


Figure 5.8: Prediction of the runoff with as set of PoNNs; One PoNN computes the runoff for a specific leading time. The intermediate values are interpolated by natural cubic splines

$$Q = w_0 + \sum_{i=1}^{I} w_i P_i \tag{5.16}$$

$$\Delta Q = w_0 + \sum_{i=1}^{I} w_i P_i \tag{5.17}$$

times (<8h) and the absolute value prediction (equation 5.16) for greater lead times.

The product vectors represent the domain-specific knowledge based feature classification with state features and hydrological response features in the preprocessing. Each polynomial is a multiplication of four factors. The first three factors are a permutation of the state features and the last one is a hydrologic response feature. Also lower degrees are possible by substituting one feature with the neutral element of the multiplication, one. Thus, with this definition the following types of polynomials are possible.

$$
\begin{aligned}
&\textbf{Degree 1S+0HR:} && P_{10n} = S_i|_{i=1}^{I} \\
&\textbf{Degree 0S+1HR:} && P_{01n} = F_j|_{j=1}^{3J} \\
&\textbf{Degree 1S+1HR:} && P_{11n} = S_i|_{i=0}^{I} F_j|_{j=0}^{3J} \\
&\textbf{Degree 2S+1HR:} && P_{21n} = S_i|_{i=0}^{I} S_k|_{k=i}^{I} F_j|_{j=0}^{3J} \\
&\textbf{Degree 3S+1HR:} && P_{31n} = S_i|_{i=0}^{I} S_k|_{k=i}^{I} S_l|_{l=k}^{I} F_j|_{j=0}^{3J}
\end{aligned}
\tag{5.18}
$$

Consider that the hydrological response features are used each threefoldly in a modified form ($F(-L)$, $F(0)$, $F(L)$). There are three possible modification methods, either phase-delayed ($F(t_{ref} - t_{Lag})$, $F(t_{ref})$, $F(t_{ref} - t_{Lag})$), exponentiated ($F(t_{ref})$, $F(t_{ref})^2$, $F(t_{ref} - t_{Lag})^3$) or with extracted root ($\sqrt{(}F(t_{ref}))$, $F(t_{ref})$, $F(t_{ref} - t_{Lag})^2$) which is in the end another form of exponentiation. Due to the huge amount of possible permutations[2], not all product vectors are considered. An optimal subset from all possible permutations is determined within the training procedure.

5.2.2 Training algorithm

The used PoNN was trained with the new algorithm of ***stepwise serial regression (SSR)***. The algorithm was especially developed for the training of PoNNs in the context of rainfall-runoff processes [Cul06] [SCG$^+$05]. It consists of a combination of regression methods [Cul06] using both *Efroymson's algorithm* as described in [Mil96] and *stepwise regression* according to [MBS70]. The goal of the algorithm is to estimate an optimal subset of product vectors.

The strategy of the algorithm is the following. First, a set P of all possible product vectors is composed by permutation of the input features. Within the PoNN, used in the PAI-OFF approach, up to three state features and one or none hydrologic response feature are permuted. Let the cardinality of P be n. Assuming that the algorithm's goal is to compute an optimal subset of polynomials O with cardinality m, it takes a small subset S from P with a cardinality approximately 30% greater than m, whose size is still manageable

[2]Applying PAI-OFF to the test catchment of the Freiberger Mulde, approximately 8000 polynomials were considered [Gör07].

for a serial regression with respect to the computational effort. Now a serial regression is performed to S according to the criterion of minimum medium square error of the general estimator of the targets. Thereby the general estimator of the target is defined a

$$\Delta(x) = w_0 + \sum_{i=1}^{m} w_i x_i \tag{5.19}$$

where w is the weight and x is the regressor. Minimizing the objective function in the selective process, the deviation of the target value and the dependent variable $\Delta(x)$) can be expressed as follows [Cul06]

$$E\{[z - \Delta(x)]^2\} \Rightarrow Min \tag{5.20}$$

Note that the regression is performed over the whole training set, which estimates the best m product vectors in S and their assigned weights. A detailed description of the regression is given in [MBS70]. After the regression, the worst members (approx. 30%) of S are rejected and filled again by still unused product vectors from P and the linear regression is carried out again. The algorithm times, if there are no more unused product vectors in P. A schematical illustration of the SSR algorithm is given in figure 5.9.

During the development of the PAI-OFF approach the SSR algorithm was improved by two new approaches. The first one is the *regional products approach*. Its idea is to multiply only features which represent the state of same or adjacent subareas. Another approach is the definition of *obligatory terms*. The underlying idea is to force the use of some product terms due to expert knowledge for the particular rainfall-runoff processes in the specific catchment. Both approaches delivered still further improvements of the runoff prediction.

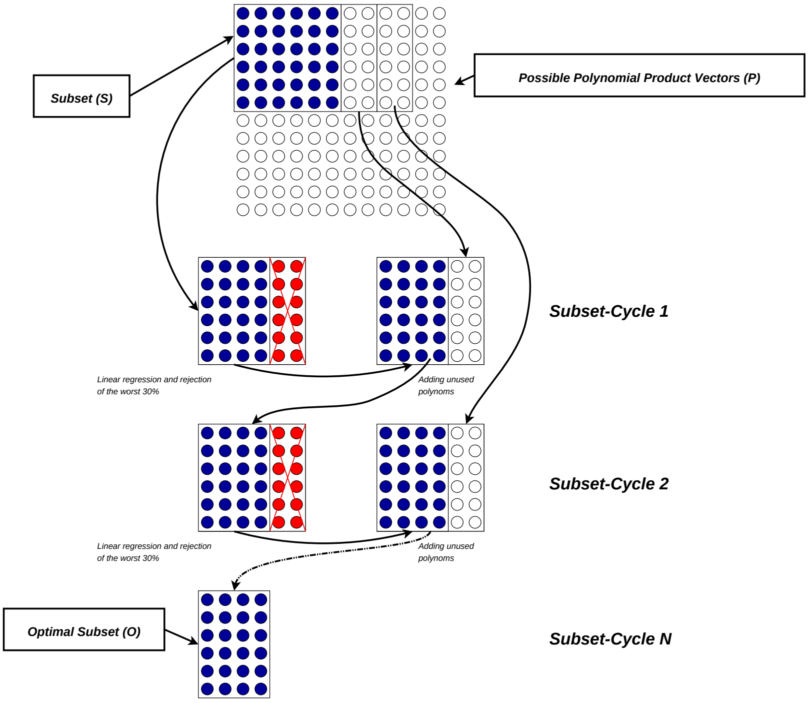

Subset (S)
Possible Polynomial Product Vectors (P)
Subset-Cycle 1
Linear regression and rejection of the worst 30%
Adding unused polynoms
Subset-Cycle 2
Linear regression and rejection of the worst 30%
Adding unused polynoms
Optimal Subset (O)
Subset-Cycle N

Figure 5.9: Scheme the of serial regression

5.3 The PAI-OFF approach - an Intelligent System

This chapter is concluded by a short summary of the PAI-OFF approach. We consider the PAI-OFF system as an intelligent agent and assume a new application to an arbitrary catchment. To use the agent as an operational flood forecasting system, the following operations have to be performed. An illustration is given in figure 5.10, definitions of used auxiliary algorithms are given in figure 5.11.

The first step is a training of the agent according to function Train-PAI-OFF-Agent. Therefor the agent takes a historical time series of weather and measured runoff and all available information of the catchment area which is relevant with respect to rainfall-runoff modelling.

1. Initially, the catchment is analyzed. As a result of the analysis, a specific set of zones and significant features is defined. Note that this step still requires a person with domain-specific knowledge. It is aspired to formalize this process, to do this process at least semi-automatically.

2. Then a suitable training database is generated by overlaying the historical time series with synthetical extreme events and computing an associated runoff with a process based rainfall-runoff model.

3. Subsequently the input data of the training database is summarized according to the defined zones (spatial preprocessing) and the operators of the defined significant features (temporal preprocessing).

4. The detected set of feature values for all reference points of the training database and the associated computed runoff form the training set which is used to train polynomial neural networks for different lead times. By now, the set of detected zones, features and the trained neural network form the new trained PAI-OFF agent.

In the operational stage, the trained agent can perform runoff predictions. thereby it takes a time series of meteorological and hydrological data of the pre-event and a time series of the weather forecast as input and computes a runoff prediction for different lead times, according to function Operate-PAI-OFF-Agent. The preprocessing method is the same as in the training algorithm. The detected features are applied to the trained neural networks for the considered lead times and a runoff prediction is computed.

```
function TRAIN-PAI-OFF-AGENT(catchment, history) returns a trained agent
    inputs:   catchment, all relevant information about the catchment
              history, a historical time-series of weather and runoff
    static:   zones, a subdivision of the catchment
              features, a set of significant features
              net, a set of polynomial neural networks, one for each lead time

RESET-AGENT()
zones ← ANALYZE(catchment)
features ← ANALYZE(catchment)
training-db ← GENERATE-TRAIN-DATABASE(history)
training-set ← []
for each m = <wea, runoff> in training-set
    hist ← COMBINE(<wea.hist, runoff.hist>), fcast ← wea.fcast
    zone-avgs ← SPATIAL-PREPROCESSING(zones, hist, fcast)
    feature-vals ← TEMPORAL-PREPROCESSING(features, zone-avgs)
    training-set[m] ← COMBINE(feature-vals, runoff.fcast)
for each lead-time in net
    net[lead-time] ← STEPWISE-SERIAL-REGRESSION(training-set)
agent ← COMBINE(<zones, features, net>)
return agent

function OPERATE-PAI-OFF-AGENT(agent, history, forecast) returns a runoff
prediction
    inputs:   catchment, all relevant information about the catchment
              history, time-series of weather and runoff of the pre-event
              forecast, time-series of current weather forecast
    static:   zones, a subdivision of the catchment
              features, a set of significant features
              net, a set of polynomial neural networks, one for each lead time

features ← agent.features
net ← agent.net
zone-avgs ← SPATIAL-PREPROCESSING(zones, history, forecast)
feature-vals ← TEMPORAL-PREPROCESSING(features, zone-avgs)
for each lead-time in net
    runoff[lead-time] ← 0
    for each polynom in net[lead-time]
        ADD(runoff[lead-time] , CALCULATE(polynom, feature-vals))
return runoff
```

Figure 5.10: The intelligent PAI-OFF agent

```
function GENERATE-TRAIN-DATABASE(history) returns a training-database
    inputs:  history, a historical time-series of weather and runoff

extremeweather ← OVERLAY-EXTREME-EVENTS(history)
extremerunoff ← RUN-PROCESS-BASED-RR-MODEL(extremeweather)
training-db ← COMBINE(<extremeweather, extremerunoff>)
return training-db


function SPATIAL-PREPROCESSING(zone, hist, fcast) returns a set of time-series
of zone averages
    inputs:  zones, a subdivision of the catchment
             hist, time-series of weather and runoff of the pre-event
             fcast, time-series of the weather forecast

zone-avgs ← []
for each z in zones
    zone-avgs[z] ← AVERAGE(z, hist, fea)
return zone-avgs


function TEMPORAL-PREPROCESSING(features, time-series) returns a set of
feature values
    inputs:  features, a set of significant features
             time-series, a set of scalar time-series

feature-vals ← []
for each fea in features
    feature-vals[fea] ← APPLY(fea, time-series)
return feature-vals


function STEPWISE-SERIAL-REGRESSION(training-set) returns a trained PoNN
    inputs:  training-set, a training set with features and target runoff

P ← PERMUTE(training-set.feature-vals)
S ← TAKESUBSET(P, S.size)
while P != empty
    S ← REMOVE(SERIAL-REGRESSION(S), 30%)
    O ← S
    S ← TAKESUBSET(P, S.size-O.size)
net ← O
return net
```

Figure 5.11: A set of auxiliary algorithms, used by the PAI-OFF system

VI

Design and Implementation

The concepts and approaches presented in the previous chapters were implemented in a flood forecasting class framework FloodNet, embedded in a platform independent program library. This chapter gives an overview of the design and implementation of the class framework FloodNet. This is followed by a short presentation of the exported program interface of the library and a discussion of an appropriate visualization of the output data, afflicted with uncertainty. This topic is not sourced out into a separate chapter, because the interpretation of uncertainty in the output data is tightly coupled to an adequate visualization.

6.1 Design

The design of the class framework FloodNet follows the principles of *modularity* and *exchangeability*, formulated in chapter 4. The issues of platform independence and performance are discussed in the coming section on details of the implementation. Nevertheless you have to consider that a good performance also bases upon an intelligent design and a clever setup and selection of fast algorithms.

Although the design is independent from the chosen programming language, the modular design of the framework can be implemented best in an *object-oriented programming (OOP)* language. OOP provides many facilities to define abstract interfaces which can be realized by many exchangeable implementations, not to mention the possibility to encapsulate data-structures and according methods in classes and to structure different classes by the principles of association and inheritance.

6.1.1 Modules

To achieve a clearly structured design and a distribution of concerns, the system was divided into different *modules*. Thereby each *module* has a clearly structured interface and a well-defined field of responsibility. All modules in

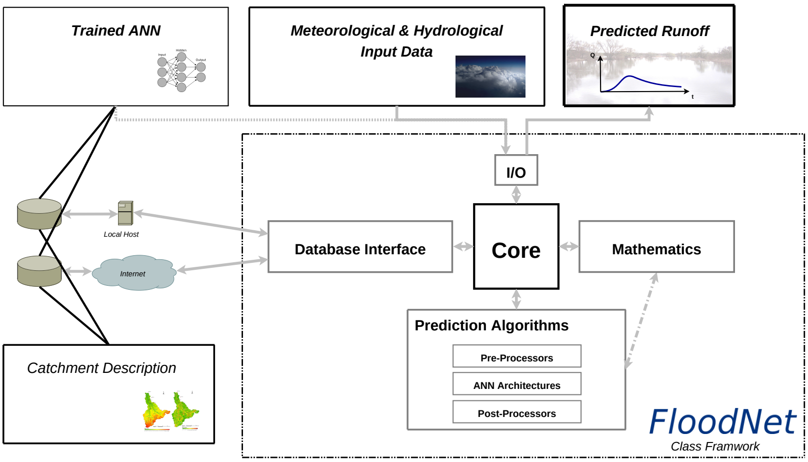


Figure 6.1: Design of the FloodNet class framework

the system are controlled by the main unit, the **core module**. It processes all calls from the user and delegates it to the other modules with the functions, they provide. An overview of the functions and interactions of the modules is presented in figure 6.1. By now, there is given a more detailed description of the different modules.

- The linchpin of the framework is the **core module**. It holds the input data in a adequate data structure, provides basic methods to perform the required operations on this data and controls all processes in the framework. Additionally it exports an interface, to use and control the framework.

- To get access to all required catchment information and the trained ANNs, the FloodNet framework uses a platform independent **database interface**. This interface can be implemented to access different architectures of relational databases at the local host or database servers in a network which hold the required information. Currently there exists no established entity-relationship model of the databases used for the PAI-OFF approach, so that it is necessary to provide an adaptable database for the framework.

- The input and output data is read and written with the **I/O module**. It enables a file format independent handoff of the data within buffers of values. The most common file formats (WaSiM-ETH file format and the new developed FloodNet file format) are supported. The design also enables the possibility to store and access trained ANNs in a specific file format with the help of the I/O module. An appropriate example is given in [Thi98].

- All needed mathematical algorithms, were implemented in a special **mathematics module**. It provides efficient mathematical algorithms, required by different prediction algorithms. Examples are the convolution operation as a sub-routine for the preprocessing or the interpolation by natural cubic splines for the interpolation of intermediate output values.

- The actual prediction is done by methods provided by the **prediction algorithms module**. It integrates a set of methods for preprocessing the input data to get a set of detected features, operating a trained ANN with the detected features and perform a post-processing on the computed output, for example an interpolation or a statistical analysis.

6.1.2 Universal operations by visitors

Besides the discussed issue of modularity, the class framework also provides a high capability of exchangeability. In terms of supported input data formats and database architectures, this is achieved by abstract interfaces which can be implemented for each database architecture or file format.

In the field of pre-processing algorithms a facility for substitution can be achieved by a global mechanism to perform operations on the input data. This mechanism is realized by the use of the ***visitor pattern*** on the fixed data structure of the input hold in the core. Since it is obvious that the variety of input data will not increase even in the remote future, we can assume the data structure to be fixed. Now for each type of operation (pre-processing, prediction with an ANN and post-processing), there has to be an abstract interface, inherited from a specific *visitor* for the data structure in the core. Concrete implementations have to implement the abstract algorithms. This mechanism provides a possibility to substitute different prediction algorithms. An illustration of this technique is given in figure 6.2.

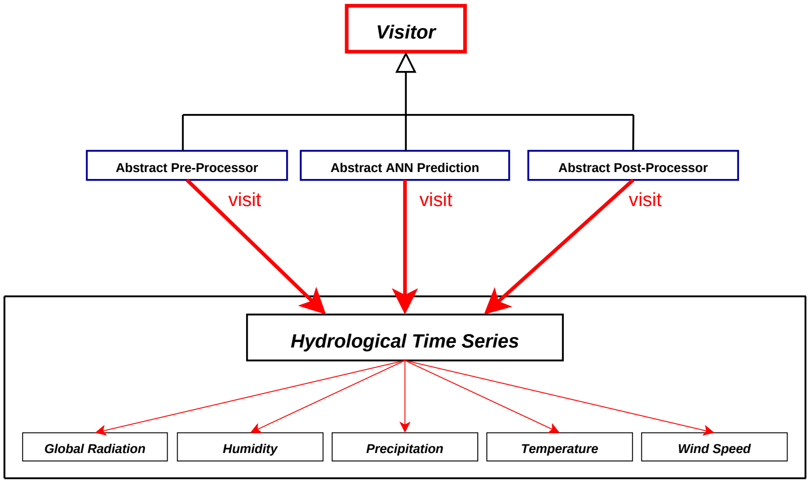


Figure 6.2: Performing substitutable operations using the visitor pattern

6.2 Implementation

This section describes important aspects of the implementation of the FloodNet class framework. It focuses on the compliance of the goals of *modularity*, *platform independence* and *performance*, and on the techniques and tools that were used to reach these goals.

6.2.1 Structure

The modular design and a distribution of concerns is realized with the implementation. That is done by the principles of OOP that are supported within C++. First, the class framework is divided in different namespaces. Each namespace consists of a set of classes that implement the functionality of the module. The different namespaces are organized in one namespace global namespace *floodnet*, which represents the FloodNet class framework (figure 6.3).

Of course the modularity and exchangeability are not limited to this global distribution, but are proceed in the implementations of the modules. Each module is implemented under the aspect of an easy substitution of various implementations using different algorithms or supporting different

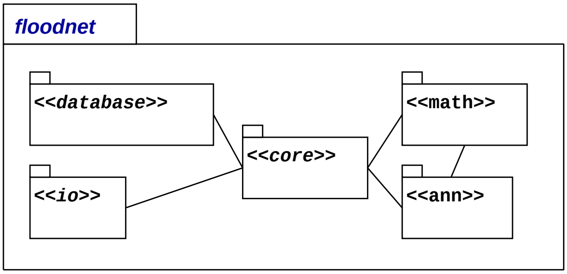


Figure 6.3: Different namespaces in the FloodNet class framework

architectures. This is done with the help of the concept of *interface* in OOP. An object's *interface* consists of a set of methods that the object must respond to. Other modules only use the object's interfaces and not the object implementations itself. Because of this, it is possible to create and use new object's implementations without any further adaptations in other modules. Two examples using this technique, the database interface and the input processor, are presented in figure 6.4.

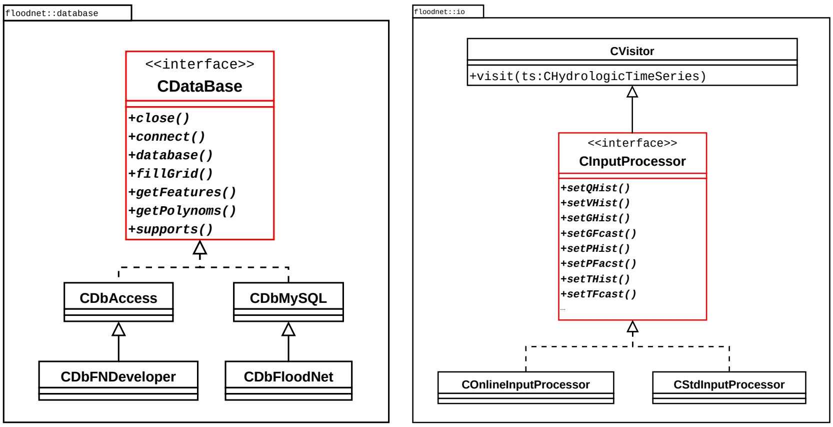


Figure 6.4: Different implementations of the interfaces of database connection (left) and input processor (right)

6.2.2 Tools

One of the most important questions at the beginning of the development process was the question of the used programming language. Alone the name "class framework" and especially the modular design presuppose the use of an object-oriented programming language. Although Java is almost the standard for platform independent OOP, C++ was the language of first choice for the implementation. It adapts perfectly to the existing structures in the planned operational area and fulfills the issues of platform independence and performance. There exist many compilers for almost all possible target platforms which produce fast machine code. With the help of cross-platform programming libraries an access to system components can be implemented in a simple way.

To cope with the challenge of a platform independent and performant implementation, the FloodNet class framework uses the following set of cross-platform programming libraries, providing efficient and useful functions:

- The C++ standard library provides an efficiently implemented set of standard data structures (arrays, lists, sets, maps etc.) and related algorithms.
- An efficient and powerful implementation of data structures and algorithms for date and time is provided by the Boost C++ libraries[1]. It also offers an implementation of a dynamic bitset.
- A platform independent access to the file system for I/O operations and a connection to different relational databases is provided by the Qt[2] framework.
- An efficient implementation of the fast Fourier transform is provided in the FFTW library. Details follow in section 6.2.3.

The actual implementation was written according to the standards of the GNU Compiler Collection (GCC)[3] and tested on the Linux and Win32 platform with the GCC and MinGW compiler[4] respectively. A developer

[1]http://www.boost.org/
[2]http://trolltech.org/products/qt
[3]http://gcc.gnu.org/
[4]http://www.mingw.org/

documentation of the complete class framework was created with the help of the source code documentation generator tool Doxygen[5].

6.2.3 Performance

One important requirement of the implementation was a high performance to be able to point out the advantages of a neural based rainfall-runoff system to physically based models. Thus, there was tried to optimize the most frequently used and computationally most intensive algorithms in the system, especially the methods in the *mathematics module*. Moreover many utilities of the language C++ were utilized to yield an efficient implementation, e.g. inline functions or call by reference to avoid unnecessary memory allocation.

A special example is the *discrete convolution operation* which is implemented in the mathematics module. It is frequently used within the temporal preprocessing of scalar time series and has a high computational complexity. For finite-domain discrete-time signals the convolution operation is defined as follows

$$h(n) = (f * g)(n) = \sum_k f(k) \cdot g(n-k) \tag{6.1}$$

where the convolution operator h takes two functions f and g and produces a third function that in a sense represents the amount of overlap between f and a reversed and translated version of g [BP84]. This operation can also be represented by the *fast convolution algorithm* which is expressed as

$$\begin{array}{lclcl} H(N) & = & \mathcal{F}(h(n)) & = & \mathcal{F}(f(n)) \cdot \mathcal{F}(g(n)) \\ h(n) & = & \mathcal{F}^{-1}(H(N)) & = & (f * g)(n) \end{array} \tag{6.2}$$

The fast convolution is computed by taking the *Fast Fourier Transform (FFT)* of f and g, multiplying them pairwise for each element of the input and computing the *Inverse Fast Fourier Transform (IFFT)* of the result [BP84]. For a length of the functions f and g of N and K respectively, this reduces the complexity from $\mathcal{O}(N \cdot K)$ to $\mathcal{O}(N \cdot \log(N))$ $(N > K)$. The reduction of complexity is obvious, because the FFT and IFFT respectively have a complexity of $\mathcal{O}(N \cdot \log(N))$ and the pairwise multiplication has

[5] `http://www.doxygen.org/`

linear complexity, whereas the regular convolution operation multiplies the whole function g with each element of f what causes quadratic complexity. Both, FFT and IFFT are efficiently implemented with C subroutines in the FFTW LIBRARY[6]. A proprietary in-house implementation provided sufficient performance for the given purposes, too.

6.3 Exported interface definition

This section gives an overview of the exported interface of the FLOODNET class framework which allows to use and control it. This is followed by a short example, using the exported functions in the right way. Consider that although the class framework itself is written in C++, the interface is exported as a C-Interface (figure 6.5). That was done due to the principle of platform independence. It enables also a use of the library with non-object-oriented programming languages. If it is necessary in the future, a direct export of the C++-Core can be done with less effort as well.

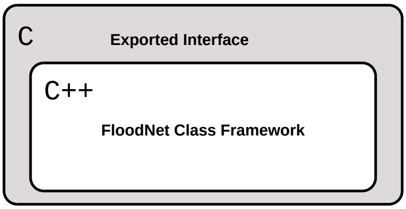


Figure 6.5: C-Interface wraps the FLOODNET class framework written in C++

The interface consists of a set of functions, to initialize and close the library, to adjust some settings of the prediction (e.g reference time, used version of feature detection, etc.), to read input data in different formats, to perform a runoff prediction and to output the computed results. For debugging issues there is also an error reporting mechanism. A listing of all defined functions is given in table 6.1.

In the following there is given a general use-case of an application of the FLOODNET class framework on the example of a sample application. This

[6] `http://www.fftw.org/`

Table 6.1: Functions of the exported C-Interface

Function name	Description
int **initLib** ()	Initialize the library
void **releaseLib** ()	Close the library and release all resources
int **getHistLen** ()	Provide required history length
int **getFcastLen** ()	Provide required forecast length
int **getVersion** ()	Provide the version of the library
int **connectDatabase** ()	Establish a database connection
int **setVersions** ()	Set up some used modules
int **setRefPoint** ()	Set the reference point
int **readQHistBuf** ()	Read the runoff history from a buffer
int **readVegetationHistBuf** ()	Read the vegetation cover from a buffer
int **readWeaHistBuf** ()	Read the weather history from a buffer
int **readWeaHistBufExt** ()	Read the weather history from a buffer
int **readWeaFcastBuf** ()	Read the weather forecast from a buffer
int **readWeaFcastBufExt** ()	Read the weather forecast from a buffer
int **readQHistWaSiM** ()	Read the runoff history from a WaSiM file
int **readVegetationHistWaSiM** ()	Read vegetation cover from a WaSiM file
int **readWeaHistFN** ()	Read the weather history from a FloodNet file
int **readWeaFcastFN** ()	Read the weather forecast from a FloodNet file
int **readWeaHistWaSiM** ()	Read the weather history from a WaSiM file
int **readWeaHistExtWaSiM** ()	Read the weather history from a WaSiM file
int **readWeaFcastWaSiM** ()	Read the weather forecast from a WaSiM file
int **readWeaFcastExtWaSiM** ()	Read the weather forecast from a WaSiM file
int **readNextTimeStep** ()	Read real-time data for the next time step
int **execPrediction** ()	Perform the prediction
int **writeResultBuf** ()	Write the result into a formatted buffer
int **writeResultFile** ()	Write the result into a WaSiM file
int **writeErrors** ()	Write all occurred errors in a text file

example is written in C++ for the WIN32 platform. A cutout of the application is presented in figure 6.6. In general case an application for runoff prediction using the FLOODNET class framework has to do the falling library calls in chronological order. The according program lines of the sample program are listed in brackets.

1. First, the all internal values have to be initialized [line 4].
2. Then a connection to a database with information about the catchment area and trained neural networks is established [line 8].
3. Set the used preprocessing version and the considered gauge [line 12].
4. Specify a reference point of the prediction [line 16].
5. Read hydrological and meteorological input data [line 20-39].
6. Perform the runoff-prediction for the given inputs and settings [line 43].
7. The computed output is written into a file or buffer and optionally a continuous real-time prediction can be started with a new input operation and prediction for each new time step [line 47].
8. At the end of each use stands the release of all allocated resources [line 51].

6.4 Displaying output data with involvement of uncertainty

After the presentation of the design and implementation, this section gives a discussion of some possibilities, how to proceed with the computed output data. In fact it is an important task to present the computed output data to the user in an adequate way. Consider an use-case of the FLOODNET system, e.g. for reservoir control, responsible for an operational flood risk management in a catchment. A responsible gatekeeper with domain-specific knowledge needs an easy and meaningful visualization of the computed output data, to be able to make a well-founded decision within short time.

The FLOODNET class framework can be driven with meteorological input data from an *ensemble prediction system (EPS)*. Thereby an *ensemble forecast* is

```
int main(int argc, char* argv[]) {
  ...

  control = initLib(argc, argv);
  if(control)
    cout << "failure" << endl;

  control = connectDatabase("../../../../mulde/databases/FbgMulde.mdb");
  if(control)
    cout << "connectDatabase() ... failure" << endl;

  control = setVersions("26", "2", "2", "1", "9", "P");
  if(control)
    cout << "setVersions() ... failure" << endl;

  control = setRefPoint("12.03.2002", "12:00");
  if(control)
    cout << "setRefPoint() ... failure" << endl;

  control = readQHistWaSiM("../../../../mulde/data/demo/Q2002.dat", NULL);
  if(control)
    cout << "readQHistWaSiM() ... failure" << endl;

  control = readVegetationHistWaSiM("../../../../mulde/data/demo/V.dat");

  if(control)
    cout << "readVegetationHistWaSiM() ... failure" << endl;

  control = readWeaHistWaSiM("../../../../mulde/data/demo/N2002.bin", NULL,
                             "../../../../mulde/data/demo/T2002.bin", NULL);
  if(control)
    cout << "readWeaHistWaSiM() ... failure" << endl;

  char** P_Fcast_Files = new char *[1]; P_Fcast_Files[0] = new char[100];
  char** T_Fcast_Files = new char *[1]; T_Fcast_Files[0] = new char[100];
  strcpy(P_Fcast_Files[0], "../../../../mulde/data/demo/N2002.bin");
  strcpy(T_Fcast_Files[0], "../../../../mulde/data/demo/T2002.bin");

  control = readWeaFcastWaSiM(P_Fcast_Files, T_Fcast_Files, 1);
  if(control)
    cout << "readWeaFcastWaSiM() ... failure" << endl;

  control = execPrediction();
  if(control)
    cout << "execPrediction() ... failure" << endl;

  control = writeResultFile("../../../../mulde/data/demo/prediction");
  if(control)
    cout << "writeResultFile() ... failure" << endl;

  releaseLib();

  ...
}
```

Figure 6.6: Sample application of the FloodNet library

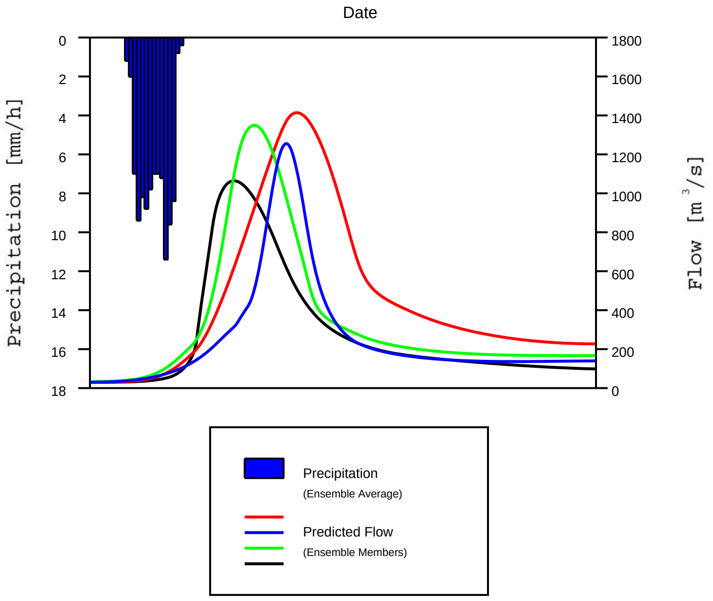


Figure 6.7: Example of a Plumes Plot

simply a collection of two or more weather forecasts verifying at the same time. [NOA06][7] is a good compendium for the complex of ensemble forecasts, EPS and possible visualizations. Further informations about the meteorological background is presented in [Atg99]. The computed hydrographs of the different members of an ensemble forecast can represent the actual uncertainty in the meteorological input data well, but this uncertainty can only be presented with the help of a visualization, including the different computed hydrographs.

From many discussions with hydrologists with a lot of domain-specific background knowledge and the state-of-the-art alternatives for visualization of ensemble output [NOA06] [Atg99] it results that a *point view* is the best way to present the estimated runoff forecast to the user. A *point view* is a visualization of ensemble output of a scalar value at one point. According to this definition, the computed output of the FloodNet class framework is

[7]`http://www.hpc.ncep.noaa.gov/ensembletraining/`

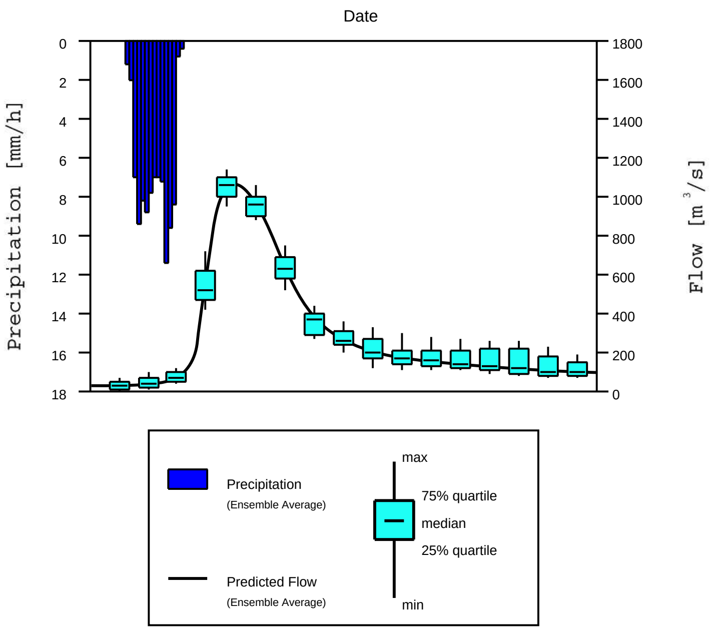


Figure 6.8: Example of a Box & Whisker Plot

the runoff at one gauge station, hence at one point. Possible realizations of a point view are **plumes** and **box & whisker plots** [NOA06]. A plume is a line plot that overlays the model output of all ensemble members in one chart. It is helpful to visualize the divergence of different members of a forecast ensemble and to detect maximum and minimum values. An example for a hypothetical gauge station is presented in figure 6.7. In contrast a box & whisker plot is a mix of a line and a bar plot. First, the the average ensemble output is plotted as a line. At specific intervals, there are some boxes, visualizing the median, the twenty-five (minimum fourth out) and seventy-five (maximum fourth out) quartiles, and whiskers, visualizing the minimum and maximum member output. In a sense, box & whisker plots are reduced plumes that compress the given information in a smart way. They are helpful to visualize the global conclusion of the estimated ensemble output and to assess the certainty of a prediction. Although in a box & whisker plot, there exists only one plot all members of an ensemble forecast, it is nevertheless possible to detect the distribution of the output and minimum and maximum values. Because of this, **it is suggested to use box & whisker plots** in a flood forecasting system, using the FloodNet class framework. An example of a box & whisker plot is given in figure 6.8.

VII

Results and Discussion

This chapter presents and evaluates some results of the implemented system, especially for the prediction of the 2002 flood event and a discussion of the achieved state. This is associated with a discussion of solved and unsolved problems.

7.1 Evaluation of the Results

7.1.1 Validation of the PAI-OFF approach

The performance of the implemented system was validated and tested in the Kriebstein catchment area, a subordinated catchment of the study area, the Freiberger Mulde catchment area [Cul06]. The training was performed using data from 1953-1971 and 1982-1999. The years 1972-1981 were excluded from the training and used for the validation process only. There exist some objective test criteria to assess the quality of hydrological models. With respect to the goal to use the FloodNet class framework as an operation flood forecasting system, the following three criteria were chosen to evaluate the system performance in the validation process [Cul06]. The first criterion is the dimensionless *Nash-Sutcliffe efficiency coefficient (NSE)* [NS70] which is defined as

$$E = 1 - \frac{\sum_{t=1}^{T} (Q_o^t - Q_m^t)^2}{\sum_{t=1}^{T} (Q_o^t - \overline{Q_o})^2} \tag{7.1}$$

where Q_o is observed discharge, and Q_m is modeled discharge. Q_*^t is discharge at time t. It is one of the most commonly used measures to assess the predictive power of hydrological models [Cul06]. A second criterion is the *peak-to-peak error (PPE)* which is defined as the difference of the peak flows of historically measured time series from a database and the predictions of the FloodNet class framework according to PAI-OFF. This relative error

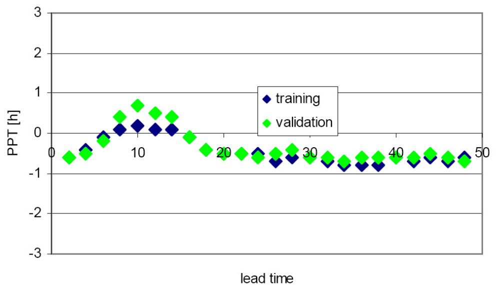


Figure 7.1: Training and validation of a PoNN with respect to PPT criterion according to [Cul06]

describes the average difference between the peak flow values for all events used in the validation process. The third criterion, the *peak-to-peak time (PPT)* describes the average error in the peak timing in hours, also over the whole validation set. Absolute values of the PPT of less than one hour (figure 7.1), a PPE of less than 4% (figure 7.2) and a NSE of more than 0.97 (figure 7.3) portray the good performance of the approach.

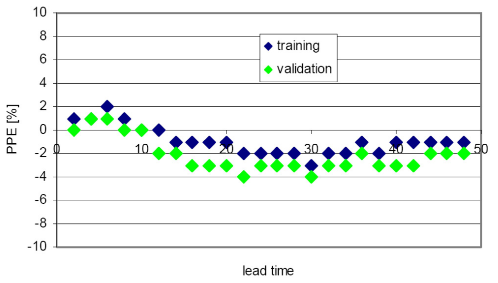


Figure 7.2: Training and validation of a PoNN with respect to PPE criterion according to [Cul06]

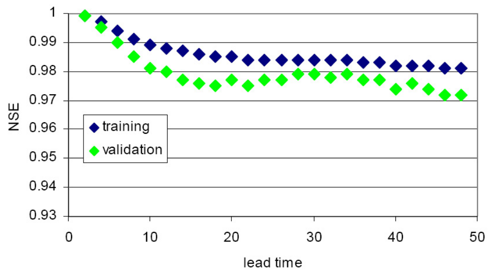


Figure 7.3: Training and validation of a PoNN with respect to NSE criterion according to [Cul06]

7.1.2 Results for the prediction of the 2002 flood event

Like the validation events from 1972-1981, the measured time series for the 2002 flood event did not feature in the training process as well. Therefore it is perfectly qualified for a presentation of achieved results. The goal is to predict the peak flow rate with an relative error of less the 10% with focus on long lead times (24h, 36h and 48h) [Cul06]. Thus, the FloodNet class framework was driven with the historical time series of meteorological and hydrological 2002 pre-event data and forecast. After this, the predicted results were compared with the observed runoff. The results for a lead time of 36h and 48h, are presented in figure 7.4 and 7.5, respectively. It can be seen that the predicted runoff stays in the defined 10% confindence interval both for shorter and longer leading times. The prediction with a 48h lead time even outperforms the one for 36h lead time, compared to the observation. That shows the stability of the PAI-OFF approach for longer leading times [Cul06].

But the FloodNet class framework is not only able to process single weather forecast scenarios, but also provides the ability to perform runoff predictions for weather forecast ensembles. To show the possibilities of this new functionality, the FloodNet library was driven with a weather ensemble forecast containing 199 members, for the 2002 flood event. The ensemble was gained by an application of a specific amount of noise to the actual distributed precipitation values. All other meteorological values are the same for all ensemble members. The results are presented in figure 7.6 and 7.7 as plumes and a box & whisker plot, respectively. Although the used forecast ensemble was synthetically generated and the members only differs in the

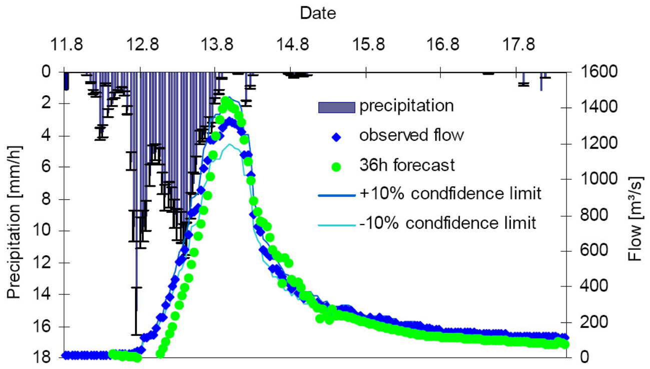


Figure 7.4: Forecast performance of the 2002 flood event at Kriebstein gauging station for 36 hours lead time according to [Cul06]

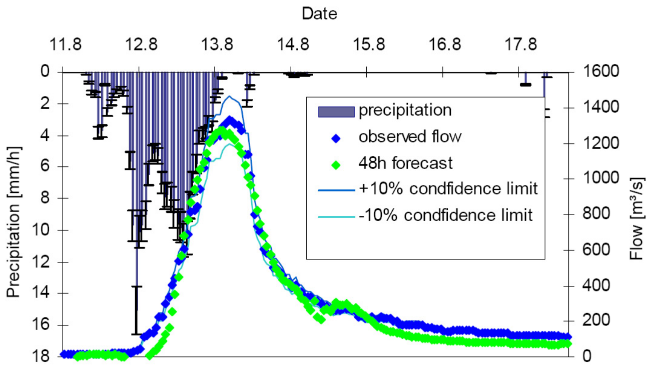


Figure 7.5: Forecast performance of the 2002 flood event at Kriebstein gauging station for 48 hours lead time according to [Cul06]

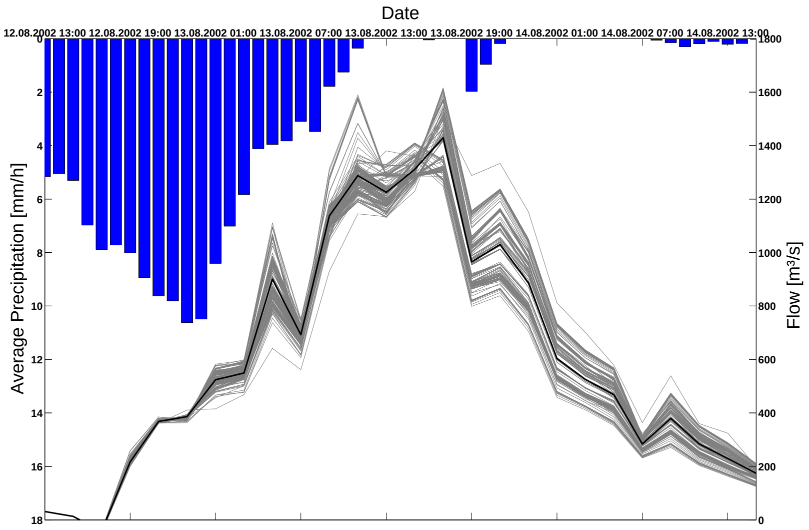


Figure 7.6: Runoff prediction of the 2002 flood event at Kriebstein gauging station driven with an ensemble forecast (Plumes)

amount and distribution of precipitation, the possibility of estimating the uncertainty of precipitation ensembles in the associated computed runoff is obvious. However this uncertainty still only expresses the uncertainty in the input data of the forecast and has no explanatory power about the uncertainty of the initial conditions in the catchment, the model uncertainty or the uncertainty of the measured characteristics of the catchment, e.g. the properties of the soil which also play an important role in the rainfall-runoff process.

However these encouraging results still leave enough potential for improvements with respect to the prediction quality. One points of criticism is for instance the choice of the architecture of the used ANN in the PAI-OFF approach. In [Cul06] only the two architectures of MLFN and PoNN were taken into account. Other neural architectures for time series prediction (section 3.3) promise maybe even better results.

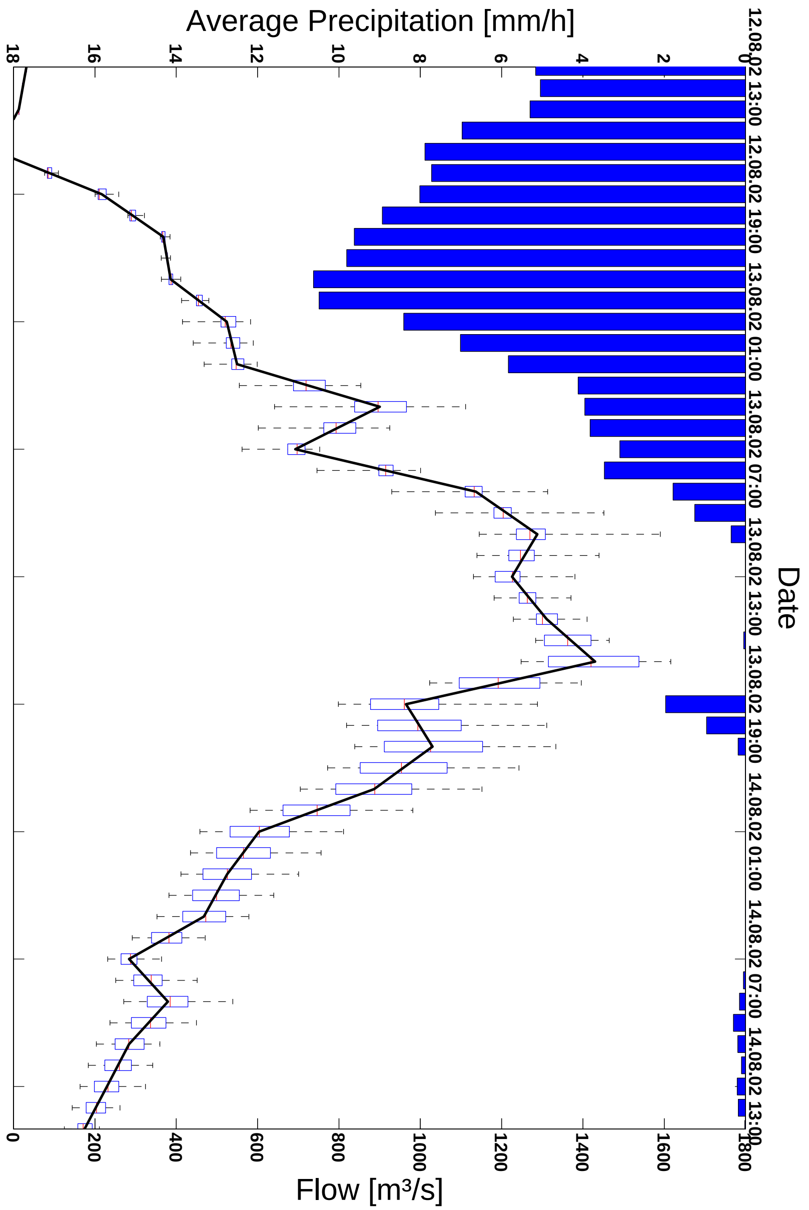


Figure 7.7: Runoff prediction of the 2002 flood event at Kriebstein gauging station driven with an ensemble forecast (Box & Whisker Plot)

7.2 Discussion of the achieved state

The FloodNet class framework provides the absolutely new possibility to perform a **stable prediction of the runoff in quickly responding catchment areas also for longer leading times** (approx. 48h) according to the PAI-OFF approach. Contrary to previous prototypes [SCG$^+$05] it provides the possibility to **perform predictions also for weather forecast ensembles** and summarizes the computed runoff scenarios for the different ensemble members in an intelligent way to enable clear and meaningful visualizations of the output data. Due to the **application of neural networks** and a **fast implementation** with efficient algorithms, ensemble predictions can be performed on the fly also for ensembles with a greater number of members. This performance even opens up the potential of incorporating Monte-Carlo-Simulations for the runoff prediction, that were not possible if using many other prediction frameworks. To deal with the huge amount of meteorological input data, especially for weather forecast ensembles, the class framework supports a **new custom-designed data format** which holds all meteorological input data for the history and the forecast in only two different containers. Additionally the class framework has a **modular and clearly laid-out design** which provides the possibility to implement other prediction methods without greater adapta

Although the system provides a lot of improvements, a number of unresolved problems remains. The most intense problem is that the system has currently **no access to real-time input data**. This includes meteorological measurements and confident forecast data. Thus the system cannot be validated with the current situation in the catchment. Another unsolved issue is the **failing support for the winter months**. During wintertime fast responding catchment areas in lower and upper mountain ranges in Central Europe are usually dominated by precipitation in solid form. Unfortunately the snow model of the underlying process based model WaSiM-ETH does not portray the snow accumulation and melt processes very well. Additionally the currently available data is not adequate to run more detailed snow models. Because of this, the process based model cannot be used to generate synthetical runoff scenarios for extreme flood events in the winter term and consequently the neural network is not able to portray these processes very well.

To use the system in operational mode, the framework also needs a clear and robust **graphical user interface**. A widely used system also needs support to **import and export data from the most common file formats**. In particular the system needs tools to convert data from the WaSiM-ETH file format to the internally used FloodNet file format and to provide the output data in as . Unfortunately there are still only less established standards in this domain.

VIII

Conclusion and Future Work

Flash floods are one of the most dangerous natural disasters. This work presents the implementation of a class framework to master this phenomen in context of online flood forecasting following the PAI-OFF approach. The system satisfies the specified requirements and shows a high potential for further extensions to use it for future research in the domain of rainfall-runoff modelling and later also as operational flash flood forecasting system. Although the results are encouraging, there are still some unsolved tasks to use the system as a online flood forecasting system in operational mode driven by real-time data.

The domain of rainfall-runoff modelling of fast responding catchment areas is still in the beginning. Currently it is still very hard to get access to a sufficient amount of input data, to represent the dynamics of the runoff processes in a fast responding catchment adequately. A lot of research effort is still needed, to apply the PAI-OFF approach to arbitrary catchments. This provides the possibility to provide stable and precise runoff predictions for longer lead times which are the base of timely and reliable warnings. Below, there is given a presentation of some approaches and ideas to resolve the most important unsolved tasks within future research.

8.1 Access to real-time meteorological input data

One main of the most precarious aspects to use the FloodNet class framework for online flood forecasting is a better access to real-time meteorological input data. This includes history and forecast as well. To get reliable weather forecast ensembles for a considered catchment area, it would be possible to operate an own mesoscale weather model, for instance the *PSU/NCAR Mesoscale Model Version No. 5 (MM5)* [DGM$^+$] [1]. MM5 is a non-hydrostatic grid-point model widely used throughout the forecasting community to simulate and estimate mesoscale and regional-scale atmospheric circulation. It was developed at the Pennsylvania State University (PSU) and the National

[1]`http://www.mmm.ucar.edu/mm5`

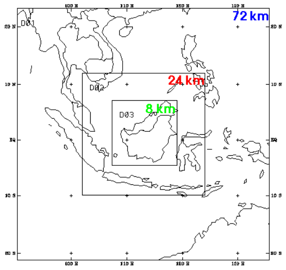


Figure 8.1: Nested domain approach in MM5[a]

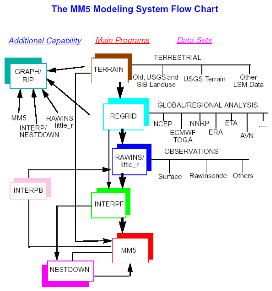


Figure 8.2: Model structure of MM5[b]

[a]http://ugamp.nerc.ac.uk/hot/rbn_mm5/mm5.htm
[b]http://www.mmm.ucar.edu/mm5

Center of Atmospheric Research (NCAR) in the USA. MM5 is driven by a global atmospheric circulation measurement grid, operated by the NCAR. For academic and non-commercial purposes there are free downloads of the real-time meteorological data, which forms the input of MM5. The downloads are renewed every six hours.

Successive nesting over a specified limited area is used in order to consistently scale down the processes playing a role in the atmospheric circulation. An illustration for the nesting approach for the isle Borneo in Indonesia is given in figure 8.1. The atmospheric circulation in the most inner nest is estimated by an iterative numeric solution of the running processes. Thereby both the outer nests determine boundary conditions for the inner nests and the processes in the inner nests influence the processes in the outer nests.

Unfortunately not only the meteorological data of the forecast is afflicted with uncertainty, but also the theoretically certain data for the history. In practice measured data from precipitation gaging stations in and around the catchment is interpolated with certain methods, as e.g. the *Kriging-Method* or the *Thiessen-polygon* approach [Hay94]. Especially subscale effects, e.g. convective precipitation events, cannot be represented with this methods in an adequate way. Recent advances in the quantitative precipitation measurement with ground-based radar observations showed that these system provide more accurate distributed information about the actual precipitation [HAK+03]. It is considerable to use this method of remote sensing in operational mode.

8.2 Using further developed prediction methods

Besides considering the uncertainty in the input data, there rests still enough potential to reduce the error ratio of the underlying prediction methods. For a better selection of the significant features, it is advisable to use well-founded physical based input factors, e.g. the potential evapotranspiration, maybe even approximated with a prefixed ANN. A good application of this approach is presented in [Pet07].

A second problem is insufficiently good snow model in the currently used process based model WaSiM-ETH. Especially the snow melt is not portrayed accurately. A very promising approach, in particular designed for hydrologic purposes in low mountain ranges is presented in [Her01]. The results are promising and a successful application would enable an application of the FloodNet library also in the winter months.

The crucial factor for the prediction quality of a neural network is an accurate training database. In a first application of PAI-OFF [Cul06], the training database was generated by overlaying a historical time series with synthetical extreme events. First the generated events are not validated by meteorologist and secondly this causes many flood events in short sequence. Therefore the long term storage units of the groundwater and soil are filled far in excess of there normal value. Both effects impend the quality of the generated training database. Another promising approach is to generate very long realistic meteorological time series (>3000 years) by a stochastic weather generator, as presented in [LBWA05]. Because of the huge length of the time series it contains sufficient extreme events, which are more realistic and without a pre-event influenced by another extreme flood.

Another problem in the rainfall-runoff modelling with neural networks is the huge amount of input data. The input has to subsumed and compressed with as less information loss as possible. A possible solution for this problem is the use of other neural architecture specific for time series prediction. Especially the RNN architecture shows a high potential, since it provides the possibility to conserve information about previous prediction steps within an internal state. For one operation step they require only information representing the current meteorological situation and the forecast, because the pre-event is already stored in the internal state of the network. Especially the new LSTM architecture, proposed in section 3.3, is very promising because it provides an internal state which represents short and long term processes. Exactly these both classes of processes are necessary to represent both the state of the catchment and the dynamics of the rainfall-runoff process as a whole in fast reacting catchments. During a catchment simulation with WaSiM-ETH for the Freiberger Mulde catchment area, it has been proven that the current runoff can be influenced by meteorological events in history up to 150 days ago [Pet07]. Other time series prediction specific neural architectures, e.g. the genetic *GMDH* approach or the architecture of *modular networks* have a potential to improve the model reliability as well. Thanks to the modular design of my system, other neural architectures can be implemented without problems. It would be preferable to carry out a comparative trial, to evaluate the proposed neural architectures in the domain of flash flood forecasting.

8.3 Development of a graphical user interface

Many possible users in the domain of flood forecasting have a reasonable amount of expert knowledge in operating conceptual or physically based models, but they do not have much operating experience with neural models. To provide confidence in spite of the black-box approach of neural networks, it is important to develop a clearly laid out and well-engineered graphical user interface. A scheme of a possible solution to visualize the uncertainty of the model output was discussed in chapter 6. The visualization can be embedded either in a desktop user interface for professional users and presentation purposes or in an online user interface available. A design idea and a comparison of existing solutions is presented in [Zim05]. As a conclusion of this work and my own experiences the online interface of the flood forecasting system of the regional office of the environment in Baden Württemberg can pass for a good example (figure 8.3).

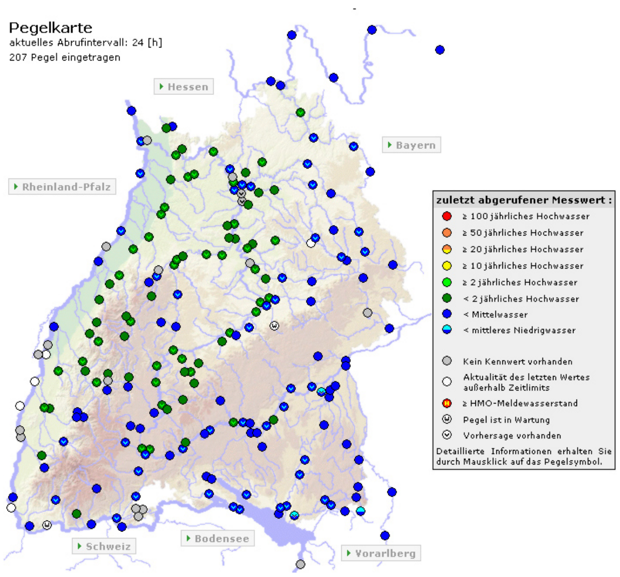

Pegelkarte
aktuelles Abrufintervall: 24 [h]
207 Pegel eingetragen
Hessen
Bayern
Rheinland-Pfalz
Schweiz
Bodensee
Vorarlberg
zuletzt abgerufener Messwert :
≥ 100 jährliches Hochwasser
≥ 50 jährliches Hochwasser
≥ 20 jährliches Hochwasser
≥ 10 jährliches Hochwasser
≥ 2 jährliches Hochwasser
< 2 jährliches Hochwasser
< Mittelwasser
< mittleres Niedrigwasser
Kein Kennwert vorhanden
Aktualität des letzten Wertes außerhalb Zeitlimits
≥ HMO-Meldewasserstand
Pegel ist in Wartung
Vorhersage vorhanden
Detaillierte Informationen erhalten Sie durch Mausklick auf das Pegelsymbol.

Figure 8.3: Example GUI of an online flood forecasting system[a]

[a] `http://www.hvz.baden-wuerttemberg.de/`

Bibliography

[AH94] ADELI, Hojjat ; HUNG, Shih-Lin: *Machine Learning: Neural Networks, Genetic Algorithms, and Fuzzy Systems: Neural Networks, Genetic Algorithms and Fuzzy Systems*. John Wiley & Sons Inc., 1994

[AMFB05] ALVISI, S. ; MASCELLANI, G. ; FRANCHINI, M. ; BÁRDOSSY, A.: Water level forecasting through fuzzy logic and artificial neural network approaches. In: *Hydrology and Earth System Sciences zimmerDiscussions* 2 (2005), S. 1107–1145

[Atg99] ATGER, Frédéric: The Skill of Ensemble Prediction Systems. In: *Monthly Weather Review* 127 (1999), Nr. 9, 1941–1953. `http://dx.doi.org/10.1175%2F1520-0493%281999%29127%3C1941%3ATSOEPS%3E2.0.CO%3B2`

[Bev01] BEVEN, Keith J.: *Rainfall-Runoff Modelling - The Primer*. John Wiley & Sons Ltd., 2001

[BL96] BAUMGARTNER, Albert ; LIEBSCHER, Hans-Jürgen: *Lehrbuch der Hydrologie*. Bd. Band 1 Allgemeine Hydrologie. Gebrüder Bornträger - Stuttgart - Berlin, 1996

[BP84] BURRUS, C. S. ; PARKS, Thomas W.: *DFT/FFT and Convolution Algorithms: Theory and Implementation*. John Wiley & Sons Inc, 1984 (Topics in Digital Signal Processing)

[Bru02] BRUNNER, Gary W. ; U.S. ARMY CORPS OF ENGINEERS INSTITUTE FOR WATER RESSOURCES HYDROLOGIC ENGINEERING CENTER (Hrsg.): *HEC-RAS River Analysis System - Hydraulic Reference Manual*. 3.1. 609 Second Street Davis, CA 95616: U.S. Army Corps of Engineers Institute For Water Ressources Hydrologic Engineering Center, November 2002

[CG05] CORANI, Giorgio ; GUARISO, Giorgio: Coupling Fuzzy Modeling and Neural Networks for River Flood Prediction. In: MARIK, Vladimir (Hrsg.): *Systems, Man and Cybernetics, Part C: Applications and Reviews* Bd. 35. IEEE Systems, Man, and Cybernetics Society, August 2005, S. 382– 390

[Cul06] CULLMANN, Johannes: *Online flood forecasting in fast responding catchments on the basis of a synthesis of artificial neural networks and process models*, Technische Universität Dresden, Diss., November 2006

[DGM$^+$] DUDHIA, Jimy ; GILL, Dave ; MANNING, Kevin ; WANG, Wei ; BRUYERE, Cindy: *PSU/NCAR Mesoscale Modeling System Tutorial Class Notes and User's Guide: MM5 Modeling System Version 3*

[Din94] DINGMAN, S. L. ; MCCONNIN, Robert (Hrsg.): *Physical Hydrology*. Prentice Hall, 1994

[DP95] DYCK, Siegfried ; PESCHKE, Gerd: *Grundlagen der Hydrologie*. Verlag für Bauwesen - Berlin, 1995

[DR90] DURBIN, Richard ; RUMELHART, David E.: Product units: a computationally powerful and biologically plausible extension to backpropagation networks. In: *Neural Computation* 1 (1990), S. 133–142

[DW98] DAWSON, Christian W. ; WILBY, Robert: An artificial neural network approach to rainfall-runoff modelling. In: *Hydrological Sciences Journal* 43 (1998), S. 47–65

[DW01] DAWSON, C.W. ; WILBY, R.L.: Hydrological modelling using artificial neural networks. In: *Progress in Physical Geography* 25 (2001), S. 80–108

[EMB$^+$01] ETCHEVERS, P. ; MARTIN, E. ; BROWN, R. ; FIERZ, C. ; LEJEUNE, Y.: SnowMIP, an intercomparison of snow models: first results. In: *Proceedings of the International Snow Science Workshop - 8th Scientific Assembly*, Centre d'Etudes de la Neige, CNRM Météo-France, Grenoble, France and Canadian Meteorological Service, Dorval, Qc, Canada and Swiss Federal Institute for Snow and Avalanche Research (SLF), Davos Dorf, Switzerland, July 2001

[Fok99] Foka, Amalia: *Time Series Prediction Using Evolving Polinomial Networks*, University of Manchester, Diplomarbeit, 1999

[GDS94] Grell, Georg A. ; Dudhia, Jimy ; Stauffer, David R.: A Description of the Fifth-Generation Penn State/NCAR Mesoscale Model (MM5) / National Centre for Atmospheric Research, Boulder, Colorado, USA. 1994. – Forschungsbericht

[GR00] Govindaraju, R.S. (Hrsg.) ; Rao, A. R. (Hrsg.): *Water Science and Technology Library*. Bd. 36: *Artificial Neural Networks in Hydrology*. Kluwer Academic Publishers, 2000

[Gör07] Görner, Wilfried ; IHM TU-Dresden (Hrsg.): *Entwicklung von KNN-basierten NA-Modellen mit dem FloodNet Developer*. Dresden: IHM TU-Dresden, January 2007

[HAK+03] Hense, Andreas ; Adrian, Gerd ; Kottmeier, Christoph ; Simmer, Clemens ; Wulfmeyer, Volker: *Quantitative Precipitation Forecast*. Priority Program of the German Research Foundation, February 2003

[Hay94] Haykin, Simon ; Griffin, John (Hrsg.): *Neural Networks - A Comprehensive Foundation*. Macmillan College Publishing Company Inc., 1994

[HDB96] Hagana, Martin T. ; Demuth, Horward B. ; Beale, Mark ; Barter, Bill (Hrsg.): *Neural Network Design*. PWS Publishing Company, 1996

[Her01] Herpertz, Dorothe: *Schneehydrologische Modellierung im Mittelgebirgsraum*, Friedrich-Schiller-Universität Jena, Diss., 2001

[HHA93] Halff, Albert H. ; Halff, Henry M. ; Azmoodeh, Masoud: Predicting runoff from rainfall using neural networks. In: *Engineering Hydrology*, 1993, S. 760–765

[HS97] Hochreiter, Sepp ; Schmidhuber, Jürgen: Long Short-Term Memory. In: *Neural Computation* 9 (1997), S. 1735–1780

[Hyd00] Hydrology, ASCE Task C. i.: Artificial Neural Networks in Hydrology. I: Preliminary Concepts. In: *Journal of Hydrologic Engeneering* 5 (2000), S. 115–123

[Kno00] Knott, Gary D.: *Interpolating Cubic Splines*. Birkhäuser Boston, 2000 (Progress in Computer Science and Applied Logic)

[KRS04] Kumar, D.Nagesh ; Raju, K. S. ; Sathish, T.: River Flow Forecasting using Recurrent Neural Networks. In: *Water Resources Management* 18 (2004), S. 143–161

[LBWA05] Leander, R. ; Buishand, T.A. ; Wit, M.J.M. de ; Aalders, P.: Estimation extreme floods of the river Meuse using a stochastic weather generator and a rainfall-runoff model. In: *Hydrological Sciences Journal* 50 (2005), S. 1089–1103

[MBS70] Meyer-Brötz, Günter ; Schürmann, Jürgen: *Methoden der automatischen Zeichenerkennung*. Akademie-Verlag, 1970. – 100–105 S.

[Mil96] Miller, Alan J.: The Convergence of Efroymson's Stepwise Regression Algorithm. In: *The American Statistician* 50 (1996), Nr. 2, S. 180–181

[MJ99] Medsker, L.R. (Hrsg.) ; Jain, L.C. (Hrsg.): *Recurrent Neural Networks - Design and Applications*. CRC Press, 1999 (Series on Computational Intelligence)

[NOA06] NOAA/ National Weather Service Hydrometeorological Prediction Center (Hrsg.): *Ensemble Prediction Systems*. Camp Springs: NOAA/ National Weather Service Hydrometeorological Prediction Center, July 2006. `http://www.hpc.ncep.noaa.gov/ensembletraining/`

[NS70] Nash, J. E. ; Sutcliffe, J. V.: River flow forecasting through conceptual models. In: *Journal of Hydrology* 10 (1970), S. 282–290

[PB00] Preissler, Günter ; Bollrich, Gerhard ; Müller, Werner (Hrsg.): *Technische Hydromechanik*. Bd. 1. VEB Verlag für Bauwesen - Berlin, 2000

[Pet07] Peters, Ronny: *Künstliche neuronale Netze zur Beschreibung der hydrodynamischen Prozesse für den Hochwasserfall unter Berücksichtigung der Niederschlags-Abfluß-Prozesse im Zwischeneinzugsgebiet*, Technische Universität Dresden, Diss., 2007

[SCG+05] Schmitz, Gerd ; Cullmann, Johannes ; Görner, Wilfried ; Lennartz, Franz ; Dröge, Werner: PAI-OFF: A new strategy of flash-flood forecasting in quickly responding catchments. In: *Hydrologie und Wasserbewirtschaftung/Hydrology and Water Resources Management* 49 (2005), October, Nr. 5, S. 226–234

[Sch81] Schmitz, Gerd: *Instationäre Eichung mathematischer Hochwasserablauf-Modelle auf der Grundlage einer neuen Lösungsprinzipes für hyperbolische Differentialgleichungs-Systeme*, Technische Universität München, Diss., 1981

[SJ06] Schulla, J. ; Jasper, K. ; ETH Zürich (Hrsg.): *Model Description WaSiM-ETH*. Zürich: ETH Zürich, 2006

[SO06] Shu, C. ; Ouarda, T.B.M.J.: *Flood frequency analysis at ungauged sites using artificial neural networks in CCA physiographic space*. May 2006. – Submitted for publication to the Water Resources Research

[SW01] Schneider, Uwe ; Werner, Dieter ; 4 (Hrsg.): *Taschenbuch der Informatik*. Fachbuchverlag Leipzig, 2001

[Thi98] Thiesing, Frank M.: *Analyse und Prognose von Zeitreihen mit neuronalen Netzen*, Universität Osnabrück, Diss., 1998

[US04] Ulrich, Maik ; Schwarze, Robert: *Modellgeschützte Entwicklung und Testung eines einfachen methodischen Ansatzes zur Bestimmung des aktuellen Feuchtezustand von Einzugsgebieten im Mittelgebirgsraum*, TU-Dresden, Diplomarbeit, 2004

[Zim05] Zimmermann, Kai: *Erarbeitung der Aufgabenstellung für die Realisierung der Nutzerschnittstelle eines Hydrologenarbeitsplatzes zur Hochwasservorhersage*, Technische Universität Dresden, Diplomarbeit, 2005