Visualization Research Center VISUS

Master Thesis

Visualization of neural networks for diagnostic hydrological modeling

Ahmed Hassan

Course of Study:	M.Sc. Informatik
Examiner:	Prof. Dr. Daniel Weiskopf
Supervisor:	Dr. Anneli Guthke, Tanja Munz, M.Sc., Manuel Alvarez Chaves, M.Sc., Nils Rodrigues, M.Sc.
Commenced:	March 21, 2022
Completed:	September 21, 2022

Abstract

Long short-term memory (LSTM) networks are known to be extremely accurate at working with temporal features and time series data, for their ability to memorize sequences of data. This is practical for predicting the rainfall-runoff process, as the input is typically a sequence of weather features for a certain duration. Furthermore a modeled system of a river catchment has an inherent memory, namely all the accumulating weather circumstances that effect the generation of runoff over time, which can be reflected in an LSTM model by its hidden states. A model has been implemented and trained using weather data from 1961-2011 extracted from the upper Neckar catchment in southwest Germany that had very satisfactory results. To assess and understand the black box nature of the LSTM model in its application in regards to rainfall runoff predictions, as well as understanding the reason for its good performance in solving problems in regards to hydrological modeling, a visualization tool has been developed. Different aspects of the neural network have been explored through the visualization tool such as the hidden states throughout testing, input sequences dimensionality reduction, correlation between individual hidden layer cells and weather features and internal model features, etc. A suggested workflow has been explored for using the tool, that derives some insights into the model and its performance. Finally different observations have been thoroughly explored and interpreted.

Keywords: Machine Learning, Deep Learning, Long Short-Term Memory (LSTM), Neural Network, Rainfall Runoff Modeling, Visualization, Interpretation, Explainable AI

Kurzfassung

Long-short-term-memory-Netze (LSTMs) sind dafür bekannt, dass sie bei der Arbeit mit zeitlichen Merkmalen und Zeitreihendaten extrem genau sind, da sie sich Datenfolgen einprägen können. Dies ist praktisch für die Vorhersage des Niederschlagsabflusses, da die Eingabe typischerweise eine Abfolge von Wettermerkmalen für eine bestimmte Dauer ist. Darüber hinaus verfügt ein modelliertes System eines Flusseinzugsgebiets über ein inhärentes Gedächtnis, nämlich alle sich im Laufe der Zeit ansammelnden Wetterumstände, die sich auf die Entstehung des Abflusses auswirken, was in einem LSTM-Modell durch seine verborgenen Zustände wiedergegeben werden kann. Es wurde ein Modell implementiert und mit Wetterdaten aus dem oberen Neckareinzugsgebiet in Südwestdeutschland aus den Jahren 1961-2011 trainiert, das sehr zufriedenstellende Ergebnisse lieferte. Um den Black-Box-Charakter des LSTM-Modells bei seiner Anwendung in Bezug auf Niederschlagsabflussvorhersagen zu bewerten und zu verstehen, sowie den Grund für seine gute Leistung bei der Lösung von Problemen im Bereich der hydrologischen Modellierung zu verstehen, wurde ein Visualisierungstool entwickelt. Mit Hilfe des Visualisierungstools wurden verschiedene Aspekte des Neuronalen Netzes untersucht, wie z.B. die versteckten Zustände während des gesamten Tests, die Dimensionsreduktion der Eingabesequenzen, die Korrelation zwischen einzelnen Zellen der versteckten Schicht und Wettermerkmalen sowie die internen Modellmerkmale etc. Es wurde ein vorgeschlagener Arbeitsablauf für die Verwendung des Tools erforscht, der einige Einblicke in das Modell und seine Leistung ermöglicht. Schließlich wurden verschiedene Beobachtungen gründlich untersucht und interpretiert.

Schlüsselwörter: Machine Learning, Deep Learning, Long Short-Term Memory (LSTM), Neuronale Netze, Niederschlagsabflussmodellierung, Visualisierung, Interpretation, Erklärbare KI

Acknowledgments

This thesis is dedicated to my father who is no longer with us. His dream was to see me graduating, I hope he can still see me from the other side.

I would like to thank my supervisors Anneli Guthke and Tanja Munz and would like to congratulate them both for giving birth to two beautiful children during the course of my thesis. I would also like to thank my two other supervisors Manuel Alvarez Chaves and Nils Rodrigues for agreeing to be replacement supervisors in such short notice, they were very encouraging and gave very helpful feedback.

I would also like to thank my manager at work Omar Haridy and Daniel Riad, who were very encouraging and understanding throughout my master studies, without their constant support it wouldn't have been possible.

Finally, I would like to thank my sister and my mother for always being there for me, and the rest of my family for their love and support. I would also like to thank my friends who had to bear with my stress and constantly rejecting activities.

Contents

List of Figures

List of Tables

1 Introduction

Ever since the beginnings of artificial intelligence, scientists and researchers have been trying to understand the complex and non-linear models they create [SMV+19]. This urge has only increased with the rise of kernel machines and deep learning and their even more complex capabilities, which can sometimes contradict the human intuition. Since we are now relying more than ever on artificial intelligence and machine learning approaches, we need to understand why certain decisions are being made, and why in some cases AI models perform as well as they do [LWB+19]. Different approaches have been developed throughout time to answer those questions. In earlier stages scientists focused on deriving connections between models and human neurodynamics [Ros61]. Later work involved trying to represent the internal states of the models and explaining its behavior by for example rule extraction [DSW+87]. An especially potent method for explaining neural networks has been visualization. Though visualization has its limitations when it comes to multi-dimensionality, which is certainly a characteristic of neural networks, scientists have found approaches to eliminate those problems. Saliency maps [MKH+95], Node-Link visualizations [Har15], hidden states projection [GMW21] and other visualization techniques have been explored. While those approaches had significant upsides, we expect that the best possible approach is a combination of many visualization tools. As neural networks, in particular the ones with complex architecture, can have multiple layers and iteratively work on improving the parameters of the model to fit a likely huge dataset, which is likely multi-dimensional, we expect the many visualization techniques to tackle different sides of the neural network model. Analysing such a model is usually an attempt to answer many different questions simultaneously and having one visualization per question is our approach to solve the mystery.

As for many scientific fields, the application of neural networks has been lately gaining popularity in hydrology. LSTMs are especially potent in discharge prediction and rainfall-runoff modelling, given the temporal dependencies inherent to the data [ABK+20; APS+21; Ayz19]. In this thesis we will be exploring Chabok's (2022) hybrid LSTM model for predicting discharge and rainfall-runoff modelling [Cha22]. The model was developed using data from the Neckar catchment located in southwest Germany from 1961 to 2010.

We created a visualization tool aimed to interpret LSTM models for regression problems. In particular we used an LSTM model trained to predict discharge and rainfall-runoff modelling. We focused on showing the performance of the model, the correlations between the different model inputs and the model output, as well as the hidden state projection representation, while maintaining the representation of multi-dimensional time series data. Alongside visualization researchers and hydrology experts the model was iteratively adjusted and improved. We want to explore the applicability of such a tool in the field of hydrology, and whether having a visualization tool for interpreting a data-driven model can eliminate the discrepancy of physical relations. After developing the tool we worked on suggesting a workflow for users in order to extract the most value for their respective research questions. Finally, we attempted utilizing the tool for gaining insights into the model and interpreting some of the phenomena we have obtained.

This thesis is divided into 8 chapters. The first chapter is a general introduction of the entire work. The second chapter is a background chapter. There we discuss the concepts and the scientific background knowledge required to understand the thesis. Different subjects such as the machine learning, neural networks and LSTMs, dimension reduction as well as rainfall-runoff modelling are discussed. In the third chapter we present the previous work related to our project, and some similar concepts that inspired us in this work. In the fourth chapter we present the model we based our work on, its architecture and its performance. In the fifth chapter we discuss the visualization tool we created and its different components we developed. Furthermore we go into detail describing the individual visualization techniques we used, and we then suggest a workflow for utilizing the visualization tool in different use cases. We then present in the sixth chapter some of the results and interpretations we have obtained after using the visualization tool. In the seventh chapter we discuss those results and try to understand the reasoning behind them. Chapter eight is a conclusion chapter, where we summarize our work and conclude the findings we have obtained throughout the thesis. Finally there is a bibliography where we cite all the articles and books we referenced in the thesis.

2 Background

2.1 Machine Learning

Machine learning is the study of allowing computers to automatically learn and gain experience in executing various tasks, by utilizing scientific knowledge of mathematics, statistics and computer science. Due to the accessibility of huge data sources as well as the reducing costs of computation power, machine learning has been gaining rapid popularity in different scientific fields [JM15].

The two major categories of ML algorithms are supervised and unsupervised learning. For supervised learning a machine learns through a mapping between a set of inputs X, also called *independent variables* or *features*, and an output variable Y, also called *dependent variable*, to apply this mapping for a set of unseen data to predict an outcome [CCD08]. The typical problems supervised learning is applied for are regression and classification problems, which are being discussed later on in this section. The second category of machine learning algorithms is unsupervised learning. In unsupervised learning the machine detects an outcome and uncovers a hidden structure in the data without being guided by a target variable to use for a certain mapping. Clustering and anomaly detection are classic problems an unsupervised learning approach is used for [CA16]. In figure 2.1 a visual example of the 3 popular ML problems can be seen.

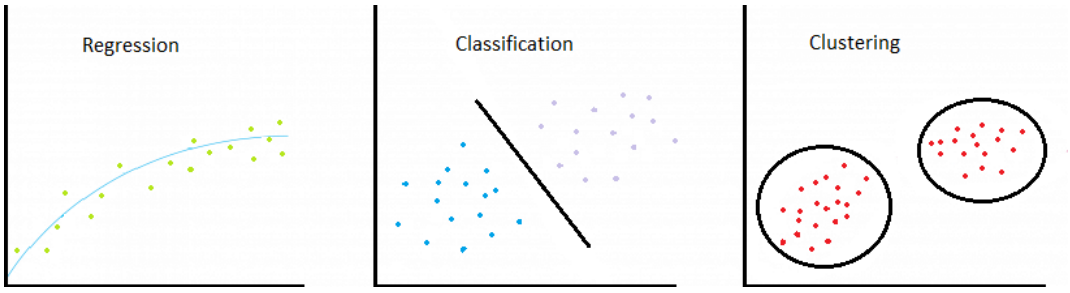

Figure 2.1: Left: A regression problem. Middle: A classification problem with 2 classes. Right: A clustering problem with 2 clusters.

2.1.1 Regression & Classification Models

In regression problems the aim is to predict a quantitative value or a vector of quantitative values based on a set of input vectors, by generating a function that best fits the data while assigning coefficients that resemble the importance of each independent variable (feature) of the input vector. In classification problems the aim is to assign an input vector to a finite number of discrete categories. [BN06]

Linear Models & Loss Function

Given a vector of inputs $X^T = (X_1, X_2, ..., X_p)$ we predict an output Y via the model

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j$$

where $\hat{\beta}_0$ is the bias and $\hat{\beta}_j$ is a value in the vector of coefficients or weights . In the regression problem the aim is to find the best fitting bias value as well as coefficients for each feature in the input variable. This model can be applied for a regression problem by predicting a quantitative output or a classification problem by assigning quantitative thresholds that indicate a qualitative output (e.g $\hat{Y}$: 0-0.49 → class_0, $\hat{Y}$: 0.5-1 → class_1).[HTFF09]

In order to fit the model to a set of training data we need to minimize what is called a Loss Function. There are many different examples for loss functions, in this example we are using the Mean Squared Error *MSE*:

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

where N is the number of samples or the count of input vectors, y_i is the observed value and $\hat{y}_i$ is the predicted value. The idea is minimizing the MSE for the model we are training through gradient based optimization methods, to be discussed later.

2.1.2 Clustering

Clustering algorithms are very popular for unsupervised learning problems. In supervised learning a mapping is created between the input set and the observed output, in unsupervised learning however the aim is to "discover groups of similar examples within the data, or to determine the distribution within the feature space", or to reduce dimensionality of the input features, without requiring a target vector of desired outputs [BN06]. In such models a common metric is the distances between the individual samples within the n-dimensional space of the data in order to find the best possible cluster assignment of the points. Examples for popular clustering algorithms are k-means and DBSCAN which will be discussed later.

2.2 Deep Learning & Neural Networks

2.2.1 Overview of Deep Learning

Deep Learning has been popularized in more recent years by the decreased cost of computational power and their increased performance through GPUs (Graphical Processing Units). A common use case for implementing deep learning models has been allowing the machines to learn how to

execute tasks which a human can do intuitively, without necessarily being able to explain how they are solved. A deep learning model learns complicated concepts by creating a "hierarchy of concepts, with each concept defined through its relation to simpler concepts" [GBC16]. This hierarchy is created through the use of what is called Artificial Neural Networks (ANN). Such models thrive on huge data sets that allow them to understand underlying patterns and phenomena, which have also been widely accessible in recent years due to the digitization and data collection techniques of the modern world.

A neural network has a graph form with its nodes being referred to as neurons, which are categorized into three different layer types; Input Layer, Hidden Layer, Output Layer. In figure 2.2 a basic feed-forward neural network is visualized on the left with two hidden layers.

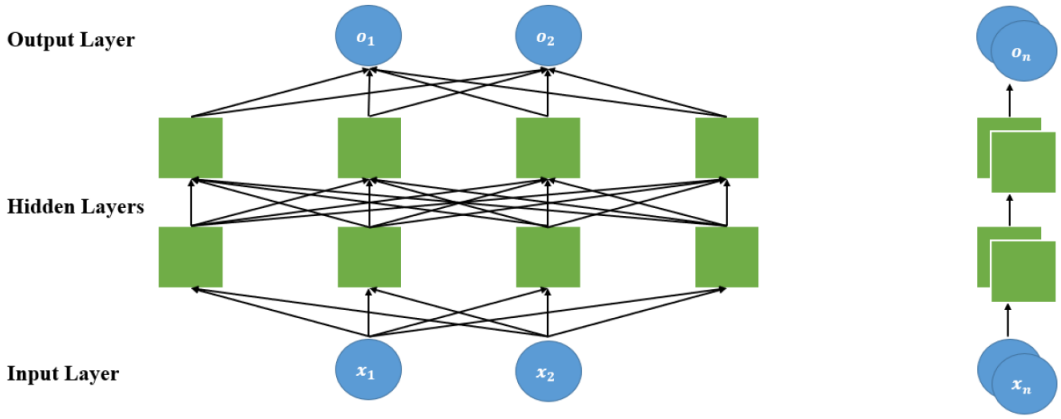

Figure 2.2: A feed forward two-layer neural network. Left: unfolded, Right: folded. [Cha22]

2.2.2 Forward Propagation

The values of the input vectors are being assigned to the input layer neurons. These inputs are being "fed forward" to the hidden layers until an output is calculated. The more hidden layers a network has, the more depth it incorporates, the more complex and precise outputs are generated, hence the term "deep learning" [GBC16]. In the same figure the same network is being represented on the right in its folded form, which is only a different method for visual representation.

A basic neural network model uses a basis function, so that each basis function itself is a nonlinear function of a linear combination of the inputs. In this scenario the coefficients in the linear combination are adaptive parameters. First, we construct M linear combinations of the input variables $x_1, ..., x_D$ in the form

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}$$

where $j = 1,...,M$ and the superscript (1) indicate the following parameters being part of the first layer of the network [GBC16]. Similar to section 2.1.1 $w_{ji}^{(1)}$ are weights and $w_{j0}^{(1)}$ are the biases. a_j are called *activations*. Each of them is going to be transformed by a different nonlinear *activation function* h(.), such that $z_j = h(a_j)$ Each activation function represents a *hidden unit* or a *hidden cell*, which are neurons of the hidden layers in figure 2.2. Every hidden layer replicates this process but with the outputs of the previous layer. This process is called forward propagation as the values of each layer are being fed to the following layer.

2.2.3 Activation Function

An activation function is a nonlinear sigmoidal function e.g:

$$Sigmoid(x) = \frac{1}{1+e^{-x}}$$

$$tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

$$ReLU(x) = max\{0, x\}$$

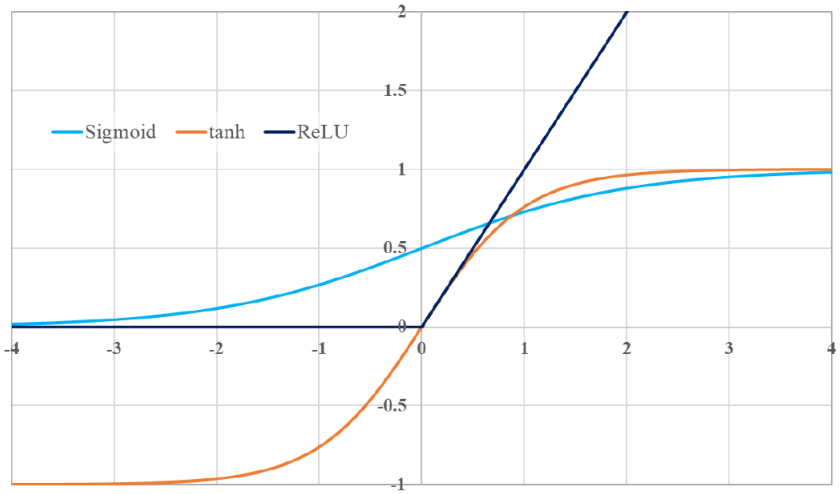

Figure 2.3: Sigmoid, tanh, and ReLU activation functions. [Cha22]

In figure 2.3 a visualization of the different activation functions can be seen.The outputs of the activation functions in the final layer are combined linearly to finally calculate *output unit activations* in the form

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)}$$

Where $k = 1,...,K$ and K is the number of outputs and (2) representing the second layer in this example and z_j being the outputs of the activation function.

2.2.4 Backward Propagation

We've discussed the Loss functions in 2.1.1 and its aim to fit the model to the data. In neural networks the concept is the same as with other machine learning models. The idea is to gradually update the weights and biases of the model, by using gradients of the loss function. This process is called back-propagation, as the errors in the output layer are used to adjust the weights of the previous layers. The advantage of back-propagation is its simplicity and local nature, where "each hidden unit passes and receives information only to and from units that share a connection", which allows for efficiency in parallel architecture computers [HTFF09]. The approach is as follows:

$$w \leftarrow w - \frac{\partial loss}{\partial w}$$

$$b \leftarrow b - \frac{\partial loss}{\partial b}$$

Where w is the vector of weights and b is the vector of biases. [Cha22]

2.2.5 Gradient-Based Optimization

We have previously mentioned that fitting the model to the data works by minimizing a given objective function. A gradient-based optimization is crucial here in order to determine the direction and amount of update in the weights and biases of the model. This process is called Gradient Descent (GD)

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)})$$

Where $\theta \in \mathbb{R}^d$ is the vector of the parameters (weights and biases), $J(\theta)$ is the objective function, $\nabla_\theta J(.)$ are the gradients of the objective function and η is the learning rate which determines the size of steps we take at a time in order to reach a (local) minimum. In the Stochastic Gradient Descent this is applied for every data point $x^{(i)}$ and $y^{(i)}$ where x is the input and y is the output at the i^{th} time step. [Rud16]

2.3 Long-Short Term Memory (LSTM)

The standard neural networks are a powerful machine learning tool, but they come with their own limitations. Since the training occurs for each data point independently, the entire state of the model is lost after each iteration. Sequential relationships or time & space dependencies between individual samples can not be detected by a standard neural network, and therefore there needs to be a state or a memory built within the neural network in order to account for this phenomenon. [LBE15] Given the sequential nature of our data and the task of rainfall-runoff modelling, we need to adjust the model. Recurrent Neural Networks (*RNNs*) are widely used to solve this problem. Recurrent networks' unique characteristic is having recurrent connections between hidden units that produce recurrent weights, allowing it to have a short-term memory between time steps. In this section we will discuss Long-Short Term Memory (*LSTM*) which is a type of RNN.

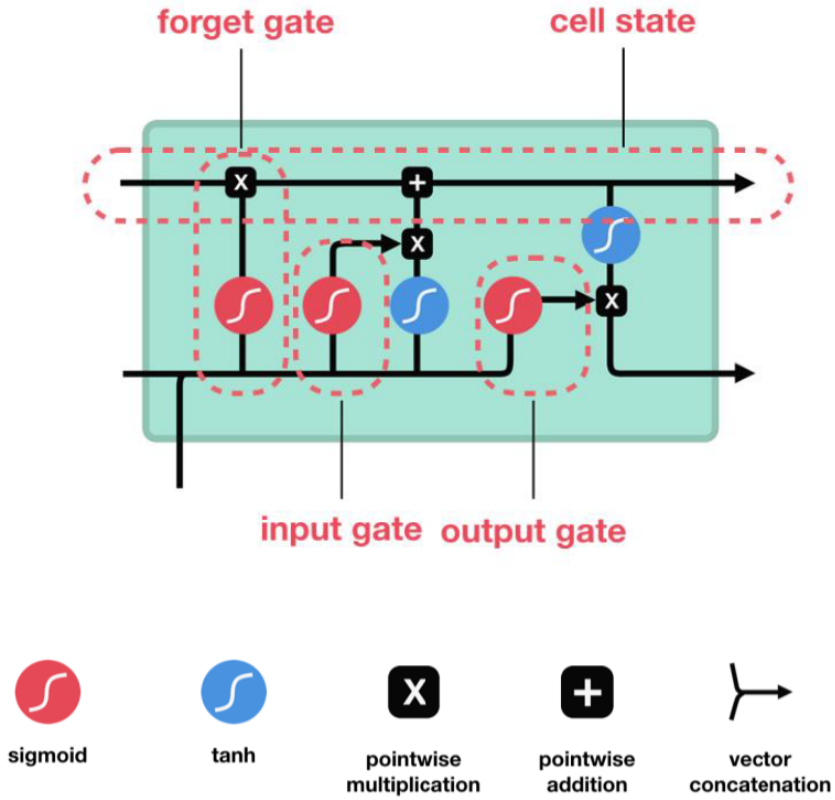

Figure 2.4: Illustration of an LSTM cell structure. Reprinted from: [Phi18]

LSTMs are commonly used in tasks that involve sequential data e.g Natural Language Processing (*NLP*) [HXY15], image classification [LZHY17] as well as speech recognition [GJM13]. They were introduced by Hochreiter & Schmidhuber in 1997 [HS97] and have been widely used since. The LSTM cell consists of three gates: forget, input and output (figure: 2.4). The aim of these gates is to indicate the importance of the variables at a time step, and whether they should be remembered or forgotten for the following steps. A sigmoid non-linear function similar to 2.2.3, produces an

output between 0 and 1 which indicates the importance, this value is later used as a multiplier for the input vector. The forget gate $f_i^{(t)}$ (for time step t and cell i) works as follows according to Goodfellow (2016) [GBC16]:

$$f_i^{(t)} = \sigma\left(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)}\right)$$

where x^t is the current input vector, h^t is the current hidden layer containing the outputs of all the LSTM cells, σ is the sigmoid unit, and b^f, U^f, W^f are biases, input weights and recurrent weights for the forget gates. The LSTM cell internal state is updated as follows:

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma\left(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)}\right)$$

where $f_i^{(t)}$ is a conditional self-loop weight, $g_i^{(t)}$ is the external input gate unit at time step t and cell i which is computed as follows:

$$g_i^{(t)} = \sigma\left(b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)}\right)$$

similar to the forget gate with a sigmoid unit, but with its own parameters. The output $h_i^{(t)}$ of the LSTM cell can also be stopped through the output gate $q_i^{(t)}$ as follows:

$$h_i^{(t)} = tanh(s_i^{(t)}) q_i^{(t)}$$

where the output of $q_i^{(t)}$ is also a value between 0 and 1 as it uses a simgoid unit, and uses its biases, input weights and recurrent weights to compute the mentioned value. This formulas and their description in this section are entirely based on Goodfellow 2016 [GBC16].

2.4 Dimensionality Reduction

"Dimensionality reduction is the transformation of high-dimensional data into meaningful representation of reduced dimensionality" [VPV+09]. In the context of machine learning and neural network explainability it is crucial, as the inputs of such models are typical high-dimensional. In order to visualize such data, dimensionality reduction techniques are needed. In this section we present two of the most popular techniques, that were used in our project.

2.4.1 Principal Component Analysis

"Principal Component Analysis (PCA) is a linear technique for dimensionality reduction, which means that it performs dimensionality reduction by embedding the data into a linear subspace of lower dimensionality". "It attempts to find a linear mapping M that maximizes the cost function $trace(M^T cov(X)M)$, where *cov(X)* is the sample covariance matrix of the data *X*. This linear mapping is formed by the *d* eigenvectors of the sample covariance of the zero-means data". In other words PCA solves the eigenproblem $cov(X)M = \lambda M$ [VPV+09]. The number of principle components (PCs) is equal to the number of features, i.e the number of dimensions of the input. In this thesis only the first and second principle components are considered, which is a two-dimensional output.

2.4.2 t-Distributed Stochastic Neighbor Embedding

t-Distributed Stochastic Neighbor Embedding (t-SNE) is the second dimensionality reduction technique we are presenting. It "minimizes the divergence between two distributions: a distribution that measures the pairwise similarity of the input objects and a distribution that measures pairwise similarities of the corresponding low-dimensional points in the embedding" [Van14]. Mathematically speaking it defines the joint probabilities $p_{i,j}$ that measures the pairwise similarity between two objects by symmetrizing two conditional probabilities as follows:

$$p_{i|j} = \frac{exp(-d(x_i,x_j)^2/2\sigma_i^2)}{\sum_{k\neq i} exp(-d(x_i,x_k)^2/2\sigma_i^2)}, p_{i|i} = 0$$

$$p_{ij} = \frac{p_{j|i}+p_{i|j}}{2N}$$

where x_i, x_j are two objects, N is the number of samples, and σ_i is the bandwidth of the Gaussian Kernel. Furthermore a d-dimensional mapping $y_1, ..., y_N$ is learned, which reflects the similarity p_{ij}. It measures the similarity q_{ij} between two of the mapped points y_i, y_j as follows:

$$q_{ij} = \frac{(1+\left|\left|y_i-y_j\right|\right|^2)^{-1}}{\sum_{k\neq l}(1+\|y_k-y_l\|^2)^{-1}}$$

Finally, the location of the points are determined by minimizing the Kullback-Leibler divergence of the distributions *P* and *Q*:

$$C(\varepsilon) = KL(P||Q) = \sum_{i\neq j} p_{ij} log\frac{p_{ij}}{q_{ij}}$$

2.5 Rainfall-runoff modeling

An important task in hydrology is rainfall-runoff modelling. Its main goal is to predict the stream flow in a river to allow decision makers to make informed decisions in regards to management, risk prevention, etc. [SHC+08]. The water resources within rivers are typically dependant on multiple weather and ecological features, being predominantly dependant on rainfall, but also other attributes such as temperature, soil moisture etc. Hydrologists use such weather attributes in order to predict the discharge at a certain time, according to their understanding of the system. In this section we will describe some of the approaches being used in order to estimate those factors.

2.5.1 Linear Reservoir

Conceptual models for rainfall-runoff modelling incorporate simplified physical elements [OCo97]. Those physical elements are represented through water storage in reservoirs. The most simple form of a conceptual model is a *linear reservoir*. "The linear reservoir concept is based on analysis of the recession limbs of drainage hydrographs" [BDWD04]. The assumption here is that the outflow happens at a constant rate $k = 1/T$ throughout time T, and by knowing the currently stored amount S, one can estimate the outflow Q at any given time as follows:

$$Q = kS$$

$$\frac{dS}{dt} = -Q$$

While this approach can be used to estimate the water buffering capacity of a storage, it is still difficult to generalize to all reservoirs, as many different constants and attributes can come into effect. Therefore more complicated non-linear concepts need to be explored.

2.5.2 Physics-Based Rainfall-Runoff Modelling

Conceptual or physics-based models attempt to solve the problem by mimicking or simulating the processes that occur in the real physical world in order to estimate the runoff. They rely on catchment characteristics as well as weather parameters in order to simulate the water trajectory after a rainfall event. A main characteristic of such models is to preserve the physical meaning of the parameters and to generate scale-independent models. An upside of these models is their generalizability as they preserve the natural attributes of a given reservoir and its environment, e.g. topography-geomorphology, soils and vegetation. A downside however for these models is their reliance on large data sets and requiring lengthy computation times. Another limitation is that setting parameters depends on characteristics that often are difficult to measure for large areas [LT02].

2.5.3 Data-Driven Rainfall-Runoff Modelling

Data-Driven models attempt to solve the problem by exploring patterns between runoff datasets and system drivers that are assumed to generate the runoff [LS20]. Such approaches are further leveraged by the rise of machine learning applications. Scientists are questioning whether data-driven models can be a feasible replacement for classic physics-based approaches. Data-driven models and especially Artificial Neural Network-based models are able to find patterns in complex non-linear underlying processes, they can eliminate noise, and are able to adapt to changes that occur over long periods of times. However, these models have disadvantages that should also be considered. Their performance relies heavily on both the quality and the quantity of the data available. Another limitation is the lack of physical concepts and relations such models come with.[App00]

As powerful as these models are, there is always some scepticism surrounding their usefulness for the reasons mentioned above, but also for their black-box behavior when it comes to their explainability.

2.5.4 Hybrid Modelling

Hybrid modelling is a method that aims to combined physics-based models and data-driven models as a means to complement each others disadvantages. Having a data-driven model that is able to identify complex patterns while still being able to preserve the physical concepts of the real world is considered to be the way forward. In chapter 4 we are going further into detail about such a model, and how a conceptual model (HBV) was used alongside an LSTM model for this project.

3 Related Work

Interpreting Deep Neural Networks (DNNs) through visualization has been a popular approach. Different visualization methods have been used to interpret different aspects of a neural network. In most approaches the common factor was to combine different plots to answer different questions for the same model, to meet the requirements to different types of users [YS18]. In this section we will present some of the related work aimed to visualize and interpret different types of neural networks in different scientific fields.

Hohman et al. (2018) addressed a general human-centered interrogative proposition for visual analytics in regards to deep learning interpretability. They created a survey trying to answer the Five W's and How (Why, Who, What, How, When, Where) as a guideline for researchers and next frontiers working on interpreting neural networks through visualization approaches. Their research went beyond visualization-focused methods to extend "a wide scope that also encompasses relevant works from top venues in AI, ML, and computer vision". [HKPC18]

Strobelt et al. (2018) developed *LSTMVis*, a visual analysis tool for recurrent neural networks with a focus on understanding hidden state dynamics for interpreting an LSTM model trained to solve an NLP task. The tool utilized a filtering feature that allows the user to focus on an input range of the data as a means to explore a predetermined hypothesis specific to a certain subset. They suggested a workflow consisting of goals and tasks to be followed by the user in order to have a beneficial experience using the tool. The tool consisted of a parallel coordinates plot to represent the hidden states, for their ability to represent multi-dimensional data. Furthermore the tool contained a heat map representation to allow the user to visually determine which hidden states are being effective for classifying certain words. They called this component the "Select View" as it enables user to understand and select the required data accordingly. The last component, the "Match View", is a combination of visualizations aimed to relate the data with the model inner states in order to evaluate the user's hypothesis. The tool effectiveness was mainly centered around online participants, with mostly usability metrics being tracked, and qualitative feedback evaluations. The tools effectiveness for model understandability and interpretability remains ambiguous. [SGPR17]

Kahng et al. (2018) worked alongside Facebook researchers to develop *ActiVis*, a Deep Neural Network visualization method, aimed for interpreting large scale models. ActiVis has been deployed to Facebook's machine learning platform and helped extract usage scenarios and insights for working with different models. The tools design goal are "to unify instance- and subset-based analysis, to tighten integration of overview of model architecture and localized inspection of activations and to enable industry-scale datasets and models". As for other similar approaches, the tool consisted of different visualization components and techniques. The first component was a computation graph summarizing the model architecture. The second component consisted of neuron activation matrices summarizing activations for instances, subsets and classes. This included a t-SNE projection of the instances activation, with color coding for representing the different classes. The third component was meant for instance selection, it consisted of a performance representation of different classes of

the model. Their work concluded that working with gradients is an effective approach for visualizing neural networks. Despite the satisfying results of their work, computational efficiency for real time analysis was a major concerns that was intended to be explored for future work. Another future work this approach might have inspired is the ability to automatically suggest interesting subsets for exploration to the users. [KAKC17]

Fong & Vedaldi (2019) tried an answer a similar question regarding Neural Network interpretability. "Why are algorithms making certain predictions?". Their solution for answering this question included two main approaches. The first approach is introducing *Meta-Predictors* as means for explanations. The second approach is creating attribution methods, which also include visual gradient attribution maps. Both methods aimed to unmask the black-box nature of a neural network model for image classification. The research "contributed new insights into the fragility of neural networks and their susceptibility to artifacts". Furthermore they suggested metrics such as *Smallest Destroying Region* (SDR) to evaluate attribution methods, as attribution methods' output can be highly effected by a network learning a poor association. Finally, the gradient selectivity maps deemed successful for detecting the most influential regions on model output. [FV19]

Garcia et al. (2021) attempted to visualize hidden states of a one layer LSTM through an interactive tool. The main idea was to allow users to inspect how hidden states store and process information throughout the feeding of input sequences for an IMDB review classification task. This was achieved through showing input trajectories alongside a PCA dimensionality reduction for the hidden states, with color mapping indicating the corresponding predicted classes. This shows an input sequence and how the prediction is determined by moving through the hidden states dimensionality reduction, until finally the prediction is created. The approach deemed promising in creating low-dimensional manifold embedding in a high dimensional space. The visualization tool was successful in allowing interpretations for causes of incorrect predictions by the model. [GMW21]

Kratzert et al. (2019) worked on interpreting LSTM models specifically trained for hydrological purposes. The model in question was used for rainfall runoff forecasting, by which a river discharge has to be predicted from meteorological observations. Questions related to the count of days influencing the output of the network given the day of the year, correlation between memory sells with hydrological states and the inputs' influence on the cells where answered. These answers were backed by a combination of visualizations such as line plots and heat maps for presenting the integrated gradient signals, time steps of influence (TSOI), median aggregations in a time series as well as correlation values between inputs and hidden states. They were able to show that "the processes learned by the LSTM matched their comprehension of a real-world environment system". They concluded that combining domain-knowledge with their proposed interpretation techniques "provide the fundamentals for designing environmental forecasting systems with neural networks". [KHK+19]

4 Model Description

In this chapter we will describe the model architecture and its hyper parameters, as well as the input data used. Furthermore we will describe the evaluation methods and the model performance analysis. The entire chapter is based on Chabok (2022) [Cha22]. As he attempted using different architectures with different input features and hyper parameters, we are only going to focus on the best performing model, and not consider the different iterations for improvements he went through. The model was based on the HBV model constructed and calibrated by the Institut für Wasser- und Umweltsystemmodellierung (IWS) at the University of Stuttgart. The HBV model is based on three meteorological data sets (Temperature, Precipitation and Potential Evapotranspiration) which generate twelve internal model variables as shown in figure 4.1.

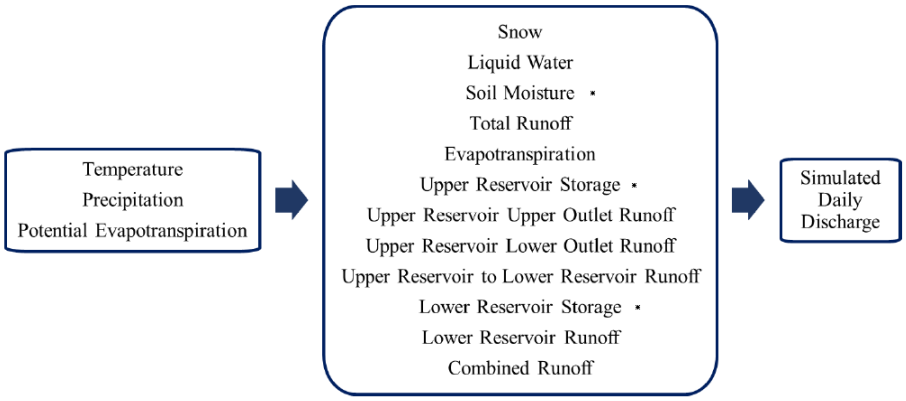

Figure 4.1: Meteorological inputs, HBV's internal states and final output. *: can be described as states [Cha22]

The processes of the conceptual model can be categorized into three units: Snow accumulation and melt, soil moisture and evapotranspiration, runoff and flow routing. **1**. Snow accumulation and melt can be generated through precipitation and temperature, by tracking weather the temperature threshold is reached, at which rain is transformed into snow, or at which snow melts. **2**. The soil moisture and evapotranspiration module is calculated through comparing evapotranspiration level with available soil moisture. This is a summation of previous soil moisture compared to previously produced liquid water, which has not turned into runoff. **3**. Runoff and flow routing unit constructs two conceptual reservoirs (upper- and lower reservoir), which are located on top of each other. The upper reservoir, being the first receiver of water in the case of heavy precipitation gets filled the fastest. It has two outlets, the top outlet which produces immediate surface runoff, and a bottom outlet which generates the interflow. The lower reservoir has only one outlet which is placed at the bottom and accounts for base flow simulation. By keeping track of the outflow of each outlet of the reservoirs, which all depend on the runoff response and their respective locations, we are able to

calculate a simulated discharge, which is a summation of the outflows of all three outlets multiplied by the catchment area. The entire structure of the HBV conceptual model can be seen in figure 4.2, the numbers 1-3 in this section correspond to the numbers in the same figure.

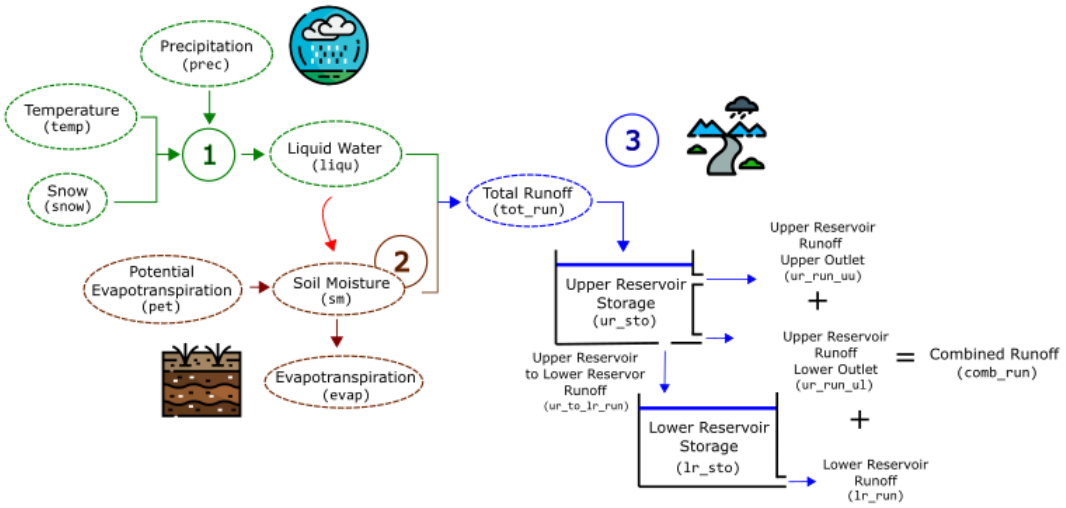

Figure 4.2: An overview of the conceptual model's internal states and their interaction.

4.1 Data

The data used for the model was collected from the upper Neckar catchment in southwest Germany from mid 1961 until the end of 2010. The input is split into training, validation and test sets according to the conceptual model's calibration and validation periods. The training and test sets are meant to cover the calibration period, the test set however is the validation period for the HBV model. The split is best described in figure 4.3. It is worth noticing that the first six months at the start of the calibration and validation periods are not taken into the ML model, as they are warm-up periods of the HBV model. This is the list of the features that were measured and used for training the model:

- **Temperature** (temp): in °C
- **Snow** (snow): in mm
- **Precipitation** (prec): in mm
- **Liquid Water** (liqu): in mm
- **Potential Evaporation** (pet): in mm/day
- **Soil Moisture** (sm): in mm
- **Evapotranspiration** (evap): in mm/day
- **Total Runoff** (tot_run): in mm/day
- **Upper Reservoir Storage** (ur_sto): in mm

- **Lower Reservoir Storage** (lr_sto): in mm
- **Upper Reservoir to Lower Reservoir Runoff** (ur_to_lr_run): in mm/day
- **Upper Reservoir Runoff/ Upper Outlet** (u_run_uu): in mm/day
- **Upper Reservoir Runoff/ Lower Outlet** (ur_run_ul): in mm/day
- **Lower Reservoir Runoff** (lr_run): in mm/day
- **Combined Runoff** (comb_run): in mm/day

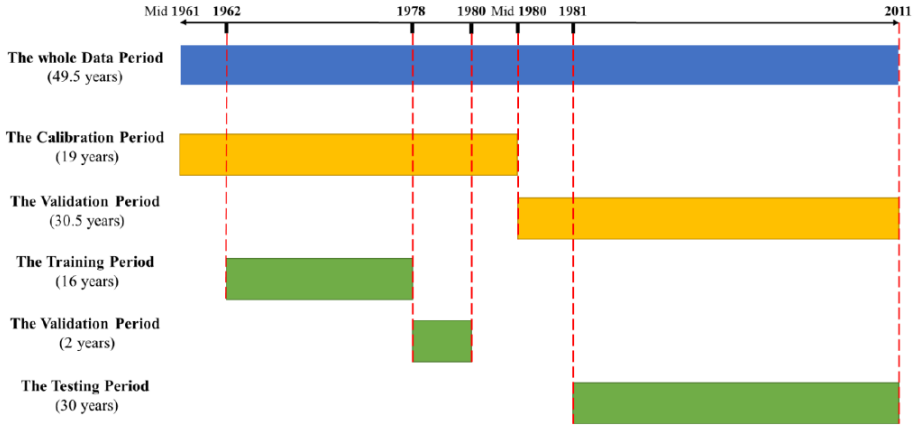

Figure 4.3: The calibration and validation periods in HBV model (orange) vs. training, validation, and testing sets for machine learning (green). [Cha22]

The input vectors were scaled using a quantile transformer. Quantile Transformation, also known as Normal Quantile Transformation *NQT* is a transformation method that is used as a means to create a Gaussian distribution function. The model uses it to boost performance due to the nature of neural networks where the values of weights and biases are rather small. After sorting the sample X in ascending order, we transform each observation x_i of X into observation $y_i = Q^{-1}$ of "the standard normal variate Y, with Q denoting the standard normal distribution and Q^{-1} its inverse, applying discrete mapping" [BPC12]. In figure 4.4 a histogram of a scaled batch using the quantile transformer can be seen for the different features. Notice the resemblance of a normal distribution with minimal skewness.

4.2 LSTM Architecture

The hybrid LSTM model, built using PyTorch [1] [PGM+19] —an open source machine learning library— consists of an input layer, 2 hidden layers with 11 hidden cells each followed by a linear layer. The input vectors had three dimensions. The different dimensions described the number of batches, the number of samples at each batch, and the number of features respectively. The batch size used had the length 365 which resembles the days of a year as each sample represents one day. The architecture was based on the following rules of thumb described in [Hea08]:

[1] https://pytorch.org/

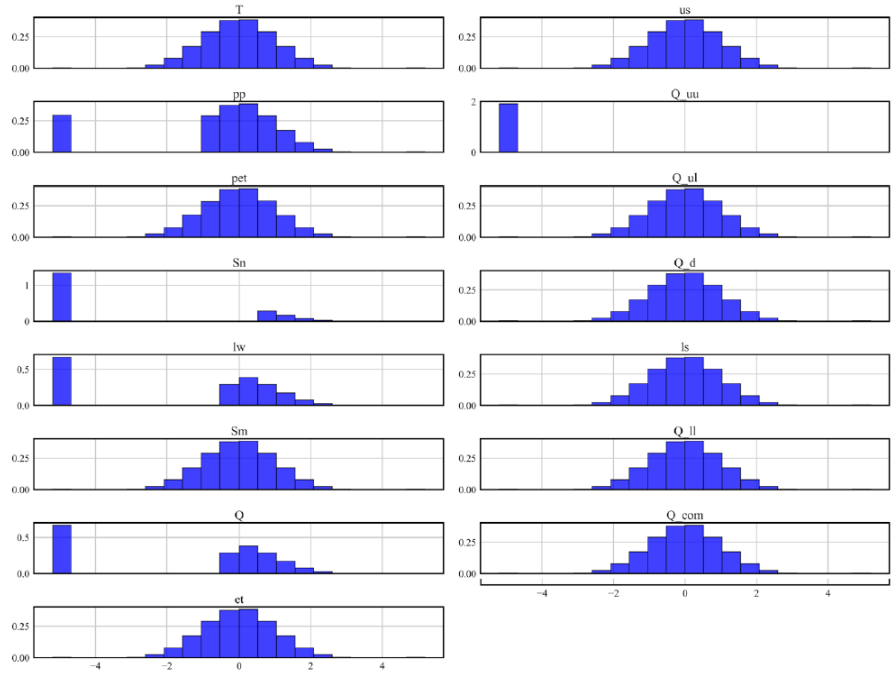

Figure 4.4: Histogram of scaled train data in a batch with the normal quantile transformer. [Cha22]

- *output_size* < *n_hidden* < *input_size*
- $n\_hidden \cong \frac{2}{3} * \ input\_size + output$
- *n_hidden* < 2 * *input_size*

4.2.1 Hidden and Cell state initialization

For hidden and cell state initialization an approach by Wenke and Fleming [WF19] was followed. The initialization was done by using the learnable encoded information of the inputs, namely the observed discharge values. This initialization method inserts more physical knowledge into the model [Cha22]. In order to avoid unnecessary complexity of the additional parameters that are a side effect of this approach, namely weights and biases of the encoder, a detach command in PyTorch was used.

$$h_0 = encode(X)$$

Where:

- h_0 is the array of initial hidden states
- X is the array of inputs
- *encode* is a linear neutral network with weights and biases

4.2.2 Weight and Bias initialization

The weight initialization for the LSTM was done using the default PyTorch technique. This can be described as follows:

$$W \sim U[(-\sqrt{\frac{6}{n+m}}, +\sqrt{\frac{6}{n+m}})]$$

Where:

- n is the number of features in the previous layer
- m is the number of features in the current layer

The bias initialization for the forget gate is initialized randomly in the range [1,2) or equal to 1. In other words, allowing the gates to not forget the inputs unless the model has learned to forget [GSC00]. This step was crucial as to not cause the model to discard valuable long term knowledge.

4.2.3 Optimization & Learning Rate Scheduler

Adam was used as an optimizer for this model, along with a cosine annealing learning rate scheduler with warm restarts. This can be found in the PyTorch learning rate scheduler module. This combination not only produces better results, but also learned in fewer iterations.

4.3 Performance analysis

After iterating over many different versions of the model, a final best performing model has been selected. This model uses a quantile transformer, Adam optimizer, dropout, weight restart, bias initialization and an encoder. All of these mentioned in section 4.2.

The Performance of the model was measured using Nash–Sutcliffe model efficiency coefficient *NSE*. The NSE is an index for assessing the performance of hydrological models. The efficiency index is a value between 0 and 1 where 1 indicates the best possible performance for a model [MKC06]. The computation of this index is defined as follows:

$$NSE = 1 - \frac{\sum_{n=1}^{N}(\hat{Y}_i - Y_i)^2}{\sum_{n=1}^{N}(Y_i - \bar{Y}_i)^2}$$

Where:

- N is the sample size
- $\hat{Y}_i$ and Y_i are predicted and measured values of the cirterion dependant variable Y, respectively
- $\bar{Y}_i$ is the mean of the measured values of Y

Model	NSE			NSE for peaks			Best Iteration
	Test	Train	Valid	Test	Train	Valid	
best performing model	0.834	0.887	0.820	0.712	0.802	0.720	451

Table 4.1: NSE values of the best performing hybrid LSTM model. [Cha22]

In table 4.1 the NSE values of the model can be seen. The table is split into two halfs, namely NSE and NSE for peaks. NSE resembles the overall performance of the model. NSE for peaks is the performance for the time steps, which have an observed discharge value greater than 95 percent confidence. The calculated NSE values for testing, training and validating for the overall performance as well as the peaks is very satisfactory. Especially relevant for us is the NSE value for testing, as our visualization tool will be enriched using the testing data set of the model. In figure 4.5 a better overview of the performance can be seen.

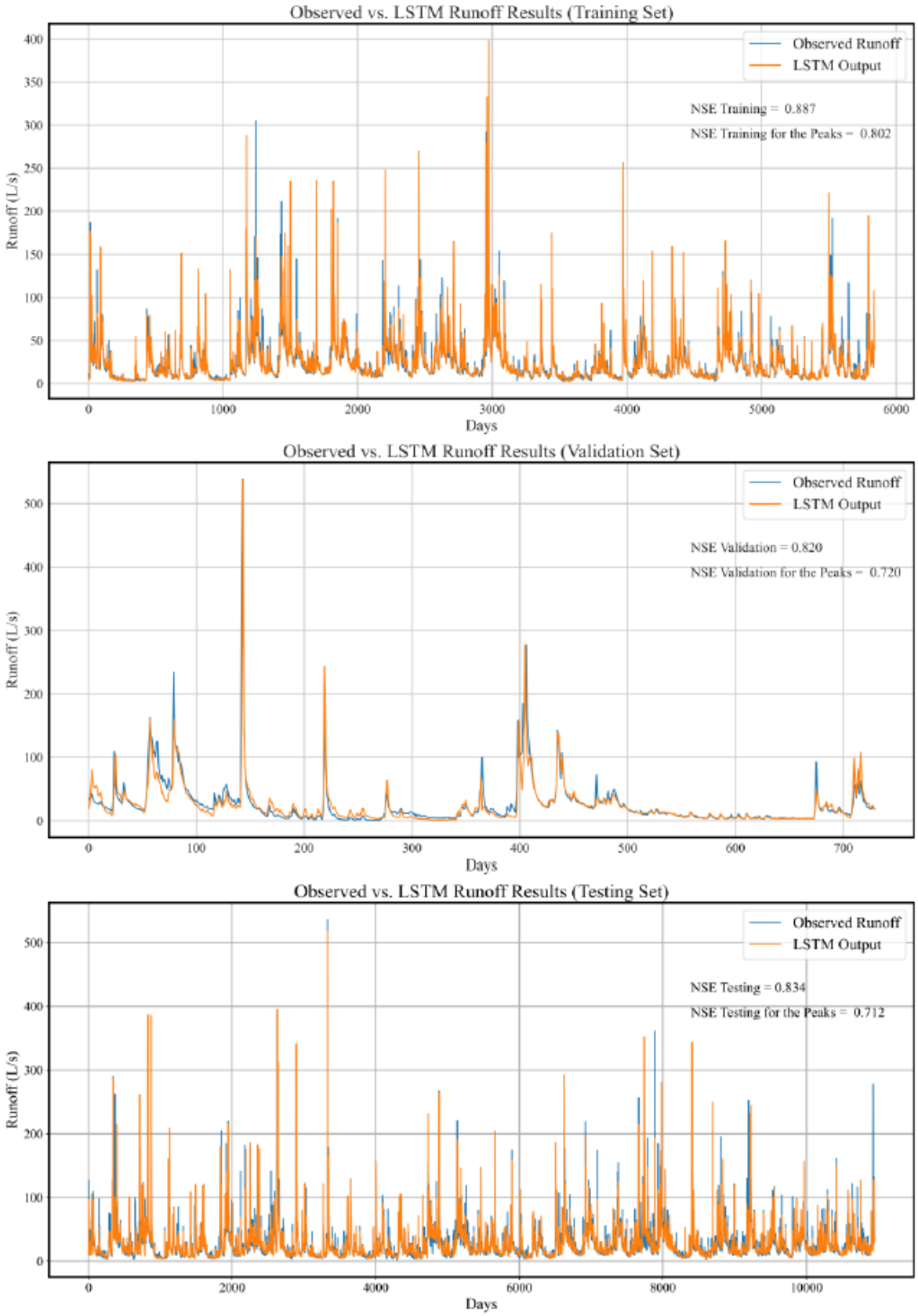

Figure 4.5: Observed vs Prediction plot for the three sets of the best performing LSTM model. [Cha22]

5 Visualization Tool

The goal of our visualization tool is to explain both the results of the model, as well as the process the model has taken during and after training in order to calculate its output. We are interested in seeing the correlations between the output and the individual inputs, as well as being able to understanding what is happening within the black box of the model. Different visualization approaches were taken in order to fulfill both needs. An important note here is, being able to visualize different dimensions and conclude how they interact together. The input is typically a sequence of 365 days which includes 16 different features (365x15), the observed values are depicted through a vector with shape (365x1) and the output is 1 value which is a prediction of stream-flow for the day as described in 4.1. Furthermore, the model consists of 2 hidden layers, each of which has 11 cells, for which we want to understand the evolution of the hidden states for every time step. In this chapter we will explore how the data was extracted for a pre-trained model, how the visualization tool is developed, the decisions we've made during development, and finally a suggestion of a workflow to use in order to get the most information from the tool.

5.1 Data Preparation

In order to generate a meaningful and comprehensible visualization, data preparation was necessary. The steps of the data preparation phase included hidden state extraction as well as dimensionality reduction. It should be noted that the first 20 years of data were used for training and validating the model, hence those years not being considered for the visualization tool as we are only interested in the testing data, which was used to measure the performance of the model. The visualization tool is mainly focused on the testing period and the first 20 years are therefore disregarded.

For the entire 30 years of testing we have a data of shape (10950x15) with observed runoff (10950x1). Since each input has the sequence of 1 year, the first day we could predict was after 365 days. This means that our predictions start for the testing data for the final 29 years (10585 days).

We used dimensionality reduction techniques to visualize the entire input data set, as well as each 365 day sequence separately. The techniques used were t-distributed stochastic neighbor embedding *t-SNE* and Principle Component Analysis *PCA* [AW10; GSH15]. The input was extracted both as raw values, as well as scaled values, the same way they were input into the model. The scaling method used was Quantile transformation using n_samples = 365.

In order to visualize this data we extracting the hidden states within the testing loop for every time step and storing them into a list that would later be stored as a csv file of all hidden states. The extraction of the inputs was done prior to launching the visualization tool in order to increase the efficiency, nevertheless with a decent system the inputs can be extracted on the spot using the user defined filters within the tool.

It's worth noticing that the first iteration of the testing, the hidden states had already values due to the training, that were imported through the final state dictionary of the pre-trained model. This is similar to a "warm up period"commonly used in conceptual models.

Furthermore, we extracted additional data to describe the model itself. Having 2 hidden layers with 11 cells each that changed with every input of a 365 days sequence resulted in a 3d matrix of shape (10585x11x2). In order to make a meaningful visualization out of this data shape a dimensionality reduction technique was required. In order to analyze both layers simultaneously we transformed the matrix to (10585x22) allowing us to keep the cells of both layers in the same dimension. We also used dimensionality reduction techniques to analyze the transformation of hidden states across time steps visually. The dimensionality reduction techniques we used were t-SNE and PCA. Additionally we calculated the Standard Precipitation index *SPI*. The SPI is a transformation method for quantifying rainfall variability. It is commonly used in monitoring drought scenarios. Its output is a numeric value, which we mapped to a color, as a means of making a statement about the year's weather, without having to read into each day's weather related variables. [KGR08]

5.2 Tool development and walk through

In this section we will explore the development process, the libraries used as well as the reasons for some of our decisions. Furthermore we will walk through the tool explaining each section, the visualization methodology as well as description of the color mapping, the interactive widgets and the data being visualized.

5.2.1 Development tools and methods

Our Web-based Visualization tool was developed using Python 3.8.8 on Jupyter labs [1]. Hvplot[2] version 0.7.3, which is a high-level plotting API that is built on HoloViews[3] and Bokeh[4], was used to create the interactive plots. HoloViews is an open-source Python library designed to make data analysis and visualization, and Bokeh is a Python library for creating interactive visualizations for modern web browsers, it helps create Javascript powered visualizations, without having to write Javascript code. The layout of the page was developed using Panel[5] version 0.13.0. Panel is a Python library for creating web-based interactive dashboards.We used panel to create all the widgets, filters and the user input fields. The reason for using this stack was its usability in Jupyter labs notebooks, ease of debugging and simplicity of use, while not having to compromise the quality of the developed tool. A different development library could've been Plotly and dash, but the trade off of not being able to use notebooks was not worth it. An earlier version of the tool was developed using Javascript and D3, but we opted for using Python due to its developing efficiency and ease of use. Though D3 allowed us to have a lot more control on micro decisions, it was not worth it as those micro decisions ended up being mostly irrelevant for the scope of this thesis.

[1] https://jupyter.org/
[2] https://hvplot.holoviz.org/
[3] https://holoviews.org/
[4] https://docs.bokeh.org/en/latest/
[5] https://panel.holoviz.org/

5.2.2 Sections and plots

General features

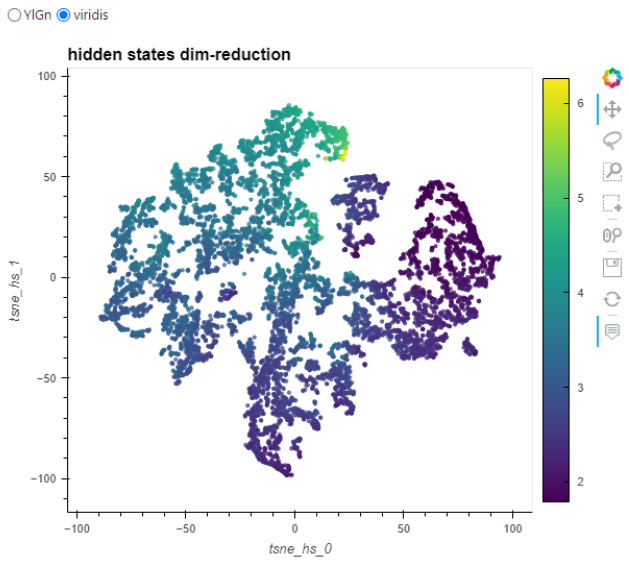

Figure 5.1: A scatter plot in Viridis color mapping. x-axis represents hidden states t-SNE dimension 0, y-axis represents hidden states t-SNE dimension 1. Color represents the log scaled prediction of the model.

The tool offers multiple features that aim to make the visualizations interactive. Those features are placed on the side bar of the tool as well as above the corresponding plots. There are two modes for the tool to be used. One mode is for free viewing, which allows the user to freely analyse multiple data points in a selected date range, while the second tool is meant for analyzing a single data point. The two modes can be active simultaneously and the features in the sidebar give the user the ability to do this selection.

The first feature is a date range selector which is available in the side bar and above the corresponding plots. It consists of two text fields, where the user can type two dates (*yymmdd*). These two dates are filters for the data points, such that the plots only contain the dates in between. There is another text field, which is used for selecting one specific date. This date works on the corresponding plots, which allow the user to analyze one specific day of the data.

Another feature available is the color map selector. It is a select widget where the user selects from a collection of different color maps, e.g Viridis and YlGr. By default the color map Viridis is selected, but once the user selects a different mapping, the plots directly swap to the selection of the user. As seen in figure 5.1 the scatter plot color mapping is Viridis, according to the selection widget.

Furthermore there are multiple selector widgets available across the tool, that allow the user to adjust the filtering of the data. Some of which are offered through toggle buttons, which change the data points to be shown in the plot. But others are offered through multi-selector widgets, which allow the user to select the variables to be shown in a plot. A third type of selectors however

is the box/ lasso selectors. Those offer the most control over which values to show, as the user can explicitly select points visually from a scatter plot, by drawing over them or by dragging and dropping in order to filter the values from a table.

Correlation Heat map Plots

For the user to analyse the correlation between the features, the prediction and the hidden states a plot was created in the form of a correlation heat map (Figure: 5.2). This heatmap uses a custom color mapping, that strengthens the difference between the values with easily identifiable colors. The color mapping used is: [#75968f , #a5bab7 , #c9d9d3 , #e2e2e2 , #dfccce , #ddb7b1 , #cc7878 , #933b41 , #550b1d]. These plots enable the user to identify positive and negative correlations between different variables, the most important features in regards to the prediction, as well as identify variables that effect specific hidden states, and potentially draw conclusions on the importance of specific hidden cells or perhaps entire hidden layers.

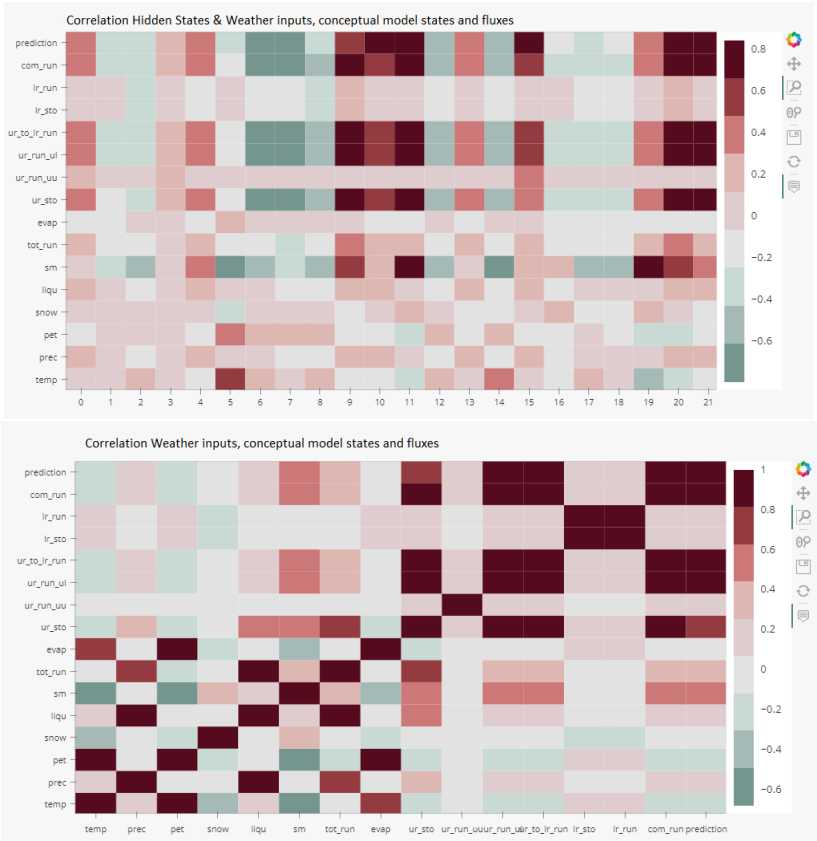

Figure 5.2: Correlation Matrices. **Top**: x-axis represents hidden states, y-axis represents the input features and prediction . **Bottom**: x-axis and y-axis represent input features and prediction. Colors represent the Pearson correlation between the values on both plots.

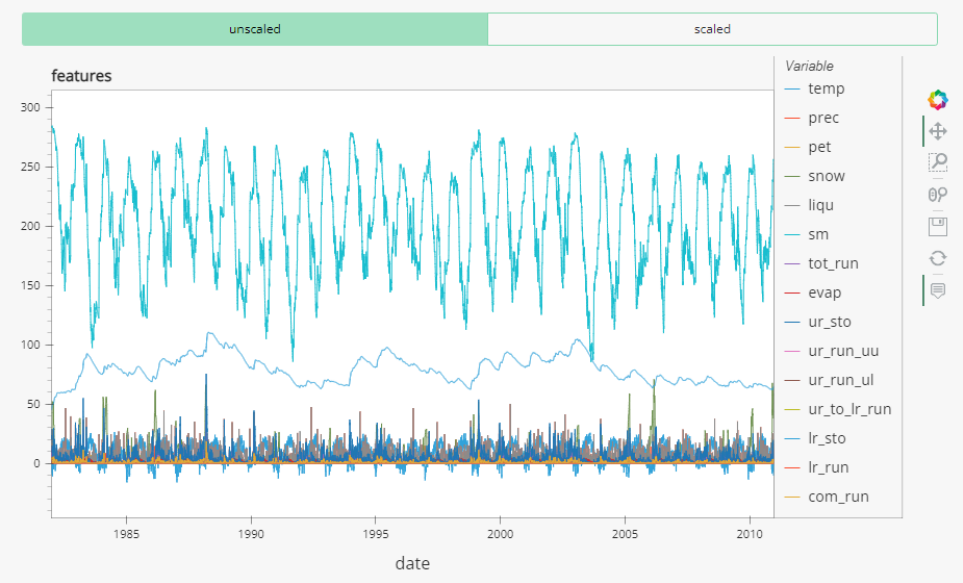

Figure 5.3: Features line plot. x-axis represents date, y-axis represents the values of each feature in its corresponding measurement unit.

Features Time Series Plot

The features plot (Figure: 5.3) is a line plot for all the input features within the selected date range. Each feature is mapped to a different color randomly assigned and labeled on the right side of the plot. The x-axis describes the value of each variable while the y-axis describes the date. All the variables are automatically selected, but the user is able to click on the labels to chose which variables to be visualized. On the top of the plot there are two buttons that can show the unscaled or the scaled values respectively. This plot allows users to have an overview of which variables are being used for training the model, as well as give intuition for their actual values.

Features vs Prediction Time Series Plot

The features vs prediction (Figure: 5.4)is also a line plot for all input features in the selected date range, the difference however from the previous plot is the ability the plot provides for the user to compare the values with the prediction of the model. There is a version for this plot for scaled values and another version for un-scaled values. On the top of the plot there is a multi selector which allows the user to select one or multiple values to be shown on the plot for the y-axis. The prediction value is always part of the plot and cannot be unselected. This plot, specially the scaled version, allows the user to compare the values with the prediction. A user can directly see the model performance by comparing the prediction with the observed value. Furthermore a correlation can be directly seen for the different variables with the prediction, whether it be a positive or a negative correlation. The prediction is always assigned the orange color, while the other values are assigned a random predetermined color.

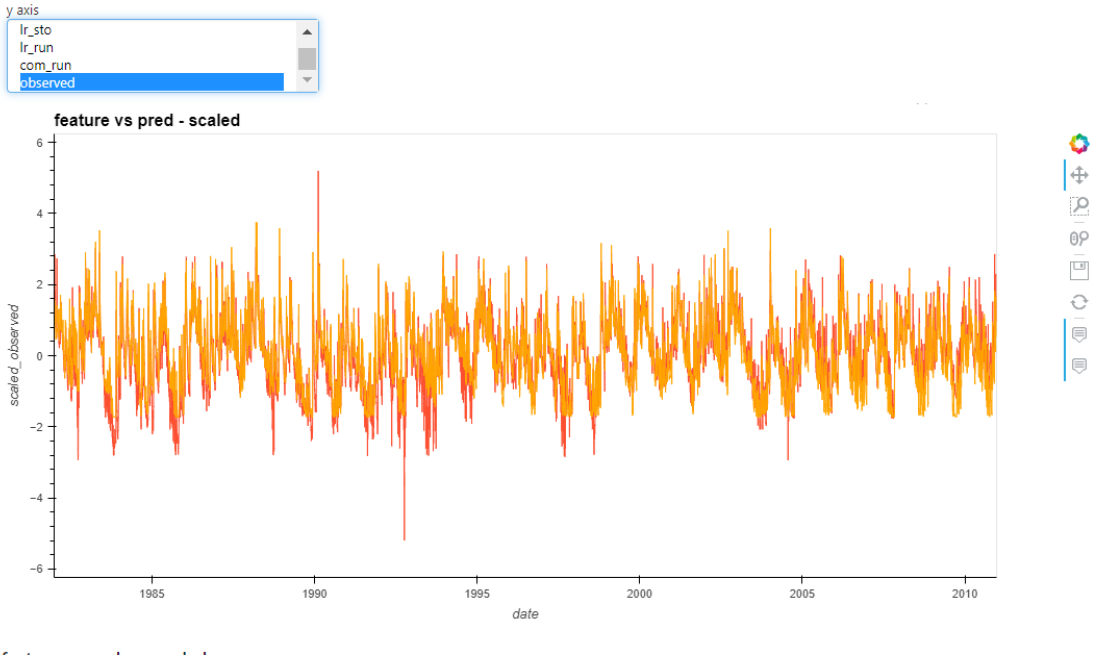

Figure 5.4: Features vs Prediction line plot. x-axis represents the date, y-axis represents the scaled value of each feature.

Dimensionality Reduction scatter plot for Hidden States & Features

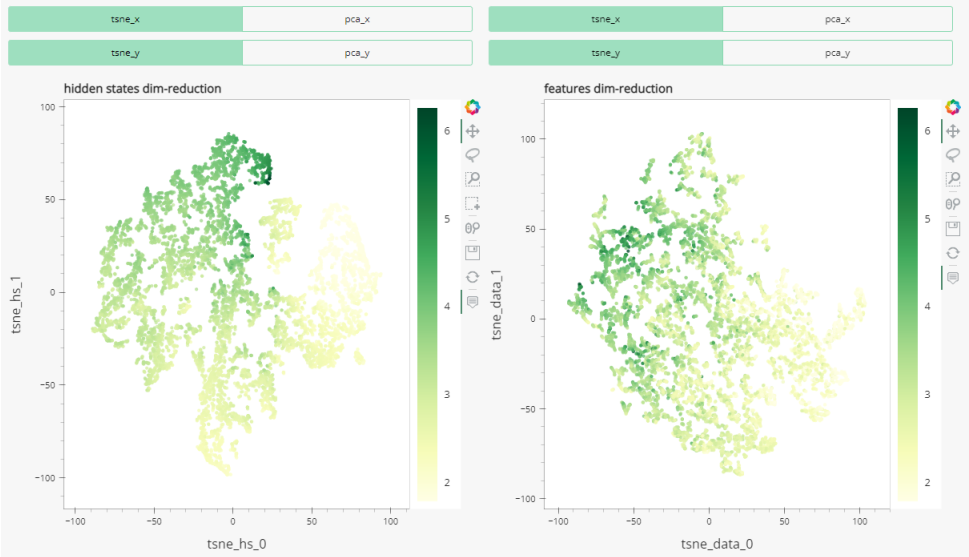

Figure 5.5: Hidden states & Features dimensionality reduction scatter plots. **Left**: Hidden states plot. **Right**: Features plot. **Both plots**: x-axis represents hidden states t-SNE dimension 0, y-axis represents hidden states t-SNE dimension 1. Color represents the log scale of the model prediction.

The dimensionality reduction (Figure: 5.5) section contains two plots side by side. The plot on the left is a dimension reduction of the 22 hidden states in the selected date range during the testing of the model. The color of the points in the scatter plot is mapped to the log-scale of the prediction similar to what Li et al. have done [LLJT14]. Without this scaling the values of the prediction were heavily skewed making the difference within the prediction values difficult to see. On the scatter plot of the right is the dimension reduction for the input variables for the same date range, using the same color mapping. Above each plot 4 buttons can be seen. Each row of buttons represents a different axis (x and y respectively). Each of the buttons changes the axis values to t-SNE and PCA. We opted for a button for each dimension due to a limitation of the visualization library we used for developing the tool. On the right a color mapping scale can be seen in order to give intuition to what each color represents.

Clusters Scatter plot

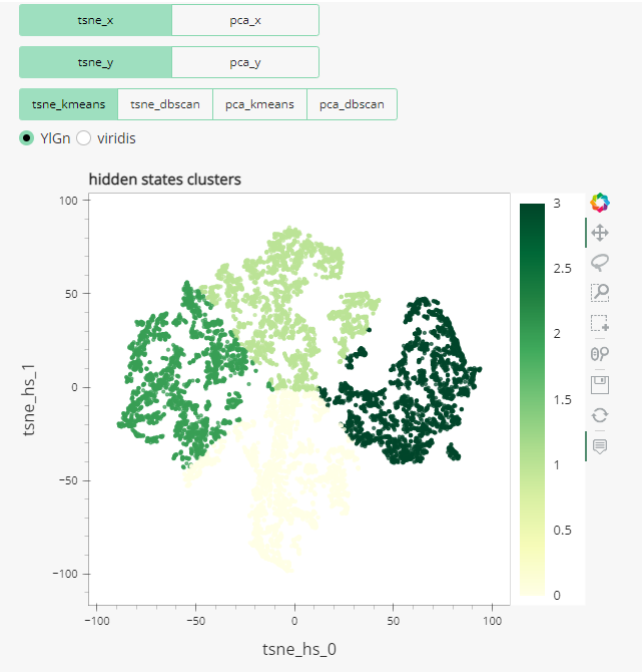

Figure 5.6: Clusters scatter plot. x-axis represents hidden states t-SNE dimension 0, y-axis represents hidden states t-SNE dimension 1. The color represents the clustering of the points.

The clusters scatter plot (Figure: 5.6) was created as a means for analyzing the similarity that can occur between certain days. As the years are passing, weather attributes can be very similar during a certain time of the year, as an effect of seasons. Those cycles can have an effect on the hidden states, that can be detected by clustering the data points. The plot offers multiple interactive tools such as the toggle button widgets that can be seen in the upper part. There are three rows on toggles, each row represents a dimension for the plot. Similar to the dimensionality reduction plot, the reduction can be shown using t-SNE or PCA. The first two rows are meant to apply this decision

respectively. The third row however, controls the color dimension. The color mapping for this plot is the assigning of the points to different clusters using two different clustering algorithms, namely k-means and DBSCAN.

K-means is an unsupervised clustering algorithm. Its idea is to define k centroids, one for each cluster, while assigning the nearest points to each centroid, to build a cluster. The algorithm iteratively changes those centroids and accordingly the associated points in order to minimize the objective function, which is typically a squared error function.

$$W(S, C) = \sum_{k=1}^{K} \sum_{i \epsilon S_k}^{K} \|y_i - c_k\|^2.$$

"Where S is a K-cluster partition of the entity set represented by vectors y_i ($i \epsilon I$) in the M-dimensional feature space, consisting of non-empty non-overlapping clusters S_k, each with a centroid $c_k (k = 1, 2, ...K)$". [KM13]

DBSCAN (Density Based Spatial Clustering of Applications with Noise) is a density based unsupervised clustering algorithm. It requires the user to define a radius (eps) that determines the minimum distance between points in the same cluster and a minimum number of samples a cluster can have. The goal of the algorithm is to select the optimal clustering in order to fulfill both parameters. [KRA+14]

Since the points and the color dimensions can be selected using different buttons, a user can freely compare different clustering methods as well as different dimensionality reduction techniques by changing the color map while keeping the same x and y axis selections. For the k-means we chose k=4, according to the elbow method we ran on the data, which is a heuristic for determining the number of clusters that maximizes the variance between the values of each cluster, while minimizing the within cluster distance of the individual samples [SKRS18]. We conclude that the 4 clusters represent the 4 seasons of the year. The other algorithm used was DBSCAN, with eps = 0.03 with samples = 7 for t-SNE and samples = 10 for PCA. Those values were selected intuitively, to allow a meaningful visualization.

Day sequence plot & Table

The day sequence section (Figure: 5.7) is a combination of a two scatter plots and a table. This section is by far the most loaded section and potentially a very insightful one. The scatter plot on the left is represents a dimensionality reduction of the input variables for a 365 days sequence used to predict the rainfall runoff for the 366th day. There is a yellow dot meant to identify the first day of the sequence and a black dot for the last day of the sequence. On top of the scatter plot there is a line that moves through the points in their chronological order. There are two color mode selections for the rest of the points. The first one shows the standard precipitation index *spi*, which is calculated as following

$$SPI = \frac{X_{ij} - X_{im}}{\sigma}.$$

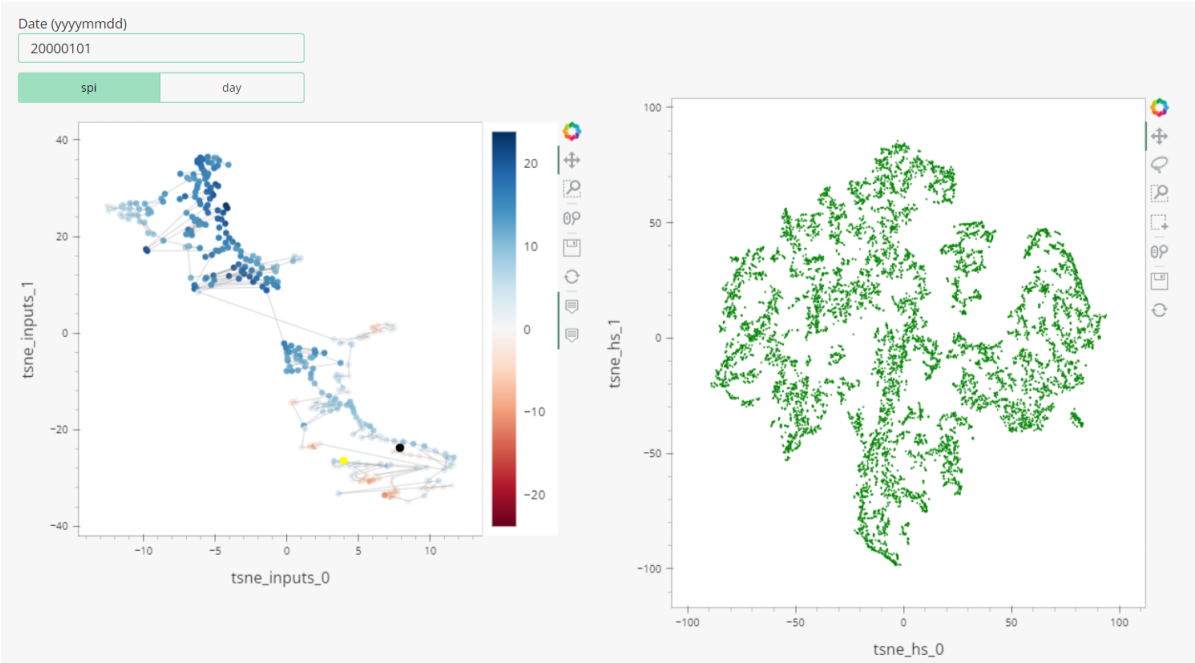

Figure 5.7: Day sequence plot. **Left**: A t-SNE dimensionality reduction of the input features of 365 days prior to the date being analysed. x-axis represents t-SNE dimension 0, y-axis represents t-SNE dimension 1. The color represents the spi value or the day mapping within the 365 days. **Right**: A t-SNE hidden states dimensionality reduction plot. x-axis represents hidden states t-SNE dimension 0, y-axis represents hidden states t-SNE dimension 1.

"Where X_{ij} is the seasonal precipitation in the i^{th} rain gauge station and j^{th} observation, X_{im} the long-term seasonal mean and σ is its standard deviation". [KGR08]

The second mode is meant to simplify the distinguishing of the chronological order of date points. The color mapping used is RdBu. The plot on the right is a remake of the hidden states dimension reduction seen in figures 5.5 and 5.6. The color mapping of this plot cannot be changed due to a limitation of the library in use. Below both plots, there is a table which contains all the input variables, the prediction and the date. The rows of these tables can be filtered using the box selection tool which can be applied on the hidden states dimensionality reduction plot.

Documentation

The tool includes a documentation section which describes every plots axis, color mapping, and if applicable its transformation methods. This section is especially useful for first time users.

5.3 Setup Guide

In the following section we will show step by step how to install the tool and do the setup properly. This guide is for Windows PCs for python 3.8. This is set up to use pip/ venv as package management tools and environment solutions but the user free to use other solutions considering the dependencies of the tool.

1. Clone the library from github.
   `git clone https://github.tik.uni-stuttgart.de/VISUSstud/MA-Hassan-HydrologicalModeling.git`
2. Create a new virtual environment in the same folder
   `python3 -m venv ./`
3. Activate the environment
   `cd Scripts/activate.bat`
4. Install the libraries: hvplot, jupyterlab, pandas, numpy, sklearn, holoviews, bokeh
   `pip install requirements.txt`
5. Deactivate the environment and install panel
   `pip install panel`
6. ***Optional*** - Activate the environment again and run jupyter lab through the cmd
   `jupyter lab`
7. ***Optional*** - Run all the cells in the notebook by clicking the "fast forward"button to make sure the libraries are properly installed.
8. ***Optional*** - Open a new terminal window from jupyter lab by pressing the big plus sign on the top left corner
9. While using the virtual environment run this command in the terminal
   `panel serve vistool.ipynb --port 5008`
10. Open this link in your browser
    `http://localhost:5008/vistool`
11. Wait for the tool to launch (takes from 3 to 5 minutes)

Note: Steps 6-8 are only optional if you want to access the tool, however they are necessary if you want to access the code and implement changes.

5.4 Workflow suggestion

In this section we will suggest ideas of methods the tools can be used in order to extract the most information from the tool. We will talk about specific order of steps as well as general concepts to work with the individual plots in contrast to one another. The suggested methods are means to understanding and analysing both the hydrological concepts and the rainfall-runoff conceptual model for the Neckar, as well as the LSTM model and its hidden states. Since the tool offers multiple filtering techniques we are going to start by using this to focus on individual subsets of the

data and compare them to one another. Later we will discuss how to understand the model error and pinpoint its direct causes. In most cases the insights we observed were a product of using multiple plots simultaneously to complement each others weaknesses while taking advantage of each plot's strengths.

5.4.1 Hydrological Year

The first orientation we use is in regards to what is called a hydrological year, or a water year. It is a 12 months period of which precipitation totals are measured. The start of a hydrological year differs from a geographical location to another, depending on the precipitation seasons throughout the year. We used the period between the first of November (01.11) and the thirty first of October (31.10) as our hydrological year, since it is the commonly used period in Germany [BS14]. We use this period as an orientation for our analysis and as a means to compare different subsets of the data, by considering each subset as a hydrological year. This can be done by typing the dates mentioned above into the date range selection filters.

5.4.2 One-day Analysis

The second orientation we suggest is a one-day analysis. A certain day can be analyzed using the single day selection filter. This yields the dimensionality reduction for the input features of the 365 prior days, that were used to predict the discharge of that specific date. An example of this can be seen in figure 5.7. The SPI color mapping is especially important for this analysis, as it gives an intuition of the weather condition throughout the year. It is worth noticing that in order to have a beneficial analysis and to build a complete idea using this approach, different dates should also be evaluated and compared, as a one day analysis for only one day can lack comparative information, which are necessary. An example for different days to analyze can be the same date for different years.

5.4.3 Dynamic time range

The user has full control of selecting any arbitrary time ranges to analyze. Using a hydrological year range is by no means necessary. A user is able to analyze a specific month for example using the date range filters. A user can also compare different months or different decades to analyze. Perhaps a user is interested in exploring the weather changes throughout the years, for this a long date range can be relevant. The options are unlimited, and therefore the analysis options are extremely diverse for different use cases. The time series plots and the dimensionality reduction scatter plots would be beneficial for this analysis type. In order to compare different time ranges, it is worth noting, that the tool offers a feature for saving current plot states as an image file on the local storage. In order to use this feature to its maximum potential, descriptive naming conventions are required.

5.4.4 Visualization Based Selection

Our last orientation is a visualization based selection. This approach is different from the others as the user doesn't need to select a specific date range. Using the clusters scatter plot, a user can filter specific points using the box selection tools. This selection filters the data in the displayed table. This approach can be used in order to analyze different clusters and come to a conclusion for why certain days are regarded similarly and have low variance in the input features. An example for an analysis that can be done using this approach is exploring, whether near points on the scatter plot resemble dates close to each other on the calendar.

5.5 Tool use cases

The tool offers a wide range of visualization techniques as well as a huge data set of weather attributes and other physical attributes relevant to rainfall runoff modeling. Different personas can utilize the model for different use cases to achieve their research goals. In the following list, some of these use cases are described.

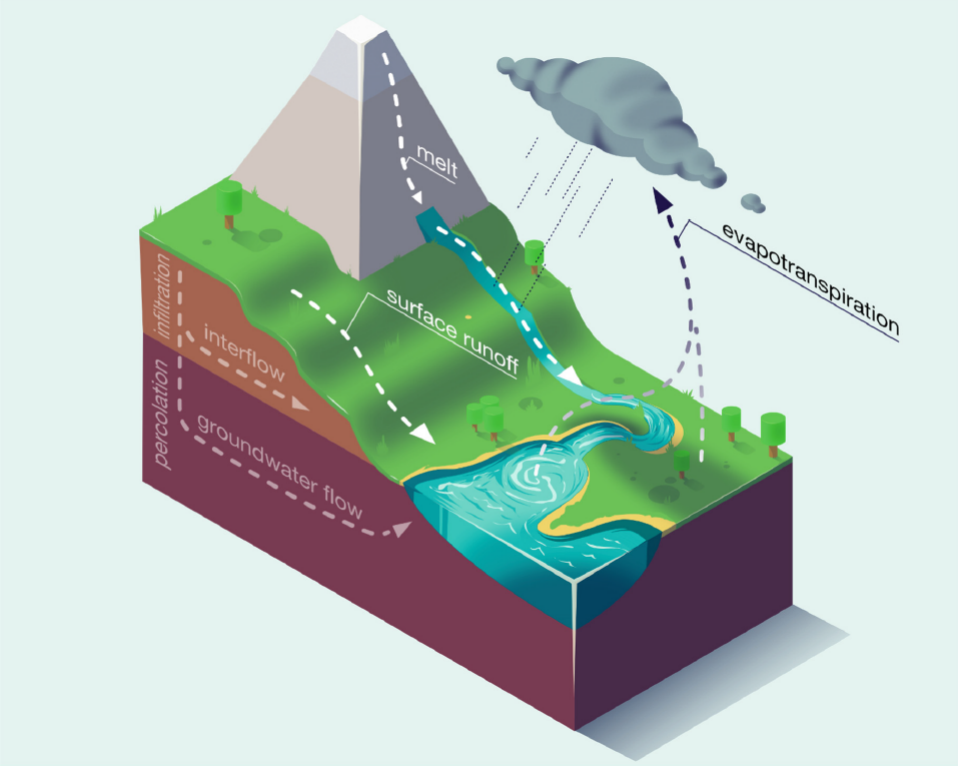

Figure 5.8: "Simplified visualization of processes and fluxes that influence the river discharge, such as precipitation, snow melt, surface runoff or subsurface flows." Reprinted from [KHK+19].

1. A water resources engineer can use the tool to study different weather attributes and their effects on discharge. Especially important for this use case would be the time series plots and the correlation heat maps. The tool can also be used for diagnosing the conceptual model.

2. A climate researcher can use the tool to study climate changes and global warming throughout the time. Having visualizations of relevant weather features for 60 years shows slight weather changes. In this use case plots such as the time series feature plots as well as the day sequence plots and tables.
3. A machine learning researcher can use the tool to generate insights about LSTMs and their application in this particular regression problem. The dimensionality reduction plots for the hidden states and their clustering, as well as the hidden state correlation heat maps can be very useful for detecting the model training as well as the hidden state changes after each time step.
4. A hydrologist can use the tool to understand the benefits and limitations of using an LSTM model for predicting discharge and for rainfall runoff modelling (figure: 5.8). Using the tool can generate insights in regards to comparing a conceptual model based on physical processes with a complete black-box model. For this use case every plot of the visualization can be useful.
5. A visualization researcher or a human computer interaction researcher can utilize the color mapping and visualization techniques to study different methods of visualizing the same data. For this purpose some changes of the code can be required.

6 Interpretation

After discussing different workflows and tool usage methods in the previous chapter, in this chapter we will present our observations we've collected using the tool, and our interpretations for the phenomena we've observed.

6.1 Soil Moisture effect on discharge

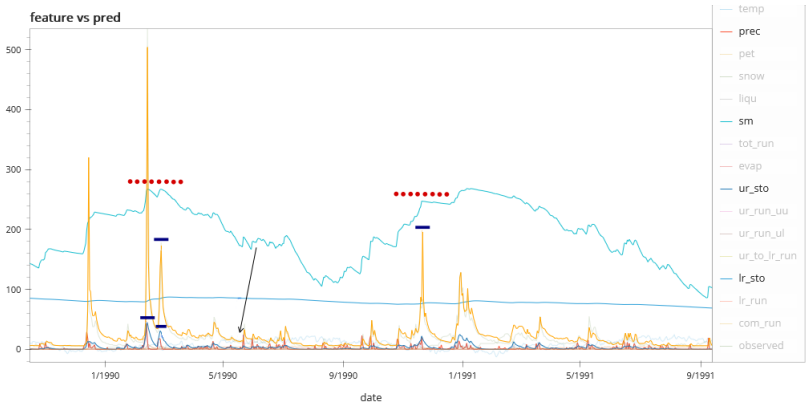

Figure 6.1: A filtered version of the time series plot showing the effect of soil moisture on high predictions. The x-axis represents the date, the y-axis represents the values for the inputs in their corresponding measurement unit.

For this example we used hydrological year 1990. The first step is to use the date range filters and inserting the time period from 19891101 to 19901031. All time series plots and most dimensionality reductions should be adjusted accordingly.

The first concept we want to understand is the conceptual model prediction and the overall behavior of the system. In figure 6.1 we can see the prediction of the model depicted in the color orange. We see a few peaks of the predicted values that are highlighted in dotted red lines and blue dashes. An observation we've had was the visually noticeable correlation between the upper reservoir storage and the discharge, every time the prediction reaches a peak, the upper reservoir storage does too, however this isn't the case vice versa. And the reason for that can be understood by observing the soil moisture line in cyan. Having a high upper reservoir storage as well as high soil moisture would mean that the soil is already saturated with water, resulting in high discharge. However, in the cases of high upper reservoir storage and low soil moisture, the soil ends up absorbing the water caused by the precipitation. The lower reservoir storage seems to not have an effect, as it's mostly constant throughout the duration. This effect of the soil moisture can not be detected right away from observing the correlation matrix heat maps in figure 5.2, as the correlation between soil moisture

and the prediction is high, however not high enough compared to different attributes. The reason for that is the delay effect, as the Pearson's correlation compares each day's value individually. In this case, the soil moisture's effect on the discharge can be observed days or even weeks after the rain (time it takes for the reservoir to fill up with water, depending on rain strength), hence lower Pearson's correlation values.

6.2 Detecting model error

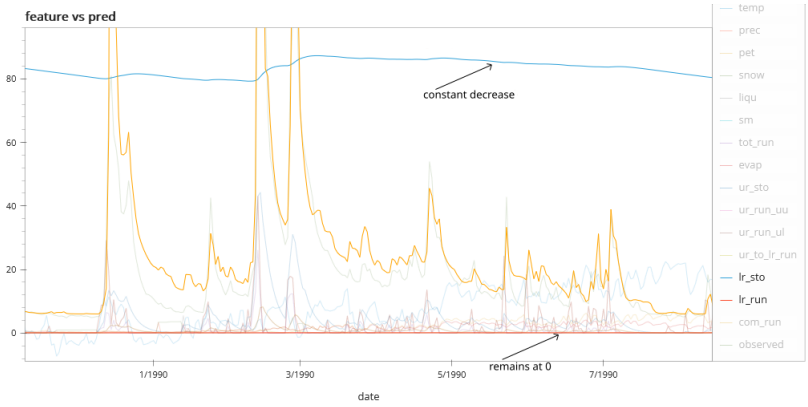

Figure 6.2: A filtered version of the time series plot showing the errors of the lower reservoir runoff. The x-axis represents the date, the y-axis represents the values for the inputs in their corresponding measurement unit.

As with anything the model is prone to errors. Those errors can be expected, for example with prediction errors or variance from the observed discharge, and also unexpected for example with visualization inaccuracy as well as data collection. In this example we will explore an unexpected error we were able to detect, while analyzing the data through the visualization tool. In our data set we can see the upper and lower reservoir storage, as well as the upper and lower reservoir runoff. While the water is being emptied from the reservoirs, the runoff accordingly should be increased, as those attributes directly correlate to each other. In figure 6.2 we can see that the lower reservoir storage in blue is increasing or decreasing based on precipitation, while the lower reservoir runoff in brown remaining at zero. This is obviously unrealistic and is due to an error. We suspect this being a cause of a mistake in the data collection, however we are unable to prove that with the visualization tool alone. It is important to detect such issues, and understand that in similar situations the prediction accuracy might suffer. Being able to detect those inaccuracies, and filtering those predictions from the model can improve the overall accuracy significantly, and further increase our trust in the model outputs, given correctly collected and prepared data. Detecting such errors can be automated in future work.

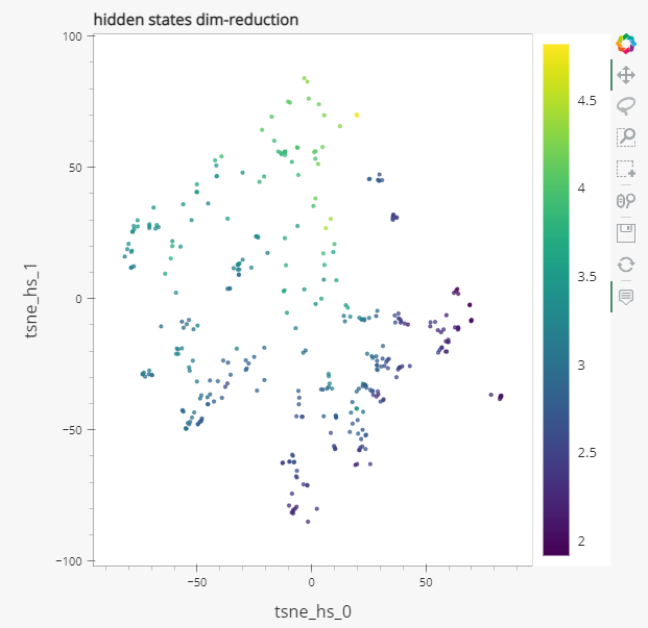

Figure 6.3: A filtered version of the hidden states t-SNE scatter plot for the hydrological year 01.11.1992 - 31.10.1993. The x-axis represents the first dimension of the t-SNE hidden states dimensionality reduction, the y-axis represents the second dimension.

6.3 Perception of discharge seasons

Previously, we discussed hydrological year in 5.4.1 and their significance in being an orientation for an analysis using the visualization tool. As for a regular year, a hydrological year has 4 seasons where discharge cycles from high to low back to high. This cyclical nature is an effect of weather conditions and precipitation levels, where rainy seasons cause higher discharge than seasons with less rain. This cyclical effect of dry and wet seasons can be seen in multiple plots. In figure 6.3 a filtered version of a hydrological year can be seen of the hidden states scatter plot of the t-SNE dimensionality reduction. Notice the difference in color of the points in the upper section of the plot in yellow and their gradual reduction in a counter clockwise circle, until they reach dark blue in the section on the right. This can be confirmed by using the box selection of the day sequence plot and table of the figure 5.7, where each month can be directly seen in the table. Filtering by hydrological year is a better confirmation, as the same pattern can be seen, even though a single year can carry independent noise inherent on itself. Looking back at the clustering plot in figure 5.6, we notice a clear definition of each cluster. The different clusters describe different groups of flow regimes, where they individual samples (days) are ranked by their stream flows.

The cyclical pattern can also be seen in the features scatter plot, but being able to identify this from the hidden states of the LSTM model is a more powerful statement, as it shows the ability of the LSTM model to understand such dependencies and adapt the hidden states accordingly. It is especially interesting because the color mapping of the plot is based on the predicted output of the model and not the observed values.

6.4 Correlations and information gain of the Conceptual model vs LSTM model

The final model we use for our visualization tool is a hybrid LSTM model consisting of both weather attributes as well as internal states of the conceptual model. An interesting experiment here is to be able to identify which features contribute the most to the final output and which features have the provide the most information. We calculated the joint entropy of the t-SNE dimensionality reductions for the hidden states as well as the input data. Joint entropy is a property which characterizes the unpredictability of a joint probability distribution, or a multi-valued random variable [Lea13]:

$$H(X,Y) = -\sum_{x \in X} \sum_{y \in Y} P(x,y) log_2 \left[P(x,y)\right]$$

Where X and Y are two random variables and $P(x,y)$ is the joint probability of x and y. The joint entropy for the hidden states using the bin counting method with 31 bins is 8.426 bits. For the entire input data set using both weather features as well as the internal states and the fluxes of the conceptual model the entropy using 31 bins is 8.534 bits. With such a small difference in entropy of 0.108, it was worth asking whether eliminating the weather attributes would reduce the entropy further, and whether those attributes are adding noise to the calculation. The next experiment was comparing the entropy of only the internal states of the conceptual model with the hidden states. The entropy using 31 bins was 8.445 bits for the former, which is still higher than the later, as it can very difficult for the hidden states of the model to understand all the minor underlying patterns within the data. One could say, that an exact entropy match would indicate two models with equal fits using different representations of the system. It is very interesting however to compare the entropy of the inputs before and after the removal of the weather attributes. Our explanation to these results is the drastic difference in information between the LSTM model and the conceptual model. This can be visually seen in the correlation heat map in figure 5.2, where the absolute correlation values of temp, prec, pet, snow, liqu and evap with the model output (prediction) are much lower than the rest of the conceptual model attributes, which have much darker colors than the rather dull colors of the weather attributes (Table: 6.1).

	joint entropy in bits
HBV inputs, states and fluxes	8.534
HBV states and fluxes	8.445
LSTM hidden states	8.426
HBV states	8.356

Table 6.1: Joint entropy results of the t-SNE dimensionality reduction of the 4 data sets.

While creating the t-SNE dimension reductions of the data sets, we were able to notice the difference of the visualizations in figures 6.4, 6.5 and 6.6. Our original hypothesis was that using the entire data set is causing unnecessary noise, and that the same system can be represented through less features (internal states and HBV states). The noise effect can be seen in the three figures, where

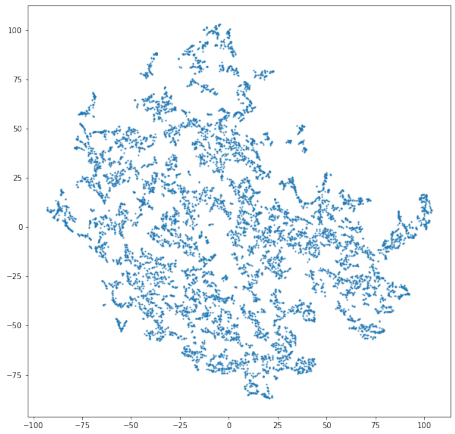

Figure 6.4: t-SNE of HBV inputs, states and fluxes.

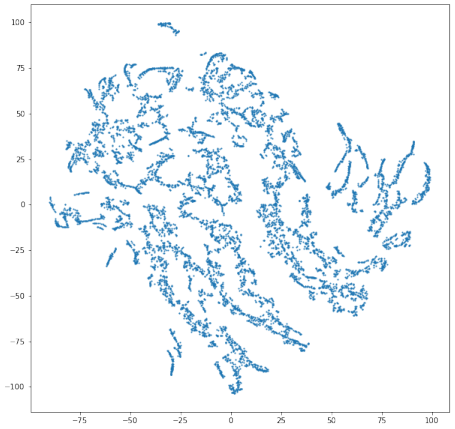

Figure 6.5: t-SNE of HBV states and fluxes.

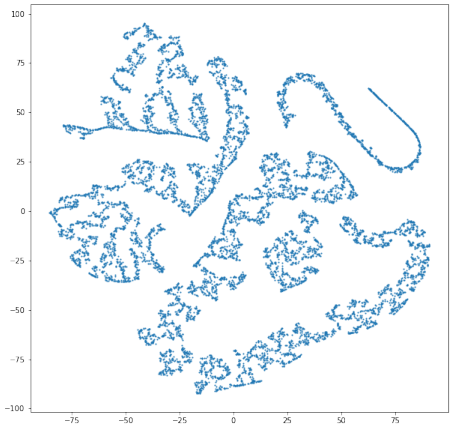

Figure 6.6: t-SNE of HBV states.

the noise is being reduced until it is almost completely eliminated in the HBV states plot. Having said that, with such small differences in the joint entropy between the three data sets, we are lead to believe that the HBV model states can be used for a complete representation of the system.

6.5 Hidden states as a representation of the system state

A further observation we were able to have is the influence of each individual hidden layer on the model output. Looking at the correlation heat map of the hidden states and distinguishing between the first layer cells (0-10) from the second layer (11-21), we were able to find higher absolute correlation values in the second layer. Given the nature of LSTM networks and the upside to adding additional hidden layers that are able to capture better insights of the data up to a certain point, this was not surprising. It was however a validation for us that the model is working as intended. We were able to validate the visual observations by calculating the sum of the absolute values of each layer separately. The sum of the first layer's correlation value absolutes was 4.76 and for the second

layer was 5.69. Adding a further layer could improve the model results, but having a simple overall architecture is very beneficial in this context, as it allows us to analyze and understand the complex structure better.

If we dig one step deeper, we can even observe the individual features which have higher absolute correlation values with the hidden states, namely the upper reservoir related features. The upper and lower reservoirs both model how the entire system works, they are however behaving differently in terms of the model. Our explanation for this can be interpreted from the physical world. The upper reservoir fills as a direct cause of precipitation, much faster than the lower reservoir, which is filled by the upper reservoir runoff. The response time of the upper reservoir being filled is quicker to the rain than the lower reservoir, which only gets emptied during the time step of the prediction, which then shows as lower correlation values. This study area is dominated by "fast" flow or a quick response time, which is confirmed by this observation.

6.6 PCA dimensionality reduction

We have opted for using PCA as an alternative dimensionality reduction technique. In order to be able to visualize the dimensionality reduced data, we used only the first two principal components in order to maintain a two-dimensional output. There is a limitation however with PCA, namely the variability that can be lost in other disregarded principle components, which provides a less coherent structure than t-SNE for example which maintains the variability in a two dimensional space. In this example we are showing an example of this phenomenon, where the output data points are split in two branches, as a cause of the variability loss (figure: 6.7). This is not unique to our data set, as we can see a very similar two-branch structure in the work of Strobelt et al. (2017) and Garcia et al (2017) [GMW21; SGPR17], where PCA was used and the first two principle components were visualized in a similar scatter plot. There is an observation to be made here however, the first observation is how the data points are arranged within our scatter plot. We can see the high flow days mainly centered around the bottom left corner, while the low flow days are in the top and right corners of the figure. It is also worth noticing, that there seems to be little to no anomalies within the point arrangement. Similar to t-SNE we can see the cycle of the hydrological year in how the points are ordered, from low- to high flow.

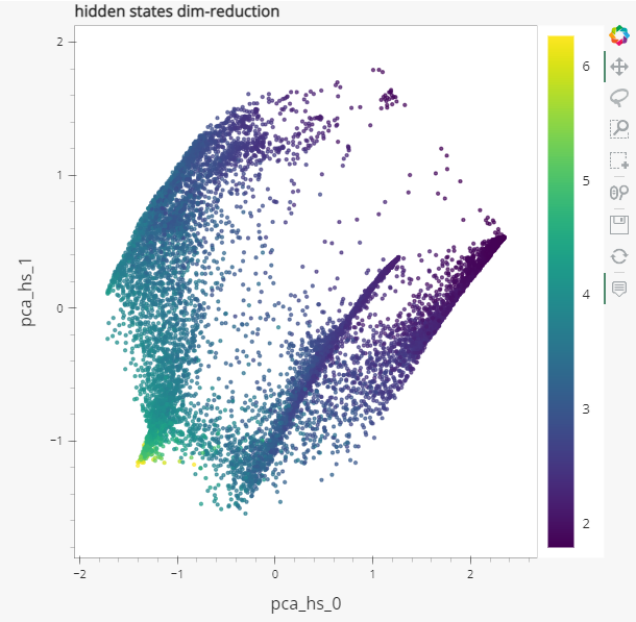

Figure 6.7: A PCA dimensionality reduction for the hidden states of the model. The x-axis represents the first principle component, the y-axis represents the second principle component. The color mapping represents the log-scale model prediction.

7 Discussion

The visualization tool we have created allowed us to explore different aspects of the LSTM model. The model creators' main priority was generating a well performing model. In order for us to visualize a rather complex model, we had to find methods to allow its visualization. If the model was developed specifically for interpretability, perhaps some performance compromises had to be done in order to simplify the model architecture which allows for an easier visualization. For future work having a simple model that is implemented and trained specifically for interpretation purposes can be a good first steps to find implications for a more complicated model. This can allow a more in depth interpretation. Nevertheless, we still were able to generate some valuable observations by using different simplification methods for a coherent and usable visualization tool without having to change the original model.

The visualization tool allowed us to detect model errors and pinpoint the exact causes for inaccurate predictions. We were able to detect inconsistencies with the lower reservoir runoff compared to its storage, which is possibly caused by an improper export of the data, or an error of the model itself. Being able to detect such errors opens up possibilities for future work. Creating an automated algorithm that is able to detect such inconsistencies and filter them from the model can be one of those ideas. Such an algorithm can further increase the trust in such data-driven model in the hydrology field.

A main concern of hydrological models is their interpretation. We were able however to visually detect correlations between the soil moisture, the reservoirs' storage, and their effect on discharge in the conceptual model. Having high discharge can be caused by high upper reservoir storage. This causality does not happen in both direction, since the soil moisture also needs to be considered. While the soil is not yet saturated after a rain event, the water in the upper reservoir fills the soil with water before the discharge is generated. This phenomenon can be observed by visualizing those three input features of the model. Another relation with the physical system we were able to observe was the perception of discharge seasons. Interpreting the data-driven LSTM model was more challenging but, through the hidden state dimensionality reduction, we were able to visually interpret the hydrological year and its seasonality. We are able to see the cyclical effect on dry and wet seasons on the hidden states and their effect on the model prediction, through the usage of color mappings. This can be interpreted using a t-SNE plot and a PCA plot. It is however worth mentioning that a t-SNE plot was providing a more coherent structure, as we only used the first two principle components, which leads to loss of variability.

For further exploration regarding the LSTM's relations with the conceptual model we used joint entropy as a metric for interpreting the information associated with each model. We concluded that using all the weather attributes and the internal states as well as the HBV model variables had the highest joint entropy. However the entropy did not reduce by a lot after disregarding the internal fluxes and the weather attributes. This shows us that the HBV model can be a representation for the entire system, and that adding weather attributes do marginally increase the variability than can

be explored by these features. This can also be seen visually by using dimensionality reduction on each input set, where using the HBV model variables significantly decrease the visual noise in the final plot. It is also worth noticing, that the joint entropy of the hidden states was more than for the HBV states. While both entropy values are measures of the variability of the system, the hidden states seem to offer more information as they have a lot more data encoded in them. We recommend using joint entropy and other information-theory based metrics for interpreting LSTM models, as these metrics benefit a lot from dimensionality reduction.

After referring to previous research and implementing our own tool, we concluded the importance of utilizing multiple visualization techniques to interpret different aspects of a neural network. In order to gain insight into the model functionality those visualizations need to be used interchangeably. To explore a research hypothesis using visualization techniques, a workflow is needed, where specific questions can be answered. Some visualizations are meant for general model explorations, while others are meant for a detailed view of the internal model structure. A workflow should include both visualization types. We suggested a workflow that can be used by different users attempting to use the tool for different objectives.

During the development of the tool, many options of different libraries and frameworks were explored. Most of which were aiming for increase of quality and robustness of the tool, but some of which also were selected in regards to development efficiency, given the limited duration in which the tool had to be done. In this sections we will discuss the limitations of the visualization tool. Those limitations are either caused by the physical world we live in and the multi dimensional space visualization challenge [Spe99], or framework and libraries related limitations. This section can be considered as a reference for future work or next steps.

The first limitation regarding the visualization tools Hvplot and Panel was the documentation and online resources. Both libraries are fairly well documented, however when used together, new issues and challenges were faced, that weren't found in official documentation sources. We would consider using different, more wide spread frameworks such as plotly and dash, which have the same functionality, except for the ability to create a web-based interface through Jupyter notebooks.

The second limitation was the amount of control we have over the visualizations. As Hvplot is built on top of Bokeh, another library for easily creating interactive and well defined plots, the general layouts are already pre-developed, giving them a visual aesthetic that is very similar across all plots. If the developer wants to have a wide choices for customization, Hvplot might not be the best option. A different method we would consider is using Javascript's library D3, which allows the developer to build the entire dashboard and its individual plots from scratch. This library was used in a similar visualization tool in [GMW21]. A disadvantage for using this approach is, the significantly increased amount of code required for recreating the same dashboard compared to python.

Generally a well performing LSTM requires a big data set. The model we based our visualization tool on is not different. Not only are we working with 60 years worth of data, we also were required to prepare and adjust the data in order to create meaningful plots. One of the plots in the visualization dashboard (figure: 5.7) is a transformation of the data into a 365 day sequence for every day in the data, which is in total more than 3 million samples. The transformation includes generating t-SNE, PCA and SPI values, which can be computationally demanding. In the best case scenario, the machine used for launching the tool would be able to run the testing, extract the hidden state and prepare the data while the tool is launching. This is however unrealistic, and can take

hours before the tool could finally launch. This limitation was solved by generating all the required files prior to launching the tool, and the launching time is on average 4 minutes. This limitation isn't very problematic, but it is worth noting, that the data preparations need to happen separately for every model that is to be used by the visualization plot.

8 Conclusion

The main objective of the thesis was to create a visual representation of an LSTM model that allows us to interpret and understand how the model learns from the inputs and predicts accurate outputs. We used an LSTM model trained for predicting discharge and for rainfall-runoff modelling. For this objective we created an interactive visualization tool that enables us to analyze different aspects of the model. We used a combination of visualization techniques in order to interpret how the hidden states are changing throughout time, how the input features effect the output, and how the hidden states correlate with the input features. In order to create understandable visualizations from multi-dimensional data, we implemented dimensionality reduction methods. The tool was used and evaluated by field experts, and the insights they provided were used for improvements. Furthermore we used the final tool in order to generate insights into the model and its implications in hydrology. A topic we focused on was the interpretibility of data-driven models for rainfall-runoff modelling and its discrepancy of physical relations. We also suggested a workflow for different users that use the tool in order to answer different questions related to this domain.

Using visualization techniques for interpreting neural networks for rainfall-runoff modelling deemed beneficial. Not only did it allow us to explain some of the behaviors of the model, but it also allowed us to find relations with the real physical world.

Bibliography

[ABK+20] A. Abbas, S. Baek, M. Kim, M. Ligaray, O. Ribolzi, N. Silvera, J.-H. Min, L. Boithias, K. H. Cho. "Surface and sub-surface flow estimation at high temporal resolution using deep neural networks". In: *Journal of Hydrology* 590 (2020), p. 125370 (cit. on p. 15).

[App00] A. T. C. on Application of Artificial Neural Networks in Hydrology. "Artificial neural networks in hydrology. I: Preliminary concepts". In: *Journal of Hydrologic Engineering* 5.2 (2000), pp. 115–123 (cit. on p. 26).

[APS+21] E. M. H. Abdalla, V. Pons, V. Stovin, S. De-Ville, E. Fassman-Beck, K. Alfredsen, T. M. Muthanna. "Evaluating different machine learning methods to simulate runoff from extensive green roofs". In: *Hydrology and Earth System Sciences* 25.11 (2021), pp. 5917–5935 (cit. on p. 15).

[AW10] H. Abdi, L. J. Williams. "Principal component analysis". In: *Wiley interdisciplinary reviews: computational statistics* 2.4 (2010), pp. 433–459 (cit. on p. 37).

[Ayz19] G. Ayzel. "Does deep learning advance hourly runoff predictions". In: *Proceedings of the V international conference information technologies and high-performance computing (ITHPC-2019), Khabarovsk, Russia*. 2019, pp. 16–19 (cit. on p. 15).

[BDWD04] W. Buytaert, B. De Bièvre, G. Wyseure, J. Deckers. "The use of the linear reservoir concept to quantify the impact of changes in land use on the hydrology of catchments in the Andes". In: *Hydrology and Earth System Sciences* 8.1 (2004), pp. 108–114 (cit. on p. 25).

[BN06] C. M. Bishop, N. M. Nasrabadi. *Pattern recognition and machine learning*. Vol. 4. 4. Springer, 2006 (cit. on pp. 17, 18).

[BPC12] K. Bogner, F. Pappenberger, H. Cloke. "The normal quantile transformation and its application in a flood forecasting system". In: *Hydrology and Earth System Sciences* 16.4 (2012), pp. 1085–1094 (cit. on p. 31).

[BS14] S. Bender, M. Schaller. *Vergleichendes Lexikon: wichtige Definitionen, Schwellenwerte und Indices aus den Bereichen Klima, Klimafolgenforschung und Naturgefahren*. CSC Climate Service Center, 2014 (cit. on p. 47).

[CA16] M. E. Celebi, K. Aydin. *Unsupervised learning algorithms*. Springer, 2016 (cit. on p. 17).

[CCD08] P. Cunningham, M. Cord, S. J. Delany. "Supervised learning". In: *Machine learning techniques for multimedia*. Springer, 2008, pp. 21–49 (cit. on p. 17).

[Cha22] K. Chabok. "Using Machine Learning Tools to Improve a Conceptual Hydrological Model of the Upper Neckar Catchment". MA thesis. Stuttgart: University of Stuttgart, Mar. 2022 (cit. on pp. 15, 19–21, 29, 31, 32, 34, 35).

[DSW+87] J. Denker, D. Schwartz, B. Wittner, S. Solla, R. Howard, L. Jackel, J. Hopfield. “Large automatic learning, rule extraction, and generalization”. In: *Complex systems* 1.5 (1987), pp. 877–922 (cit. on p. 15).

[FV19] R. Fong, A. Vedaldi. “Explanations for attributing deep neural network predictions”. In: *Explainable ai: Interpreting, explaining and visualizing deep learning*. Springer, 2019, pp. 149–167 (cit. on p. 28).

[GBC16] I. Goodfellow, Y. Bengio, A. Courville. *Deep learning*. MIT press, 2016 (cit. on pp. 19, 20, 23).

[GJM13] A. Graves, N. Jaitly, A.-r. Mohamed. “Hybrid speech recognition with deep bidirectional LSTM”. In: *2013 IEEE workshop on automatic speech recognition and understanding*. IEEE. 2013, pp. 273–278 (cit. on p. 22).

[GMW21] R. Garcia, T. Munz, D. Weiskopf. “Visual analytics tool for the interpretation of hidden states in recurrent neural networks”. In: *Visual Computing for Industry, Biomedicine, and Art* 4.1 (2021), pp. 1–13 (cit. on pp. 15, 28, 56, 60).

[GSC00] F. A. Gers, J. Schmidhuber, F. Cummins. “Learning to forget: Continual prediction with LSTM”. In: *Neural computation* 12.10 (2000), pp. 2451–2471 (cit. on p. 33).

[GSH15] A. Gisbrecht, A. Schulz, B. Hammer. “Parametric nonlinear dimensionality reduction using kernel t-SNE”. In: *Neurocomputing* 147 (2015), pp. 71–82 (cit. on p. 37).

[Har15] A. W. Harley. “An interactive node-link visualization of convolutional neural networks”. In: *International Symposium on Visual Computing*. Springer. 2015, pp. 867–877 (cit. on p. 15).

[Hea08] J. Heaton. *Introduction to neural networks with Java*. Heaton Research, Inc., 2008 (cit. on p. 31).

[HKPC18] F. Hohman, M. Kahng, R. Pienta, D. H. Chau. “Visual analytics in deep learning: An interrogative survey for the next frontiers”. In: *IEEE transactions on visualization and computer graphics* 25.8 (2018), pp. 2674–2693 (cit. on p. 27).

[HS97] S. Hochreiter, J. Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780 (cit. on p. 22).

[HTFF09] T. Hastie, R. Tibshirani, J. H. Friedman, J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer, 2009 (cit. on pp. 18, 21).

[HXY15] Z. Huang, W. Xu, K. Yu. “Bidirectional LSTM-CRF models for sequence tagging”. In: *arXiv preprint arXiv:1508.01991* (2015) (cit. on p. 22).

[JM15] M. I. Jordan, T. M. Mitchell. “Machine learning: Trends, perspectives, and prospects”. In: *Science* 349.6245 (2015), pp. 255–260 (cit. on p. 17).

[KAKC17] M. Kahng, P. Y. Andrews, A. Kalro, D. H. Chau. “A cti v is: Visual exploration of industry-scale deep neural network models”. In: *IEEE transactions on visualization and computer graphics* 24.1 (2017), pp. 88–97 (cit. on p. 28).

[KGR08] S. Khan, H. Gabriel, T. Rana. “Standard precipitation index to track drought and assess impact of rainfall on watertables in irrigation areas”. In: *Irrigation and Drainage Systems* 22.2 (2008), pp. 159–177 (cit. on pp. 38, 45).

[KHK+19] F. Kratzert, M. Herrnegger, D. Klotz, S. Hochreiter, G. Klambauer. "NeuralHydrology–interpreting LSTMs in hydrology". In: *Explainable AI: Interpreting, explaining and visualizing deep learning*. Springer, 2019, pp. 347–362 (cit. on pp. 28, 48).

[KM13] T. M. Kodinariya, P. R. Makwana. "Review on determining number of Cluster in K-Means Clustering". In: *International Journal* 1.6 (2013), pp. 90–95 (cit. on p. 44).

[KRA+14] K. Khan, S. U. Rehman, K. Aziz, S. Fong, S. Sarasvady. "DBSCAN: Past, present and future". In: *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*. IEEE. 2014, pp. 232–238 (cit. on p. 44).

[LBE15] Z. C. Lipton, J. Berkowitz, C. Elkan. "A critical review of recurrent neural networks for sequence learning". In: *arXiv preprint arXiv:1506.00019* (2015) (cit. on p. 22).

[Lea13] E. G. Learned-Miller. "Entropy and mutual information". In: *Department of Computer Science, University of Massachusetts, Amherst* (2013), p. 4 (cit. on p. 54).

[LLJT14] M. Li, F. Liu, M. Juusola, S. Tang. "Perceptual color map in macaque visual area V4". In: *Journal of Neuroscience* 34.1 (2014), pp. 202–217 (cit. on p. 43).

[LS20] H. Lange, S. Sippel. "Machine learning applications in hydrology". In: *Forest-water interactions*. Springer, 2020, pp. 233–257 (cit. on p. 26).

[LT02] Z. Liu, E. Todini. "Towards a comprehensive physically-based rainfall-runoff model". In: *Hydrology and Earth System Sciences* 6.5 (2002), pp. 859–881 (cit. on p. 25).

[LWB+19] S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek, K.-R. Müller. "Unmasking Clever Hans predictors and assessing what machines really learn". In: *Nature communications* 10.1 (2019), pp. 1–8 (cit. on p. 15).

[LZHY17] Q. Liu, F. Zhou, R. Hang, X. Yuan. "Bidirectional-convolutional LSTM based spectral-spatial feature learning for hyperspectral image classification". In: *Remote Sensing* 9.12 (2017), p. 1330 (cit. on p. 22).

[MKC06] R. H. McCuen, Z. Knight, A. G. Cutter. "Evaluation of the Nash–Sutcliffe efficiency index". In: *Journal of hydrologic engineering* 11.6 (2006), pp. 597–602 (cit. on p. 33).

[MKH+95] N. J. Morch, U. Kjems, L. K. Hansen, C. Svarer, I. Law, B. Lautrup, S. Strother, K. Rehm. "Visualization of neural networks using saliency maps". In: *Proceedings of ICNN'95-International Conference on Neural Networks*. Vol. 4. IEEE. 1995, pp. 2085–2090 (cit. on p. 15).

[OCo97] K. O'Connor. *Applied hydrology I-deterministic*. Unpublished Lecture Notes. Department of Engineering Hydrology, National University of Ireland, Galway. 1997 (cit. on p. 25).

[PGM+19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32 (2019) (cit. on p. 31).

[Phi18] M. Phi. *Illustrated Guide to LSTM's and GRU's: A step by step explanation.* 2018. URL: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21 (cit. on p. 22).

[Ros61] F. Rosenblatt. *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. Tech. rep. Cornell Aeronautical Lab Inc Buffalo NY, 1961 (cit. on p. 15).

[Rud16] S. Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016) (cit. on p. 21).

[SGPR17] H. Strobelt, S. Gehrmann, H. Pfister, A. M. Rush. "Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks". In: *IEEE transactions on visualization and computer graphics* 24.1 (2017), pp. 667–676 (cit. on pp. 27, 56).

[SHC+08] S. Sorooshian, K.-l. Hsu, E. Coppola, B. Tomassetti, M. Verdecchia, G. Visconti. *Hydrological modelling and the water cycle: coupling the atmospheric and hydrological models*. Vol. 63. Springer Science & Business Media, 2008 (cit. on p. 25).

[SKRS18] M. Syakur, B. Khotimah, E. Rochman, B. D. Satoto. "Integration k-means clustering method and elbow method for identification of the best customer profile cluster". In: *IOP conference series: materials science and engineering*. Vol. 336. 1. IOP Publishing. 2018, p. 012017 (cit. on p. 44).

[SMV+19] W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, K.-R. Müller. *Explainable AI: interpreting, explaining and visualizing deep learning*. Vol. 11700. Springer Nature, 2019 (cit. on p. 15).

[Spe99] W. M. Spears. "An overview of multidimensional visualization techniques". In: *Evolutionary Computation Visualization Workshop*. Citeseer. 1999 (cit. on p. 60).

[Van14] L. Van Der Maaten. "Accelerating t-SNE using tree-based algorithms". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 3221–3245 (cit. on p. 24).

[VPV+09] L. Van Der Maaten, E. Postma, J. Van den Herik, et al. "Dimensionality reduction: a comparative". In: *J Mach Learn Res* 10.66-71 (2009), p. 13 (cit. on pp. 23, 24).

[WF19] S. Wenke, J. Fleming. "Contextual Recurrent Neural Networks". In: *arXiv preprint arXiv:1902.03455* (2019) (cit. on p. 32).

[YS18] R. Yu, L. Shi. "A user-based taxonomy for deep learning visualization". In: *Visual Informatics* 2.3 (2018), pp. 147–154 (cit. on p. 27).

All links were last followed on September 21, 2022.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature