Approximation and Optimization of Compute-Intensive Environmental Simulations through Machine Learning Methods

Zur Erlangung des akademischen Grades einer

Doktorin der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Elnaz Azmi
aus Tabriz, Iran

Tag der mündlichen Prüfung:	8. Februar 2024
Erster Gutachter:	Prof. Dr. Achim Streit
Zweiter Gutachter:	Prof. Dr. Erwin Zehe

Erklärung zur selbständigen Anfertigung der Dissertationsschrift

Hiermit erkläre ich, dass ich die Dissertationsschrift mit dem Titel

Approximation and Optimization of Compute-Intensive Environmental Simulations through Machine Learning Methods

selbständig angefertigt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht und die Regeln zur Sicherung guter wissenschaftlicher Praxis am Karlsruher Institut für Technologie (KIT) beachtet habe.

Ort, Datum Elnaz Azmi

To my family.

Abstract

Despite technological advancements in computing, executing simulations for large and intricate systems remains time and energy-intensive. Computational optimization approaches are widely applied across diverse fields to achieve enhanced solutions for various problems. These optimizations strive to minimize adverse effects, reduce costs and computation time, and improve efficiency and reliability. In environmental sciences, simulations play a crucial role in understanding complex natural systems, often demanding substantial computing resources due to their high spatio-temporal resolution. This thesis explores the challenges of compute-intensive environmental simulations and proposes approximation and optimization approaches aimed at augmenting their efficiency. The aim is to develop a computationally efficient approximation method using machine learning that yields useful outputs for domain scientists.

To achieve this goal, my proposed approach integrates unsupervised machine learning into simulations to identify similarities in their properties, reducing model redundancies and computation complexities. Specifically, this approach involves applying clustering methods to group functionally similar model units. The simulation is then executed only on a small yet representative subset of each cluster, scaling their outputs to the remaining cluster members. The study focuses on balancing the uncertainty of the simulation output resulting from simulating representative model units and output scaling with the computational effort involved. This research introduces an evolutionary approach that dynamically clusters and selects the representatives based on the dynamics enforced to the model units during the simulation. Results of application of this approach to a hydrological simulation indicate notable speed-up with low deviations from original simulations. Additionally, the criteria for applying this approach to various environmental simulations are discussed.

For simulations unsuited to the integrated evolutionary approach in original simulations using unsupervised machine learning, supervised machine learning is explored, introducing a neural network-based surrogate model to replace compute-intensive simulations with forecasts generated by a trained model. Supervised machine learning methods and specially deep learning can provide an accelerated approximation of simulations while requiring fewer computing resources compared to the original one. The training and test datasets are generated by execution of an atmospheric chemistry simulation. Applying the trained model to forecast a test dataset results in a good fit of the forecast values to the target dataset.

An extensive evaluation shows that both unsupervised and supervised approaches can approximate the environmental simulations with acceptable output for domain scientists. More importantly, the evaluation reveals that the approximation reduces the computational complexity of simulations, thereby reducing their computing resources.

This thesis introduces a universal evaluation method based on information theory concepts, applicable across domains. This work pursues two distinct yet interrelated goals. Firstly, it proposes a practical method for measuring computational complexity using a general system call tracer, which counts the total number of memory visits during the execution of a model. Secondly, it introduces an evaluation approach that combines measuring computational complexity with assessing model performance through information loss relative to observations, both unified within a single unit of bits.

Zusammenfassung

Trotz des technologischen Fortschritts in der Datenverarbeitung ist die Durchführung von Simulationen für große und komplexe Systeme nach wie vor zeit- und energieaufwändig. Computergestützte Ansätze zur Optimierung von Berechnungen werden in vielen verschiedenen Bereichen eingesetzt, um bessere Lösungen für verschiedene Probleme zu finden. Diese Optimierungen zielen darauf ab, unerwünschte Effekte zu minimieren, Kosten und Rechenzeit zu reduzieren und die Effizienz und Zuverlässigkeit zu verbessern. In den Umweltwissenschaften spielen Simulationen eine entscheidende Rolle für das Verständnis komplexer natürlicher Systeme und erfordern aufgrund ihrer hohen räumlichen und zeitlichen Auflösung oft erhebliche Rechenressourcen. Diese Arbeit untersucht die Herausforderungen rechenintensiver Umweltsimulationen und schlägt Näherungs- und Optimierungsansätze vor, um deren Effizienz zu verbessern. Das Ziel ist es, unter Verwendung von maschinellem Lernen, eine rechnerisch effiziente Annäherungsmethode zu entwickeln, die verwertbare Ergebnisse für Fachwissenschaftler liefert.

Um dieses Ziel zu erreichen, wird in dem von mir vorgeschlagenen Ansatz unüberwachtes maschinelles Lernen in Simulationen integriert, um Ähnlichkeiten in ihren Eigenschaften zu identifizieren und so Modellredundanzen und die Berechnungskomplexität zu reduzieren. Konkret beinhaltet dieser Ansatz die Anwendung von Clustering-Methoden, um funktional ähnliche Modelleinheiten zu gruppieren. Die Simulation wird dann nur auf einer kleinen, aber repräsentativen Teilmenge jedes Clusters ausgeführt, wobei die Ergebnisse auf die übrigen Mitglieder des Clusters skaliert werden. Die Studie konzentriert sich darauf, die Unsicherheit der Simulationsergebnisse, die sich aus der Simulation repräsentativer Modelleinheiten und der Skalierung der Ergebnisse ergibt, mit dem damit verbundenen Rechenaufwand abzuwägen. Diese Forschungsarbeit führt einen evolutionären Ansatz ein, der dynamisch Cluster bildet und die Repräsentanten auf der Grundlage der Einflüsse, die während der Simulation auf die Modelleinheiten wirken, auswählt. Die Ergebnisse der Anwendung dieses Ansatzes auf eine hydrologische Simulation zeigen eine merkliche Beschleunigung bei einer geringen Abweichung von der ursprünglichen Simulation. Außerdem werden die Kriterien für die Anwendung dieses Ansatzes auf andere Umweltsimulationen diskutiert.

Für Simulationen, die für den integrierten evolutionären Ansatz mit unüberwachtem maschinellem Lernen ungeeignet sind, wird die Verwendung von überwachtem maschinellen Lernen untersucht. Dabei wird ein auf einem neuronalen Netzwerk basierendes Ersatzmodell eingeführt, um rechenintensive Simulationen durch Vorhersagen zu ersetzen, die von einem trainierten Modell erzeugt werden. Methoden des überwachten maschinellen Lernens, und insbesondere des Deep Learnings, können eine beschleunigte Annäherung an die Simulationen bieten und benötigen dabei weniger Rechenressourcen als die ursprüngliche Simulation.

Die Trainings- und Testdatensätze werden durch das Ausführen einer Simulation der Atmosphärenchemie erzeugt. Die Anwendung des trainierten Modells zur Vorhersage eines Testdatensatzes resultiert in einer guten Annäherung der Vorhersagewerte an den Zieldatensatz.

Eine umfassende Bewertung zeigt, dass sowohl unüberwachte als auch überwachte Ansätze die Umweltsimulationen mit akzeptablen Ergebnissen für Fachwissenschaftler approximieren können. Noch wichtiger ist, dass die Auswertung zeigt, dass die Annäherung die Rechenkomplexität der Simulationen reduziert und damit die Rechenressourcen schont.

In dieser Arbeit wird eine universelle Bewertungsmethode vorgestellt, die auf Konzepten der Informationstheorie basiert und Domänenübergreifend anwendbar ist. Diese Arbeit verfolgt zwei unterschiedliche, aber miteinander verknüpfte Ziele. Erstens wird eine praktikable Methode zur Messung der Rechenkomplexität mithilfe einer gängigen Software zur Überwachung von Systemaufrufen vorgeschlagen, welche die Gesamtzahl der Speicherzugriffe während der Ausführung eines Modells zählt. Zweitens wird ein Bewertungsansatz vorgestellt, der die Messung der Rechenkomplexität mit der Bewertung der Leistung des Modells durch den Informationsverlust im Vergleich zu den Beobachtungen kombiniert und in der Einheit bits vereinigt.

Acknowledgments

This thesis owes its completion to the invaluable help of several individuals, to whom I would like to give my sincere thanks and appreciation.

Firstly, I would like to express my heartfelt gratitude to Prof. Achim Streit for granting me the opportunity to pursue my doctoral degree under his guidance, for his support and valuable feedback during this journey, and especially for providing an environment where I could explore every aspect of research. Additionally, I would like to sincerely thank Prof. Erwin Zehe for agreeing to be my co-supervisor, engaging in valuable discussions about challenges in hydrological simulations to guide my research focus, and providing me with research materials and datasets that greatly enhanced my work.

I am particularly thankful to Jörg Meyer for all the time, energy, and patience he devoted to supervising my work. He consistently provided me with the right perspective and an open ear from the outset that I came to SCC until the final submission of this thesis. Besides aiding me in scientific discussions, his support has been invaluable in discussions spanning various topics on multiple levels. In addition to reviewing my work, I appreciate his continuous support, constructive discussions and his ability to look at the big picture, which helped me to discover and shape the topic of this thesis.

An exceptional thank goes to Marcus Strobl, whose contribution to successfully finishing this work cannot be overstated. From the start of my work in the V-FOR-WaTer project, countless bike rides to work, German speaking and writing corrections, coaching me at bouldering sessions to being a friendly office mate and tango partner, he made the doctorate pursuit less lonely. I am especially grateful for his invaluable support and patience in reading and correcting my papers, proofreading this thesis, editing plots and references, and moral support during the thesis writing, which helped me stay sane and successfully reach the finish line.

I would like to express my gratitude to Uwe Ehret for his invaluable help in finding an interesting topic, engaging in profound discussions about hydrological simulations, providing neat and organized board writings and instructions, and for his valuable co-authorship on my papers. I also want to thank him for introducing me to Rik van Pruijssen, with whom I collaborated to optimize the CAOS simulation. Additionally, my thanks go to Rik van Pruijssen for his patience during our long discussions about the simulation.

I owe many thanks to Uğur Çayoğlu for constructive discussions at each level of my work, supporting me in tackling Python and LaTeXproblems, proofreading this thesis, and for his sense of humor and moral support throughout the thesis writing process.

I would like to send my great thanks to Eileen Kühn for constructive discussions and meetings, proofreading this thesis, and sharing our common interest in plants, to Markus Götz for the invaluable neural networks sessions and constructive discussions on their application in my work, and to Isabella Bierenbaum for patiently proofreading the entire thesis.

I would like to specially thank Michael Weimer for his support in integrating my approach into the ICON-ART simulation, engaging in extensive discussions to understand the simulation, and co-authoring the related paper. Furthermore, I am grateful to Stefan Versick, for providing the benchmark code of the ICON-ART simulation and his support in its compilation, to Ole Kirner for interpreting the forecasting results, to Roland Ruhnke that made the ICON-ART model and its related data files available, to Björn-Martin Sinnhuber for insightful discussions about the utilization of neural networks to accelerate the computation of chemicals, and to connecting me to Helmholtz AI consultants for Earth and environment from German climate computing center (DKRZ), to whom I would also like to thank for discussions about AI in climate simulations.

I am specially thankful to Maren Hattebuhr for being a patient and enthusiastic swimming couch for me, as well as our joyful time we shared during and outside of work. I would also like to thank our swimming team: Marcus Strobl, Michael Weimer, Marco Berghoff, Bastian Härer, Haykuhi Musheghyan, Melvin Strobl, Christof Wendenius, Isabella Bierenbaum and Diana Gudu. Additionally, I extend my thanks to our bouldering team: Marcus Strobl, Melvin Strobl, Marco Berghoff, Mehmet Soysal and Maren Hattebuhr. A special thank goes to Mehmet Soysal for biking together on the way to work, socializing and helping me with HPC jobs management.

Furthermore, I would like to thank my current and former colleagues, friends and doctoral researchers for organizing helps, pleasant project meetings, delicious cakes, countless enjoyments, lunch and coffee break discussions, all of which created a warm and inspiring atmosphere. Also thank you: Parinaz Ameri, Sibylle Haßler, Mirko Mälike, Diana Gudu, Peter Krauß, Benjamin Ertl, Max Fischer, Manuel Giffels, René Caspart, Matthias Schnepf, Valentin Kozlov, Daniel Coquelin, Oscar Taubert, Anis Farshian Abbasi, Arvid Weyrauch, Juan Pedro Gutiérrez Hermosillo Muriedas, Samuel Braun, Holger Obermaier, Lisana Berberi, Khadijeh Alibabaei, Gholamali Hoshyaripour, Jos van Wesel, Uros Stevanovic, Anja Müller and everyone else I forgot to mention. I have had a wonderful time with you, from social evenings to sports activities.

A very Special thanks to my parents and my family, Raheleh, Rasoul and Ramin, for their unwavering love and support on this extensive journey, from the moment I chose to pursue my scientific passion in Germany. I would not have reached this point without their support. Sağolun!

Elnaz Azmi
Karlsruhe, December 2023

Contents

List of Figures

List of Tables

List of Abbreviations

AIT	Algorithmic Information Theory.	24
ANN	Artificial Neuronal Network.	58
ART	Aerosols and Reactive Trace gases.	18
AS	Active Storage.	70
CAOS	Catchments as Organised Systems.	14
CMIP	Coupled Model Intercomparison Project.	42
CNNs	Convolution Neural Networks.	32
cos SZA	cosine of the solar zenith angle.	20
D	difference.	21
DWD	Deutscher Wetterdienst, en. German Weather Service.	18
ECMWF	European Centre for Medium-Range Weather Forecasts.	42
EFU	Elementary Functional Unit.	14
EMAC	ECHAM/MESSy Atmospheric Chemistry.	42
Eps	radius.	28
ERA	ECMWF Re-Analysis.	42
FCM	fuzzy C-mean.	45
FNNs	Feedforward Neural Networks.	31
GCM	General Circulation Model.	41
HPC	high-performance computing.	2
HSL	Hillslope.	14
ICON	ICOsahedral Nonhydrostatic modeling framework.	18
ICONET	ICON Neural Network-based approach.	98
K	number of clusters.	72
K-AEL	Auto-K tests using K-determiner with the elbow method.	88
K-ARC	Auto-K tests using K-determiner with the rmse-ctime method.	88
KGE	Kling-Gupta Efficiency.	22
LSTM	Long Short-Term Memory.	8
MinPts	minimum number of points.	28
MLPs	multilayer perceptrons.	31
MOR	Model Order Reduction.	34
MPI-M	Max Planck Institute for Meteorology.	18
MSE	Mean Squared Error.	22
NSE	Nash-Sutcliffe Efficiency.	39
NWP	numerical weather prediction.	43
PBIAS	percent bias.	40
PCA	Principal Component Analysis.	29

PCC	Pearson Correlation Coefficient.	23
POD	Proper Orthogonal Decomposition.	35
RE	relative error.	22
Resnet	residual convolutional neural network.	42
RIV	River element.	14
RMSE	Root Mean Squared Error.	22
rmse-ctime	RMSE-Computation-Time.	72
RNNs	Recurrent Neural Networks.	32
RSR	RMSE-Standard deviation Ratio.	40
SHM	Simple Hydrological Model.	17
SSE	Sum of Squared Errors.	27
SVMs	Support Vector Machines.	30
t-SNE	t-Distributed Stochastic Neighbor Embedding.	29
TE	Time to Equilibrium.	70
VC	Vapnik-Chervonenkis.	38
VMR	volume mixing ratio.	20
WF	with forcing.	87
WOF	without forcing.	87

The important thing in science is not so much to obtain new facts as to discover new ways of thinking about them.

Sir William Bragg

1 Introduction

Computational optimization approaches are used in many research fields, such as engineering disciplines, economics, energy systems, environmental science, and data science, to achieve the optimal solution for a given problem (Alonso et al., 2020, Herzel et al., 2021). The ultimate objectives of these optimizations are minimizing undesirable effects, reducing costs and computation time, providing useful solutions with enhanced efficiency and reliability, and increasing profits corresponding to various criteria or constraints (Merrill et al., 2008). Computational optimization methods can be applied to the solutions for handling various issues such as large-dimensional problems, search and big data handling problems, feature extraction and computationally expensive simulations. Computer-based modeling and simulations are widely used techniques in scientific research to analyze and understand real-world systems, as well as to design and develop high-performing products (Yin & McKay, 2018). However, the execution of simulations for large and complex systems consumes a significant amount of time and energy. Considering the perspective of energy saving, and despite the availability of modern and powerful computing technologies, there is a need "to address issues such as the complexity and scale of the systems that need to be modeled today" (Fujimoto et al., 2017).

Environmental science is a multidisciplinary domain that focuses on the study of the natural world and the complex interactions between humans and the environment. Environmental scientists engage in researching and evaluating environmental conditions, investigating the origins and consequences of environmental challenges such as pollution, climate change, and habitat loss, and developing policies to alleviate or resolve these issues. Environmental simulations play a crucial role in this context. However, conducting these simulations can be compute-intensive, depending on factors such as simulation complexity, required level of detail, and the scale of the simulated area or system. Contributing to the V-FOR-WaTer[1] project, I was involved in working with spatio-temporal data and the related simulations in environmental science. The objective of this project is to create a virtual research environment specifically designed for storing and analyzing hydrological data (Azmi et al., 2017, Meyer et al., 2019, Strobl et al., 2021, 2023). Engaging in this project led me to delve into environmental simulations and gain insights into the current challenges in this domain.

Environmental simulations in high spatio-temporal resolution, which consist of large-scale dynamic systems, are computationally intensive. At first glance, parallelization, and the usage of high-performance computing (HPC) resources could tackle this issue. The growth of HPC resources over the last few decades has made it possible to increase the resolution of environmental simulation models. This allows for the direct resolving of more and more processes, rather than relying solely on their effects through parametrization (Weimer et al., 2021). However, there are still several issues associated with computation of environmental simulations on HPC resources in parallel, which I will discuss in the following.

Problem 1

Parallelization of existing sequential simulations.

Parallelization of sequential simulations may necessitate significant design modifications or code reprogramming in a modern language, demanding specialized software, advanced computational resources and programming expertise from domain scientists. Modelers often tend to avoid parallelization due to its complexity, emphasizing the need for seamless parallel computing tools. Moreover, system-level design often requires collaboration among experts from various scientific and application domains (Pfaffe, 2020, Suslov et al., 2020).

Problem 2

Efficient parallelization of heterogeneous simulations.

Heterogeneous simulations often require exchanging data among various model units (levels of model details representing entities or phenomena), or dealing with dependent variables, which pose challenges in parallelization regarding data consistency and synchronization (Bieniusa et al., 2018). Efficient parallelization of these simulations is challenging due to their

[1] Virtual research environment for water and terrestrial environmental research – https://vforwater.de

heterogeneous nature and the need for partially sequential execution of the model units (Linford & Sandu, 2011, Álvarez-Farré et al., 2021). Inefficient parallelization or high communication overhead among processors can limit the benefits of parallel processing in these simulations (Meyer, 2014, Fujimoto, 2016, Fitzgerald et al., 2019).

Problem 3

Parallelized simulations are still compute-intensive.

While parallelization can significantly accelerate simulations, they can still be compute-intensive depending on the complexity and scale of the simulation, the size of the problem domain, and the available computing resources. For instance, solving differential equations for each individual model unit in a meteorological simulation, despite parallelization using spatial decomposition, encompasses a significant portion of the computation time. This challenge becomes even more pronounced as the resolution increases further.

Problem 4

Limited availability of HPC resources.

The availability of HPC resources can be restricted due to physical, technological, and operational constraints as well as high expenses. Moreover, there is an intense competition and high demand for the finite amount of resources available. HPC facilities are frequently utilized by multiple research groups or institutions, leading to extended resource queues for individual projects or researchers. This is a common challenge in computational science and research. Due to the limited number of available resources, various resource allocation strategies are developed to manage limited resources among competing applications (Qureshi et al., 2020).

Problem 5

HPC resources are energy-consuming.

HPC systems show immense computational capabilities to perform complex calculations and simulations, yet they exhibit a rapidly increasing energy consumption (Dorier et al., 2016, Morán et al., 2020). The processors and GPUs at the heart of these systems consume vast amounts of electricity. This issue not only translates into significant energy expenses but also raises environmental concerns due to the resultant increased carbon emissions and climate warming.

In this work, I focused on research, development, and application of computational optimization approaches in environmental simulations. In this context, I proposed several approaches to tackle the aforementioned challenges by leveraging the similarities or patterns found in the properties of simulation models and applying machine learning methods to reduce redundancies within the models. Ultimately, these approaches aid in reducing the computational complexity of the simulation, including the utilization of computational resources such as time, processing power, memory, and storage. In the following, I will discuss the research questions related to the previously mentioned challenges and outline my corresponding contributions. Throughout the remainder of this chapter, the term "simulations" will refer to large-scale and compute-intensive environmental simulations.

1.1 RESEARCH QUESTIONS

During my doctoral research, I studied how machine learning methods can be used to approximate and optimize compute-intensive environmental simulations. I conducted all of my work in the context of real-world use cases from environmental sciences. Throughout my study, I addressed several key research questions related to the challenges concerning parallelization of compute-intensive simulations and the need for programming expertise, efficient parallelization of heterogeneous simulations, solving time-consuming differential equations, and limited availability of HPC resources as well as their high energy consumption. The main question underlying this thesis is:

How to develop an effective and computationally efficient approximation of simulations using machine learning methods that yields outputs acceptable for domain scientists?

To address all the aforementioned challenges, this question can be broken down into five distinct research questions.

Research Question 1

Is there a universally applicable approach for evaluating simulations?

Simulation models can vary widely in their purpose, complexity, and the domain for which they are used. This heterogeneity increases the need for a general approach to compare and evaluate the simulation models. To address this issue, general principles can be employed to assess the quality and validity of simulations across various contexts. Thus, it is necessary to investigate these principles and develop a universally applicable approach for evaluating simulations.

Research Question 2

How to effectively identify and leverage similarities within simulation model units to approximate simulations?

Domain knowledge can help to understand the structure of simulation models and identify existing redundant functions throughout the simulation processes. An extensive analysis is required to identify similarities in simulation model units and utilize them to optimize the simulations. The existing approaches may need to be adapted, or unsupervised machine learning methods could be applied to discover the similarities.

Research Question 3

How can unsupervised machine learning methods be used to approximate and optimize simulations?

An investigation is necessary on how unsupervised machine learning methods can approximate simulations to reduce their computational complexity. Further analysis is needed to consider external simulation dynamics in the approximation approach.

Research Question 4

What are the necessary criteria for simulations to be met in order to be approximated using unsupervised machine learning methods?

To answer this question, it is essential to employ the approximation approach across various scenarios to demonstrate its versatility. These analyses will help to establish criteria for identifying appropriate compute-intensive simulations where the developed approximation approach can be effectively utilized.

Research Question 5

How can supervised machine learning methods be used to approximate and optimize simulations?

It is conceivable that certain simulations may not meet the criteria of the developed approximation approach through unsupervised machine learning. Thus, a comprehensive analysis becomes necessary to identify appropriate and useful existing supervised machine learning methods. Existing approaches need to be adapted to the modeled problem, or new approaches need to be developed.

1.2 SCIENTIFIC CONTRIBUTIONS

Overall, my research integrates the fields of simulation, computational optimization, and machine learning. By combining these areas of expertise, I aim to advance the state of the art in simulation techniques, making them more efficient, and applicable to a broader range of problems. My contributions as a doctoral researcher can be divided into three main parts, namely: i) A general simulation evaluation method on computational complexity and performance, ii) Approximation and optimization of simulations using unsupervised-, and iii) supervised machine learning methods. In the following, I will discuss each contribution in more detail.

Contribution 1

A practical and general approach to evaluate simulations based on their computational complexity and performance.

One of the main goals of environmental modeling scientists is the development of well-performing yet parsimonious models for natural systems. Measuring the performance and parsimony (complexity) of simulation models is theoretically key and practically challenging for environmental science. To address this issue, I introduced a practical and general method, applicable to simulation models from any domain, leveraging information theory for evaluation of environmental models. In this approach, I used two evaluation measures: 1. *performance*: this refers to the agreement of a simulation model with a real-world system as well as the model's ability to reduce predictive uncertainty about an object of interest, and 2. *computational complexity*: this refers to the simulation's resource consumption in terms of run time or resource usage to provide the desired output. It is crucial to note that this complexity measure is distinct from the algorithm's time or space complexity. I quantified performance by measuring information loss relative to observations, and computational complexity by tracing the size of read and write system calls using a Linux system call tracer, both in the unit of bits. The evaluation is sensitive to model characteristics in terms of the size of a model and its input data, as well as the model's numerical scheme (explicit and iterative) and temporal resolution. Additionally, I presented an application of the method to watershed models, representing a wide diversity of modeling strategies such as simple, and advanced process-based methods, autoregressive, and neural network models (Chapter 4). This contribution was published in Azmi et al. (2021), and addresses Research Question 1.

Contribution 2

A study of using unsupervised machine learning methods for approximation and optimization of simulations.

In this part of my work, I contributed to approximating large-scale and compute-intensive environmental simulations, aiming to reduce their computational complexity by leveraging unsupervised machine learning methods. My primary focus is on efficiently simulating accurate, reduced models that approximate the underlying high-resolution simulations. In the proposed approach, I used unsupervised machine learning methods to cluster functionally similar model units and simulated a representative model unit for each cluster. This reduced the number of similar computations and, consequently, the computational complexity of the simulation. The underlying principle is that simulation dynamics depend on the static properties, current state, and meteorological forcing of model units. Based on this, it is assumed that similar model unit settings lead to similar simulation dynamics. Applying this principle in the use case of a hydrological simulation, namely CAOS (Zehe et al., 2014), I clustered similar model units, ran the simulation model on a small yet representative subset of each cluster, and scaled the simulation output of the cluster representatives to the remaining cluster members. In my analyses, I investigated the performance of three clustering methods namely, K-means (MacQueen, 1967), K-medoids (Kaufman & Rousseeuw, 1987), and DBSCAN (Ester et al., 1996) on the CAOS simulation without considering any meteorological forcing on model units (Chapter 5). This contribution addresses Research Question 2, and was published in Azmi (2018b) and Azmi et al. (2019).

Contribution 3

An evolutionary approach of clustering for approximation and optimization of simulations.

This contribution extends the best performed clustering method from Contribution 2, considering additionally a meteorological forcing on model units in the use case of the CAOS simulation. In this approach, I introduced a method to make use of the static properties, current state, as well as meteorological forcing of model units to reduce computational complexity in the simulation. The approach consists of several steps, mainly employing an evolutionary approach of clustering, and scaling of the simulation output of the cluster representatives to the remaining cluster members. The clustering process includes utilizing K-means clustering together with a K-determiner that automatically defines a suitable number of clusters (Chapter 5). This contribution assists in answering Research Question 3, and was published in Azmi et al. (2020).

Contribution 4

A study of the application generality of the evolutionary approach.

To identify the essential conditions to generalize the evolutionary approach, I investigated its applicability to two other use case simulations. The first simulation is within the hydrological domain, called SHM (Ehret et al., 2020), and the second one is the atmospheric chemistry model from the ICON-ART simulation (Rieger et al., 2015) in meteorology. The results of the experiments on the ICON-ART simulation showed that the optimization approach itself introduced a significant computational overhead, which hindered the speedup of the simulation. A trade-off between the accuracy of the result and computational complexity was also observed. Through this analysis, several key criteria have been revealed regarding the approach's suitability and broader applicability (Chapter5). This contribution answers Research Question 4.

Contribution 5

A supervised machine learning approach for approximation and optimization of simulations.

In Contribution 4, I observed that the initially introduced approximation approach might not be appropriate for certain simulations (e.g., ICON-ART). Thus, to address this constraint, I proposed an alternative approximation approach using supervised machine learning methods, focusing on deep neural networks. I developed a neural network-based approach (ICONET) consisting of a multi-feature Long Short-Term Memory (LSTM) model (Hochreiter & Schmidhuber, 1997) for forecasting atmospheric chemistry. This approach replaces the compute-intensive chemistry simulation of about two million atmospheric cells with a trained neural network model to forecast the concentration of trace gases at each cell and to reduce the computation complexity of the simulation (Chapter 6). This contribution addresses Research Question 5, and was published in Azmi et al. (2023).

1.3 LIST OF PUBLICATIONS

The contributions in this thesis except Contribution 4 have already been published in peer-reviewed conference proceedings and journals. Own publications relevant to the content presented in this thesis are listed in the following.

- **Elnaz Azmi**. *On Using Clustering for the Optimization of Hydrological Simulations.* 2018 IEEE International Conference on Data Mining Workshops (ICDMW), Sentosa, Singapore, 17-20 Nov 2018, pages 1495–1496, IEEE, 2018.
doi:10.1109/ICDMW.2018.00215

- **Elnaz Azmi**, Uwe Ehret, Jörg Meyer, Rik van Pruijssen, Achim Streit, and Marcus Strobl. *Clustering as Approximation Method to Optimize Hydrological Simulations.* European Conference on Parallel Processing (Euro-Par), Göttingen, Germany, 26-30 Aug 2019, pages 256–269, Springer, 2019.
doi:10.1007/978-3-030-29400-7_19

- **Elnaz Azmi**, Marcus Strobl, Rik van Pruijssen, Uwe Ehret, Jörg Meyer, and Achim Streit. *Evolutionary Approach of Clustering to Optimize Hydrological Simulations.* International Conference on Computational Science and Its Applications (ICCSA), Online, 1-4 Jul 2020, pages 617–633, Springer, 2020. **[best paper award]**
doi:10.1007/978-3-030-58799-4_45

- Uwe Ehret, Rik van Pruijssen, Marina Bortoli, Ralf Loritz, **Elnaz Azmi**, and Erwin Zehe. *Adaptive clustering: reducing the computational costs of distributed (hydrological) modelling by exploiting time-variable similarity among model elements.* Hydrology and Earth System Sciences (HESS), volume 24, pages 4389–4411, Copernicus, 2020.
doi:10.5194/hess-24-4389-2020

- **Elnaz Azmi**, Uwe Ehret, Steven V. Weijs, Benjamin L. Ruddell, and Rui A. P. Perdigão. *Technical note: "Bit by bit": a practical and general approach for evaluating model computational complexity vs. model performance.* Hydrology and Earth System Sciences (HESS), volume 25, pages 1103–1115, Copernicus, 2021.
doi:10.5194/hess-25-1103-2021

- **Elnaz Azmi**, Jörg Meyer, Marcus Strobl, Michael Weimer, and Achim Streit. *Approximation and Optimization of Global Environmental Simulations with Neural Networks.* Platform for Advanced Scientific Computing Conference (PASC), Davos, Switzerland, 26-28 Jun 2023, ACM, 2023.
doi:10.1145/3592979.3593418

- **Elnaz Azmi**, Uwe Ehret, Jörg Meyer, Rik van Pruijssen, Achim Streit, and Marcus Strobl. *The optimization of hydrological simulations using dynamic clustering.* EGU General Assembly, Geophysical Research Abstracts volume 20, Vienna, Austria, 8–13 Apr 2018, EGU2018-10596, 2018.
https://meetingorganizer.copernicus.org/EGU2018/EGU2018-10596.pdf

- Rik van Pruijssen, **Elnaz Azmi**, Erwin Zehe, and Uwe Ehret. *Representative computation: How to make use of hydrological similarity?* EGU General Assembly, Geophysical

Research Abstracts volume 20, Vienna, Austria, 8–13 Apr 2018, EGU2018-10210, 2018.
https://meetingorganizer.copernicus.org/EGU2018/EGU2018-10210.pdf

- Uwe Ehret, Rik van Pruijssen, Marina Bortoli, Ralf Loritz, **Elnaz Azmi**, and Erwin Zehe. *Dynamical clustering: A new approach to make distributed (hydrological) modeling more efficient by dynamically detecting and removing redundant computations.* EGU General Assembly, Online, 4–8 May 2020, EGU2020-4772, 2020.
  doi:10.5194/egusphere-egu2020-4772
- **Elnaz Azmi**, Jörg Meyer, Marcus Strobl, Michael Weimer, and Achim Streit. *Approximation and Optimization of Atmospheric Simulations in High Spatio-Temporal Resolution with Neural Networks.* EGU General Assembly, Vienna, Austria, 24–28 Apr 2023, EGU23-6287, 2023.
  doi:10.5194/egusphere-egu23-6287

1.4 CODE AND DATA AVAILABILITY

The implementations and some example data of own methods introduced in this thesis and previously listed papers are available under the MIT license at the following addresses:

- https://github.com/elnazazmi/hyda (Azmi, 2018a, 2020)
- https://github.com/KIT-HYD/model-evaluation (Azmi & Ehret, 2021)
- https://github.com/elnazazmi/iconet (Azmi, 2023)

1.5 THESIS OUTLINE

The remainder of this thesis is organized as follows:

In Chapter 2, the relevant background to the research in this thesis is presented. I introduce the fundamentals of environmental simulations, outline their use in real-world applications, and lay out the challenges related to high spatio-temporal and compute-intensive simulations. Additionally, the use case simulations from the following chapters are described in detail. Next, I present the simulation evaluation concepts and the generality of the existing evaluation techniques. The chapter ends with an introduction to different machine learning algorithms and their role in approximation and optimization of environmental simulations.

Chapter 3 presents related works on key principles for evaluating environmental models, simulation approximation and optimization approaches, as well as the application of different machine learning methods in optimization of the environmental simulations. As some approximation methods benefit from similarities in the simulation model units, I present the related work on the identification of such similarities.

In Chapter 4, I investigate the evaluation of the simulations based on their performance and computational complexity.

In Chapter 5, I investigate how unsupervised machine learning methods can be used to approximate and optimize compute-intensive environmental simulations. Furthermore, I propose an evolutionary approach using clustering to tackle variable meteorological forcing during the simulation time steps.

Chapter 6 presents how supervised machine learning methods can be used to approximate and optimize compute-intensive environmental simulations.

Chapter 7 summarizes the research done in this thesis, focusing on my contributions. I conclude with an overview of open questions and sketch possible directions for future research.

A theory is the more impressive the greater the simplicity of its premises is, the more different kinds of things it relates, and the more extended its area of applicability.

Albert Einstein

2 Background

In this chapter, the relevant background and principles to understand the research described in this thesis are presented. Section 2.1 gives a brief overview of environmental simulations, introducing the general concepts and technical terms of the simulations utilized in the following chapters. Section 2.2 defines the key aspects of model evaluation and proceeds with the introduction of several model evaluation methods and metrics. Section 2.3 provides an overview of the basic concepts of the machine learning algorithms used in this thesis. Finally, Section 2.4 presents approaches to approximate complex simulations, thereby optimizing them in terms of their computational complexity. The background provided in this chapter has been partly published in Azmi (2018b), Azmi et al. (2019, 2020, 2021, 2023), and Ehret et al. (2020).

2.1 ENVIRONMENTAL SIMULATIONS

Modeling and simulation involve creating physical, mathematical, or logical representations of systems, phenomenons, or processes (Trenholme, 1994, Hestenes, 1997). These representations are used across various fields to understand, analyze, and forecast the behavior of complex systems. Models and simulations offer a means to explore real-world systems within controlled environments, enabling experimentation with infeasible or impractical scenarios. Computer-based simulations involve the process of mathematical modeling performed on a computer, utilizing various models, such as physics-based or data-driven models (Winsberg, 2022). Additionally, they support iterative design and optimization, along with the generation of substantial synthetic data for training machine learning models or conducting statistical analyses (Bărbulescu, 2016). One application of computer-based simulations is in the realm of environmental science (Anderson et al., 2015). In this context, computational models are employed to investigate and forecast physical phenomena or the functionality of environmental systems. Environmental simulations rely on a broad range of scientific, mathematical, computational, and engineering expertise and techniques. They find practical use in various environmental fields such as hydrology, oceanography, and meteorology, among others (Bennett et al., 2010). In the following, I will provide the principles and concepts behind the environmental simulations utilized in this thesis.

2.1.1 HYDROLOGICAL MODEL CAOS

The Catchments as Organised Systems (CAOS) model (Zehe et al., 2014) is one of the hydrological models used in this work. This model simulates water related dynamics at the lower meso-scale ($10 - 200$ km^2). The CAOS model provides a high-resolution and distributed process-based simulation of water- and energy fluxes in the near surface atmosphere, the Earth's surface, and subsurface. Hydrological simulations, such as CAOS, are generally applicable in various fields of hydrological research, including agricultural water demand estimation, erosion protection, flood forecasting, and assessment of the impact of land use changes or climate variations on water availability and quality.

The term "catchment", in hydrology also known as a watershed or drainage basin, refers to the area where all precipitation within its boundaries flows into a single point or outlet along a river (Roper, 1979). This outlet represents the lowest elevation within the catchment, demonstrating how water naturally moves from higher elevated regions to lower ones. Hydrologists identify and delineate natural catchment areas based on the physical landscape and hydrological characteristics.

In the CAOS model, the functioning of catchments is controlled by a hierarchy of three main model units, namely Elementary Functional Unit (EFU), Hillslope (HSL), and River element (RIV) (Figure 2.1). The smallest main model units are soil columns, referred to as EFUs. Each EFU is composed of several components such as soil surface, soil layers, and vegetation. In an EFU, all vertical water movements (e.g, vertical soil water flow, and evapotranspiration) are modeled. The next higher level of the hierarchy consists of HSLs containing and connecting all EFUs along the downhill path from a ridge line which is a natural barrier, guid-

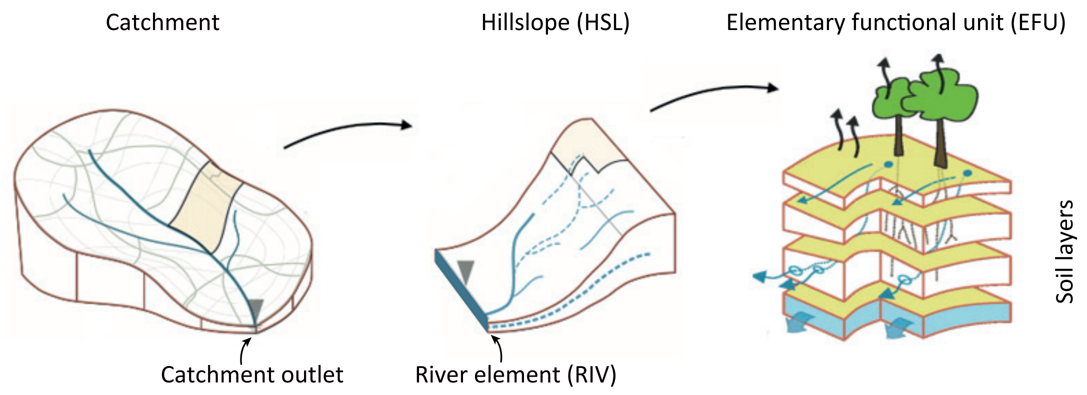


Figure 2.1: Simplified hierarchy of the CAOS model units. Left: A catchment is an area that collects and drains water to a common outlet (gray triangle at the bottom of the catchment). Middle: Each catchment is divided into HSLs, where water flows into a RIV (blue line on the bottom of the HSL). Right: Each HSL consists of laterally connected EFUs with components such as soil surface, soil layers, and vegetation, wherein water flows. Modified after Zehe et al. (2014).

ing water into a RIV depending on the slope and the surrounding topography. In an HSL, all lateral downhill flow processes (surface flow and groundwater flow) are modeled in network-like flow structures on the surface and within the subsurface (blue lines in Figure 2.1, middle and right sketch). The third main model units, RIVs, are linear elements along the lower edge on the surface of an HSL. They are parts of a river, connected sequentially to each other and transport the water of a catchment to the lowermost point, the catchment outlet. Before executing the CAOS simulation, each catchment is divided into HSLs based on the flow network derived from a digital elevation model.

HSLs act completely independent of each other, and they are subdivided into laterally connected EFUs. EFUs within the same HSL may interact due to the backwater effect, which refers to the phenomenon where water is obstructed or delayed in its typical flow, deviating from its regular or natural course (Langbein & Iseri, 1960). The hierarchy of model units is represented as a network model, including various objects and their interactions, for better understanding of the dependencies among the model units (Figure 2.2).

The structure of the CAOS model is specified based on domain knowledge. The resulting simulation dynamics depend on three main factors: 1. model unit properties (static), 2. model unit state (current discharge), and 3. meteorological forcing (rainfall or radiation). The underlying principle of the model is that similar properties, states and meteorological forcing of the model units lead to similar simulation dynamics (Zehe et al., 2014). Exploiting this concept, it might be possible to reduce computational complexity of the CAOS model by

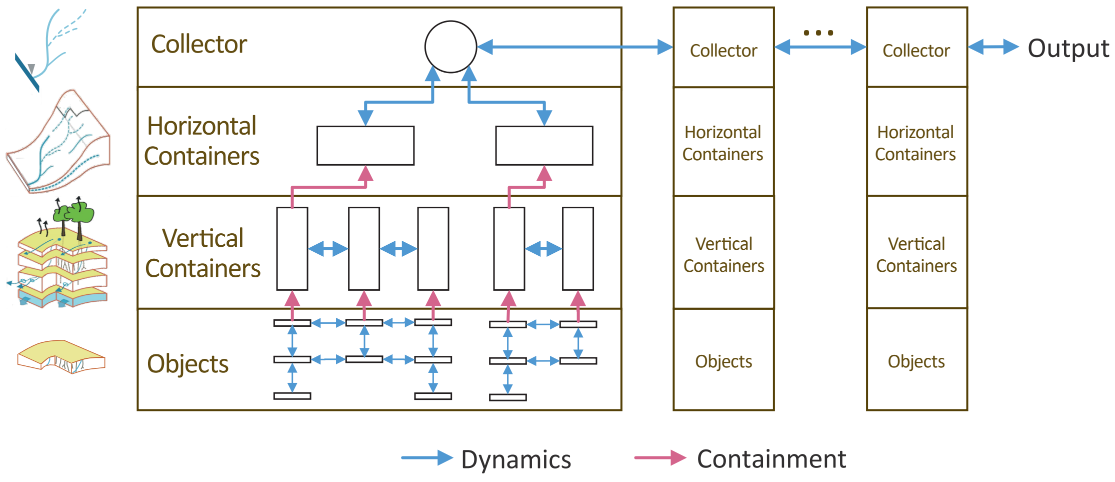


Figure 2.2: The abstracted hierarchical network model from the CAOS model. Modified after Azmi (2018b).

eliminating these redundancies from the computation process. To explore this hypothesis, I developed an optimization approach and tested it on the CAOS model.

The study area used to develop and test the hydrological model is the Attert catchment, located in the Grand Duchy of Luxembourg. Since the computation of the CAOS model is time-consuming, a representative subset of the Attert catchment (Figure 2.3), the Wollefsbach catchment, was used for the development of my proposed approach presented in Chapter 5. The CAOS simulation is implemented in MATLAB. To establish a benchmark for the simulation time, I executed the CAOS model for the Wollefsbach catchment for January 2014 at a five-minute resolution (Section 5.1.2). The Wollefsbach catchment holds 1.8 % of the area, and 2.4 % of the number of HSLs of the Attert catchment, with an execution time of 50.6 hours (Table 2.1). The simulation execution time of the entire Attert catchment could be estimated at 3.5 months, but has not been determined yet. Access to the model and the required dataset was granted by the CAOS project[1].

Table 2.1: Study area properties. Taken from Azmi et al. (2019).

Catchment	Area [km^2]	# HSLs	Run Time [h]
Attert	247.0	9716	–
Wollefsbach	4.5	232	50.6

[1] Catchments as Organised Systems – https://caos-project.de

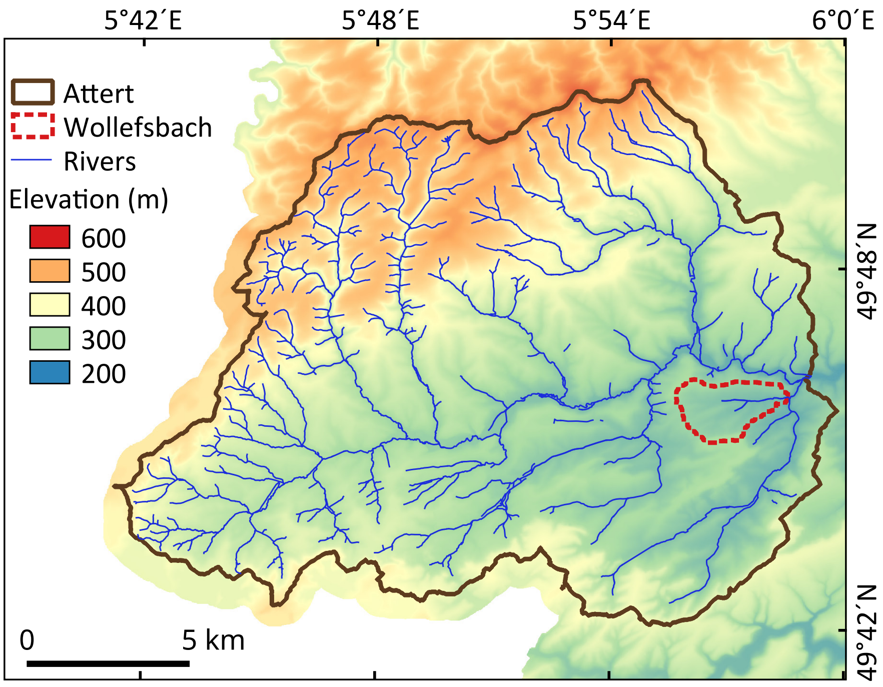


Figure 2.3: Digital elevation model of the Attert catchment (study area, brown line) and the Wollefsbach catchment (use case, red dashed line). Taken from Azmi et al. (2019).

2.1.2 HYDROLOGICAL MODEL SHM

The Simple Hydrological Model (SHM) (Ehret et al., 2020) is closely related to established hydrological models, such as the HBV hydrology model from the Swedish Meteorological and Hydrological Institute (Bergström, 1976). The SHM model contains a simple structure, conceptual, yet distributed architecture tailored to the structure and hydrological function of the Attert catchment. In this model, the catchment is divided into subcatchments and river elements using a digital elevation model. The water stocks and fluxes in each subcatchment are represented in a conceptualized manner by a set of linked linear reservoirs (Figure 2.4). The choice of the type, number, and linkage of reservoirs is based on the insights about the hydrological functioning of the Attert catchment and suitable conceptualizations reported by Fenicia et al. (2014, 2016). In the schematic illustration of the SHM model, the first storage (top left in Figure 2.4) represents the unsaturated zone reservoir of a subcatchment. Precipitation falling onto a subcatchment (s_u) is divided into direct runoff ($q_{u,out}$) and soil moisture replenishment as a nonlinear function of current soil moisture. Evapotranspiration draws water

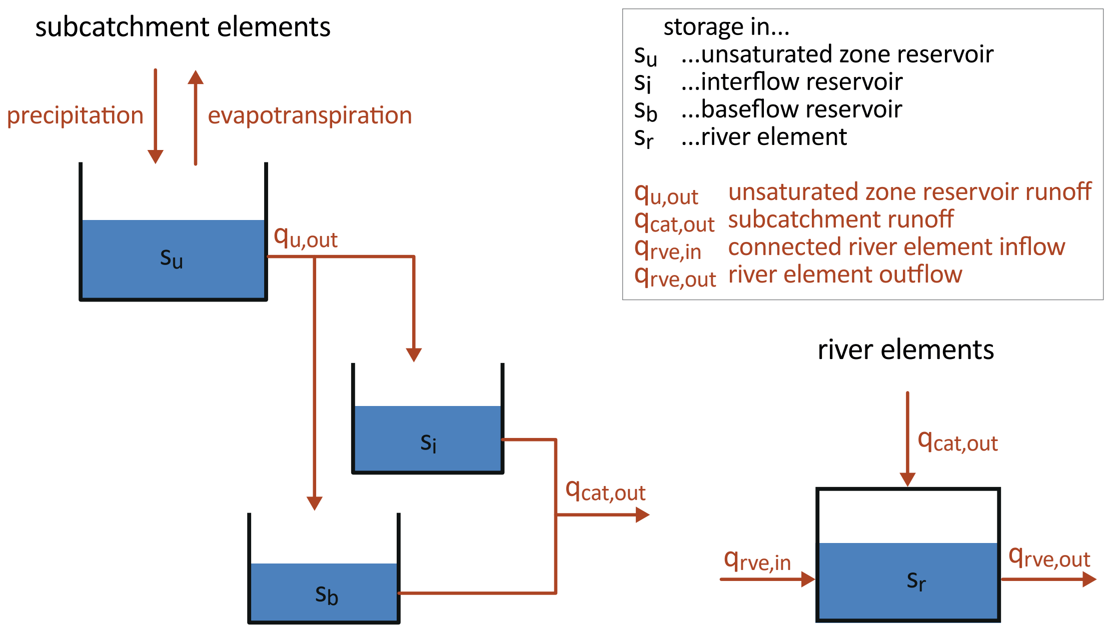

subcatchment elements
precipitation
evapotranspiration
s_u
$q_{u,out}$
s_i
s_b
$q_{cat,out}$
storage in...
s_u ...unsaturated zone reservoir
s_i ...interflow reservoir
s_b ...baseflow reservoir
s_r ...river element
$q_{u,out}$ unsaturated zone reservoir runoff
$q_{cat,out}$ subcatchment runoff
$q_{rve,in}$ connected river element inflow
$q_{rve,out}$ river element outflow
river elements
$q_{cat,out}$
$q_{rve,in}$
s_r
$q_{rve,out}$

Figure 2.4: Illustration of the SHM model consists of the structural elements, state variables (black), and fluxes (red). Modified after Ehret et al. (2020).

from the unsaturated zone storage. Direct runoff is split by a constant factor and replenishes two linear reservoirs, with one representing interflow (s_i) and the other representing base flow (s_b). Runoff from the interflow and base flow reservoirs ($q_{cat,out}$) are merged and then enters the river system. The river system is represented by a linear reservoir cascade (bottom right in Figure 2.4), where each unit represents a river stretch of about one km. The SHM model is implemented in MATLAB. The numerical scheme is non-iterative forward in time, and the time stepping is hourly.

2.1.3 ATMOSPHERIC MODEL ICON-ART

The meteorological simulation use case of this work is ICON-ART, the ICOsahedral Non-hydrostatic modeling framework (ICON) (Zängl et al., 2015) with its extension for Aerosols and Reactive Trace gases (ART) (Rieger et al., 2015, Weimer et al., 2017, Schröter et al., 2018). The ICON model has been jointly developed by the German Weather Service (dt. Deutscher Wetterdienst, DWD) and the Max Planck Institute for Meteorology (MPI-M). It is a unified global numerical weather prediction and climate modeling system, encompassing all time and spatial scales that are relevant for the atmosphere. Since 2015, the ICON model has been used

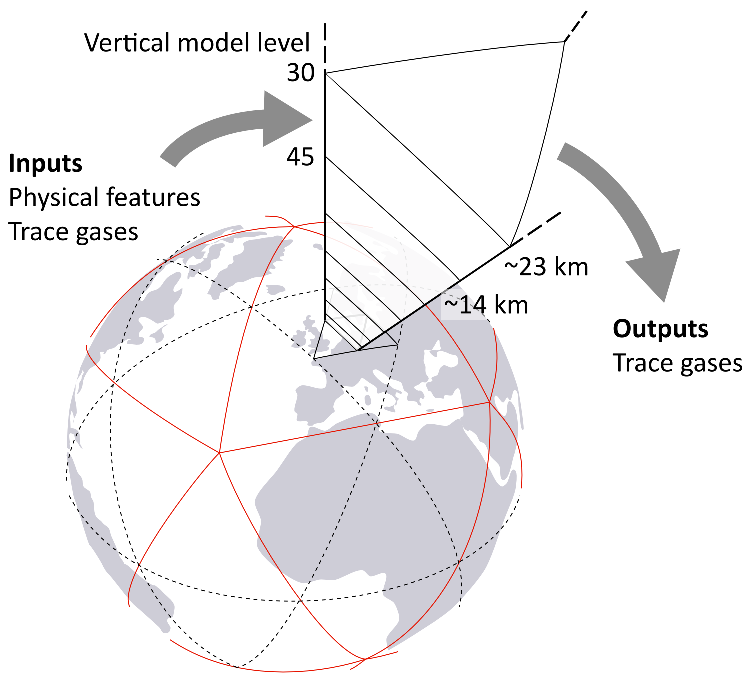


Figure 2.5: Schematic illustration of global horizontal triangular grids and a grid cell with its vertical model levels as used by ICON-ART. The input and output features are from the studied atmospheric chemistry simulation of ICON-ART. The various line styles and colors of grid cells (triangles) on the globe show the grids in different resolutions. Provided by courtesy of Marcus Strobl.

for operational weather forecasting at the German Weather Service (Zängl et al., 2015).

The ICON–ART model incorporates the interactions of atmospheric trace substances (gases and particles) and the state of the atmosphere within a numerical weather prediction model from the global to regional scale (Rieger et al., 2015). The chemistry and photolysis rates in ICON-ART are calculated using the box model CAABA/MECCA and CLoudJ for each grid cell (Sander et al., 2011, Prather, 2015, Schröter et al., 2018). In other words, the differential equations for the chemical reactions in ICON-ART are solved separately in each grid cell (Figure 2.5). As the atmospheric chemistry simulation is a compute-intensive component in the ICON-ART model, it serves as a suitable use case for this work. In this simulation, the chemistry is calculated separately for each grid cell, independently of the neighboring grid cells. To minimize the influence of other components in the ICON-ART simulation on the atmospheric chemistry model, the simulation is executed without transportation of trace gases. This preserves a closed system within the model, eliminating interactions with neighboring grid cells.

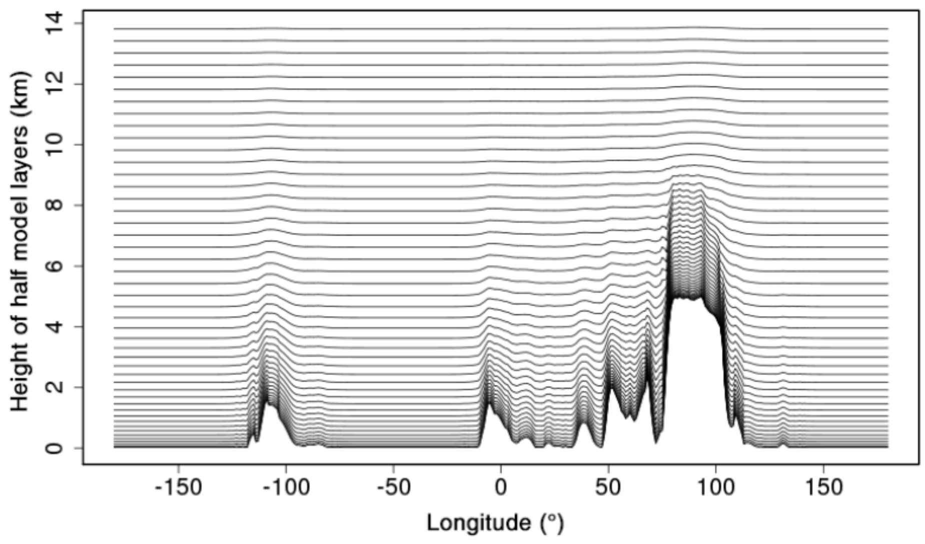


Figure 2.6: Schematic illustration of vertical model levels in ICON-ART. Height of the lowest 46 ICON model layers at 33° N in the configuration with 90 total model layers. Taken from Weimer et al. (2017).

In this study, I use ICON-ART version 2.1 of the operational setup at DWD, which incorporates 90 vertical model levels (Figure 2.6) from near ground (level 90) up to 75 km (level 1), and a horizontal resolution of about 160 km. This configuration results in approximately 1.8 million grid cells (Zängl et al., 2015). The studied simulation scenario focuses on a chemical mechanism for ozone in the stratosphere, spanning from vertical model level 45 to 30 (16 levels) (Reinert et al., 2021). This simulation employs a subset of 23 reactions for oxygen- and nitrogen-related species (Schröter et al., 2018). From the fourteen gases involved in these reactions, N_2 and O_2 are no trace gases as they are present in excess. Therefore, they are treated as constants and are not included in this study. Consequently, each output data file of the simulation contains the volume mixing ratio (VMR) of twelve trace gases and the values of three physical features for all grid cells. The twelve trace gases are HNO_3, HO_2, H_2O, NO, NO_2, NO_3, N_2O, N_2O_5, OH, O_3, $O(^1D)$, and $O(^3P)$. The physical features comprise temperature, pressure, and the cosine of the solar zenith angle (cos SZA). The solar zenith angle refers to the angle between the surface normal (an upright line extending vertically from the Earth's center above the surface) and the Sun's direction (Jacobson, 1999). These features are provided at a six-minute resolution over the course of one day. The studied simulation covers the data from the years 2013 and 2014. The ICON-ART simulation is implemented in FORTRAN. The computation environment is defined in Section 6.4.

2.2 MODEL EVALUATION FACETS AND METRICS

One of the goals of natural sciences is to create models that are both efficient and effective in describing natural phenomena and systems. The evaluation of these models involves considering two key aspects, namely performance and complexity. In terms of performance, these models should be able to produce results that align with real-world observations, demonstrating high levels of accuracy and overall correctness (Kirchner, 2006). From the complexity aspect, the models should be concise, explainable, comprehensible, compact, and energy-efficient (Solomonoff, 1964). This section introduces the main concepts of model evaluation and commonly used metrics that are utilized in this thesis.

2.2.1 PERFORMANCE

Performance refers to a model's ability to reduce predictive uncertainty about an object of interest. In other words, it pertains to the extent of agreement or disagreement between the model's prediction with the observations of the related real-world system (commonly known as ground truth). Since the aim of this thesis is to approximate and optimize environmental simulations, the term "ground truth" refers to the original use case simulation and its output data that serves as a reference point for evaluating the approximated and optimized simulations.

Different evaluation measures are used for each type of predictive model, depending on whether it is a classification model (for nominal or binary data) or a regression model (for continuous data). As the models used in this work belong to regression models, I will describe several common techniques and metrics to measure the performance of these models. These techniques and metrics could be used individually or in combination. Selecting an appropriate quality metric for forecasting models is challenging due to the various aspects that need to be considered when assessing a model's performance. In this discussion, I will present the advantages and disadvantages of relevant metrics.

One of the basic evaluation metrics is the difference D (James et al., 2023) (Equation 2.1) between the forecast $\widehat{Y}$ and the ground truth Y values, as follows:

$$D(Y, \widehat{Y}) = \widehat{Y} - Y. \tag{2.1}$$

This metric directly indicates the extent to which the forecast values deviate from the ground truth, and it shares the same unit as the calculated values. However, it has a limitation, as it does not allow for a fair comparison between the deviations of variables on different scales.

An alternative metric is the relative error *RE* (Abramowitz & Stegun, 1972) (Equation 2.2), which measures the absolute difference between the forecast $\widehat{Y}$ and the ground truth Y values divided by the ground truth value. The relative error is mathematically presented as:

$$RE(Y, \widehat{Y}) = \frac{|\widehat{Y} - Y|}{Y}\,. \tag{2.2}$$

While this metric can handle variables on different scales, it is not defined for $Y = 0$.

The Mean Squared Error *MSE* (Bennett et al., 2013) (Equation 2.3) is a regression metric that measures the mean squared difference between the forecast $\widehat{Y}$ and the ground truth Y values. This metric is highly sensitive to outliers. *MSE* assigns greater significance to large errors compared to smaller ones. For a vector of n forecast values derived from a sample of n data points, *MSE* is formally expressed as:

$$MSE(Y, \widehat{Y}) = \frac{1}{n}\sum_{i=1}^{n}(\widehat{Y}_i - Y_i)^2\,. \tag{2.3}$$

Another common metric is the Root Mean Squared Error *RMSE* (Barnston, 1992) (Equation 2.4), which is obtained by taking the square root of *MSE*, see Equation 2.3. *RMSE* reduces the impact of larger deviations in comparison to *MSE*. This metric is commonly preferred due to its interpretability, since it expresses the error in the same unit as the target value. *RMSE* values range from zero to positive infinity, where values closer to zero indicate a more accurate estimation. *RMSE* is the standard deviation of the forecast errors, formally defined as:

$$RMSE(Y, \widehat{Y}) = \sqrt{MSE(Y, \widehat{Y})}\,. \tag{2.4}$$

Although *RMSE* effectively measures the distance between two curves, it does not measure how accurately the forecast follows the trend of the ground truth curves. Thus, I choose an appropriate metric for this, which is the Kling-Gupta Efficiency *KGE* (Gupta et al., 2009) (Equation 2.5). This widely used measure in hydrological modeling evaluates the goodness-of-fit. *KGE* values range from negative infinity to one, and the values closer to one indicate a better fit. The mathematical representation of *KGE* is as follows:

$$KGE(Y, \widehat{Y}) = 1 - \sqrt{(\rho - 1)^2 + (\alpha - 1)^2 + (\beta - 1)^2}\,, \tag{2.5}$$

with:

$$\rho = \frac{COV(Y, \widehat{Y})}{\sigma_Y \cdot \sigma_{\widehat{Y}}}, \qquad \alpha = \frac{\sigma_{\widehat{Y}}}{\sigma_Y}, \qquad \beta = \frac{\mu_{\widehat{Y}}}{\mu_Y}, \tag{2.6}$$

where $COV(Y, \widehat{Y})$ is the covariance between the forecast and the ground truth values, σ is the standard deviation, and μ is the mean. In other words, ρ is the Pearson Correlation Coefficient *PCC* (Pearson, 1896) between the forecast and the ground truth values, α is a measure of variability in the data values, and β is equal to the mean of the forecast values over the mean of the ground truth values.

PCC (ρ in Equation 2.6) is a linear model used to test the individual effect of each of many regressors, and it serves as a scoring function commonly employed in feature selection procedures. Generally, it is a common practice to calculate *PCC* as a standardized measure of the predictive accuracy of a model (Daetwyler et al., 2008). It is the ratio between the covariance of two variables and the product of their standard deviations. Thus, it is a normalized measure of the covariance, yielding a value within $[-1, 1]$, where 1 defines the highly positive linear correlation, 0 is the non-linear correlation and -1 is for a highly negative linear correlation. *PCC* does not provide any information about the difference between dependent and independent variables, nor about the slope of the curves. Its primary function is to determine whether a relationship exists between the variables.

In information theory, the *entropy* state function quantifies the amount of required information in the system to fully specify the microstate of the system. Information theory is a unified field of mathematics and computer science that focuses on the quantification, storage, communication, and processing of information (Shannon, 1948, Reza, 1961, MacKay, 2003). The principles of information theory are utilized in machine learning for tasks such as model evaluation, feature selection, and clustering. It provides a framework for measuring and understanding the fundamental properties of information, where *entropy* emerges as a key measure. In the context of performance evaluation of classification models, *entropy* serves as a metric to assess the diversity of class predictions. In the context of communication, *entropy* is the measure of the amount of missing information before reception. Information *entropy*, often called Shannon *entropy*, was originally devised by Shannon (1948) to study the size of information of a transmitted message. Formally, information is defined as the negative logarithm of the probability of an event $p(x)$. Information *entropy* $H(X)$ is defined as the expected or average value of information (Equation 2.7) of a specific value of a data set $X = \{x_1, x_2, ..., x_n\}$.

$$H(X) = -\sum_{x \in X} p(x)\, log_2\, p(x)\ , \tag{2.7}$$

where the base of the logarithm determines the units, e.g., the binary logarithm corresponds to bits (units of information). The *entropy* measured in bits yields a value within $[0, \infty]$. The minimum *entropy* of zero bits indicates that an event is certain and carries no uncertainty. The fundamental concept behind information theory is that, learning the occurrence of an improbable event is more informative than learning the occurrence of a probable event (Balian, 2004). In other words, high *entropy* or high information implies high uncertainty or low event probability.

2.2.2 COMPLEXITY

In the hydrological sciences, complexity is widely employed in a sense of its dictionary definition (see the Oxford Learner's Dictionaries definition of *complexity*[2]) to refer to "the state of being formed of many parts; the state of being difficult to understand" (see also Gell-Mann (1995) on various interpretations of complexity). Complexity in the context of mathematics and computer science comprises two aspects, namely descriptive complexity and computational complexity:

1. Descriptive complexity is related to inference quality, generality, and the size of the model. In the framework of Algorithmic Information Theory (AIT), descriptive complexity of a model is measured by its size expressed in bits (binary digits), when stored as instructions for a computer (Solomonoff, 1964, Chaitin, 1966, Kolmogorov, 1968, Solomonoff, 1978). It is therefore a formalization of Occam's razor[3]. This philosophical razor, a bedrock principle of science, argues that the least descriptively complex model is preferable, at a given level of predictive performance that is adequate to the question or application at hand. Furthermore, the same concept of descriptive complexity can also be directly applied to data. The complexity of data is formalized as its shortest description length, and the best model for the data is that shortest description: the shortest computer program that has the data as an output (Solomonoff, 1964, Wallace & Boulton, 1968, Rissanen, 1978, MacKay, 2003). In all these approaches that employ Occam's razor, an emphasis is placed on descriptive complexity and performance, but is completely independent of any practical considerations such as lim-

[2]https://www.oxfordlearnersdictionaries.com/definition/english/complexity

[3]Commonly attributed to the fourteenth-century scholastic philosopher William of Ockham, but emphasized about twenty years before Ockham by John Duns Scotus (Li & Vitányi, 2008)

ited storage space or computing power, in other words, it ignores computational complexity. So while Occam's razor promotes models that achieve effective compression of experimental data, compression for the sake of meeting constraints in a storage-limited world is not the primary goal. Instead, the reverse is true: finding the shortest description is the process of inference, achieved by distilling patterns from data in order to find general predictive laws.

2. Computational complexity is the model's effort to provide output, and is often a practical and economic concern for limited computing resources. It provides an abstract understanding of how an algorithm's performance scales with input size. The efficiency at which models generate their output is subject to the discipline of Analysis of Algorithms (Aho et al., 1975, Sedgewick & Flajolet, 2013). In this framework, computational complexity can be measured in terms of two finite resources that are required for a computation: time and/or space. Time complexity describes the amount of time a computer needs to run an algorithm as a function of the input size. It can be measured in terms of clock cycles, number of floating point operations or steps of a Turing machine, and often it is the scaling with the input size that is of interest. Space complexity is related to the amount of space or memory used by an algorithm to run as a function of the input size. Computational complexity can also be measured in terms of the run time of an algorithm. Run time is a concrete measure, referring to the actual time taken by an algorithm to complete on a particular input, considering the available computational resources. It is important to note that, run time depends on the actual hardware and software environment in which the algorithm is executed. In model evaluation, speed-up serves as a metric to evaluate the enhancement in computational complexity, run time or processing speed achieved by a particular optimization method in comparison to a reference implementation. The speed-up is calculated as follows:

$$\textit{Speed-up} = \frac{\textit{Reference run time}}{\textit{Optimized run time}} \,. \tag{2.8}$$

It measures how many times faster one algorithm is compared to another, either for a specific input instance or across different hardware configurations.

2.3 MACHINE LEARNING ALGORITHMS

Machine learning refers to the utilization of algorithms that automatically analyze data, with the aim of detecting the underlying patterns in order to predict future data or make decisions based on the discovered patterns. Generally, machine learning can be classified into three main categories, distinguished by the learning method and training approach: 1. unsupervised (descriptive), 2. supervised (predictive), and 3. reinforcement learning (Silvestrini & Lavagna, 2022). In the following, I will provide an overview of the first two categories considered in this thesis.

2.3.1 UNSUPERVISED LEARNING METHODS

Unsupervised learning algorithms use the properties of data points, such as distance- or density distribution, to identify clusters of similar data points based on the provided similarity measurement. Due to the lack of labeled data (ground truth values), validating the output of unsupervised learning algorithms is commonly difficult, and heuristic arguments are often used to judge the quality of the results (Hastie et al., 2009). There are several methods within unsupervised learning, including density estimation, clustering, dimensionality reduction, generative models, and anomaly detection. In the following, I will describe two of these methods in detail: clustering and dimensionality reduction, which are used in this thesis.

CLUSTERING

Clustering is one of the main machine learning methods, categorized as unsupervised learning (Hastie et al., 2009, Dubes & Jain, 1976). This method learns to detect similar data points that can be grouped together. Based on the type of input data, clustering can be divided into similarity-based and feature-based clustering. The similarity-based clustering takes a dissimilarity or distance matrix as input, while the input of the feature-based clustering is a feature or design matrix (Murphy, 2012). Clustering methods can also be categorized regarding their output types. Partitional clustering produces a set of clusters, while hierarchical clustering generates a nested tree of clusters (Kaufman & Rousseeuw, 2009). These widely studied major categories of clustering can be further divided into sub-categories like distance-based, density-based, grid-based and model-based clustering (Aggarwal & Reddy, 2013). In the following, the clustering methods used in this work are described.

K-means Clustering is one of the most widely used partitional clustering algorithms (Aggarwal & Reddy, 2013, Hastie et al., 2009, Ding & He, 2004). It is categorized as distance-based clustering method. K-means aims to create clusters of similar data points, with each cluster represented by a central point called centroid. The algorithm starts with the initialization of K centroids uniformly at random or using a specific initialization method, such as K-means++ (Arthur & Vassilvitskii, 2007). K-means++ selects one initial centroid uniformly at random and the next $K-1$ centroids with the probability defined by Equation2.9, as follows:

$$p(x') = \frac{D(x')^2}{\sum_{x \in X} D(x)^2}, \tag{2.9}$$

where $D(x)$ represents the shortest distance from a data point x to the closest centroid, and x' is the next centroid selected from a set of data points X. The K-means++ initialization method improves the convergence, speed, and accuracy of K-means, therefore I used it in this study. In the next step of the K-means algorithm, each data point in the dataset is assigned to its closest centroid. Subsequently, the centroids are recomputed for each cluster by calculating the mean value of the data points assigned to those centroids. The process is repeated until the convergence criterion is met, indicating that the centroids no longer change (Aggarwal & Reddy, 2013, Arthur & Vassilvitskii, 2007). The goal of the algorithm is to minimize the Sum of Squared Errors (SSE) of the centroids. The K-means algorithm requires the number of clusters, denoted as K, as an input parameter. However, determining the optimal number of clusters poses a challenge. Approaches introduced by Kassambara (2017), such as the elbow, average silhoutte (Rousseeuw, 1987), and gap statistic (Tibshirani et al., 2001) methods, aim to determine the optimal value of K.

K-medoids Clustering is a variation of the K-means algorithm that is more robust to noise and outliers in the dataset, as medoids correspond to real data points and are less influenced by extreme values (Aggarwal & Reddy, 2013). Unlike K-means, it selects actual data points as centroids and aims to minimize the absolute error rather than the SSE (Aggarwal & Reddy, 2013, Hastie et al., 2009). K-medoids receives the number of clusters, denoted as K, and the distance matrix of points as input parameters. The other steps of the algorithm are similar to K-means. It is important to note that K-medoids might demand higher computational resources than K-means, particularly when dealing with large datasets, due to the need of computing distances between all data points and medoids during each iteration.

DBSCAN Clustering is classified as density-based partitional clustering method. This widely used clustering algorithm aims to create clusters of data points that exhibit close proximity in a high-dimensional space. DBSCAN is particularly advantageous when dealing with complex shapes and noise, finding its utility in domains such as spatial data analysis and anomaly detection. Unlike K-means and K-medoids, which require a predefined number of clusters, DBSCAN determines the number of clusters automatically based on the underlying data distribution. The algorithm starts by selecting a random data point and counting the number of neighboring data points relative to a core point within a given radius (*Eps*). The points within this radius are considered to be part of the same cluster. If the density of the neighboring points is greater than or equal to the specified minimum number of points (*MinPts*) required to form a dense region, a cluster is formed (Ester et al., 1996). The points that do not belong to any cluster are considered as noise (Aggarwal & Reddy, 2013). The algorithm continues to explore data points iteratively until all points are either assigned to clusters or labeled as noise. Selecting appropriate values for *Eps* and *MinPts* requires domain knowledge or experimentation. Furthermore, DBSCAN is sensitive to the density variations in the dataset, facing challenges when dealing with clusters of varying densities or those separated by less dense regions. This sensitivity can lead to fragmented clusters or failure to identify certain clusters.

Clustering Evaluation Metrics or clustering indices are used to compare different clustering solutions and measure how well the data points are grouped into clusters. These metrics can be classified in two categories: external and internal indices. External indices are used to evaluate how well each cluster matches the ground truth clusters provided by human experts. Some of these measurements include cluster purity, Cluster Separation Measure, Fowlkes-Mallows Index, F-measure, Normalized Mutual Information and Entropy (Zhang et al., 2006, Han et al., 2012). However, internal indices measure the similarity of objects within each cluster using features and cohesion in the dataset. These indices typically consider factors like the compactness of clusters and the separation between clusters within the dataset. Examples of internal indices are SSE, Silhouette Coefficient, RMSE, Dunn Index and Root Mean Square Standard Deviation (Han et al., 2012, Aghabozorgi et al., 2015).

DIMENSIONALITY REDUCTION

Dimensionality reduction techniques are one of the commonly used unsupervised learning methods. They aim to decrease the number of features within a dataset while retaining as much relevant information as possible. This is often used to mitigate the challenges posed by high dimensionality and enhance the efficiency of subsequent analyses. Principal Component Analysis (PCA) (Vidal et al., 2016), t-Distributed Stochastic Neighbor Embedding (t-SNE) (Van der Maaten & Hinton, 2008), and PCC (Pearson, 1896) are widely used methods for achieving dimensionality reduction.

PCA is a commonly used method in unsupervised learning (Saul & Roweis, 2003, Ghodsi, 2006) that effectively reduces the dimensionality of data by transforming it into a new set of orthogonal (uncorrelated) variables known as principal components. PCA serves the purpose of retaining essential data insights while minimizing interference from irrelevant data and reducing computational complexity. PCA can be perceived as a feature extraction method, where the principal components serve as new features for further analysis.

2.3.2 SUPERVISED LEARNING METHODS

The supervised learning methods are algorithms trained on labeled datasets, where each data point in the dataset is associated with the correct target or output. The goal of supervised learning is to learn a mapping from inputs to outputs, enabling accurate predictions or classifications on new, unseen (test) data. In supervised learning, tasks are divided into two main categories determined by the nature of their outputs: classification and regression (Nasteski, 2017, Gupta et al., 2022, Silvestrini & Lavagna, 2022). In classification tasks, the objective is to assign input data points into predefined categories or classes. The output consists of discrete labels denoting the corresponding class for the given input. Tasks such as image recognition and email spam detection are examples of classification tasks. In regression tasks, the focus is on forecasting a continuous numeric output value, representing a quantity or measurement, using input data. Examples of regression tasks include weather forecasting and predicting stock prices.

Regression models are a type of supervised learning, encompassing a range of statistical approaches utilized for forecasting a continuous target variable by considering one or more input features. These techniques serve the purpose of both prediction and uncovering associations among variables. The primary objective of regression analysis is to construct a model that characterizes the relationship between the input features and the target variable, thereby enabling forecasts on new data. For instance, polynomial regression is a common method to estimate and model complex relationships between variables by incorporating higher-order terms in the regression equation. It fits a nonlinear relationship between the value of an independent variable and the expected value of a dependent variable (Stigler, 1974, Magee, 1998).

Support Vector Machines (SVMs) (Cortes & Vapnik, 1995) are used for classification and regression tasks, where the algorithm learns from labeled data. They are commonly employed in binary classification scenarios, with the aim of categorizing data points into one of two classes. Nonetheless, they can also be expanded to address tasks involving the classification of data into multiple classes. SVMs utilize polynomial input transformations to map the input vectors to a high-dimensional feature space. A linear decision surface is created in the feature space to enable the machine learning classification and prediction of data.

Generally, supervised learning involves the following steps:

1. **Data collection** involves gathering a dataset containing input features and their corresponding correct output labels.

2. **Data preprocessing** is the process of cleaning, transforming, and structuring the data to ensure their suitability for training.

3. **Feature extraction** is the selection of relevant features from the input data or extracting new features to effectively represent the data.

4. **Model selection** involves selecting an appropriate machine learning model based on factors, such as data type (e.g., numerical, categorical), problem complexity, and the availability of computing resources.

5. **Training** is the process of feeding the labeled features to the selected model and enabling it to learn the underlying patterns and connections between inputs and outputs.

6. **Validation** evaluates the model's performance using a validation dataset (a subset of the training dataset not used during training). This helps in hyperparameters tuning and preventing overfitting, where the model performs well on the training dataset but falters on an unseen dataset.

7. **Testing** is the final evaluation of the trained model on a separate test dataset (previously unseen by the model). This step provides an estimate of the model's ability to generalize.

8. **Forecasting** is the last step in which the trained model is utilized to forecast or classify new, unseen data.

The selection of a supervised learning method relies on several factors, including the type of data, the complexity of the tasks, and the targeted performance. Commonly used supervised learning algorithms include linear regression, decision trees, support vector machines, and neural networks. Given that the use case data in this study consists of large datasets of time series, and the pursued task is time series forecasting, I conducted an investigation into appropriate supervised learning methods applicable to this scenario. Time series forecasting is the process of predicting future values by leveraging domain knowledge and previously observed data (Hyndman & Athanasopoulos, 2018). The forementioned factors led me to select neural networks as a promising supervised learning method for my use case.

NEURAL NETWORKS

Neural networks are among the most widely used methods in supervised learning, holding the potential to achieve remarkable performance in various applications, including image recognition (Krizhevsky et al., 2012), natural language processing, recommendation systems (Goodfellow et al., 2016), as well as time series analysis and forecasting (Sønderby et al., 2020). Neural networks are inspired by the structure and functioning of the human brain's neural networks, enabling them to identify patterns and correlations within data. A neural network consists of layers of interconnected nodes, referred to as neurons, which process and transform input data to produce output.

Neural networks are constructed from basic components, including an input layer, hidden layers, weights and biases, activation functions and an output layer. Algorithmically, the input layer feeds the raw data into the network, where each node in the input layer corresponds to a feature or attribute of the input data. Then, each neuron in a hidden layer performs a computation involving weighted inputs, followed by the application of an activation function, to model complex relationships within data. Common activation functions include Sigmoid, Tanh, and ReLU. The weights and biases associated with the connections between neurons determine the strength of the links and the influence of a neuron on the neurons in the following layer. The neural network learns these weights and biases from training data, aiming to enhance its performance in a particular task. Finally, the output layer produces the final result or desired prediction. Based on their network type, neural networks are classified into several main categories, such as Feedforward and Recurrent Neural Networks, as well as Transformers. These architectures and their usage in this work are described briefly in the following.

FEEDFORWARD NEURAL NETWORKS (FNNs), also often called multilayer perceptrons (MLPs), stand as a fundamental variant of neural networks (Silvestrini & Lavagna, 2022). FNN arranges the layers sequentially, allowing information to flow in only one direction, starting from the input layer, passing through the hidden layers, and reaching the output layer (Goodfellow et al., 2016). FNNs are powerful models capable of approximating complex functions and learning intricate patterns in data. However, they face challenges in handling sequential and temporal data due to their lack of memory. Convolution Neural Networks (CNNs) are a specialized kind of FNNs that learn and extract relevant features automatically from input data using convolutional kernels or filters (Silvestrini & Lavagna, 2022). CNNs leverage a specialized layer called convolutional layer, which performs a mathematical dot product between the convolution kernel and the layer's input matrix. As the convolution kernel slides along the input matrix of the layer, the convolution operation generates a feature map. This feature map contributes to the input of the next layer. The kernels capture different visual patterns, such as edges, textures, and more complex structures. Following the traversal through several convolutional and pooling (dimensionality reduction) layers, the final feature maps are flattened and fed into fully connected layers. CNNs are specifically designed to analyze structured grid-like data, such as images, videos, and map data.

Recurrent Neural Networks (RNNs) and Long Short-Term Memory networks (LSTMs) are widely used for analyzing sequences and time series data (Agarwal et al., 2021, Van Houdt et al., 2020, Lara-Benitez et al., 2021). RNNs are networks with loops that utilize information learned from previous inputs to generate outputs. LSTM is an extension of RNNs, employing additional gates and cell states to learn long-term dependencies (Hochreiter & Schmidhuber, 1997, Hochreiter, 1991). LSTMs consist of memory cells that contain gates using Sigmoid and Tanh activation functions. These functions change the cell's state and determine which information to retain for forecasts. In the LSTM model, the last hidden state (short-term memory) and cell state (long-term memory) are passed to the next step of the sequence, thereby retaining the information from previously observed data and utilizing it to forecast future data (Hochreiter & Schmidhuber, 1997). LSTMs are particularly appropriate for tasks that involve sequential data, such as natural language processing, speech recognition, and time series forecasting. However, the fundamental sequential nature of RNNs and LSTMs prevent parallel processing within training examples. This limitation becomes critical when dealing with longer sequence lengths, since memory restrictions hinder batching across examples (Vaswani et al., 2017).

Transformers are a modern type of neural networks (Vaswani et al., 2017) that rely entirely on a self-attention mechanism to compute representations of its input and output. In the self-attention mechanism, each word (token) in the input sequence is associated with representations obtained from the token's input embedding. Subsequently, attention scores are calculated for each token in relation to all other tokens present in the sequence. These scores indicate the level of emphasis each token should receive. This mechanism empowers transformers to effectively capture long-term dependencies and contextual information, enabling parallel processing and scaling to large datasets, and requiring less training time compared to RNNS or CNNs. They have achieved superior results on several natural language processing tasks, including machine translation, question answering, and text generation. Although transformers offer utility in time series forecasting, they might not consistently outperform conventional time series forecasting methods, especially for smaller datasets or simpler patterns. Furthermore, discussions regarding the effectiveness of transformers in time series forecasting are ongoing. This uncertainty arises because self-attention, a fundamental aspect of transformers, does not inherently consider sequence order, which is an important feature in time series forecasting. Despite the inclusion of positional and temporal embeddings, existing transformers still face the challenge of preserving temporal information (Zeng et al., 2023).

Selecting the most appropriate neural network architecture is challenging and depends on factors, such as time series characteristics, data volume, seasonal patterns, the complexity of patterns, and the availability of computing resources. It is often a good practice to start the process with straightforward and proven methods and gradually progress to more complex models like transformers, if necessary. Additionally, it is essential to engage in empirical trials of different techniques and compare their performance on a validation dataset to find the best approach for a specific forecasting task. In this study, I employed the LSTM model within the approach introduced in Section 6, that is a reliable and appropriate choice for time series forecasting.

2.4 APPROXIMATION AND OPTIMIZATION OF SIMULATIONS

Simulation refers to the representation of the behavior of a real-world system as it evolves over time. Advances in modeling techniques and the availability of computing resources have enabled scientists to conduct large-scale and high-resolution simulations. However, the execution of such simulations might be expensive in terms of time and resources. Consequently, tasks such as engineering design optimization, design space exploration, sensitivity analysis, conducting hypothetical assessments, and performing ensemble runs become unfeasible due to the need for thousands or even millions of simulation runs. Therefore, there is a need to limit the number of simulations executed when searching for optimal parameters (Amaran et al., 2016). This is where the concept of optimization becomes relevant. Optimization involves modifying the algorithms, parameters, or settings of a simulation to enhance its efficiency or tailor it for a specific purpose.

Simulation optimization is a term used to describe computational techniques or algorithms utilized to optimize stochastic simulations that involve randomness or uncertainty (Amaran et al., 2016). It involves the usage of simulations to identify the best input variable values among all possibilities, without explicitly evaluating each one. Simulation optimization aims to achieve an optimal balance between resource utilization and information acquisition during a simulation process. Simulation optimization methods are categorized into several classes, such as gradient-based search, heuristic and statistical methods, as well as stochastic optimization (Carson & Maria, 1997).

Considering the definition of simulation optimization, the work presented in this thesis does not primarily focus on defining any simulation optimization method. Instead, this research introduces approaches to approximate complex and compute-intensive simulations, thereby optimizing them in terms of their computational complexity. The process of approximating simulations generally involves finding simpler models or techniques capable of delivering similar results as the full simulation, but with reduced computational complexity. Numerous approaches exist for simulation approximation, such as reduced-order models, surrogate models, adaptive mesh refinement (Berger & Colella, 1989), time stepping strategies (Demirel et al., 2015), multiscale modeling, and parallelization (Kiesling, 2005). The choice of an approximation method depends on the specific goals of the simulation, the accuracy required, available computational resources, and the nature of the system being simulated. In the following, I will focus on two widely used categories of simulation approximation approaches: reduced-order models and surrogate models, both of which are used in this thesis.

2.4.1 REDUCED-ORDER MODELS

Reduced-order models are approximated models that use Model Order Reduction (MOR) techniques (Benner & Faßbender, 2013) to reduce the computational complexity of simulations while retaining their necessary functionalities. These techniques significantly reduce the run time and memory usage of large-scale simulations. MOR aims to automatically simplify dynamical models that consist of numerous equations and variables, capturing the essential properties and dominant dynamics of the original model in a more compact approximation (Schilders et al., 2008). The applications of MOR include various fields such as design, simulation, optimization, and real-time dynamic systems. MOR techniques can be categorized into several classes based on different approaches, such as time domain, frequency domain, time-frequency domain, optimization techniques and artificial intelligence, and data-driven methods (Kumar & Ezhilarasi, 2023).

The Proper Orthogonal Decomposition (POD) method, which is conceptually similar to Principal Component Analysis (PCA), is a powerful and effective MOR method in the time domain. Its objective is to extract the most important components from a high-dimensional complex system by using a few proper orthogonal modes (Lu et al., 2021). In this statistical method, the POD reduction function is derived by solving the eigenvectors of the autocorrelation matrix, constructed from snapshot signals obtained through numerical simulations or experimental data from the original system. While POD modes effectively capture dominant features of the data, decoding and interpretation of their physical meaning might be challenging, especially in complex systems. Additionally, POD-based models are constructed based on the available data and might not generalize well to new or unseen scenarios.

While MOR techniques accelerate complex simulations, it is important to consider the trade-off between performance and computational complexity, the necessity of understanding the details of the simulation code, and the limitations in applicability and generalizability of the reduced order models.

2.4.2 SURROGATE MODELS

Another approach to approximate complex and compute-intensive simulations involves the utilization of surrogate models or emulators. These models are constructed using statistical, data-driven or machine learning approaches and serve as efficient replacements for computationally expensive simulations, while providing reasonably accurate predictions of the outputs of the original simulation model. Surrogate models offer a benefit over reduced-order models by not requiring knowledge or comprehension of the precise mechanisms within the simulation code. Instead, their focus lies solely on capturing the input-output relationship. Surrogate models use data generated from the original or high-fidelity simulation models to provide rapid approximations (Queipo et al., 2005). These data consist of input parameters and corresponding output values from the simulation runs. Various techniques can be used to create surrogate models, including polynomial regression, Gaussian process regression, Support Vector Machines (SVMs), decision trees, and neural networks. These models essentially learn the underlying relationships between inputs and outputs of the simulation by utilizing supervised learning methods.

Environmental simulations and forecasting models are usually compute-intensive, particularly when considering ensemble simulations at high spatio-temporal resolution. With the growing capabilities of HPC systems containing GPU resources, there is a potential of benefiting from forecasting methods based on machine learning, especially neural network models (Dueben & Bauer, 2018, Scher, 2018, Rasp & Thuerey, 2021, Van Houdt et al., 2020, Lara-Benitez et al., 2021). Once such a model is trained on an HPC system, it can be reused multiple times for forecasting. Consequently, the overall process is faster and computationally less expensive than simulating physical models. Additionally, there is potential for forecasting at time scales and in locations where physical models perform poorly. However, this is the case when the network is trained with observational data.

Incorporating physical principles, governing laws and domain knowledge into machine learning models, or physics-informed machine learning (Karniadakis et al., 2021), aims to create physically consistent and scientifically sound predictive models. This serves to accelerate the model training process, to improve the generalizability of models in making reliable predictions for unseen scenarios, and to enhance transparency and interpretability, thereby making the models more trustworthy. A widely used method for including physics and domain knowledge in surrogate models is through regularization with custom loss functions. The relative weights of the physics-based losses are adjustable hyperparameters (Kashinath et al., 2021).

The evaluation of surrogate models involves a trade-off between their performance and computational complexity. As these models serve as representations of original simulations, there might be some loss of accuracy in exchange for faster computation. The degree of accuracy depends on factors, such as the choice of surrogate model technique, the size, and quality of the training dataset, and the complexity of the underlying simulation model. The surrogate model's accuracy can deteriorate outside the range of the training data or if the underlying simulation model undergoes significant changes. Regular updates and validations against the original simulation model are necessary to ensure the surrogate model's reliability.

Research is to see what everybody else has seen, and to think what nobody else has thought.

Albert Szent-Györgyi

3 Related Work

Following the presentation of the relevant background in Chapter 2, this chapter features an overview of related scientific work connected to the research conducted in this thesis. The provided overview is not intended to be all-encompassing. The related work about key principles for evaluating environmental models are presented in Section 3.1, presenting the utilized measures for accuracy and efficiency of the models. Section 3.2 provides the related work and approaches using reduced-order models that have been employed for the approximation and optimization of environmental simulations. The related work on the application of supervised learning in the context of surrogate models is presented in Section 3.3. As certain approximation methods leverage the similarities between simulation model units, Section 3.4 introduces the strategies and unsupervised learning methods used for identifying similarities in the simulation structures. The description of the related work provided in this chapter has been partly published in Azmi (2018b) and Azmi et al. (2019, 2020, 2021, 2023)

3.1 MODEL EVALUATION

Evaluation of environmental models is an essential principal step involved in modeling to measure the accuracy, reliability, and efficiency of the models in replicating and forecasting environmental processes and phenomena (Aral, 2010, Bennett et al., 2010). Assessment of a model's performance and its ability to accurately represent the system being studied has resulted in various approaches and debates about the suitability of these techniques. The most suitable approach depends on factors such as the model type, available data, modeling objectives, and, in some cases, a combination of methods might be necessary to achieve the most comprehensive understanding and effective decision support Bennett et al. (2010). Many approaches exist to guide model development, and they differ by the way they handle and the emphasis they put on each of the two key aspects, namely performance and complexity (Schoups et al., 2008), discussed in Section 2.2.

Model selection has also been done by applying complexity penalization measures. Complexity penalization measures, such as the Akaike Information Criterion (Akaike, 1974) or the Bayesian Information Criterion (Schwarz, 1978), are a formalization of the principle of parsimony which can be applied to make models of varying complexity comparable in terms of performance. Many discipline- and purpose-specific variants for complexity penalization exist to build parsimonious models. Hydrological systems have been described and analyzed in terms of their complexity by Jenerette et al. (2012), Jovanovic et al. (2017), Ossola et al. (2015), and Bras (2015). Similarly, hydrological time series complexity was investigated by Engelhardt et al. (2009).

Complexity measures have been used for classification of hydrological systems by Pande & Moayeri (2018), who used the Vapnik-Chervonenkis (VC) dimension (Vapnik & Chervonenkis, 1979) from statistical learning theory (Cherkassky & Mulier, 2007) and machine learning. The VC dimension measures the capacity or complexity of a set of functions or classifiers that can be learned by a statistical binary classification algorithm. This is yet another view on model complexity as its flexibility to classify arbitrary data.

Other complexity-based classification was done by Sivakumar et al. (2007) and Sivakumar & Singh (2012). In this context, many complexity measures have been proposed based on information entropy (Zhou et al., 2012, Castillo et al., 2015), wavelets (Sang et al., 2011), correlation dimension of system output (Sivakumar & Singh, 2012), and dynamic source analysis (Perdigão, 2018, Perdigão et al., 2019).

In hydrological modeling, model complexity most often refers to the number of processes, variables, or parameters a model comprises, and many authors have investigated the relation of model complexity and predictive performance (Gan et al., 1997, Schoups et al., 2008, Arkesteijn & Pande, 2013, Förster et al., 2014, Finger et al., 2015, Orth et al., 2015) and proposed ways to build or select models of minimally adequate complexity or parsimonious models (Atkinson et al., 2002, Sivapalan, 2003, McDonnell et al., 2007, Schöniger et al., 2015, Höge et al., 2018).

Weijs & Ruddell (2020) call Occam's parsimony a "weak parsimony", because it identifies a set of parsimonious models rather than a single most parsimonious model. They further argue that a single, "strongly parsimonious" model could be identified by considering, in addition to model descriptive complexity, also model performance, and to express them in the language of Algorithmic Information Theory (AIT) as two additive terms which together are the description length of the data in bits. Thus, Performance becomes part of parsimony by collapsing them into the single dimension of description length. A strongly parsimonious model in the terms of Weijs & Ruddell (2020) perfectly or losslessly reproduces experimental observations in the smallest number of bits, after adding together the compressed size of the model and the compressed corrections needed to adjust the model's predictions to equal the observations. Such a model balances minimum model size and minimum information loss, and maximum generalizability outside the observed datasets used to construct and test the model. The latter claim is based on insights from AIT, where shorter descriptions have been shown to be more likely to be generalizable. This is expressed through the concept of algorithmic probability, assigning higher prior probability to simple models, and convergence of induction systems based on this formalization was shown in Solomonoff (1978). An alternative perspective on this concept is to utilize all structure within the data, while finding the minimum description length. This was used by Kolmogorov (1968) to define randomness as absence of structure and therefore as incompressibility.

The approach proposed by Weijs & Ruddell (2020), drawing on the minimum description length principle (Rissanen et al., 2007, Grünwald, 2007), not only has the advantage of favoring models with a good trade-off between descriptive complexity and performance. Application of a single measure, expressed in bits, to quantify both of these aspects also offers the advantage of rigor and generality over more contextually defined performance measures (Bennett et al., 2013), such as RMSE, KGE, Nash-Sutcliffe Efficiency (NSE) (Nash & Sutcliffe, 1970). This more generalized strategy helps to guide model preference, especially in automated environments for learning models from data, starting with the widest class of all computable models, and making very few prior assumptions on structure. At the same time, the lack of prior assumptions is also a weakness of this framework in contexts where considerable prior information is available. In hydrology, this is typically the case, therefore, practical application of this framework is still an open challenge.

Moriasi et al. (2007) produced guidelines for systematic model evaluation, including a list of recommended evaluation methods and performance metrics based on a thorough review of relevant literature. For model evaluation, they recommended three quantitative statistics, NSE, percent bias (PBIAS), and the RMSE-Standard deviation Ratio (RSR), in addition to graphical techniques.

Validation set approaches are a standard procedure in hydrological model development. Among a set of competing models, the model is preferred that performs best on the unseen data during model parameter estimation. The fact that model performance is evaluated on a validation set promotes general models, i.e, models that have captured the essential workings of the natural system they represent, and demotes models overfitted to the calibration data, which are likely to be models with unnecessarily high descriptive complexity. It is therefore an implicit form of model complexity control.

My research contributes to the large existing body of complexity studies in environmental science, by expressing the key aspects of computer-based models, namely performance, descriptive complexity and computational complexity in the single general unit, facilitating comprehensive model evaluation and optimization (Chapter 4).

3.2 REDUCED-ORDER MODELS

In the context of reduced-order models, Sun et al. (2020) introduced a method based on Proper Orthogonal Decomposition (POD) to predict flow and heat transfer of oil and water using numerical reservoir simulations. They generated a reduced set of POD basis functions from offline full-order simulations and predicted new physical fields online without directly solving the full-order governing equations. The proposed reduced-order model is dozens of times faster than that of the finite difference method in online calculation speed, with a relative error of less than 1.3% and 1.5% for water saturation and temperature fields, respectively.

In a numerical simulation of wind and pollutants, the calculation of flow in large-scale urban areas is time-consuming. The reduced-order model proposed by Ding & Yang (2021), which is based on POD and radial basis function interpolation, resulted in a 99% reduction in CPU time at the cost of 0.1% information loss, compared to the traditional approach of computational fluid dynamics.

While reduced-order models provide fast and efficient simulations, they might suffer from a high accuracy loss compared to the full-order model or original simulation. Numerical simulations for computational fluid dynamics require millions of degrees of freedom and several days of computing resources. In their work, Lassila et al. (2014) presented challenges and perspectives of model reduction methods for incompressible fluid dynamics, contributing to the understanding of the types of reduced-order models that may be best suited for particular fluid dynamics applications. Difficulties arise when attempting to reduce the model order of unsteady flows, especially when accurately predicting long-term transient behavior and considering more complex features like turbulence or multi-physics phenomena.

A general framework for projection-based model order reduction assisted by deep neural networks is proposed by Daniel et al. (2020). They introduced a concept, in which a suitable local reduced-order model is recommended from a dictionary through deep neural networks. The dictionary of local reduced-order models is constructed through clustering of simplified simulations, enabling the identification of the subspaces in which the solutions evolve for different input tensors. It has been shown that direct clustering of the input space may result in clusters that cannot be exploited to define local reduced-order models. This issue can be circumvented by defining a reduced-order model oriented dissimilarity based on the results of simplified numerical simulations. Online cluster assignment can be performed using a classifier based on deep neural networks to bypass the need for numerical simulations, thereby reducing the computation time by a factor of 60.

Though most of the Model Order Reduction (MOR) techniques capture the essential features and dynamics of the full-order simulations, I introduced a hybrid approach based on MOR and unsupervised machine learning, that preserves representative simulation model units within discrete clusters of similarly functional model units (Chapter 5).

3.3 SURROGATE MODELS

Surrogate models are utilized to provide faster forecasts instead of running compute-intensive simulations. Regression models can approximate the behavior of the original model using its input-output data. In their work, Chang et al. (2010) proposed a method for efficiently training and testing SVM for low-degree polynomial data mappings. They successfully applied the proposed method to a natural language processing application by improving the testing accuracy under some training/testing speed requirements. While polynomial regression can capture complex relationships, high-degree polynomial terms might lead to overfitting so that the models fit the training data extremely well but might not generalize to new data.

Scher (2018) used a deep convolutional neural network to emulate the complete physics and dynamics of the General Circulation Model (GCM), typically employed in weather prediction and climate science. The network learns from the dynamics of the GCM and forecasts the weather for multiple time steps ahead (up to 14 days). The neural network forecast demonstrated a reduced RMSE compared to both the persistence and climatological forecasts when assessing its accuracy against the true state of the GCM. The study indicated that the neural network learns the time evolution and dynamics of a simple GCM principally, but the studies need to be continued for more complex models, including the incorporation of meteorological forcing.

A fully connected multi-layer neural network model for forecasting of the atmosphere's global weather was developed by Dueben & Bauer (2018). They showed that the model outperforms a simple persistence model in terms of forecasting and produces competitive forecasts compared to coarse-resolution ($6^\circ \approx 668$ km) atmosphere models of similar complexity, at least for short lead times. However, the forecasts exhibit instability and deterioration after a few days. According to the authors, a close collaboration is required between computer scientists and meteorologists to include the physical knowledge and deep understanding of the Earth system into the neural network model.

For medium-range weather forecasting, Rasp & Thuerey (2021) defined a data-driven method that uses a deep residual convolutional neural network (Resnet). They trained models to forecast geopotential, temperature, and precipitation at $5.625^\circ \approx 626$ km resolution up to five days ahead. When compared to physical models, Resnet achieves comparable scores to a physical model at a similar resolution. They used a dataset spanning 150 years from the Coupled Model Intercomparison Project (CMIP) (Eyring et al., 2016) to pretrain the model, which was then fine-tuned using the ERA data. ERA (ECMWF Re-Analysis) refers to a series of climate reanalysis datasets produced at the **E**uropean **C**entre for **M**edium-Range **W**eather **F**orecasts. Given that the current CMIP models run at around 100 km resolution, this model cannot be used effectively for forecasts at higher resolutions. The goal of this work is to explore the feasibility of data-driven approaches in weather forecasting.

In order to overcome the high computational costs while attaining comparable quality in their results, Albrecht et al. (2021) presented a fully connected neural network. They used a dataset generated from the global numerical **E**CHAM/**M**ESSy **A**tmospheric **C**hemistry (EMAC) model to make predictions of chemical tendencies. This work showed a proof of concept that neural networks can predict atmospheric chemistry tendencies. However, hyperparameter tuning is required to overcome modeling problems arising from seasonal trends in the data. This challenge is associated with using a neural network model in comparison to the use of physical models.

Schultz et al. (2021) presented a review and discussion of the opportunities provided by deep learning approaches in the field of weather prediction. They focused on the possibility to replace numerical forecast models and presented models that are limited to short-term forecasting of less than 24 hours. The authors acknowledge the potential of using deep learning for weather forecasting, but emphasize the importance of overcoming challenges, particularly in terms of accuracy and reliability, before replacing traditional numerical models. These challenges arise from the characteristics of meteorological data, including varying probability distributions, dynamic behavior on various spatio-temporal scales, rapid feature changes, extreme events, and the complexity of atmospheric interactions.

The execution of current physics-based numerical weather prediction (NWP) at high resolution is compute-intensive. Kurth et al. (2023) presented FourCastNet, a data-driven deep learning Earth system emulator capable of predicting global weather and generating medium-range forecasts, approximately five orders of magnitude faster than traditional NWP simulations, while maintaining high accuracy. They trained the deep learning model on the ERA5 (ERA fifth generation) dataset at the native resolution of $0.25^\circ \approx 25$ km, covering the years $1979-2017$. The input tensor has dimensions $(20 \times 720 \times 1440)$, representing 20 prognostic variables supposed to have the greatest impact on near-surface winds and temperatures. The model produces a six-hour single time step forecast with the same dimensions. FourCastNet scales efficiently only on three HPC systems utilizing the NVIDIA A100 GPUs. FourCastNet produces accurate instantaneous weather predictions for up to a week in advance and enables the generation of extensive ensembles that could significantly improve predictions of rare weather extremes.

The use of customized loss functions in physics-informed machine learning models has been employed by Daw et al. (2022), where they modeled the temperature of water in a lake at varying depths and times. This framework combines physics-based simulations and observational features in a hybrid model, leveraging neural networks to generate predictions. This approach not only reduces errors in the training set but also ensures the consistency of predictions with the known laws of physics in both training and unlabeled data.

Since surrogate models serve as efficient replacements for computationally expensive simulations, I utilized this concept in my method, which replaces a compute-intensive global atmospheric chemistry simulation with an LSTM model to forecast the volume mixing ratio of twelve trace gases (Chapter 6).

3.4 IDENTIFICATION OF SIMILARITIES

In environmental science, especially in hydrology, the identification of similarities plays a key role in detecting patterns in the environment, extracting correlations and improving the forecast of future events (Corzo & Solomatine, 2007, Sawicz et al., 2011, Ali et al., 2012, Ehret et al., 2014, Zehe et al., 2014). To define hydrologic similarity, Wagener et al. (2007) suggested a framework that is both descriptive and predictive. Their metrics for defining hydrologic similarity or dissimilarity between catchments include static characteristics and dynamic responses of catchments to their meteorological forcing. They discussed the demand for a catchment classification system based on the structure and hydro-climatic conditions of catchments, as well as their functional response to the precipitation input. Following this, Sawicz et al. (2011) derived signatures from precipitation, temperature, and streamflow data to apply a Bayesian clustering and to identify groups of similar catchments. In my approach (Chapter 5), I incorporated such similarity features of model units to represent their properties.

Classification and clustering are the most commonly used methods in environmental science for detecting patterns in datasets, making decisions, and extracting the required information using similarity measurements (Türkeş & Tatlı, 2011, Zarnani et al., 2014, Arroyo et al., 2015, Netzel & Stepinski, 2016). These two methods identify groups of similar objects using already labeled data or object neighborhood properties such as distance or density (Murphy, 2012, Kassambara, 2017).

K-means, Clara, HClust and Fuzzy clustering algorithms were studied by Zarnani et al. (2014) to analyze the uncertainty of weather situations. The proposed method led to a decrease in the RMSE of point forecasts by up to 10%. To predict the minimum and maximum weather-based meteorological data, Shobha & Asha (2017) compared the application of the K-means and hierarchical clustering using internal validation measures. Türkeş & Tatlı (2011) used the spectral clustering to determine the coherent precipitation regime regions. They could obtain spatial patterns of the precipitation regions, offering new hydro-climatological insights for a deeper understanding of hydrological systems.

Time series are one of the main types of input data in environmental science (Bărbulescu, 2016), and dealing with these data requires additional preprocessing steps. There are four main steps involved in time series clustering: dimensionality reduction or representation method, distance measurement, clustering algorithm and evaluation (Aghabozorgi et al., 2015). When considering time series as discrete objects, several approaches are applied to do the time series preprocessing for clustering. One such method involves converting time series data to make them compatible with the conventional clustering algorithms. The feature-based method converts time series into equal-length feature vectors, which are then used as input data for the clustering algorithm (Liao, 2005, Hautamaki et al., 2008). This approach serves as a dimensionality reduction technique, aiming to reduce memory requirements and computation complexities (Keogh & Pazzani, 2000). An alternative approach for the transformation of time series data is known as the model-based method. It fits a parametric model to each time series to calculate a model-model distance for clustering (Liao, 2005, Hautamaki et al., 2008). An additional strategy, referred to as the shape-based method, focuses on matching the shape of time series through non-linearly stretching and contracting the time axes (Hautamaki et al., 2008, Aghabozorgi et al., 2015). Feature selection techniques also belong to dimensionality reduction approaches that filter the redundant or irrelevant features. Measures such as correlation (Hall, 1999), mutual information (Battiti, 1994), incremental orthogonal centroid (Yan et al., 2006) and structural similarity (Mitra et al., 2012) select a proper subset of the original features to reduce the dimensionality. After representing time series, one of the clustering methods is applied based on the requirements of the study case. In the last step of the clustering process, the accuracy of the extracted clusters is evaluated, which is a challenging issue without having a labeled data set or ground truth.

Classification methods provide efficient pattern detection and help in a deeper understanding of hydrological systems (Ley et al., 2011, Sawicz et al., 2014). However, they require labeled data to classify unseen data. Due to the lack of labeled data, I used clustering methods to identify similar hydrological model units in the CAOS simulation model (Chapter 5). Thus, I do not go through the classification methods. The selection of an appropriate clustering method depends on various parameters, such as the type of input data and clustering output, scalability and robustness, thus it is use case dependent. Kar et al. (2012) compared the hierarchical clustering fuzzy C-mean (FCM) and K-means to analyze regional flood frequency and the underlying distribution. The results of both clustering methods for their application were nearly identical. Therefore, they concluded that the choice of the best clustering method depends on the specific use case. Netzel & Stepinski (2016) presented a clustering-based classification of climate data that resulted in internally more homogeneous and externally more distinct climate types compared to the types in the rule-based Köppen-Geiger classification, which is the de facto standard in global climate classification. Zhang et al. (2016) successfully defined regions with clear boundaries of homogeneous precipitation regions with highly varied spatio-temporal patterns using K-means on a gridded dataset for automatic delineation.

The greatest enemy of knowledge is not ignorance; it is the illusion of knowledge.

Stephen Hawking

4 Evaluation of Simulations on Performance and Computational Complexity

This chapter addresses Research Question 1, defined in Section 1.1. Namely, I investigate general principles that can be employed to assess the quality and efficiency of simulations, aiming to develop a universally applicable approach for their evaluation. Section 4.1 presents key aspects of model evaluation and provides a guideline for model development. Section 4.2 describes an evaluation approach based on measuring model performance and computational complexity, and its application to various test models in a real-world study area. This is followed by the demonstration of the models' evaluation results and discussions of the key insights derived from the evaluation approach in Section 4.3. The implementation environment of the tests is described in Section 4.4. Finally, Section 4.5 summarizes the chapter. The presented approach in this chapter has been published in Azmi et al. (2021).

4.1 MOTIVATION

The main goal of environmental sciences is to acquire a deeper understanding of the natural world and its governing mechanisms. Scientists investigate phenomena, make predictions, and advance theories that can be validated by comparison to observations (Kerr & Goethel, 2014). Models and simulations, for example, provide tools for understanding, analyzing, and predicting complex systems, while enabling the examination of diverse scenarios and the assessment of potential results of various choices. Their significance becomes especially apparent when addressing situations that involve conducting time-consuming, expensive, or impractical real-world experiments or observations. It is important to ensure that the models representing natural phenomena and systems are both concise and effective in accurately capturing their essence.

These models, on the one hand, should align with observations of real-world systems, performing well regarding the level of accuracy, precision, and overall correctness (Kirchner, 2006). Evaluating such models involves considering their complexity. On the other hand, these models should be concise, elegant, explainable, understandable, communicable, teachable, and compact (Solomonoff, 1964). An ideal model, exemplified by mathematical analytical models like Newton's laws, achieves a balance between performance (exhibiting high accuracy and precision when compared to a wide range of experimental observations) and appropriate complexity (displaying elegance, brevity, and communicability). Efficiency in producing output is another crucial aspect of model complexity, particularly for large models employed in operational settings where computational effort or computation times are significant factors. These essential components of model evaluation are labeled as "performance", "descriptive complexity", and "computational complexity" (Figure 4.1).

Occam's razor offers a guideline to promote models that effectively describe patterns in data and to distil laws that allow effective compression of experimental data. Additionally, it serves as a guideline for inference, which refers to the process of drawing conclusions or making predictions based on the information contained in a set of data or observations. When applying Occam's razor, the parsimonious among the well-performing models are identified. However, for model selection, this principle alone does not allow for comparisons between models of different complexities.

Occam's razor and the AIT-based extension proposed by Weijs & Ruddell (2020) are designed with a focus on inference, specifically distilling small and universal laws from experimental data, while validation set approaches primarily emphasize performance. Neither of these approaches directly considers the effort required by a model to make its predictions. However, this effort can be an important quality of a model in settings where computing resources are limited. In earth science modeling, considering this effort is more the rule than the exception for the following reasons: i) the scales of earth systems cannot be separated easily, and in some cases, not at all, hence it might be necessary to simulate large systems in high spatio-temporal resolution, even for local questions, ii) calibration of model parameters from

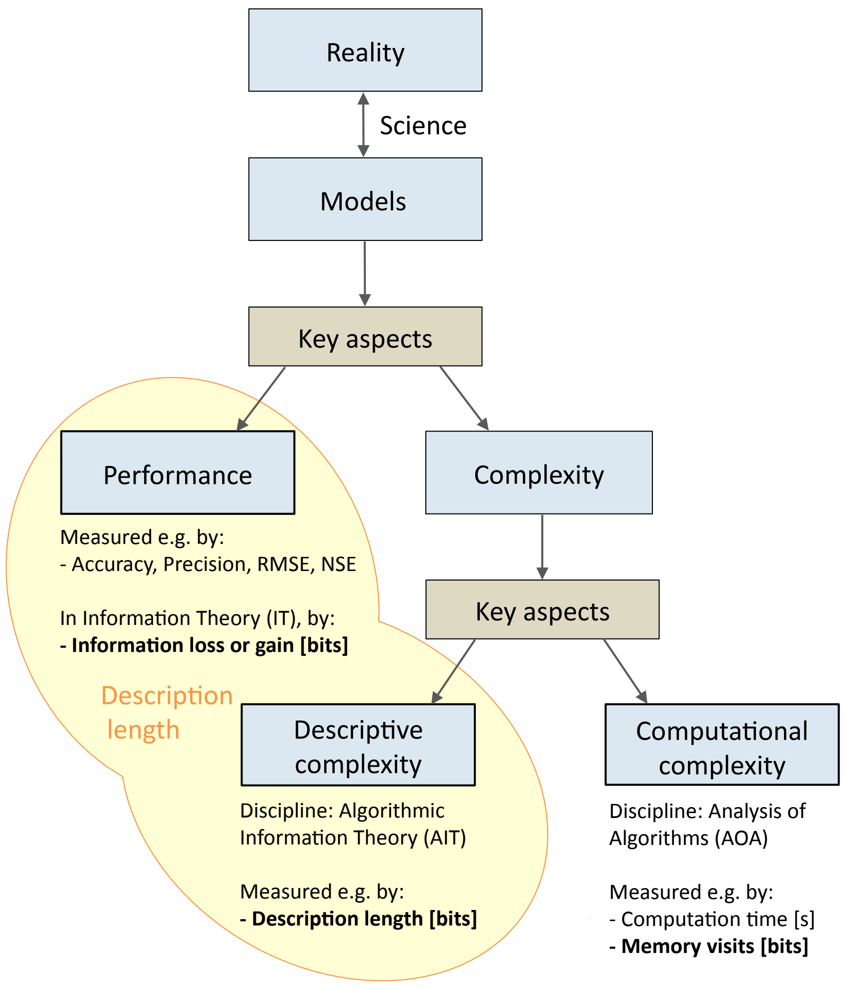


Figure 4.1: Key aspects of model evaluation. In the framework of AIT, the sum of a model's performance and descriptive complexity can be expressed as its description length (yellow area). Modified after Azmi et al. (2021).

data requires many repeated model runs for parameter identification, and iii) models used in optimal decision-making require repeated use to identify the optimal alternative.

4.2 METHODOLOGY

In the context of the guidelines for model development discussed in Section 2.2, this work has two distinct but related goals. The first goal is to propose a practical method for measuring computational complexity by counting the total number of memory visits during the execution of a model. This measurement is sensitive to all aspects of a model, including its size, the input data it processes, its numerical scheme and time stepping, and runtime environment. The second goal is to demonstrate how the measurement of computational complexity can be combined with the performance measurement by information loss relative to observations. This approach allows for the joint evaluation of all key aspects of a model, specifically, descriptive complexity, computational complexity, and performance.

In the proposed method, I employ a validation set approach inspired by Weijs & Ruddell (2020) to quantify model performance in terms of information loss in bits. Therefore, the evaluation approach consists of explicitly evaluating a model in terms of computational complexity and performance, with both metrics quantified in bits. Descriptive complexity is implicitly considered within the framework of the validation set approach. Measuring computational complexity using a monitoring and profiling tool is a general practice, applicable to any model executed on a computer. Likewise, the evaluation approach is universal, as it measures two key aspects of a model within the single unit of bits.

For the demonstration of this method, I run hydrological models of various types (simple and advanced process-based, both approximate and exact restatements of experimental observations, autoregressive, and neural network models), all of which aim to perform the same task of predicting discharge at the outlet of a watershed. Similar to Weijs & Ruddell (2020), I examine possible trade-offs between computational complexity and information loss. It is important to note that the primary purpose of the model comparison here is not to identify the best among the different modeling approaches, but rather, it serves as a demonstration of how the measuring process is sensitive to all facets of a model, and how differences among the models can be explained by their setup, aligning with expectations.

4.2.1 STUDY AREA

The real-world system I seek to represent with the hydrological models is the Dornbirnerach catchment in western Austria (Figure 4.2). This catchment, located upstream of river gauge Hoher Steg (Q_Host), the target of the models' predictions, covers 113 km^2. The catchment's rainfall-runoff dynamics reflect its Alpine setting, including winter snow accumulation, spring snowmelt, high and intensive summer rainfall, and rapid rainfall-runoff response due to the steep terrain. The meteorological dynamics of the system are represented by precipitation observations at a single rain gauge, Ebnit (P_Ebnit), located within the catchment. Time series data of Q_Host and P_Ebnit are available in hourly resolution for ten years (1 January 1996 – 31 December 2005). No additional dynamical or structural data were used

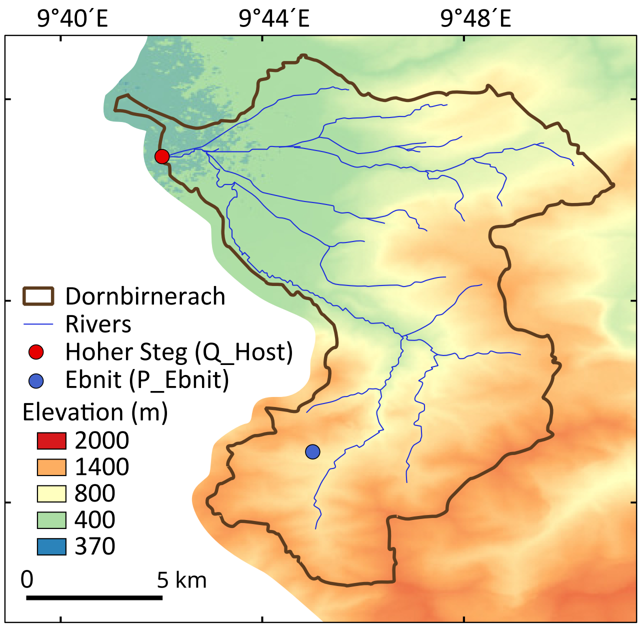


Figure 4.2: Digital elevation model of the Dornbirnerach catchment (study area, brown line). Provided by courtesy of Marcus Strobl.

in the model set-up. Although this dataset might be considered data-scarce for building the best possible hydrological model, it is sufficient for demonstration of the evaluation approach.

4.2.2 EVALUATION APPROACH

In this work, I use the term "complexity" in a specific way to refer to different characteristics of a model. I have adopted the term "descriptive complexity" from AIT to express the parsimony of a model by its size in bits when stored on a computer. Additionally, I use the term "computational complexity" from Analysis of Algorithms to express the efficiency at which a model generates its output by the number of memory visits during program execution. All models are evaluated in terms of the two criteria described in Section 2.2: "performance" referring to the model's ability to reduce predictive uncertainty about the target, and "computational complexity" representing the effort required to make the model generate a prediction about the target. Similar to Weijs & Ruddell (2020), I express both quantities in bits. This approach allows for an investigation into whether direct comparison or combining both quantities in a single measure aids in the interpretation of the values.

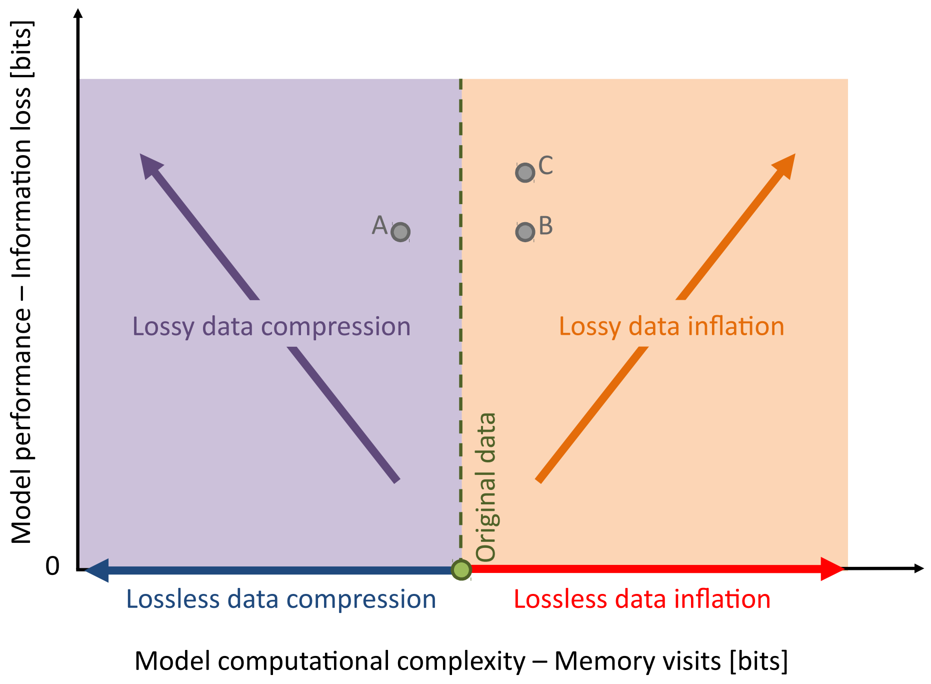

Model performance – Information loss [bits]
0
A
C
B
Lossy data compression
Lossy data inflation
Original data
Lossless data compression
Lossless data inflation
Model computational complexity – Memory visits [bits]

Figure 4.3: Model evaluation approach in terms of performance and computational complexity. The points A, B and C are dummy model measurements. Provided by courtesy of Uwe Ehret.

In the evaluation space (Figure 4.3), different models can be compared based on their positions in the diagram. The horizontal axis represents the computational complexity of the models during their simulation, including the computer memory used to read and process the input data, as well as write the output data. The vertical axis indicates the uncertainty of the model output, which is the amount of missing information needed to obtain output values that match the observations. The maximum information loss corresponds to the loss of the entire original data. As the position of a dummy model measurement moves relative to the original data and shifts horizontally to the left in the diagram, the invested data are compressed. This indicates that the model needs a smaller amount of data than the original data (Figure 4.3). However, data compressibility is limited by its entropy, and data cannot be compressed to zero computational complexity. Therefore, no model can be positioned in the bottom-left corner of the diagram.

To facilitate the understanding of the evaluation approach, three dummy model points, namely *A*, *B* and *C* are included for comparison in this diagram (Figure 4.3). Model *B* outperforms model *C* because it contains more information while requiring the same amount of data or memory visits (computational complexity). The comparison between models A and B indicates that, model A is more efficient than model B because of its lower data demand, while containing the same amount of information (performance).

4.2.3 MODEL PERFORMANCE MEASURE

I express model performance in terms of information loss, as described in Weijs & Ruddell (2020). In information theory, entropy is a measure of the uncertainty about the outcome of a random draw from a distribution before it is revealed. All that is known a priori is the data distribution. If the outcome is known beforehand, then the a priori known data distribution reduces to a Dirac function (Dirac & Fowler, 1927) with $p = 1$ for the outcome and $p = 0$ everywhere else. The entropy of such a distribution, and thus the uncertainty, is zero. In model performance evaluation, one could use this "perfect prediction" case as a benchmark to compare other states of prior knowledge against in terms of added uncertainty (or information loss). In the case described above, where all that is known a priori is the data distribution, the information loss compared to the benchmark case equals the entropy of the distribution. In other cases, one might have useful side information e.g., predictions of a model, which reduces information loss. In such a case, information loss can be quantified by conditional entropy (Equation 4.1), where X represents the targets, Y the model predictions (predictor), x and y are a particular target and prediction, and $H(X|Y)$ is the conditional entropy in bits.

$$H(X|Y) = -\sum_{y \in Y} p(y) \sum_{x \in X} p(x|y)\, log_2\, p(x|y)\,. \tag{4.1}$$

Note that for models providing single-valued (deterministic) predictions, in order to construct a predictive distribution for which an entropy could be calculated, one has to assume that the state of knowledge for each prediction is given by the subset of observations jointly occurring with the particular prediction, i.e., the conditional distribution of X for a particular y. If the models were to provide probabilistic predictions, one could employ a relative entropy measure such as Kullback-Leibler divergence (Kullback & Leibler, 1951, Cover & Thomas, 2006), which would lead to fairer assessments of information loss (Weijs et al., 2010). However, it is worth noting that models providing probabilistic predictions are not yet a standard practice in hydrology.

Alternatively, instead of measuring information losses of model predictions compared to an upper benchmark (observations), as described above, it is also possible to measure information gains compared to a lower benchmark (entropy of a uniform distribution), which expresses the minimum prior knowledge. In the work by Weijs & Ruddell (2020), to which I refer throughout the text, information losses were used because they directly translate to a description length. For the sake of comparability, I applied the same concept here.

To avoid fitting theoretical functions to the empirical data distributions, I calculated the conditional entropy of discrete (binned) distributions, i.e., normalized empirical histograms. The choice of the binning scheme has important implications for the values of the information measures derived from the binned distributions. While the lower bound for entropy ($H = 0$ for a Dirac distribution) is independent of the number of bins n, the upper bound ($H = log_2(n)$ for the maximum-entropy uniform distribution) is a function of n. The choice of n is typically guided by the objective of balancing resolution and sufficiently populated bins, and different strategies that have been proposed by Knuth (2019), Gong et al. (2014) and Pechlivanidis et al. (2016). In this context, several estimators for discrete distributions based on limited samples have been proposed that both converge asymptotically towards the true distribution and at the same time provide uncertainty bounds as a function of sample size and binning choice. For instance, Darscheid et al. (2018) presented both a Bayesian approach and a Maximum-Likelihood approach to estimate discrete distributions from samples.

I applied uniform binning as it introduces minimal prior information (Knuth, 2019), and it is simple and computationally efficient (Ruddell & Kumar, 2009). I uniformly split the value range of ($0 \; - \; 150 \; m^3/s$), which covers all observed and simulated values of Q_Host ($0.05 \; - \; 137 \; m^3/s$) into 150 bins, each with a width of $1 \; m^3/s$. Compared to the typical error associated with discharge measurements in small Alpine rivers, which may be as high as $10 \;\%$, this is an adequate resolution which neither averages away the data-intrinsic variability nor fine grains to resolutions potentially dominated by random errors. When calculated in this manner, a lower bound and two upper benchmarks for the values of conditional entropy can be stated as follows. If the model predicts perfectly the true target value, the conditional entropy will be zero. Non-zero values of conditional entropy precisely quantify the information loss when using an imperfect prediction. If predictor and target are independent, the conditional entropy will be equal to the unconditional entropy of the target, which in this case, is $H(Q\_Host) = 3.46 \; bits$. In the worst-case scenario, where no paired data of target and predictors are available for model calibration, and the only known a priori information is the physically feasible range of the target data, the most honest guess about the target value would be a uniform (maximum entropy) distribution. For the 150 bins I used, the entropy of a uniform distribution is $H_{uniform} = log_2(150) = 7.23 \; bits$.

4.2.4 MODEL COMPUTATIONAL COMPLEXITY MEASURE

I quantify computational complexity by tracking the total number of memory read and write operations (in bits) during the execution of a model. In the context of Information Theory, both these bit counts and the bits representing the model's performance measured through conditional entropy in the previous section, can be interpreted as the number of binary Yes/No questions that were either already asked and answered during the model run (performance) or still need to be asked (computational complexity) in order to fully reproduce the data. Counting memory visits while running a computer program can be conveniently done by using a troubleshooting and monitoring utility such as *Strace* (Kranenburg et al., 2018). *Strace* is a Linux system call tracer to diagnose, debug and trace interactions between processes and the Linux kernel. It is executable alongside codes written in many programming languages such as Python, C++ or R. *Strace* serves as an example tracer tool, and the proposed method can be utilized on any alternative operating system equipped with a compatible system call tracer.

I used *Strace* to monitor the test models written in Python by counting the total number of bits read during the model execution from a file stored in the file system into a buffer, and the total number of bits written from a buffer into a file stored in the file system. These counts reflect the entire effort of the model to produce the desired output. This includes reading input files, writing output files, reading the program itself and all system functions called during its execution, as well as the computations involved in numerical iterations within the program and the efforts to read and write state variables during runtime. Therefore, *Strace* penalizes models that require large amounts of meteorological forcing data, run at high-resolution time stepping or spatial resolution, or employ unnecessarily high-iterative numerical schemes. In short, *Strace* evaluates all memory-related components of a model in the broadest sense.

To evaluate the reproducibility of the measurements, I repeated each model execution 100 times, clearing the memory cache between individual runs. Since the measurements of these executions were similar to each other, I calculated the average of all runs as a single value representing the model's computational complexity. In the application of *Strace*, I traced the "read()" and "write()" system calls of the models while executing their code in Python and wrote them into a log file running the following command in the Linux command line:

```
strace -o target.log -e trace=read python model.py
```

In this command, *Strace* is the system call tracer, "-o target. log" is the option to set the log file path, "-e trace= read" traces the "read()" system calls that returns the number of bits read from the required files during the model execution into the system buffer, "python" is the path to the Python interpreter and "model.py" is the path to the model code. Additionally, I used "-e trace= write" to trace the "write()" system calls that returns the number of bits written from the system buffer into the output file. The total number of bits, representing the computational complexity of the model, is obtained by summing all the read and write operations collected in the target log file.

4.2.5 MODELS' SETUP

I selected altogether eight modeling approaches with the aim of covering a wide range of model characteristics such as type (ignorant, perfect, conceptual-hydrological and data-driven), structure (single and double linear reservoir), numerical scheme (explicit and iterative) and precision (double and integer). The models are listed and described in Table 4.1, with additional information provided in Figure 4.4. I trained/calibrated each model on a five-year calibration period (1 January 1996 – 31 December 2000) and validated them over the subsequent five-year validation period (1 January 2001 – 31 December 2005).

Table 4.1: Models used in the study and their characteristics. Time stepping of the models is one hour, except for Model-02b, which is one minute. Variable precision of the models is double, except for Model-02c, which is integer. Modified after Azmi et al. (2021).

Name	Description	Scheme	Training	Input
Model-00	An (almost) ignorant model, which predicts for each time step the observed time series mean (4.86 m^3/s).	–	Q_Host	–
Model-01	A perfect model representing full prior knowledge contained in the experimental observations. For each time step, the observed value of Q_Host is read as input and provided as output.	–	Q_Host	Q_Host(t)

Name	**Description**	**Scheme**	**Training**	**Input**
Model-02	A simple conceptual hydrological model (Figure 4.3(a)), representing the catchment's rainfall-runoff by a single linear reservoir (single-bucket) with a constant $K = 64\ h$ found by calibration, and a state variable S.	explicit	Q_Host P_Ebnit	P_Ebnit (t)
Model-02a	Same as Model-02, but $K = 120\ h$ is an uncalibrated initial value.	explicit	Q_Host P_Ebnit	P_Ebnit (t)
Model-02b	Same as Model-02, but time stepping is one minute.	explicit	Q_Host P_Ebnit	P_Ebnit(t)
Model-02c	Same as Model-02, but all variables are in integer precision.	explicit	Q_Host P_Ebnit	int(P_Ebnit(t))
Model-02d	Same as Model-02, but the numerical scheme is iterative.	iterative	Q_Host P_Ebnit	P_Ebnit(t)
Model-03	A more advanced conceptual model (Figure 4.3(b)). Precipitation input is split by an intensity threshold $T = 3.5\ mm/h$, and enters into double linear reservoirs (double-bucket) with $K_1 = 10\ h$ and $K_2 = 80\ h$ found by calibration.	explicit	Q_Host P_Ebnit	P_Ebnit(t)
Model-04	An LSTM model with a single hidden layer of five neurons and rolling window of size 20 along the time axis, using P_Ebnit(t) to predict Q_Host(t). In the learning, it uses the "Adam" optimizer.	–	Q_Host P_Ebnit	P_Ebnit(t)

Name	Description	Scheme	Training	Input
Model-05	A simple third-order autoregressive model, which predicts Q_Host(t) by a linear combination of previous observations in the form $Q(t) = c_0 + c_1Q(t-1) + c_2Q(t-2) + c_3Q(t-3)$. All coefficients were discovered by calibration ($c_0 = 0.0536, c_1 = 1.9916, c_2 = -1.3130, c_3 = 0.3104$). Testing models of various order, I found that adding observations beyond lag-3 improved predictive power only marginally.	–	Q_Host	Q_Host(t-3) Q_Host(t-2) Q_Host(t-1)
Model-06	An Artificial Neuronal Network (ANN) model with a single hidden layer of five neurons, using Q_Host(t-1, t-2, t-3) to predict Q_Host(t). In the learning, it uses the "RMSprop" optimizer.	–	Q_Host	Q_Host(t-3) Q_Host(t-2) Q_Host(t-1)

4.3 RESULTS AND DISCUSSION

As mentioned earlier, the primary purpose of the model comparison presented here it is not to identify the best among a set of competing models for a particular purpose. Instead, it serves as a demonstration and proof of concept, illustrating how sensitive a system call tracer is to various facets of a model. Furthermore, it demonstrates how applicable the evaluation approach is to a wide range of modeling approaches, and how it might be used to guide model optimization and model selection. This presentation is carried out across six use cases described in Section 4.3.2, each representing different steps along the iterative process of model building and evaluation, as described in Gupta et al. (2008).

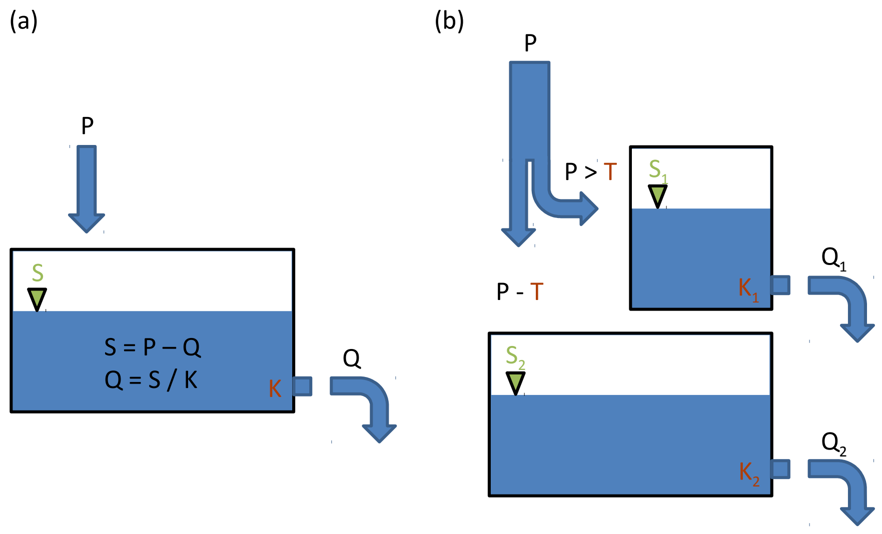


Figure 4.4: (a) Schematic illustration of Model-02, a single-bucket model with state variable *S* and constant *K* found by calibration. The reservoir is replenished by precipitation *P* and drained by discharge *Q*. (b) Schematic illustration of Model-03, a double-bucket model. Precipitation input is split by intensity threshold *T*. Taken from Azmi et al. (2021).

4.3.1 SIMULATION VS. EXPERIMENTAL OBSERVATION

In this section, I provide a short and exemplary visualization of the model predictions to illustrate their general behavior. The observed precipitation (P_Ebnit) is shown in Figure 4.5(a), and the observed and simulated discharge time series of all models (Q_Host) are depicted in Figure 4.5(b). These data correspond to a rainfall-runoff event in June 2002, which falls within the validation period. The observed hydrograph (Figure 4.5(b) Observation) shows a flood peak of 71 m^3/s due to a 14-hour rainfall event. The ignorant Model-00 (orange) is unable to reproduce these dynamics and remains at its constant mean value prediction. As expected, Model-01 (olive) perfectly matches the observations, and likewise, the autoregressive Model-05 (green dashdotted) and the ANN Model-06 (red) show almost perfect agreement. The single-bucket Model-02 (purple) reproduces the observed rise and decline of discharge overall but fails in the details. The rise is too slow and too small, as is the decline. Apparently, a single-bucket model cannot adequately represent the catchment's hydrological behavior, irrespective of the time stepping and the numerical scheme. Discharge simulations by the high-resolution Model-02b (pink dashed) and the iterative Model-02d (pink dotted) are almost

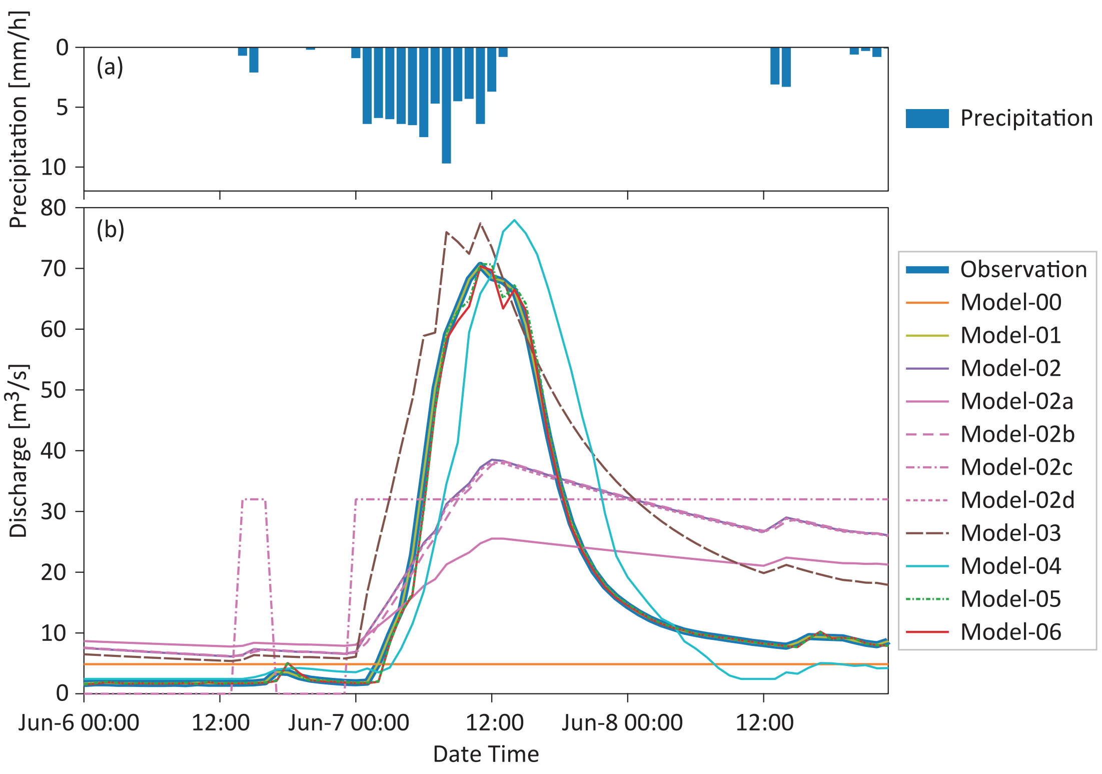


Figure 4.5: (a) Observed precipitation (P_Ebnit). (b) Observed discharge (Q_Host) and simulations thereof by Model-00 to Model-06 for a rainfall-runoff event in June 2002. Modified after Azmi et al. (2021).

identical to that of Model-02. Model calibration and data precision, however, do play a role. The uncalibrated Model-02a (pink solid) shows clearly worse performance than its calibrated counterpart Model-02. The same applies to Model-02c (pink dashdotted), which is identical to Model-02, except for a switch from double to integer precision for all variables. For Model-02c, the hydrograph is only coarsely reproduced by a two-step series. Among all the bucket models, the double-bucket Model-03 (brown dashed) performs best, correctly reproducing the overall course of the event. The LSTM Model-04 (cyan) also provides a good representation of the event's rise, recession, and peak discharge magnitude but shows a delayed response with a lag of about three hours.

4.3.2 PERFORMANCE VS. COMPUTATIONAL COMPLEXITY

As described in Section 4.2.3, model performance is expressed as the remaining uncertainty, at each time step, about the observed data D given the related model simulation M by con-

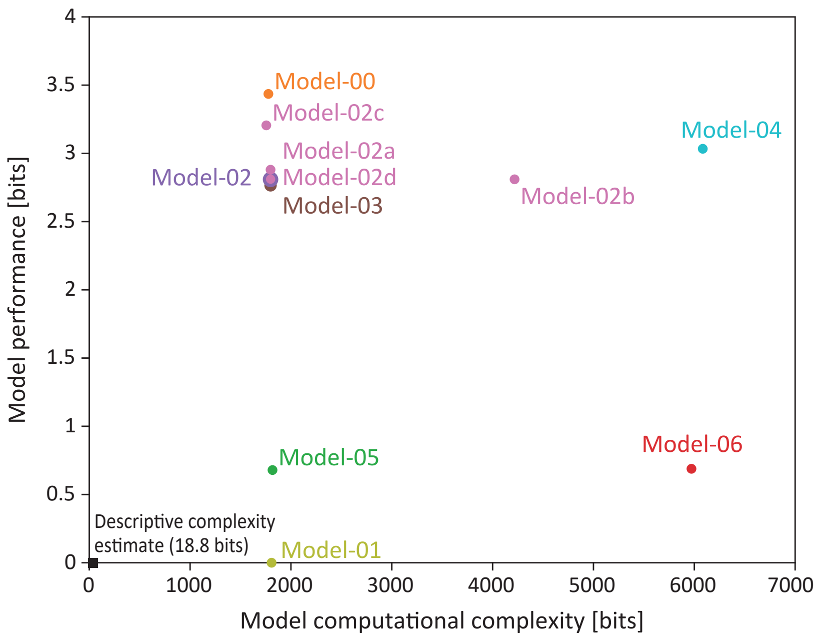


Figure 4.6: Model performance vs. model computational complexity. Model performance is expressed by its inverse, information loss per time step, measured by conditional entropy in bits. Model computational complexity is measured by the average number of memory visits per time step in bits. Modified after Azmi et al. (2021).

ditional entropy $H(D|M)$. Model computational complexity is expressed as the total number of memory read and write operations during model execution, as counted by *Strace*. For easier interpretation, I have shown the average computational complexity per time step by dividing the total number of visits by the length of the validation period (43802 time steps). Figure 4.6 shows the computational complexity and performance of all models. The theoretical optimum, which represents zero information loss despite zero modeling effort, lies in the lower-left corner. There, the black square indicates a loose upper bound on the descriptive complexity of a single recording of the target discharge series (Q_Host). The value of 18.8 *bits* was calculated by dividing the size of the Q_Host validation dataset by the number of time steps, representing the raw size of a single data point in the series without any compression. In a use case, one can compare it to the computational effort of generating a single data point by any of the models. Clearly, the descriptive complexity is much smaller than the computational complexity. In the following, I will discuss six use cases for hydrologically meaningful interpretation of model comparisons.

Use Case 1: the comparison of the single-bucket Model-02 to benchmark Model-00 and Model-01, providing a perspective on the range of possible performance results. In terms of computational complexity, the models differ only slightly (Model-00: 1776 *bits*, Model-02: 1797 *bits*, and Model-01: 1808 *bits*). The main difference lies in model performance. As expected, Model-01, which accurately reproduces the observations, exhibits perfect model performance with zero information loss. The mean Model-00 shows the worst performance among all the models. Taken together, these two models provide a reference against which the performance of other models can be evaluated. For instance, the single-bucket Model-02 (standard model) demonstrates better performance compared to the mean Model-00, but it still falls short of perfection.

Use Case 2: the comparison of two versions of the single-bucket model. Model-02 is calibrated with $K = 64\ h$, where Model-02a is uncalibrated initiating with a reasonable default $K = 120\ h$ (Table 4.1). This use case corresponds to a situation during model calibration, where the conceptual model is fixed, and optimal parameters are determined by parameter variation. Both models are equal in terms of computational complexity, but their performance difference (2.8 *bits* for the former and 2.88 *bits* for the latter) reveals the benefit of calibration. This shows that model performance, expressed by information loss, can be used as an objective function during model calibration in a validation set approach.

Use Case 3: the comparison of the single-bucket Model-02 to its variations in terms of time stepping (Model-02b), variable precision (Model-02c), and numerical scheme (Model-02d). This use case corresponds to a situation where an adequate numerical model for a given conceptual and symbolic model is sought. Increasing the temporal resolution (Model-02b) only leads to higher computational complexity (from 1797 *bits* to 4217 *bits*) without any improvement in performance. Obviously, for the given system and data, hourly time stepping is sufficient. The precision of variables is crucial for performance. Model-02c, which uses integer precision variables, performs noticeably worse than Model-02 and even underperforms the uncalibrated Model-02a. The corresponding computational complexities 1755 *bits* for Model-02c differs slightly from 1797 *bits* for Model-02. Despite the expectations, implementation of an iterative numerical scheme (Model-02d) has almost no effect on both performance and computational complexity. Investigation of the iterative model during runtime revealed that, for hourly time stepping, results were usually stable on the first try, such that on average, only 1.8 iterations per time step were required, increasing computational complexity slightly from

1797 *bits* (Model-02) to 1798 *bits* (Model-02d). The reason lies in the pronounced autocorrelation of the hydrological system response, such that in just a few cases, mainly at the onset of floods, iterations were actually needed to satisfy the chosen iteration precision limit of 0.001. For the models included in this use case, the effect of the numerical solver on computational complexity is small. However, for other systems and models, this can be more significant. Overall, this use case demonstrates how different numerical implementations of the same model can be evaluated with the evaluation approach, which can be helpful when both performance and computational effort of a model are crucial, such as in the context of global-scale and high-resolution weather models.

USE CASE 4: the comparison of the single-bucket Model-02 to the more advanced double-bucket Model-03. This corresponds to a scenario where a modeler compares competing process hypotheses formulated within the same modeling approach (conceptual hydrological models). The double-bucket model performs slightly better (2.76 *bits* instead of 2.80 *bits*), at the cost of an increase in computational complexity from 1797 *bits* to 1798 *bits*. Given this minimal additional computational cost, users would likely prefer the conceptually advanced Model-03 in this context.

USE CASE 5: the comparison of two competing modeling approaches, the single-bucket Model-02 and the LSTM Model-04. Both models take the same input (precipitation) to assure comparability. Interestingly, the bucket model not only outperforms the LSTM model (2.80 *bits* instead of 3.03 *bits*), but also is significantly more efficient. The computational complexity of the LSTM model is almost three times higher than that of the bucket model (1797 *bits* instead of 6083 *bits*). In this context, the obvious choice for a modeler is the bucket Model-02.

USE CASE 6: the comparison of two competing modeling approaches, the autoregressive Model-05 and the ANN Model-06. Both models use previously observed discharge as input and effectively leverage the high information content in this data, resulting in performance measurements of 0.67 *bits* for Model-05 and 0.68 *bits* for Model-06. Their performance is significantly better than those of all other models, except for the perfect Model-01. However, they differ in computational complexity. The autoregressive Model-05 only requires 1817 *bits*, while the ANN Model-06 uses 5971 *bits*. As the autoregressive Model-05 performs slightly better and operates more efficiently, modelers would likely prefer it over the ANN Model-06.

4.4 IMPLEMENTATION ENVIRONMENT

All the test models have been executed using Python 3.6 with packages, including Numpy, Pandas, Scipy, Keras and H5py. The experiments were conducted on a 16-core Intel(R) Xeon(R) CPU E5-2640 v2 @ 2.00 GHz processor with Red Hat Enterprise Linux Server release 7.4. All the scripts and relevant data for the test models are available under the MIT license at `https://github.com/KIT-HYD/model-evaluation` (Azmi & Ehret, 2021).

4.5 SUMMARY

This chapter started by stating one of the main objectives of the environmental sciences, i.e., the development of well-performing yet parsimonious models for natural phenomena and systems. For evaluation of these models, three key aspects need to be considered: descriptive complexity, computational complexity, and performance. Further, I described several paradigms to guide model development. Occam's razor emphasizes descriptive complexity, considers performance as a threshold filter, but it ignores computational complexity. Weijs & Ruddell (2020) express both model performance and descriptive complexity in bits and, by adding the two, obtain a single measure for what they call "strong parsimony". Validation set approaches focus on performance, and indirectly promote general and parsimonious models by evaluating models on unseen data during calibration. Neither of these approaches directly incorporates computational complexity. I suggested closing this gap by applying a system call tracer like *Strace*, which measures computational complexity by the total number of memory visits during the execution of a model. Furthermore, I proposed an evaluation approach that combines measuring computational complexity by *Strace*, and measuring model performance by information loss relative to observations, all in bits, similar to Weijs & Ruddell (2020).

For a proof of concept, I applied the evaluation method in combination with a validation set approach to consider descriptive complexity indirectly. I used various watershed models, including simple and advanced process-based models with various numerical schemes, as well as autoregressive and neural network models. Among the models tested, a third-order autoregressive model demonstrated the best trade-off between computational complexity and performance, while the LSTM and a conceptual model operating in high temporal resolution showed very high computational complexity. For all models, computational complexity exceeded the model performance by approximately three orders of magnitude. Additionally, I compared a simple upper bound of descriptive complexity of the target data set to model computational complexity and found that the latter was approximately two orders of magnitude higher.

In addition to the mentioned results, the proposed approach has the following main outcomes: i) measuring computational complexity using a system call tracer is a general technique that can be applied to any model executed on a computer, ii) this approach is sensitive to all aspects of a model, including its size, the input data it processes, its numerical scheme and time stepping, and iii) the evaluation approach is general in the sense that it measures two key aspects of a model in a single unit of bits, allowing them to be used together for guiding model analysis and optimization in a Pareto trade-off manner, particularly in the context of incremental learning. It can be useful, especially in operational settings where the speed of information processing is critical.

Unlike approaches that estimate computational complexity based on model execution time, measuring memory visits using a system call tracer remains unaffected by other ongoing processes that compete for CPU time. In general, this approach increases the reproducibility and clarity of the results. However, different operating systems may have slightly different behaviors or parameter requirements to handle the system calls, hence the behavior of the system calls might vary marginally. The evaluation approach can help to promote better model code in two ways: computational complexity highlights inefficient coding, while performance reveals issues with erroneous coding. This is particularly relevant as computer models in the earth sciences have become increasingly complex recently. Thus, efficient, modular, and error-free code is a prerequisite for further progress (Hutton et al., 2016).

Although descriptive and computational complexity describe distinct characteristics of a model, they are interconnected, as the evaluation approach quantifies both the size of a program and the computational effort required to execute it. Similar to performance measured by information loss, the descriptive complexity of a model is typically orders of magnitude smaller than its computational complexity. This makes a simple additive combination of the two into a single, overall measure of model quality impractical. Nevertheless, I propose that combining the approach outlined by Weijs & Ruddell (2020) with the measurement of computational complexity using a system call tracer is worthwhile for future exploration. It has the potential to offer a comprehensive and multi-faceted method for model evaluation applicable across the earth sciences, where all key aspects of a model can be expressed in a single unit of bits.

You don't understand anything until you learn it more than one way.

Marvin Minsky

5 Approximation of Simulations using Unsupervised Machine Learning Methods

This chapter mainly addresses Research Questions 2, 3 and 4, defined in Section 1.1, namely the investigation of the structure of environmental simulation models, the exploitation of the potential similarities in functionality of simulation model units to reduce the computational complexity of the simulation utilizing unsupervised machine learning methods, and the establishment of criteria for identifying appropriate compute-intensive simulations where the developed approximation approach can be effectively utilized. Section 5.1 presents the analysis of different unsupervised machine learning methods applied to a use case simulation and their evaluation using various test cases. Section 5.2 describes the utilization of the best proposed approach from the previous section in the same use case, and the development of an evolutionary approach to approximate the simulation. Section 5.3 demonstrates the generalizability of the unsupervised machine learning approach and the necessary criteria for simulations to be met in order to be approximated using this approach. The contributions presented in this chapter include a study of the balance between the uncertainty of the simulation output of the approximated model and its computational complexity. The presented methods and their evaluation have been published in Azmi (2018b) and Azmi et al. (2019, 2020).

5.1 ANALYSIS

Physics-based and highly detailed models of hydrological processes are employed to enhance our understanding of the nature of hydrological systems (Schulz et al., 2006). These models are spatially heterogeneous and consist of a hierarchy of units (Grayson & Blöschl, 2001, Zehe et al., 2014). Thus, the simulation of these models at a high spatio-temporal resolution is compute-intensive. One of the most widely used approaches to address this challenge involves leveraging HPC and parallel processing of hydrological model units (Jones & Woodward, 2001, Kollet et al., 2010, Maxwell et al., 2015). However, parallelization of these models is challenging because of their heterogeneous nature and a demand for partially sequential execution of the model units. The tight interconnections between the model units and the necessary communication of the units per time step complicate the efficient process of parallelization. Furthermore, parallelization may require significant design modifications or code reprogramming in a modern language, demanding specialized software, advanced computational resources and programming expertise from domain scientists.

In the following, I provide a simple parallelization approach as a reference to speed up a sequential hydrological simulation, namely CAOS (Section 2.1.1). This approach involves the parallelization of independent model units (HSLs) and the execution of the simulation on multicore processors. The simulation is conducted for the Wollefsbach catchment (Figure 2.3), over the course of one month (January 2014). Temporal resolution of the HSLs' outputs (discharge) was set to five minutes. To implement this approach, I distributed the execution of HSLs on multicore processors. Porting the sequential code to a parallel version, which only executes independent steps in parallel, requires less implementation effort than introducing communication between computational parts, such as using the Message Passing Interface. Since the original simulation code was implemented by domain scientists in MATLAB, I used its *parfor*[1] functionality for parallelization. The simulation is initially executed sequentially without any parallelization on a computing system (Section 5.1.2), with an average execution time of 50.6 *hours*. The average execution time of the parallel simulation, under the same computing environment using all 16 cores of the processor, was 5.4 *hours*. This exhibits a 9.4 times speed-up compared to the sequential execution, leveraging more computing resources. I will not delve into the details of the aforementioned parallelization approach, which might require significant HPC resources. In the remaining of this chapter,

[1]https://de.mathworks.com/help/parallel-computing/parfor.html

I will present the development and evaluation of a computationally efficient approximation of simulations, which is applicable on a single core processor.

5.1.1 METHODOLOGY

The main objective of this study is to develop an effective and computationally efficient approximation of environmental simulations, thereby reducing the utilization of computing resources. To accomplish this goal, I introduce an approach, referred to as the evolutionary approach. It can be categorized as Model Order Reduction techniques (Section 2.4.1). These techniques aim to reduce the computational costs by dimensionality reduction and computation of an approximation of the original model.

I leverage hydrological similarity (Ehret et al., 2014) to reduce the model complexity and computational efforts. Principally, the similarities in simulation structure and dynamics lead to redundancies stemming from the natural behavior of the model units and the simplifications resulting from the model choice. The underlying concept of this approach is that similar model units function similarly when they possess similar static properties and current states, being exposed to similar meteorological forcing. Throughout the remainder of this chapter, the term "forcing" will refer to the meteorological forcing of the use case simulations.

To investigate the underlying concept of the evolutionary approach, I apply clustering algorithms to create clusters of functionally similar model units (HSLs in this use case). Subsequently, I select only one representative from each cluster, and demonstrate how the uncertainty of the approximated model can be controlled by adjusting the number of clusters while considering the corresponding computation time. In the following, I will provide a detailed explanation of how this concept is utilized in the evolutionary approach.

PREPROCESSING

The first step of this approach is data preparation for clustering. These data should accurately represent the properties and functionalities of the HSLs. The configuration and initialization of the CAOS simulation consists of various input data, constants, and initial states of variables. These settings describe different types of model units' static properties and the corresponding applied forcing. To define a representation of the static properties of the HSLs, such as structure, size, slope, soil profile and drainage, I conducted a drainage test and generated discharge time series for each HSL (Figure 5.1). These time series represent hydrological characteristics that provide an insight into the functionality of the HSLs.

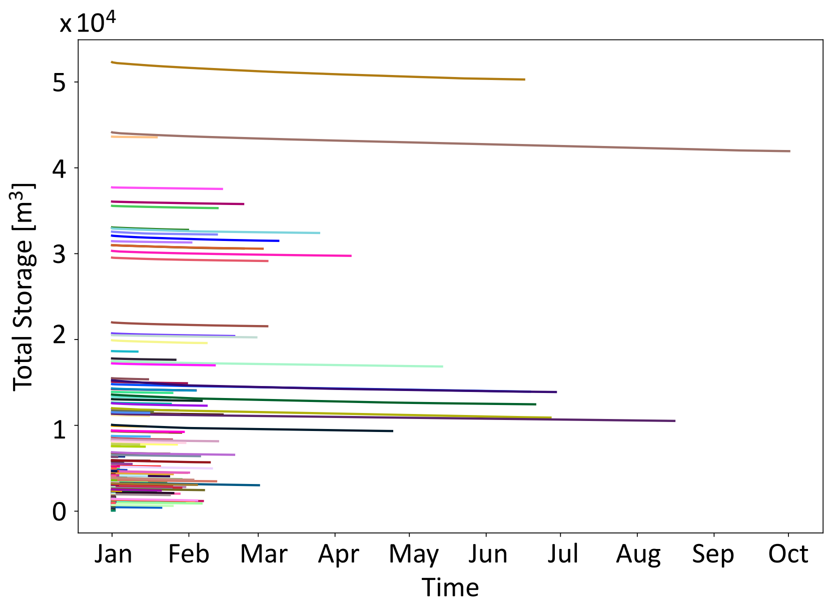


Figure 5.1: Time series of Total Storage (water volume) of HSLs, obtained from drainage test. The time stepping is five minutes. Each line represents a single HSL, and its color enhances the distinction. Taken from Azmi et al. (2019).

In the drainage test, the simulation model is executed for all the HSLs initiated with full storage of water (maximum capacity) and drained over time. During the test, no forcing is applied. The test is conducted on the Wollefsbach catchment (Figure 2.3), starting from an arbitrary time (January) and continuing until the drainage of all the HSLs reaches their equilibrium. This time duration is referred as *Time to Equilibrium (TE)* of the HSLs. The time series of Total Storage obtained from the drainage test (Figure 5.1) serve as integral signatures of HSL size, slope, soil (HSL structure), and drainage properties, which are further expressed by two key features: *TE* and *Active Storage (AS)*. The second feature (*AS*) is extracted from the Active Storage time series, which are calculated from the Total Storage time series. Active Storage time series (Figure 5.2) represent the accumulated volume of water flowing out of an HSL at each time step (*initial Total Storage – current Total Storage*), normalized to the initial Total Storage of that HSL. *AS* is the Active Storage at the equilibrium time step, displayed as the last data point of each line in Figure 5.2.

Feature Extraction From the Active Storage time series (Figure 5.2), I extracted a total of seven features as input data for the clustering approach. The three hydrologically meaningful features are *TE*, *AS*, and the gradient of the first time step of the time series, referred to as *1st Gradient*. The third feature is valuable as the speed of drainage, especially at the first steps

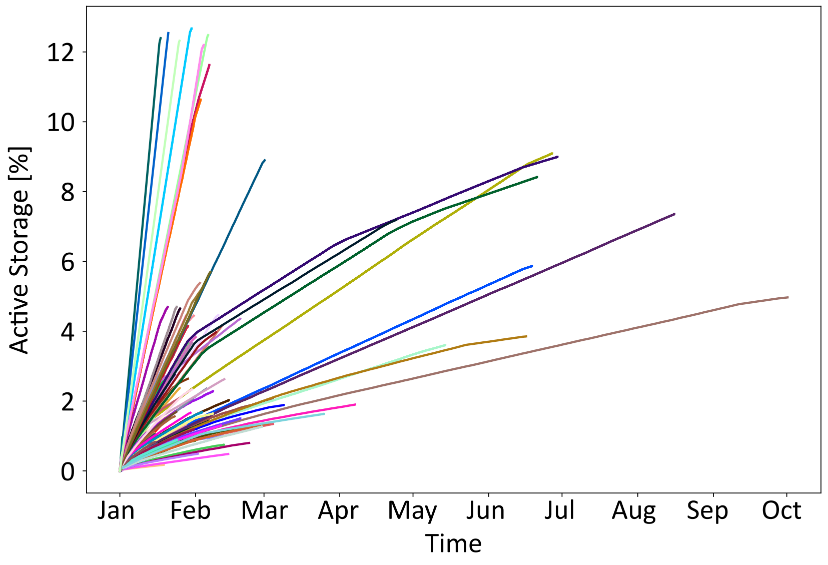


Figure 5.2: Time series of Active Storage of HSLs, calculated from the Total Storage time series. The time stepping is five minutes. Each line represents a single HSL, and its color enhances the distinction. Taken from Azmi et al. (2019).

of the test, characterizes the HSLs. The other features describing the Active Storage time series are the mathematical moments that express the shape of the distribution. Specifically, I extracted four moments: *Mean*, *Variance*, *Skewness* and *Kurtosis*.

Feature Scaling Following the extraction of features, all values of the features are transformed to a consistent scale within the dataset to ensure that all features contribute equally to the model. This is a necessary preprocessing step when working with datasets that contain features of varying ranges, units of measurement, or orders of magnitudes. For this feature set, I use the standardization scaling method (Equation 5.1), where the values are centered around the mean with a unit standard deviation, formally expressed as:

$$X' = \frac{X - \mu}{\sigma}\,, \tag{5.1}$$

where X' is the standardized feature set, μ is the mean and σ is the standard deviation of the feature set X.

Dimensionality Reduction An important method for reducing the computational complexity of clustering is dimensionality reduction of the feature set. Therefore, I filter out the highly correlated features, in other words, redundant ones since they carry similar infor-

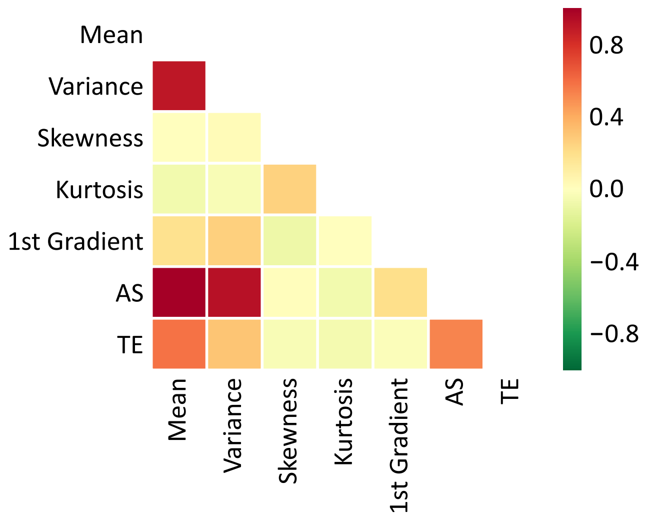


Figure 5.3: *PCC* values for each pair of the features in the feature set. Taken from Azmi et al. (2019).

mation. To perform this, the *PCC* (Section 2.2.1) was calculated for each pair of the extracted features (Figure 5.3). The results indicate that the extracted features are mostly non or positively correlated, and there are no pairs of features with a highly negative correlation. Since *AS* and *TE* are the hydrological key features, I filter out the features that are highly correlated with these two with a filtering threshold of 0.95. Regarding Figure 5.3, *Mean* and *Variance* fall into the exclusion criteria. Therefore, the final feature set consists of five dimensions: *Skewness*, *Kurtosis*, *1st Gradient*, *AS* and *TE*.

CLUSTERING

Following the generation and preprocessing of the feature set as input data, I apply conventional clustering methods (Section 2.3.1 under Clustering), namely K-means, K-medoids and DBSCAN to these input data and evaluate the methods using various test cases.

The only required parameter for the K-means and K-medoids clustering is the number of clusters (K). Conventional methods, such as *elbow* (Satopaa et al., 2011), and *silhouette* (Zaki et al., 2014) methods are used to determine an appropriate value for K based solely on the distribution of the data points to be clustered. However, considering an additional constraint led me to introduce another approach, referred to as *RMSE-Computation-Time (rmse-ctime)* method (Azmi et al., 2019), for determining an appropriate value for K.

In this work, I use the term "K-determiner" to refer to the process of selecting an appropri-

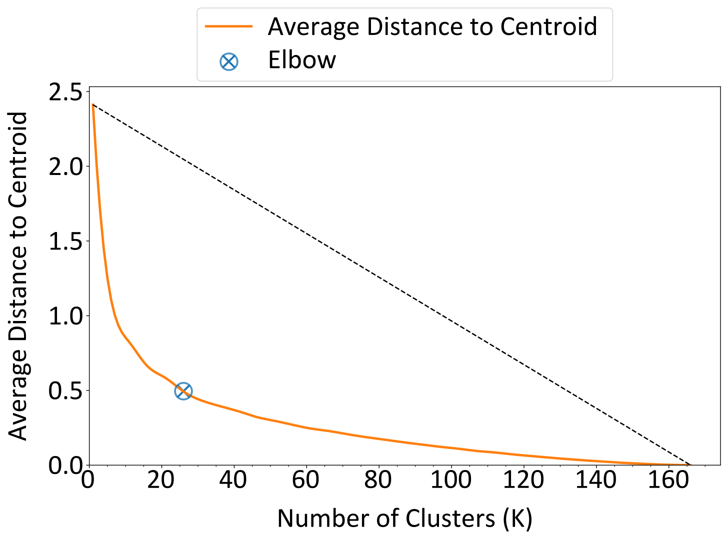


Figure 5.4: Application of the *elbow* method to the input feature set. The appropriate number of clusters (Elbow) is determined at $K = 26$. Taken from Azmi et al. (2020).

ate K. The following section describes how the K-determiner uses the *elbow* and *rmse-ctime* methods with K-means clustering, for instance.

ELBOW METHOD The *elbow* method applies K-means clustering to a given dataset for a range of K values, and for each value of K, calculates the average distance from data points to the centroid of each cluster. As K increases, the average distance to the centroid decreases rapidly until the elbow or maximum curvature of the calculated curve, which is the optimal K (Figure 5.4). The K-determiner defines the elbow by using the point with the maximum distance from the straight line connecting the end-points of the curve (Satopaa et al., 2011). To reduce its run time, the K-determiner executes the clustering for a range of K values in a predefined interval between the minimum and maximum potential number of clusters for the given dataset. Then, it interpolates the resulting average distance to the centroid values into all potential number of clusters and calculates the Elbow point among them (Figure 5.4).

RMSE-CTIME METHOD The *rmse-ctime* method is a customized *elbow* method that allows scientists to decide which K to use, based on an additional constraint. It considers the balance between K, *RMSE* of each cluster member and a selected representative of that cluster, along with the computation time of representatives (Figure 5.5). Here, I demonstrate how the *rmse-ctime* method determines the optimal number of K for the feature set obtained from the use

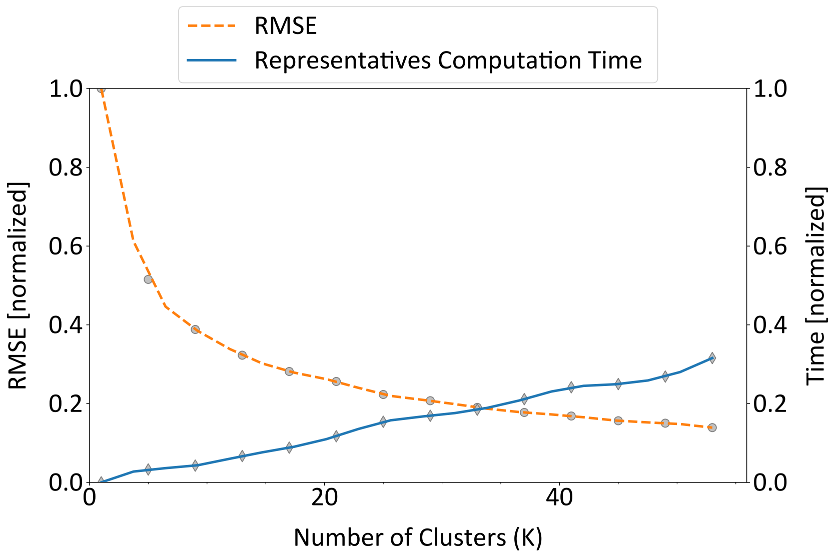


Figure 5.5: Application of the *rmse-ctime* method to the input feature set. The appropriate number of clusters is determined at $K = 33$. Taken from Azmi et al. (2020).

case simulation. The method applies K-means clustering to the preprocessed feature set with variable K values. For each variation of K, the *RMSE* is calculated within each cluster between the cluster members and a selected representative of that cluster. I define the cluster representative as the Medoid data point, which is the point with the minimal average dissimilarity to the other points of the cluster (Kaufman & Rousseeuw, 1987). Formally, the Medoid of a cluster $X = \{x_1, x_2, ..., x_n\}$ is defined as:

$$x_{medoid} = argmin_{y \in X} \sum_{i=1}^{n} d(y, x_i) \, , \tag{5.2}$$

where $d(y, x_i)$ is the distance between y and x_i. According to this, the *RMSE* measure was calculated between the Active Storage time series of the cluster members and their representative. Thus, there is one *RMSE* measurement per HSL for each K variation. Finally, the *total-RMSE* measure of all HSLs corresponding to the number of data points in the feature set (n) is calculated using Equation 5.3:

$$\sigma_{total-RMSE} = \sqrt{\sum_{i=1}^{n} (RMSE_i)^2} \, . \tag{5.3}$$

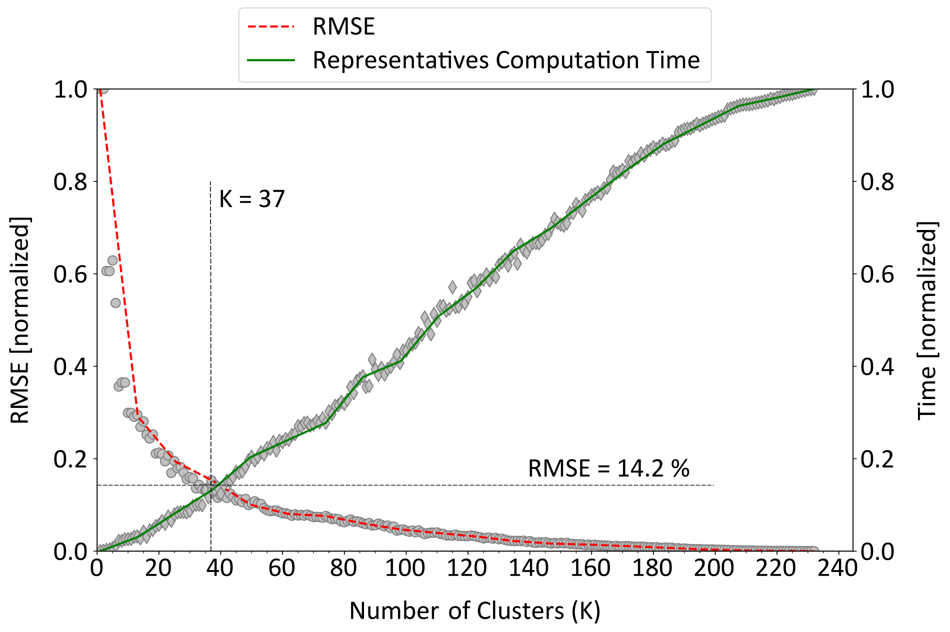


Figure 5.6: *RMSE* and computation time of representatives employing K-means clustering with variable *K*. The gray markers show the original values, and the curves in red and green represent their smoothed trend. Taken from Azmi et al. (2019).

The K-determiner applies this method to a range of *K* values in a predefined interval, spanning from one cluster to one third of all potential number of clusters, and interpolates the resulting *RMSE* values into the remaining potential number of clusters (Figure 5.5). This approach reduces the run time of the K-determiner through interpolation instead of clustering and calculating *RMSE* for all potential number of clusters. The K-determiner calculates the intersection point of the curves, which is a balance of *RMSE* and computation time, as the appropriate *K*.

5.1.2 EVALUATION

In this section, I detail the experimental studies to evaluate the proposed method on the CAOS simulation. For each clustering method, I illustrate the relationship between the performance of the approximated model and its computational complexity. For these evaluations, I performed the CAOS simulation and measured the computation time of each HLS.

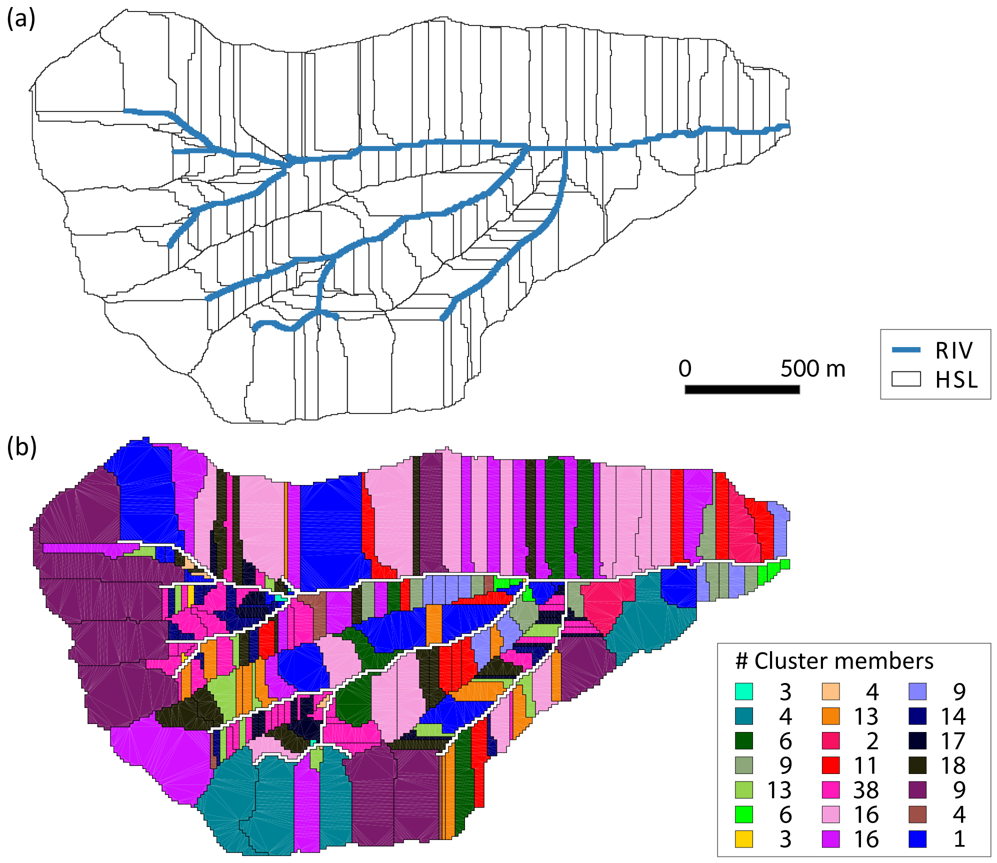


Figure 5.7: (a) Wollefsbach catchment is divided into HSLs of variable shape and size, delineated in black. Each HSL has an edge connected to a RIV in blue. Taken from Azmi et al. (2020). (b) Spatial distribution of K-means clusters at $K = 37$ applied to Wollefsbach catchment. Each color indicates a cluster. The 17 single member clusters are shown in blue. Taken from Azmi et al. (2019).

APPLICATION OF K-MEANS CLUSTERING

The results of application of the *rmse-ctime* method using K-means clustering to the preprocessed feature set of CAOS are shown in Figure 5.6, where the horizontal axis represents K values and the vertical axes represent the $RMSE$ measurement and sum of the computation time of representative HSLs of each cluster normalized by Min-Max. Evidently, as K increases, the corresponding $RMSE$ decreases while the computation time increases (Figure 5.6). The main goal of the approach is to achieve the best trade-off between computation time and simulation uncertainty. In Figure 5.6, a range of the intended compromise between $RMSE$ and computation time is recognizable where the curves intersect. As K-means places the initial centroids randomly, the output of its executions with the same number of K differs slightly.

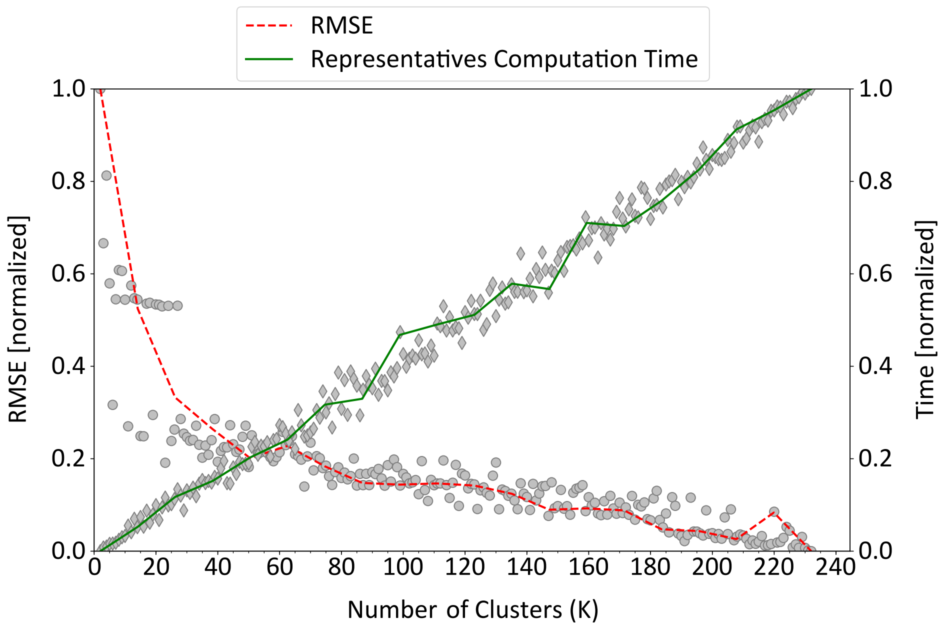


Figure 5.8: *RMSE* and computation time of representatives employing K-medoids clustering with variable *K*. The gray markers show the original values, and the curves in red and green represent the smoothed trend. Taken from Azmi et al. (2019).

Thus, the intended compromise occurs where K ranges from 32 to 42, the corresponding $RMSE$ values range from 0.14 to 0.12 of the maximum $RMSE$ (39.2 *percentage points*) and the computation time ranges from 0.10 to 0.16 of total computation time (31.8 days). As an example, the spatial distribution of the K-means clustering at $K = 37$ which corresponds to the best compromise between $RMSE$ and computation time in Wollefsbach catchment is shown in Figure 5.7. All the single member clusters are the HSLs that do not fit into any cluster. According to domain scientists, this map demonstrates a plausible clustering of HSLs, considering the hydrological parameters like the structure, size, and location.

APPLICATION OF K-MEDOIDS CLUSTERING

Another variant of K-means is the K-medoids algorithm that uses the actual data points as cluster centers. It takes K and the distance matrix of points as input parameters. The algorithm was executed for various values of K, and the results are demonstrated in Figure 5.8. The trend in the results is similar to K-means clustering, as K increases, the corresponding $RMSE$ decreases while the computation time increases. However, the diagram indicates that the intended compromise range between $RMSE$ and computation time occurs in a significantly higher K values ranging from 58 to 78, the corresponding $RMSE$ values range from

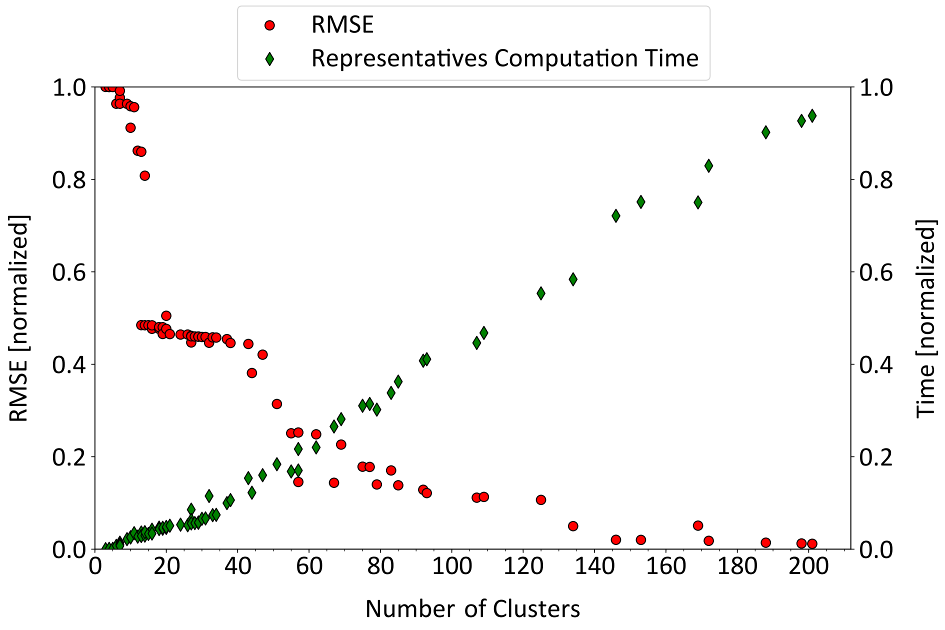


Figure 5.9: *RMSE* and computation time of representatives employing DBSCAN clustering with variable *Eps* and *MinPts*. Some DBSCAN parameter sets generate the same number of clusters (# clusters = 57). Taken from Azmi et al. (2019).

0.35 to 0.17 of the maximum *RMSE* (31.8 *percentage points*) and the related computation time is between 0.23 and 0.34 of the maximum computation time (31.8 days). For *K* values between 5 and 30, the *RMSE* values exhibit minimal decrease, indicating slight changes in the resulted clusters and their selected representatives.

APPLICATION OF DBSCAN CLUSTERING

DBSCAN clustering requires two parameters as input, namely radius (*Eps*) and minimum number of points (*MinPts*). To find a set of optimal parameters, a grid search is conducted by applying DBSCAN clustering with variable values of *Eps* and *MinPts*. As in previous clustering methods, the same approach is employed to determine and visualize *RMSE* along with the computation time when using DBSCAN clustering. For each set of parameters, the number of created clusters is calculated. Noise clusters are considered as one cluster. The results shown in Figure 5.9 indicate that the intended compromise range between *RMSE* and computation time is achieved where the number of clusters ranges from 51 to 62, *Eps* values are from 0.3 to 0.7, *MinPts* values are from 1 to 21, the corresponding *RMSE* values range from 0.31 to 0.15 of the maximum *RMSE* (38.6 *percentage points*) and the computation time falls within the range of 0.18 and 0.23 of the maximum computation time (31.8 days). The

Table 5.1: Parameters and achievements of different clustering methods. Taken from Azmi et al. (2019)

Parameters	**K-means**	**K-medoids**	**DBSCAN**
K (# clusters)	$32 - 42$	$58 - 78$	$51 - 62$
Eps	–	–	$0.3 - 0.7$
$MinPts$	–	–	$1 - 21$
$RMSE$ [normalized]	$0.14 - 0.12$	$0.35 - 0.17$	$0.31 - 0.15$
Max $RMSE$ [percentage points]	39.2	31.8	38.6
Computation Time [normalized]	$0.10 - 0.16$	$0.23 - 0.34$	$0.18 - 0.23$
Max Computation Time [day]	31.8	31.8	31.8

optimal range of clusters in DBSCAN clustering yields lower *RMSE* values compared to K-medoids clustering and higher values than those resulting from K-means clustering. For the number of clusters between 15 and 45, the *RMSE* values exhibit minimal decrease, indicating slight changes in the resulted clusters and their selected representatives.

CONCLUSIONS

In this study, I applied clustering algorithms to identify groups of functionally similar model units in the CAOS simulation. By simulating only one representative from each cluster, I demonstrated how the uncertainty of the approximated model can be controlled by adjusting the number of clusters while considering the corresponding computation time. The direct comparison of the three applied clustering methods is illustrated in Figure 5.10, clearly indicating that K-means clustering outperforms the others in the studied case, featuring the lowest *RMSE* for computations spanning up to 18 days.

As presented in the summary of all results in Table 5.1, K-means clustering achieves the intended compromise between *RMSE* and computation time of the representatives in smaller number of clusters, and consequently, less computation time for representatives compared to the other studied clustering methods. K-medoids clustering exhibits the poorest performance. DBSCAN clustering yields promising results but falls short of the performance achieved by K-means clustering. The main challenge in the application of DBSCAN is to find the right balance between the *Eps* and *MinPts* parameters.

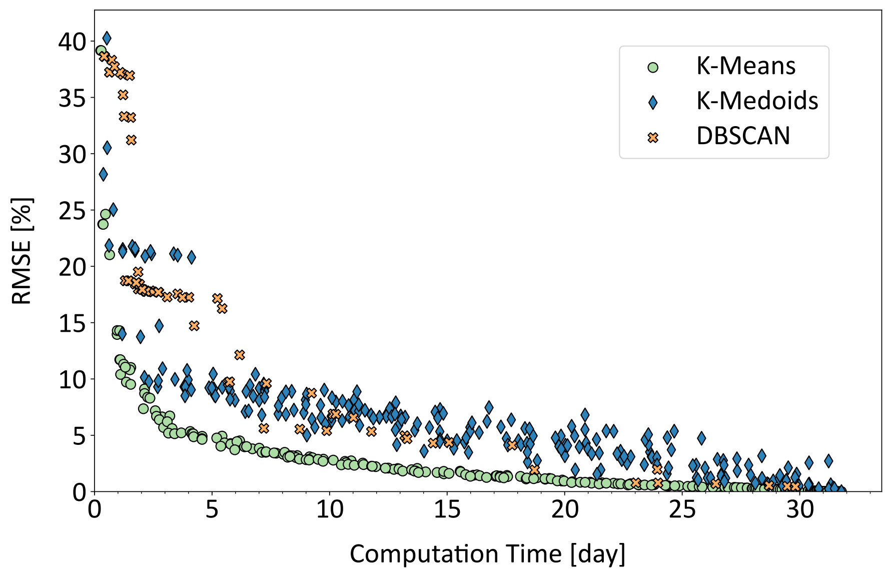


Figure 5.10: Comparison of the *RMSE* and computation time of all analyses. Taken from Azmi et al. (2019).

IMPLEMENTATION ENVIRONMENT

All the analysis methods are implemented in Python and executed on a four-core 64-bit Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz processor with Ubuntu 16.04.4 LTS running the Linux kernel 4.4.0-127-generic. The benchmarking of simulation model parallelization was conducted on a 16-core Intel(R) Xeon(R) CPU E5-2640 v2 @ 2.00GHz processor with Red Hat Enterprise Linux Server release 7.4 running the Linux kernel 3.10.0-693.11.6.el7.x86_64. The K-medoids source code is taken from Alspaugh (2018). All the scripts and related data for the analyses are available under the MIT license at `https://github.com/elnazazmi/hyda` (Azmi, 2018a).

5.2 EVOLUTIONARY APPROACH

In Section 5.1, I presented a proof of concept for reducing the computational complexity of a hydrological simulation using clustering, and generating a performant approximated model. In that analysis, I simplified the simulation and removed any forcing during the simulation. In the following, I extend the analysis to develop an evolutionary approach to approximate and speed up the use case simulation CAOS, using the best-performing clustering method from the previous section, i.e, K-means clustering.

5.2.1 Methodology

The concept behind the evolutionary approach is to reduce the computational costs by reducing redundant computations and approximating the original model dynamics closely. The steps of this approach are integrated into the original simulation and executed together. The preprocessing of the initial feature set is as described in Section 5.1.1. During the feature set preprocessing, I identified some outliers in the *Mean*, *Kurtosis*, *1st Gradient*, and *Active Storage* features (see arrows in Figure 5.11). I excluded these outliers from the clustering process and treated them as individual clusters to speed up the clustering.

The evolutionary approach starts with the identification of similarities in static features of the HSLs by clustering of the initial feature set (Section 5.1.1 under Preprocessing). Subsequently, I select a representative HSL of each cluster and execute the simulation only on these representatives. The next step involves mapping the output of the representatives to the remaining cluster members. This approach helps in avoiding the need to run the simulation for all model units, thereby reducing computation time. The degree of simulation output fluctuation can be controlled by the number of clusters and the clustering frequency. In the subsequent stages of the simulation, I update the initial feature set by adding two features, namely current state (discharge) and forcing. Forcing is defined as the volume of rainfall enforced to HSLs within a specific time frame. Consequently, I continually update the representatives through redoing the clustering with an updated feature set during the ongoing simulation. Finally, the calculated output is saved, and the simulation is completed. Further, I compare the results obtained from the evolutionary approach with those from the original simulation, in terms of performance and speed-up.

The entire approach is outlined step by step in Figure 5.12. The original simulation com-

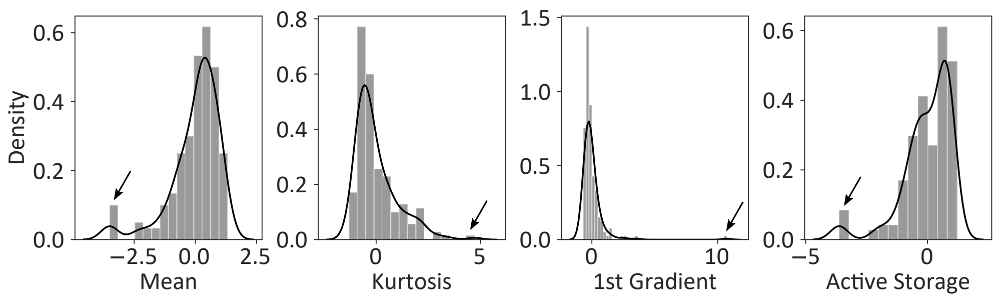


Figure 5.11: Histograms and the kernel density estimate fits of the initial feature set. The arrows show the outliers in each feature. Taken from Azmi et al. (2020).

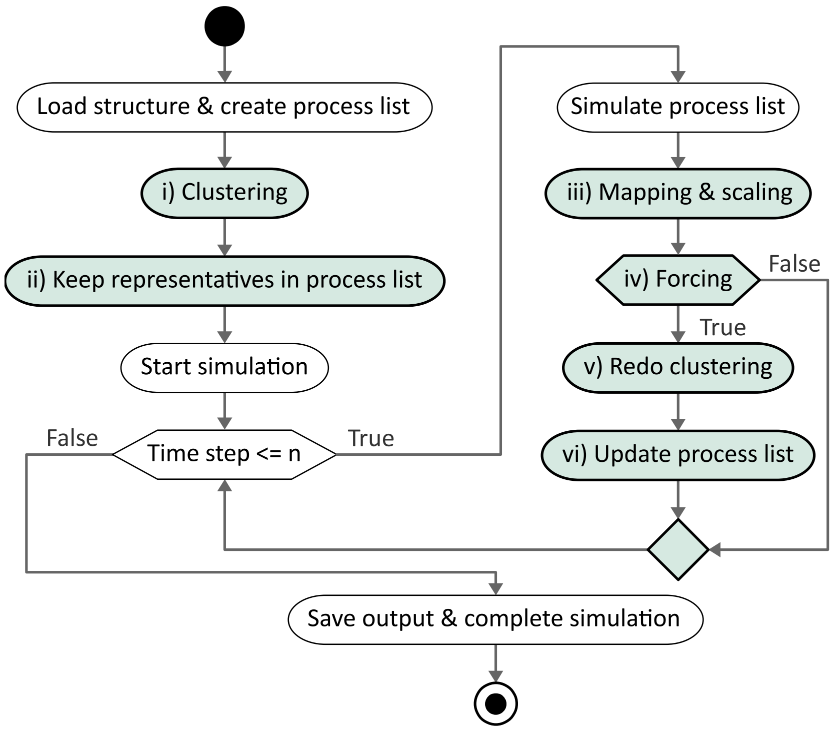


Figure 5.12: Simulation workflow including the original and evolutionary approach. The evolutionary approach integrated in the original simulation is shown in the highlighted boxes. Modified from Azmi et al. (2020).

prises the major steps shown in the white boxes: 1. loading the catchment structure and creating a list of processes or interactions between the model units, 2. starting the simulation for a predefined number of time steps (n), 3. executing the simulation based on the processes list, and 4. saving the output and finalizing the simulation.

The steps of the evolutionary approach shown in the highlighted boxes are as follows:

i) Determine the initial value of K using the K-determiner (Section 5.1.1 under Clustering), and perform clustering on the initial feature set.

ii) Select a representative HSL (Medoid) of each cluster, and keep only these representatives in the process list to simulate the first time step.

iii) Perform the output mapping and scaling (as described in the following section), and update the status of all HSLs.

iv) If there is no forcing in the next time steps, continue the simulation using the already defined representatives.

v) At the beginning and end of a forcing time block (a time frame with active forcing), rerun the K-determiner with the updated feature set and redo the clustering. Because the feature set values change significantly with variable forcing over time, avoid using the K-determiner at each time step within the forcing time block to reduce overhead.

vi) Update the process list based on the new representatives and continue the simulation.

Mapping and Scaling The application of clustering to the initial feature set is followed by the selection of a representative member (HSL) for each cluster. I select the Medoid data point of each cluster as its representative, whose average dissimilarity to other data points in the cluster is minimal. At each time step of the simulation, only the representative HSLs are simulated. The next step of the evolutionary approach is the mapping and scaling of the output of the representative HSL to the member HSLs of the same cluster. The scaling accounts for different areas of the HSLs within a cluster. Therefore, the discharge (output) of the representative HSL, already computed in the simulation, is used to calculate the discharge of other HSLs using Equation 5.4. In the applied use case simulation, the outputs and the area are measured in $\frac{m^3}{s}$ and, m^2 respectively.

$$Cluster\,member\,output = Representative\,output \times \frac{Cluster\,memeber\,area}{Representative\,area}\,. \tag{5.4}$$

5.2.2 EVALUATION

In this section, I present the results of the proposed approach applied to the study case Wollefsbach. I use three metrics, either individually or in combination, to demonstrate the quality of the evolutionary approach, specifically, how closely the outputs align with the original simulation. These metrics are *RMSE*, *PCC* and *KGE* (Section 2.2.1). In addition to the quality metrics, I present the computational complexity of the evolutionary approach as the speed-up in the approximated model's simulation run time compared to the original model's simulation run time. All evaluation results are shown in tables 5.2 - 5.7, sorted in ascending order based on *RMSE*. Additionally, as simulation of time blocks with forcing is more compute-intensive, I provide the *RMSE* for time blocks with forcing as *RMSE-WF* and without forcing as *RMSE-WOF* separately.

Table 5.2: Evaluation results of the original simulation with different random seeds. The best value for each metric is highlighted. Taken from Azmi et al. (2020).

Tests	RMSE	RMSE-WOF	RMSE-WF	PCC	KGE
Test-1	0.001 55	0.001 36	0.002 14	0.990	0.960
Test-3	0.001 63	0.001 43	0.002 21	0.988	0.983
Test-2	0.002 13	0.001 91	0.002 81	0.984	0.954

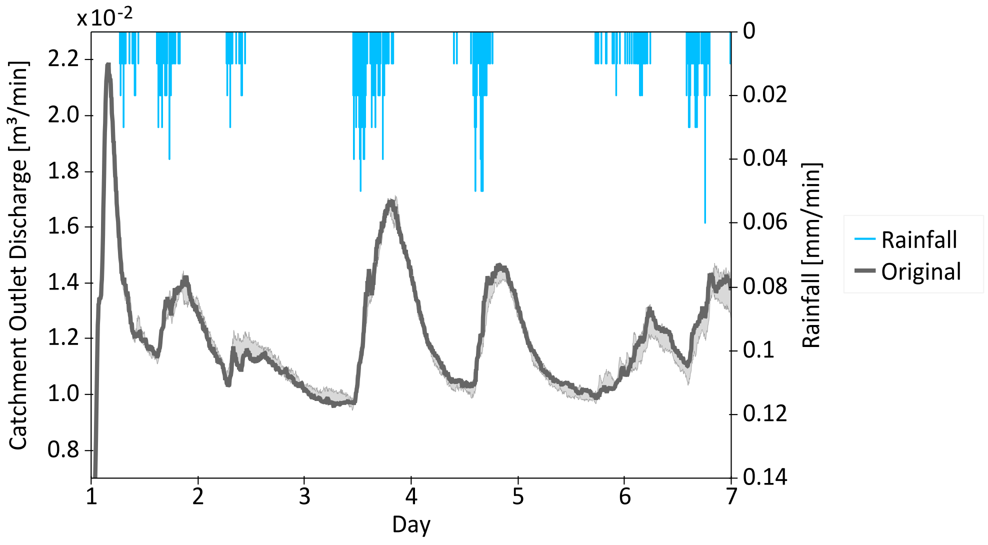


Figure 5.13: Output of the original simulation with different random seeds. The gray band shows the minimum and maximum of all tests. Taken from Azmi et al. (2020).

INFLUENCE OF RANDOM SEED

The original simulation executes processes of the model units in random order to keep the model close to the natural behavior of a hydrological system. Changing the random seed in the original simulation results in slightly different curves. I conducted the original simulation four times with different random seeds to estimate the systematic error resulting from the choice of a random seed. As it is illustrated in Figure 5.13, the horizontal axis shows the simulation time of one week (1st to 7th of January). The left vertical axis shows the discharge at the catchment outlet, and the right vertical axis from top to bottom the amount of rainfall during the simulation. The gray band shows the minimum and maximum of the three test executions. The line labeled as "Original" is used as ground truth for all following tests, hence all evaluation results (Table 5.2 - 5.7) are relative to this line. Table 5.2 presents the evaluation of the three tests relative to the ground truth.

Table 5.3: Evaluation results of the tests with different random seeds. The best value for each metric is highlighted. Taken from Azmi et al. (2020).

Tests	RMSE	RMSE-WOF	RMSE-WF	PCC	KGE
Test-6	0.0045	0.0044	0.0050	0.920	0.765
Test-4	0.0056	0.0050	0.0073	0.891	0.776
Test-3	0.0056	0.0054	0.0064	0.868	0.729
Test-1	0.0078	0.0075	0.0087	0.907	0.803
Test-5	0.0096	0.0091	0.0110	0.858	0.748
Test-2	0.0097	0.0091	0.0117	0.814	0.713

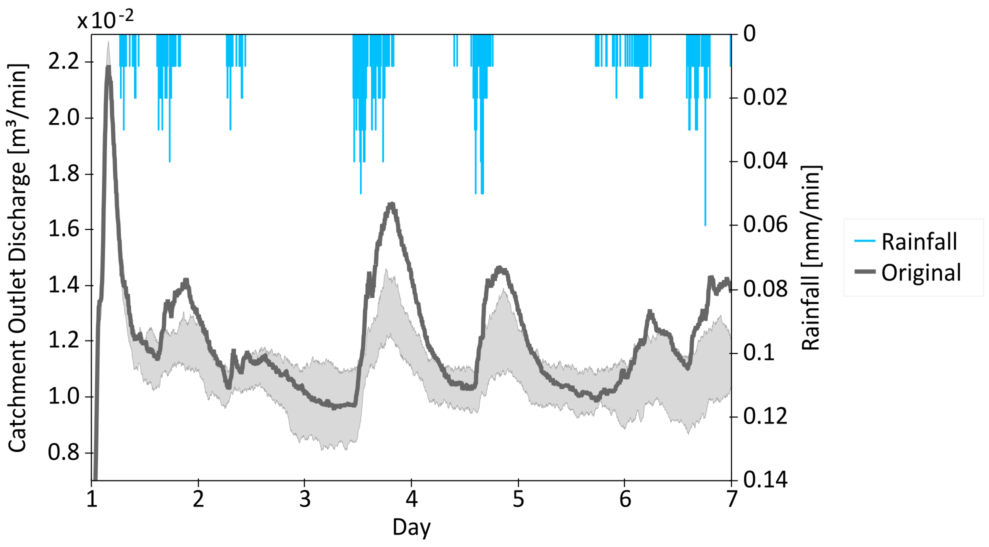


Figure 5.14: Simulation output of the tests with different random seeds for K-means. The gray band shows the minimum and maximum of all tests. Taken from Azmi et al. (2020).

In addition to the randomness of the original simulation, the evolutionary approach uses the K-means clustering that generates its initial centroids randomly. To evaluate the influence of a variable random seed of the clustering on the simulation output, I conducted six simulations with the same random seed for the original simulation and using different random seeds for K-means clustering (Figure 5.14). The clustering was performed once using the initial feature set with $K = 9$ without any further clustering. All tests, represented by the gray band, exhibit a similar trend to the original simulation and achieve an acceptable *KGE* value from a hydrological perspective (Table 5.3). To make K-means clustering deterministic and retain reproducible test results in the next sections, I use a fixed random seed for both the original simulation and clustering.

Table 5.4: Evaluation results of the Constant-K tests. The best value for each metric is highlighted. Taken from Azmi et al. (2020).

Tests	RMSE	RMSE-WOF	RMSE-WF	PCC	KGE	Speed-up
K-42	0.0037	0.0035	0.0042	0.954	0.818	2
K-50	0.0039	0.0039	0.0040	0.943	0.794	1.9
K-9	0.0045	0.0044	0.0050	0.920	0.765	3.2
K-34	0.0066	0.0061	0.0081	0.909	0.775	2.3
K-17	0.0070	0.0070	0.0068	0.927	0.806	2.8
K-25	0.0092	0.0088	0.0103	0.923	0.792	2.4

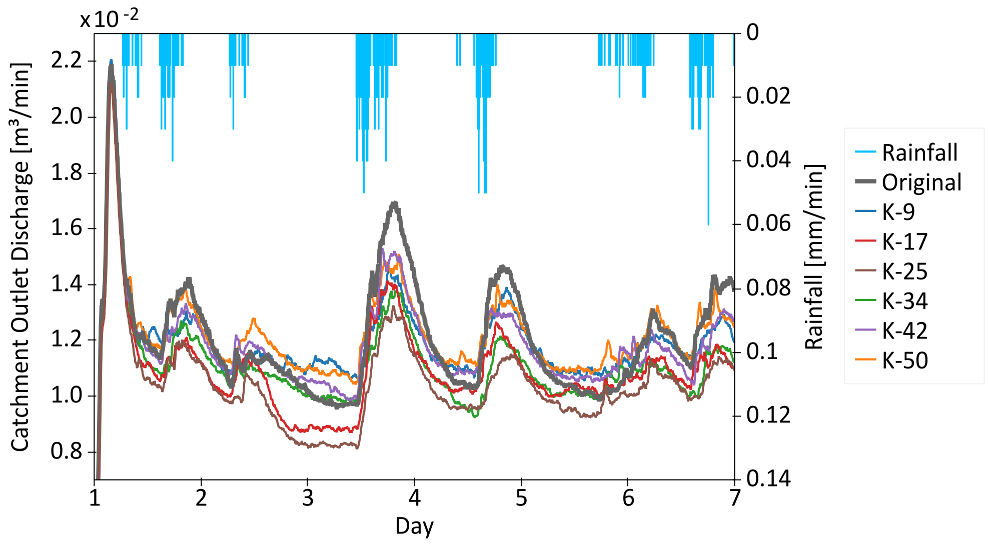


Figure 5.15: Simulation output of the Constant-K tests. Taken from Azmi et al. (2020).

The following sections describe detailed tests to evaluate the evolutionary approach.

CONSTANT-K

In order to reveal the similarities in the static model unit properties of the hydrological model, I designed a test that applies clustering once to the initial feature set at the first time step of the simulation without any further clustering. The simulation continues using the same representatives of the initial feature set clustering. The catchment outlet discharge is calculated as the output for each time step (Figure 5.15), and the evaluation results are shown in Table 5.4. I have repeated the approach for a range of K values defined as a percentage of total HSLs (5 - 30 %, corresponding to $K = 9$ - 50) to test the effect of the parameter K on the simulation output. The curves follow the trend of the original simulation (Figure 5.15). Evaluation of the results reveals that the quality measures have no strong correlation with *K*, and the *RMSE* order does not fit one by one to the *KGE* metric (Table 5.4). This means that the test with a lower *RMSE* is not always the more efficient one. However, the smaller the value of *K*, the higher the speed-up. To further increase the speed-up while keeping the *RMSE* value low and the *PCC* as well as *KGE* values high, I designed the following Variable-K tests.

VARIABLE-K

In the Constant-K tests, I demonstrated clustering of the initial feature set based on the static properties of the model units. Subsequently, I incorporated the current state (discharge) and forcing of the model units into the feature set and applied the clustering to the updated feature set. To determine the frequency of redoing the clustering during the simulation, I considered the intensity of the occurring dynamics. As dynamic processes become more pronounced during the simulation of forcing, I split the simulation time into "with forcing (*WF*)" and "without forcing (*WOF*)" time blocks and redo the clustering exclusively during the forcing time blocks. Table 5.4 presents a trend of improved results for higher *K* value, at the cost of longer run times.

Leveraging the findings from the Constant-K tests, I designed the Variable-K tests that execute the simulation using updated representatives resulting from redoing the clustering. To achieve high-quality results during high dynamics combined with a short run time, I adjusted the value of *K* during the simulation, according to the on- and offset of forcing (Figure 5.16). This means that I divided the simulation run into two parts: i) using a high *K* value at *WF* time blocks, and ii) using a low *K* value at *WOF* time blocks. I paired higher *K* values and lower *RMSE* values for *WF* time blocks and lower *K* values and lower *RMSE* values for *WOF* time blocks according to the results of the Constant-K tests (Table 5.4). The simulation is started with the clustering of the initial feature set using the best *K* of *WOF* time block. Then according to the forcing, the simulation continues with a low *K* at *WOF* time blocks, and with a higher *K* at *WF* time blocks (Figure 5.16). Clustering is only performed when switching between *WF* and *WOF* time blocks and vice versa, because of the overhead of clustering. The trend of the curves for Variable-K is difficult to interpret, but the tests indicate a tendency toward higher *RMSE* and lower *PCC* as well as *KGE* compared to the Constant-K tests (Table 5.5).

Table 5.5: Evaluation results of the Variable-K tests. The best value for each metric is highlighted. Taken from Azmi et al. (2020)

Tests	RMSE	RMSE-WOF	RMSE-WF	PCC	KGE	Speed-up
K-17-50	0.0061	0.0058	0.0071	0.823	0.719	1.9
K-17-42	0.0079	0.0076	0.0090	0.720	0.714	2
K-9-42	0.0101	0.0095	0.0122	0.693	0.591	2.1
K-9-50	0.0121	0.0123	0.0113	0.509	0.462	2.1

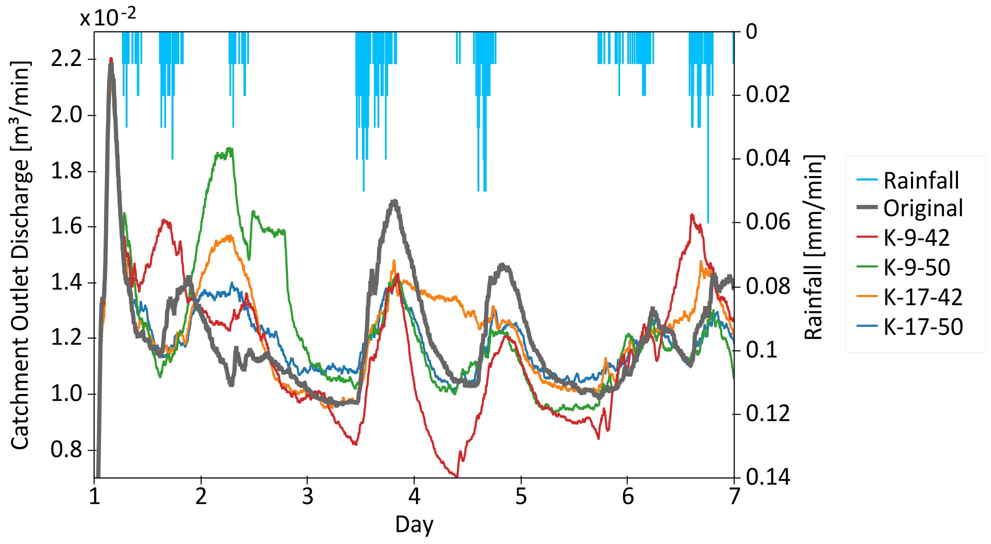


Figure 5.16: Simulation output of the Variable-K tests. Taken from Azmi et al. (2020).

AUTO-K

The Variable-K tests revealed significant variation of the simulation output when changing K during the simulation. As a solution, I applied the K-determiner during the simulation in Auto-K tests to automatically select an appropriate K value and reduce the high fluctuations. The Auto-K tests share the same settings as Variable-K tests, with the difference that the K value is not set manually. Instead, the K-determiner is employed once at the beginning and end of each forcing time block to automatically determine an appropriate K value. This process is based on the evolving feature set generated from the dynamics occurred during the simulation. In the Auto-K tests, the K-determiner applies the *elbow* (K-AEL) and *rmse-ctime* (K-ARC) methods, respectively. The trends of their curves fit well with the original simulation. The gray band, representing six tests for K-ARC with different random seeds, narrows during the

Table 5.6: Evaluation results of the Auto-K tests. The best value for each metric is highlighted. Taken from Azmi et al. (2020).

Tests	RMSE	RMSE-WOF	RMSE-WF	PCC	KGE	Speed-up
K-ARC	0.0049	0.0046	0.0059	0.894	0.80	1.8
K-AEL	0.0061	0.0061	0.0058	0.828	0.72	2

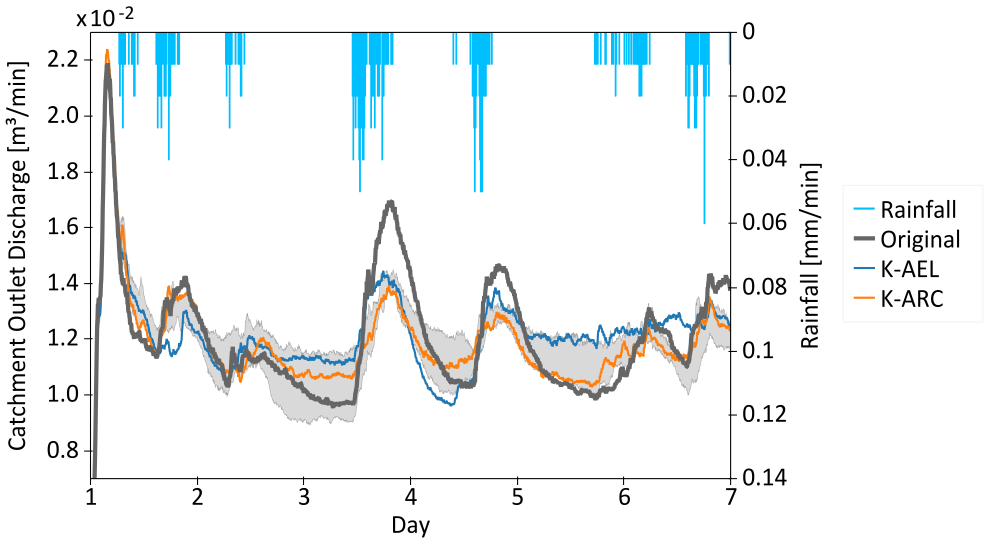


Figure 5.17: Simulation output of the Auto-K tests. The gray band shows the minimum and maximum of six tests for K-ARC with different random seeds. Taken from Azmi et al. (2020).

forcing time blocks, demonstrating a reproducible peak discharge (Figure 5.17). The *RMSE* for both tests, K-ARC and K-AEL, is acceptably low. Although K-ARC results in a lower *RMSE* and higher *PCC* as well as *KGE* compared to K-AEL, its speed-up is lower than K-AEL since K-ARC employs higher K values (Table 5.6).

Additionally, to test the variability of the best K, I executed the Auto-K tests with K-determiner at all time steps of the simulation and obtained a narrow frequency distribution of the selected best K values. The K values ranged between 17 and 30, with the most frequent $K = 25$, when using the *elbow* method (Figure 5.18(a)) and between 30 and 38, with the most frequent $K = 34$, when using the *rmse-ctime* method (Figure 5.18(b)). Using the K-determiner at each time step, rather than only once at the beginning and end of each forcing time block, resulted in slightly better performance. However, it comes with a higher computational complexity, which increases the total simulation run time compared to the original simulation (Table 5.7).

Table 5.7: Evaluation results of the Auto-K tests when the K-determiner is applied at all time steps of the simulation. The best value for each metric is highlighted. Taken from Azmi et al. (2020).

Tests	RMSE	RMSE-WOF	RMSE-WF	PCC	KGE	Speed-up
K-AEL	0.0051	0.005	0.0055	0.950	0.825	0.6
K-ARC	0.0053	0.005	0.0060	0.954	0.843	0.6

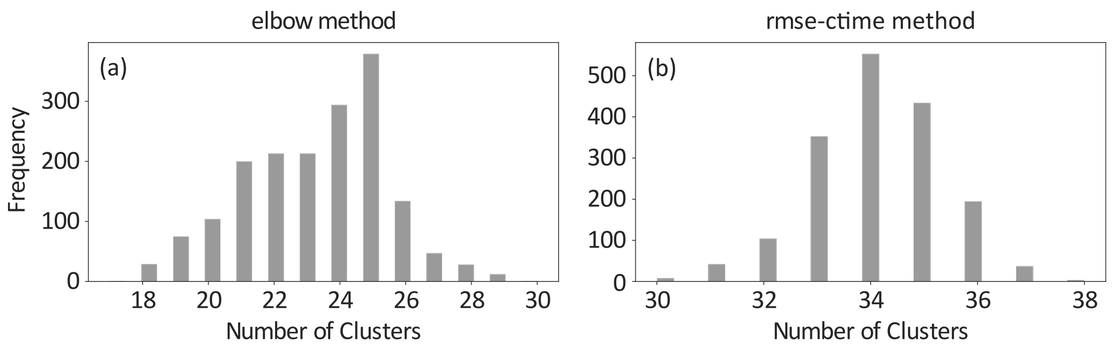


Figure 5.18: Histogram of the selected K values by K-determiner using (a) *elbow* and (b) *rmse-ctime* methods when it is applied at all time steps of the simulation. Taken from Azmi et al. (2020).

CONCLUSIONS

In this study, I demonstrated an evolutionary approach consisting of several main steps: clustering the feature set obtained from the static properties of model units, simulating a representative of each cluster, mapping and scaling the outputs to the remaining cluster members, and dynamically clustering the updated feature set based on the enforced forcing during the simulation. I integrated the evolutionary approach into the original CAOS simulation and presented its evaluation through several tests. I applied three different test settings (Constant-K, Variable-K and Auto-K) to accelerate the simulation with K-means clustering. The metrics indicate minor differences between the Auto-K and Constant-K tests, considering several test runs with different random seeds for the K-means clustering (Table 5.3, 5.4 and 5.6).

The results of Auto-K, which clusters at every time step of each forcing time block, show lower fluctuations, resulting in a more reliable output at these time blocks compared to the Constant-K tests (Figure 5.14 and 5.17). The Auto-K tests utilize the K-determiner to dynamically determine the appropriate K based on the given feature set and automatically configure the clustering parameters during the simulation. The additional clustering steps of the Auto-K tests yield better values for *RMSE*, *PCC* and *KGE* metrics, along with a lower speed-up.

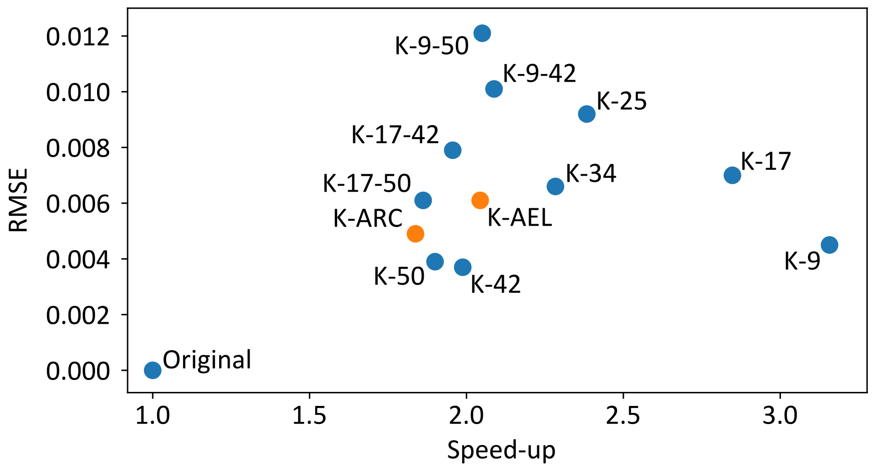


Figure 5.19: Simulation run time speed-up and *RMSE* of all tests. The orange markers highlight the Auto-K tests. Taken from Azmi et al. (2020).

Since K-ARC frequently selects $K = 34$ as the appropriate K, its comparison with the K-34 test from the Constant-K tests confirms this observation (Table 5.4 and 5.6).

The direct comparison of all tests is illustrated in Figure 5.19, indicating the overall efficiency and effectiveness of the evolutionary approach. Constant-K tests (on the right side of the diagram) presented a higher speed-up, although the ideal choice for achieving a high speed-up together with a low *RMSE* appears to be unpredictable. This means that, although K-42 from Constant-K tests has the lowest *RMSE* value among all tests and two times of speed-up, prediction of such a favorable K without running tests for a particular use case is not possible. Therefore, I conclude that the evolutionary approach using a K-determiner (Auto-K) is the best compromise between the performance and computational complexity of the simulation.

IMPLEMENTATION ENVIRONMENT

The CAOS simulation's original scripts are implemented in MATLAB R2019a. The evolutionary approach is implemented in Python. All tests are executed on a Red Hat Enterprise Linux Server release 7.4 on a 16-core Intel(R) Xeon(R) CPU E5-2640 v2 @ 2.00 GHz processor. All the scripts and related data for the analyses are available under the MIT license at `https://github.com/elnazazmi/hyda` (Azmi, 2020).

5.3 GENERALIZATION

To demonstrate the generality of the evolutionary approach and to establish the requirements and criteria for any applicable environmental simulation, I applied the approach to two further use cases. It cannot be guaranteed that the approach is suitable for all compute-intensive environmental simulations, but these use case simulations assisted in delineating some general requirements to narrow the simulation choices down. In the following, I provide an overview of the application of the evolutionary approach to these use cases and the conclusions drawn.

5.3.1 SIMPLE HYDROLOGICAL USE CASE

The hydrological simulation SHM (Section 2.1.2) simulates the discharge of various catchments over a specific time. For this simulation, I generated a feature set based on the drainage test results as described in Section 5.1.1. Following the preprocessing of this feature set, I applied the evolutionary approach to the SHM simulation, and designed several test cases as follows:

i) K-9 to K-25: Constant-K tests with various predefined K values ranging from 9 to 25.

ii) K-E-16: Constant-K test employing the K-determiner once using the elbow method, which determined $K = 16$.

iii) K-AEL: Auto-K test employing the K-determiner using the elbow method.

iv) Adaptive clustering: A simulation approximation method using similarities in states of the model units.

Adaptive clustering is another simulation approximation method that clusters similarly functioning model units based on normalized and binned transformations of the states and fluxes of model units. This method simulates a few representatives per cluster and maps the updated states to the cluster members (Ehret et al., 2020).

The evaluations of the test cases presented in Table 5.8 exhibit significantly lower *RMSE* and higher *PCC*, as well as *KGE* values for Constant-K and Auto-K tests when compared to the Adaptive clustering. The speed-up of K-25 with the best *RMSE* and *KGE* values is slightly higher than that of the Adaptive clustering (Figure 5.20). Although the K-AEL test exhibits a low *RMSE* value, it requires more computation time compared to the original simulation. In

Table 5.8: Evaluation results of test cases from SHM simulation. The best value for each metric is highlighted.

Tests	RMSE	PCC	KGE	Speed-up
K-25	0.0	1.0	1.0	3.5
K-17	16.27	1.0	0.999	4.9
K-13	228.28	1.0	0.993	6.1
K-E-16	230.56	1.0	0.991	5.4
K-9	244.50	1.0	0.995	8.7
K-AEL	333.89	1.0	0.996	0.5
Adaptive clustering	2515.16	0.987	0.929	3.1

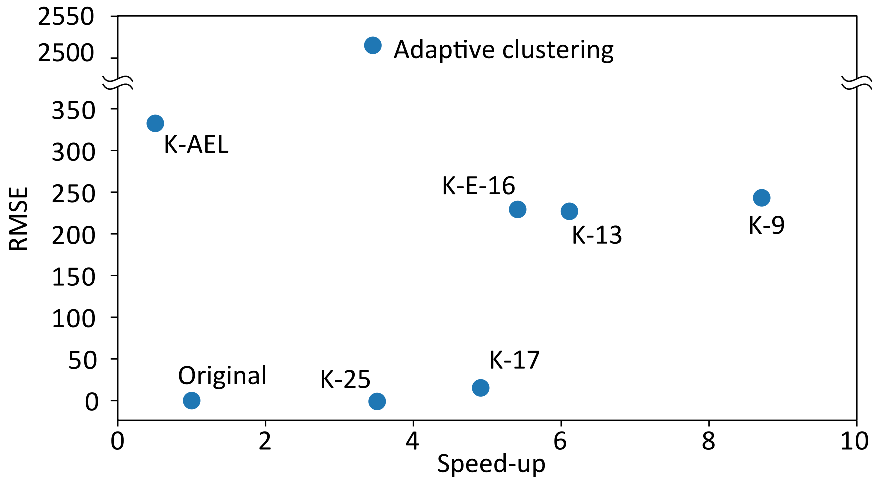


Figure 5.20: SHM simulation run time speed-up and *RMSE* of all tests.

such a simple simulation with low computational complexity, the overhead of frequent clustering hinders a speed-up. Nevertheless, the evolutionary approach outperforms the adaptive clustering.

5.3.2 METEOROLOGICAL USE CASE

The meteorological simulation ICON-ART (Section 2.1.3) was employed in this experiment. The core concept of the evolutionary approach is to detect independent model units with similar properties and functionalities. In the use case of ICON-ART atmospheric chemistry, the atmosphere is divided into triangular grid cells and height levels. The simulation inputs the physical properties and the initial volume mixing ratio (VMR) of trace gases into each cell and provides the current VMR of the trace gases as output. Thus, I have used these input parameters as the feature set for the evolutionary approach. I applied the approach in several test cases utilizing variously preprocessed feature sets, such as application of PCA and autoencoder for dimensionality reduction of the feature set. The results of these tests indicated a longer simulation run time compared to the original simulation. The computational complexity of K-means clustering, integrated into the evolutionary approach, increases exponentially based on the number of clusters and the dimensionality of the feature set. Consequently, even when running the simulation solely on a small, representative subset of the grid cells, the overhead of executing the evolutionary approach on a feature set of approximately 1.8 million grid cells during the simulation hinders any speed-up. While the performance of the approximated model improves as the number of clusters increases, employing the approach in this use case becomes futile when a plausible outcome is only achieved without any speed-up.

5.3.3 CONCLUSIONS

Application of the evolutionary approach to different use cases has helped in establishing certain criteria for selecting appropriate simulations for this approach. First, the simulation model units should be capable of parallel processing without interacting with each other in the original simulation. Second, the simulation model units should exhibit similarities in properties or functionality based on domain knowledge. Finally, individual model units should be compute-intensive, ensuring that the simulation execution time of the representative model units, when combined with the overhead of the evolutionary approach, is shorter than that of the original simulation.

5.4 SUMMARY

In this chapter, I introduced an evolutionary approach to leverage static properties and dynamics of environmental models to reduce the computational complexity of their simulations. To explore and identify similarities in the static properties of model units within a use case simulation in hydrology (CAOS), I applied three conventional clustering methods, namely K-means, K-medoids and DBSCAN to a feature set extracted from a drainage test time series conducted using the CAOS simulation. According to the analysis, K-means clustering outperforms the other applied clustering methods in terms of both performance and computational complexity.

The evolutionary approach consists of several steps, primarily clustering and scaling of the simulation output of the cluster representatives to the remaining cluster members. I employed K-means clustering in conjunction with the K-determiner, which automatically determines a suitable number of clusters using the *elbow* or *rmse-ctime* methods. The results of the tests demonstrated that the K-ARC approach has a promising *RMSE* of 0.0049, a *PCC* of 0.89 and a *KGE* efficiency of 0.8 which closely approximates the original simulation output. Additionally, K-ARC has a simulation run time speed-up of 1.8 which is approximately half of the original simulation run time.

To demonstrate the generality of the evolutionary approach, I applied it to two further use cases, namely a simple hydrological model (SHM) and a meteorological model for atmospheric chemistry (ICON-ART). The approach approximates and optimizes the SHM simulation when clustering is not applied frequently during the simulation. It outperforms a similar approximation method in terms of quality metrics. In the case of the ICON-ART simulation, the approximated model exhibits no speed-up compared to the original simulation. This overhead is caused by clustering of millions of grid cells during the simulation. Based on these investigations, I established certain criteria for selecting appropriate simulations for the evolutionary approach.

The best way to predict the future is to invent it.

Alan Kay

6 Approximation of Simulations using Supervised Machine Learning Methods

Numerical environmental simulations, especially those conducted at high spatio-temporal resolutions and involving large-scale dynamical systems, are compute-intensive and demand HPC resources. Atmospheric chemistry modeling, for example, requires solving a system of hundreds of coupled ordinary partial differential equations that describe the concentration changes of atmospheric trace gases due to chemical reactions in the atmosphere (Sander et al., 2011, Emmons et al., 2010). Generally, the growth of HPC resources over the last few decades has made it possible to increase the resolutions (i.e., number of grid cells) of atmospheric models. This allows resolving more and more processes numerically, rather than accounting only for their effect with so-called parametrizations (Weimer et al., 2021). However, solving the system of chemical differential equations for each grid cell constitutes a significant portion of the computation time in atmospheric chemistry models, which is a challenge as resolution increases.

In Chapter 5, I showed an approximation of compute-intensive environmental simulations using unsupervised machine learning methods, specifically clustering. However, the discussion in Section 5.3 revealed that for certain environmental simulations applied to millions of model units, the approach yields implausible results without overall acceleration in the simulation. This led to determining specific criteria to be fulfilled to efficiently utilize the evolutionary approach.

This chapter addresses the Research Question 5, defined in Section 1.1, namely I propose an approach to reduce the computational complexity and high demand for computing resources in environmental simulations by approximating them using supervised machine learning methods, with a focus on neural networks. The goal of this study is to investigate the feasibility, opportunities but also challenges and pitfalls of replacing a compute-intensive simulation like the ICON-ART atmospheric chemistry model (Section 2.1.3) with a trained neural network model to reduce the computational complexity of the simulation. Section 6.1 describes the approach that creates a surrogate model (Section 2.4.2) using a dataset of past simulations to forecast volume mixing ratio (VMR) of atmospheric chemicals iteratively in the future. Section 6.2 presents an evaluation of the surrogate model compared to the original numerical simulation. Section 6.3 demonstrates the generalizability of the supervised machine learning approach and the necessary criteria for simulations to be met in order to be approximated using this approach. The implementation environment of the approach is described in Section 6.4. Finally, Section 6.5 summarizes the chapter. The approach and its evaluations presented in this chapter have been published in Azmi et al. (2023).

6.1 METHODOLOGY

In the following, I present the development of a surrogate model (Section 2.4.2), referred to as the ICON Neural Network-based approach (ICONET). Surrogate models aim to replace compute-intensive simulations through computing an approximation of the original model by using data-driven models and supervised machine learning methods. The use case simulation targeted for evaluation of the ICONET is a meteorological simulation of the atmospheric chemistry from ICON-ART (Section 2.1.3). Within ICONET, a multi-feature LSTM model (Section 2.3.2 under Recurrent Neural Networks) is employed to forecast future values based on its input features. The core process of ICONET learns a function that maps a sequence of values from previous time steps to the subsequent time step. By utilizing output data from prior simulations, ICONET forecasts future values without computing differential equations for individual grid cells. The data-driven nature of ICONET makes it applicable to environmental simulations, independent of their spatio-temporal resolution. Figure 6.1 illustrates the ICONET workflow divided into four main steps: preprocessing (red), training (yellow), forecasting (blue) and post-processing (green). The following sections describe each step of the ICONET workflow in detail.

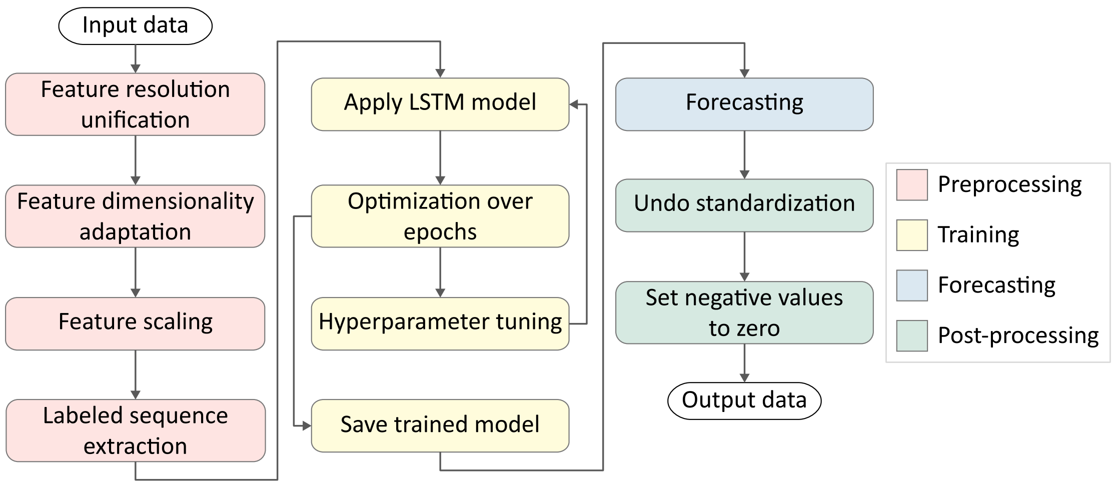


Figure 6.1: ICONET workflow illustrating development of the surrogate model. Taken from Azmi et al. (2023).

6.1.1 PREPROCESSING

I generated the input data for training, validating, and testing of ICONET from ICON-ART simulation outputs. Every data file consists of four dimensions: vertical model levels, grid cell locations, time steps, and features. Each file contains 90 vertical model levels, 20480 grid cell locations in six-minute resolution during one day (240 time steps), encompassing the VMR of twelve gases (chemical features) and three physical features (in total 15 features). To handle training of these large data files, selecting a good representation (sample) of the entire dataset is required. For efficient data sampling and loading during the training of ICONET, I split every daily data file of $\sim$ 27 GiB into 256 files, each $\sim$ 94 MiB in size (Figure 6.2). I shuffled all grid cell locations and randomly distributed them among the generated subfiles to potentially cover data from all spatial regions. I utilized two of these subfiles for training and one for validation. Given the significance of the temporal dependency in the simulation, I preserved the original temporal dimension.

FEATURE RESOLUTION UNIFICATION

The twelve trace gases[1] used for the atmospheric chemistry simulation scenario are HNO_3, HO_2, H_2O, NO, NO_2, NO_3, N_2O, N_2O_5, OH, O_3, $O(^1D)$, and $O(^3P)$. The physical features

[1]All the gases listed are trace gases, except for H_2O. For simplicity, I will use the term "trace gas" for all chemical features mentioned in the remainder of this chapter.

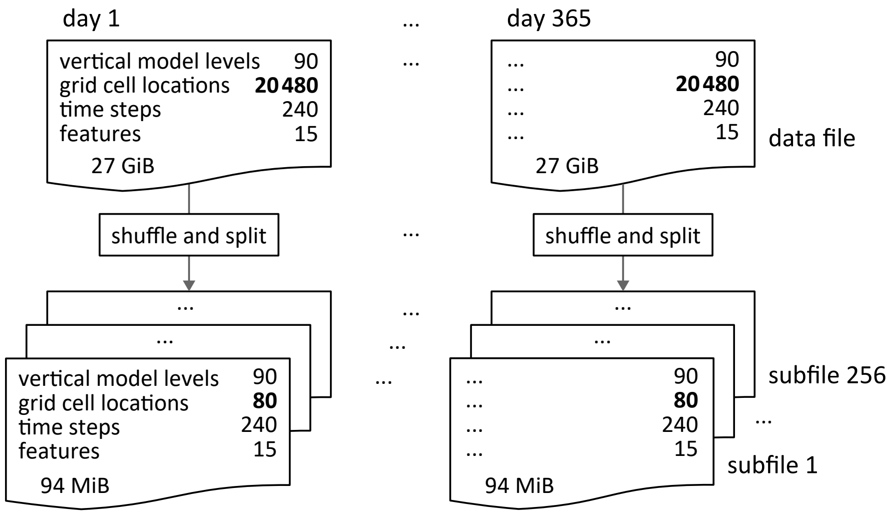


Figure 6.2: Splitting schema for simulation data files. Taken from Azmi et al. (2023).

comprise temperature, pressure, and the cosine of the solar zenith angle (cos SZA). All these features are available in the same temporal resolution, except for cos SZA. To improve the accuracy of ICONET, I interpolated this feature linearly to the six-minute resolution, aligning it with the other features.

FEATURE DIMENSIONALITY ADAPTATION

LSTM layers are designed for processing of three-dimensional data structured as: i) samples, referred to as sequences (or batch size in mini-batch learning), ii) time steps, representing the length of the sequences, and iii) features. As the four-dimensional simulation data are not suitable for the input layer of LSTM, I adapted their dimensionality by excluding the first two dimensions, vertical model levels and grid cell locations. The vertical model levels are determined by pressure values that are already included as a physical feature. Since I exclusively focus on replacing the atmospheric chemistry, in which the grid cells have no interaction with each other, their locations lack significance in this use case. Therefore, the final feature set consists of two dimensions (time steps and features) for each grid cell. The creation of the first dimension (batch size) will be described in the section of Labeled Sequence Extraction.

FEATURE SCALING

Equally scaled input data are important for the learning performance in machine learning algorithms (Bishop, 1995). Among several feature scaling techniques, such as normalization

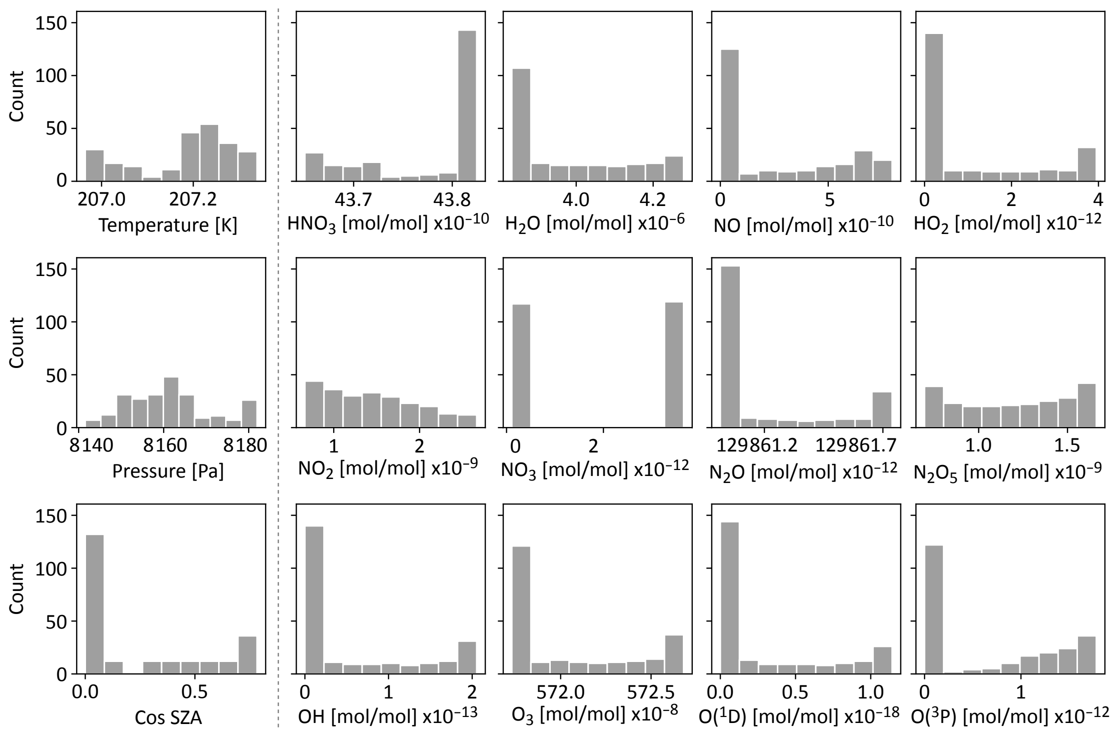


Figure 6.3: Feature set distribution of an exemplary grid cell located above the Indian Ocean from 12 September 2013. The physical features are shown on the left side of the dotted line. Taken from Azmi et al. (2023).

and standardization, I conducted experiments by training a model using differently scaled feature sets. As a result, I selected the standardization scaling (Equation 5.1), which resulted in the best performance. Figure 6.3 displays the value range and distribution of all features from an example grid cell located above the Indian Ocean on 12 September 2013. It demonstrates how the features vary in several magnitudes before scaling. In the following, I refer to this grid cell as an exemplary grid cell.

LABELED SEQUENCE EXTRACTION

The final preprocessing step is to extract sequences and labels (ground truth) from the feature set. For this purpose, I employed the rolling window technique to create sequences of fixed length and their respective labels from all one-day feature sets. For labelling, I use the subsequent time step after each sequence. To determine the rolling window size (sequence length = n time steps), I conducted experiments by training a model using a rolling window of variable length over all one-day feature sets, and consequently assigned the corresponding subsequent time step of each sequence ($(n+1)$th time step) as the respective label. The best performing model is achieved for a sequence length of 10 time steps, resulting in 230 sequences (batch size), for every one-day feature set. Batch size and sequence length are directly dependent on each other and add up to 240 time steps per day, due to the six-minute resolution. These sequences define the first dimension of the feature set for the input layer of the LSTM model.

6.1.2 TRAINING

The network structure and training configuration of neural network models are defined as hyperparameters that must be set before training. The training of a neural network model (Section 2.3.2) begins with the random initialization of its weights and hyperparameters. Naturally, this initial configuration is unlikely to yield satisfactory results. By adjusting these hyperparameters, which are variables that determine both the structure of the network (e.g., number of hidden states) and how the network is trained (e.g., learning rate), the objective is to obtain optimal weights for the model, leading to improved forecasts. Throughout the training process, the underperforming neural network is transformed into a highly accurate one by optimizing the loss function and minimizing the loss between the target and forecast values by the end of the training. This optimization is achieved by iteratively adjusting the network's weights, effectively modifying its function.

LSTM MODEL

ICONET includes an LSTM model with one input layer, one hidden layer that consists of 15 hidden states representing 15 chemical and physical input features, and one output layer. During the training, ICONET reads an input sequence into the LSTM model and gets the final hidden state of the last time step in the input sequence as an output or forecast value. This output is then compared with the labels in the loss function. This process is iterated over epochs to learn a plausible model.

OPTIMIZATION OVER EPOCHS

Following the preprocessing steps, the input data are prepared for training of ICONET. To load the input sequences into the training step, I employed the Pytorch Lightning Dataloader with four workers (subprocesses) per GPU. Increasing the number of workers enables asynchronous data loading, facilitating overlap between training and data loading, which opti-

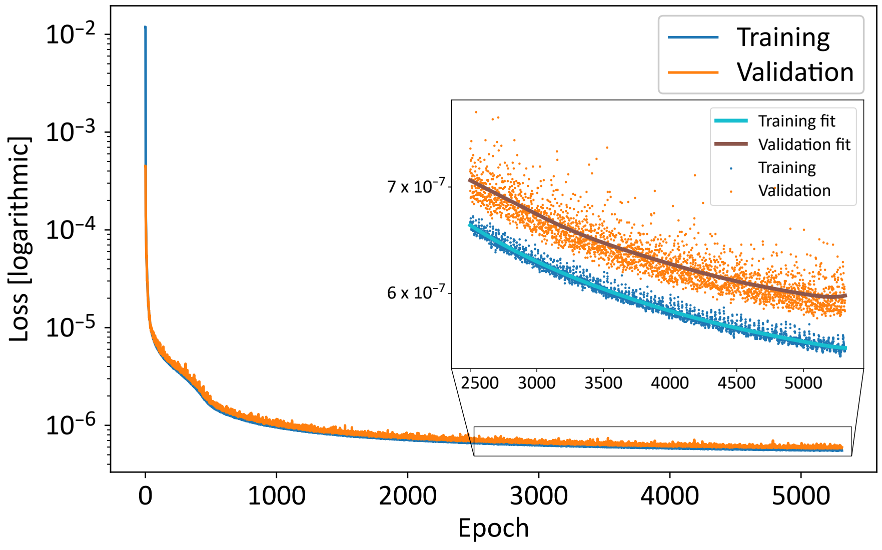


Figure 6.4: Training and validation loss per epoch. Taken from Azmi et al. (2023).

mizes the training performance. While the Dataloader loads the next sequences and labels into the training step, the multi-feature LSTM of ICONET generates an output $\widehat{Y}$ (Equation 6.2). The loss is calculated between the ground truth Y and the model output $\widehat{Y}$ using *MSE*. The loss value is minimized in several iterations (Epochs) of adjusting the weights and biases of the model.

To avoid overfitting, I employed early stopping, a regularization strategy that determines the optimal point to stop training, ensuring the model's ability to generalize to larger or unseen feature sets (Patterson & Gibson, 2017). Figure 6.4 displays the epoch loss (*MSE*) during the model's training on both the training and validation feature sets. In the zoomed-in portion of Figure 6.4, it becomes evident that the gap between the training and the validation fit curves begins to widen around epoch 3500, indicating that the model is not learning anymore.

HYPERPARAMETER TUNING

I conducted a greedy search over a range of values to find a suitable combination of hyperparameter values for achieving an accurate forecast. For instance, I varied the number of LSTM layers between one and four. While more layers perform better in reproducing the trend of the curves compared to a single layer, they produce a higher oscillation around the ground truth (not smoothed curves), without significant performance and efficiency improvements.

Table 6.1: Hyperparameters of ICONET training.

Hyperparameter	**Search range**	**Final selection**
Learning rate	$1 \times 10^{-3} - 1 \times 10^{-2}$	3×10^{-3}
Batch size	$239 - 210$	230 (samples)
Sequence length	$1 - 30$	10 time steps (1 hour)
# Epochs	$500 - 14000$	3500
# LSTM hidden layers	$1 - 4$	1
# LSTM hidden states	$10 - 40$	15
# Features in loss function	12, 15	15 (all chemical and physical features)
λ for mass conservation	$0.5 - 1.0$	1
Feature scaling	Normalization, Standardization	Standardization
Feature resolution unification	yes, no	yes
Spatial subset (# grid cells)	$15 - 180$	126 for training, 54 for validation
Temporal subset	24, 365	24 days (2 days per month in 1 year)

Furthermore, the training and forecasting execution times increase notably compared to using only one layer. The search range and final hyperparameters are listed in Table 6.1.

One of the main objectives during the development of the ICON simulation was to achieve improved mass conservation compared to other meteorological models (Zängl et al., 2022). Mass conservation should be given for every closed system in chemistry, which in this use case, refers to a single grid cell without the transportation of trace gases. Any surrogate model or simulation modification should preserve the quantities that are essential for the model accuracy. In this use case, from all considered trace gases, only the VMR of Nitrogen (N) remains constant and is conserved throughout the simulation period (Figure 6.5). The VMR of Hydrogen (H) and Oxygen (O) are not conserved because there is no closed system available for H_2O, which influences the VMR of H and O. I calculate the VMR of N using Equation 6.1.

$$C_N = \sum_{g=1}^{12} n_g \cdot C_g, \qquad (6.1)$$

where C_N is the VMR of N in $\frac{mol}{mol}$, n_g is the number of N atoms in each molecule of a trace gas and C_g is the VMR of trace gas g in $\frac{mol}{mol}$. To assess the quality of mass conservation in ICONET forecast, I calculated the VMR of N in the forecast values generated by a trained model with

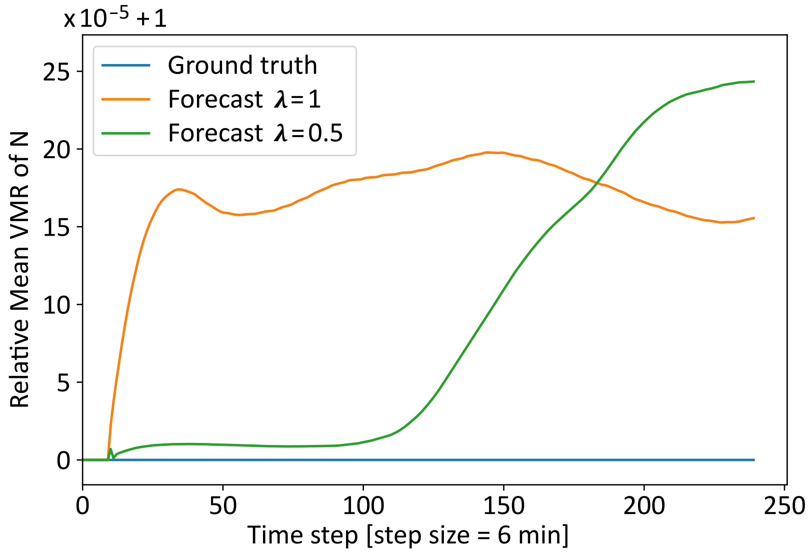


Figure 6.5: Mean VMR of N for all grid cells relative to ground truth of the middle stratospheric vertical model level on 12. September 2014. Taken from Azmi et al. (2023).

a variable λ in the loss function, formally expressed as follows:

$$Loss(Y, \widehat{Y}) = \lambda \cdot MSE(Y, \widehat{Y}) + (1 - \lambda) \cdot MSE(C_N, \widehat{C_N}), \tag{6.2}$$

where Y is the ground truth, $\widehat{Y}$ is the forecast, C_N is the VMR of N in $\frac{mol}{mol}$ calculated from the ground truth, $\widehat{C_N}$ is the VMR of N in $\frac{mol}{mol}$ calculated from the forecast, λ is a hyperparameter for penalizing the deviation from mass conservation and MSE is the mean squared error (Equation 2.3). The loss function consists of two parts, the loss of the forecast and of the mass conservation. I varied λ in the range of 0.5 to 1.0 and evaluated the test results from two aspects. First, I assessed the stability of the VMR of N, and second, I examined the quality of the forecast. Figure 6.5 demonstrates that the conservation of N in the forecast, using the loss function with $\lambda = 0.5$, closely aligns with the ground truth for the first 100 time steps, in contrast to the forecast using $\lambda = 1.0$. However, the forecast of trace gases using the loss function with $\lambda = 0.5$ exhibits higher $RMSE$ values compared to those of the loss function with $\lambda = 1.0$. The VMR of N for $\lambda = 1.0$ raises directly at the beginning of the forecast, but it remains stable during the test day. Analyzing the test results with different λ values revealed that the VMR of N varies during the test day when using λ values except 1.0. Therefore, I concluded to set λ to 1.0 and, thereby, removed the second part of the loss function.

I trained and validated ICONET using a subset of 180 randomly located grid cells in total.

I used 70 % of this subset for training and the remaining 30 % for validation. The temporal subset for training and validation contains in total 24 days distributed over one year at 15-day intervals. Finally, I saved the trained model at the epoch 3500 where the validation epoch loss starts to converge.

6.1.3 FORECASTING

After training the model on a small subset of the stratospheric grid cells, it can be used to forecast all twelve trace gases for all grid cells of any stratospheric vertical model level on any date of a year. To test its performance, I utilized the trained model to forecast all twelve trace gases for all grid cells of one vertical model level in the middle stratosphere on 12. September 2014, hence one year after the training feature set. This is an exemplary spatio-temporal case for testing, referred to as the test case. The three physical features serve as input features for the atmospheric chemistry simulation, and therefore, they are not included in the forecast values. The preprocessing of the test case follows the same procedures as for the training feature set. In the forecasting step, I input all features of ten time steps (sequence length) to ICONET and forecast only the trace gases for the next time step. Afterward, I use the forecast values as input, thereby the next time step is forecasted. In the remainder of this chapter, I use the term "forecast" for this iterative forecast.

6.1.4 POST-PROCESSING

In the post-processing step, I transform the standardized output of the neural network back to the physical units by undoing the standardization (Equation 5.1) applied during the preprocessing. Forecasts with very small values compared to the distribution of the training data might get transformed back to negative values. Since VMR values must be positive numbers, I set these negative nearly zero values to zero.

6.2 EVALUATION

In the following, I present the forecasts resulting from the application of the trained ICONET to the test case and evaluate them in terms of performance and computational complexity.

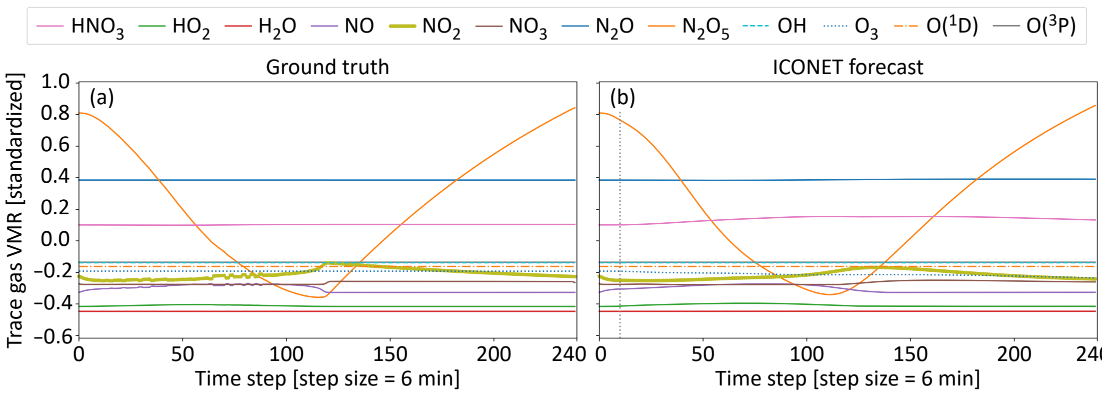


Figure 6.6: (a) ICON-ART simulation output (ground truth) and (b) ICONET forecast for VMR of all trace gases in the exemplary grid cell on 12 September 2014. The vertical dotted line in (b) indicates the start time step of forecasting. Taken from Azmi et al. (2023).

6.2.1 METRICS

ICONET is a regression model which outputs continuous variables. Therefore, I employ an evaluation metric appropriate to assess regression models. Considering the sensitivity to outliers and other disadvantages of the metrics discussed in Section 2.2.1, I use two metrics, alone and in combination, to evaluate the performance of the model. Both metrics measure how closely the forecast values align with the simulation outputs (ground truth). The first metric is *RMSE* (better when closer to zero) and the second one is *KGE* (better when closer to one). I present the *RMSE* values in both the original and standardized scales. The *RMSE* values in the original scale aids interpretation for domain scientists, whereas the standardized values allow comparison across different variables on a unified scale.

In addition to the performance metrics, the computational complexity of ICONET is demonstrated by its run time speed-up compared to the run time of the atmospheric chemistry simulation of ICON-ART, both using the same computing resources.

6.2.2 RESULTS AND DISCUSSION

In the following, I present the results of forecasting the test case and its evaluation. Figure 6.6 exhibits the standardized VMR of all trace gases in the exemplary grid cell during the test day. All trace gases in the forecast show a plausible fit to the ground truth values. Additionally, the shapes of the forecast curves appear smoothed compared to the ground truth.

To illustrate the deviation of the test results from the ground truth across all grid cells, I

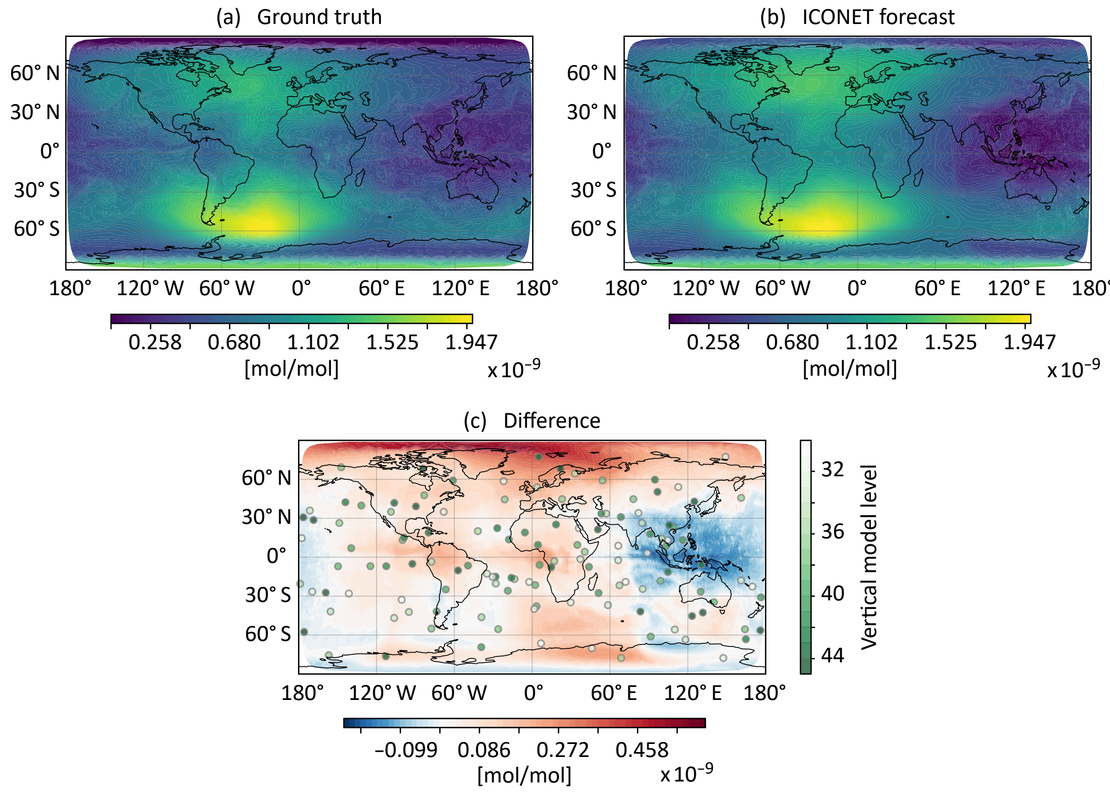


Figure 6.7: N_2O_5 VMR for all grid cells in an arbitrary time step. Points on (c) show the location, and their colors show the vertical model levels (45 to 30) of the trained grid cells. Taken from Azmi et al. (2023).

depict N_2O_5 values as an example in Figure 6.7. The values represent the VMR of N_2O_5 at each grid cell in an arbitrary time step of the test day. The difference map demonstrated in Figure 6.7(c) exhibits numerous grid cells where the difference is nearly zero (less than $\pm 10^{-11} \frac{mol}{mol}$) displayed in white, while the highest differences (in red) primarily appear in polar regions. The points on Figure 6.7(c) highlight the locations, and their colors denote various vertical model levels of the trained grid cells. As depicted, I trained a subset of randomly distributed grid cells from different stratospheric vertical model levels.

I present and interpret the *RMSE* values from two perspectives. First, I evaluate the *RMSE* distribution of all grid cells for each trace gas during the test day in one diagram (Figure 6.8). Second, I analyze these *RMSE* values, which are illustrated in a map view for each trace gas during the test day presented in Appendix B.

From the first perspective, I compared the performance of ICONET forecast between dif-

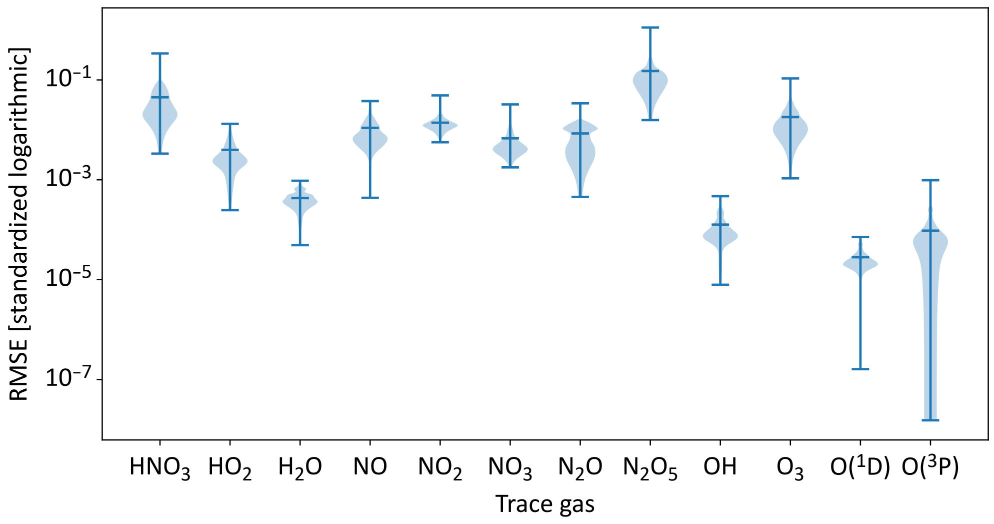


Figure 6.8: *RMSE* distributions for all grid cells of the test case. The horizontal lines depict the minimum, mean, and maximum values. Each shaded area corresponds with the approximate frequency of values. Taken from Azmi et al. (2023).

ferent trace gases (Figure 6.8). In this diagram, N_2O_5 exhibits the highest mean *RMSE* value (0.15), while $O(^1D)$ has the lowest value (2.8×10^{-5}) among the others (Appendix A, Table A.1). The *RMSE* values of $O(^3P)$ are very low but widely distributed, indicating the instability of the model forecast spatio-temporally (Figure 6.8). Furthermore, *RMSE* values for $O(^1D)$ display some outliers, concluding that the model fails to forecast this trace gas accurately in some grid cells.

From the second perspective, I compared the performance of ICONET forecast globally. The maps in Appendix B visualize the *RMSE* between the ground truth and ICONET forecast for VMR of all trace gases of the test case. These indicate mostly higher *RMSE* in polar regions. HNO_3, N_2O_5, O_3 and $O(^3P)$ show a similar spatial distribution of *RMSE* values. This demonstrates that the model exhibits varying performance across distinct spatial regions.

Another metric that I used in the performance evaluation is *KGE*. Figure 6.9 shows a similar diagram as Figure 6.8, illustrating *KGE* values. Among these values, described in Table A.2 from Appendix A, NO_2 shows the best mean *KGE* value (0.71), whereas $O(^3P)$ exhibits the lowest mean value (-3213.45). Furthermore, I compared the *KGE* and *RMSE* results and their relation with each other. Both metrics present a plausible forecast for HO_2, NO, NO_2, and NO_3 with high *KGE* and low *RMSE* values (Appendix A), which demonstrates the model's ability to learn well and forecast plausibly for these trace gases. Despite very low *RMSE* values, OH, $O(^1D)$, and $O(^3P)$ exhibit low *KGE* values. Although the re-

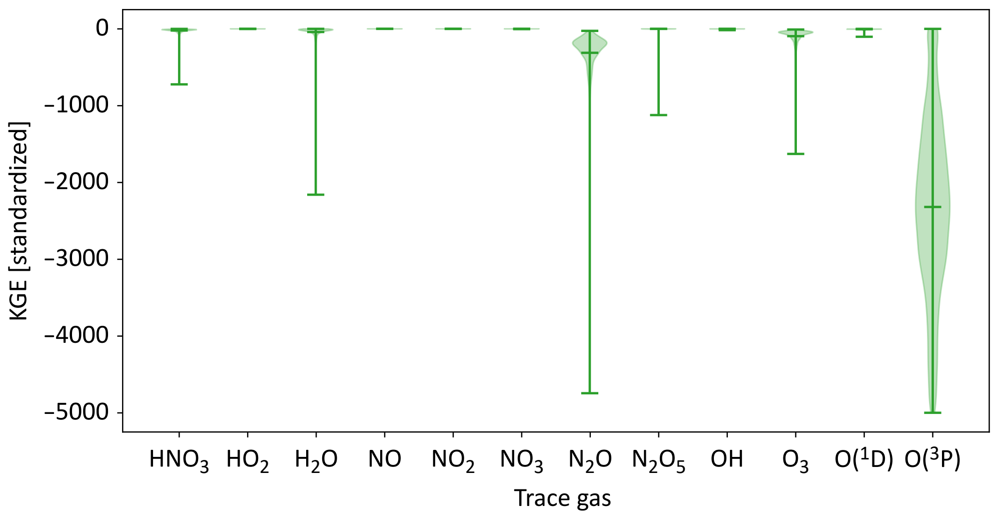


Figure 6.9: *KGE* Distributions for all grid cells of the test case. The horizontal lines depict the minimum, mean, and maximum values. Each shaded area corresponds with the approximate frequency of values. Some outliers smaller than -5000 are removed from the diagram for better visualization of the values close to one. Taken from Azmi et al. (2023).

sults show an overall low *RMSE* and plausible fit for most of the trace gases, there are exceptions. For instance, N_2O shows plausible *RMSE* values but low *KGE* values with numerous outliers, indicating insufficient model learning for this trace gas. N_2O_5 demonstrates plausible *KGE* values in contrast to its high *RMSE* values. Additionally, $O(^3P)$ though a very low mean *RMSE* (9.6×10^{-5}), shows very low and highly scattered *KGE* values, concluding the model's inability to learn the trend of this variable.

During the evaluation of ICONET, I compared the model's forecasts with a simple persistence model as a reference. The persistence model forecasts the future value of a time series under the assumption that nothing changes between the current and the forecast time (Hong & Pula, 2020). This means that the values at time step $t + 1$ (forecast) are identical to those at the current time step t. As the persistence model forecasts account for the next time step only, I generated a comparable ICONET forecast for one time step without iteration, referred to as ICONET one-ts forecast. This forecast ensures a fair comparison with the persistence model. Figure 6.10 illustrates the ground truth values together with the forecast values of the ICONET and the persistence model. These values are the VMR of two trace gases N_2O_5 and N_2O from the exemplary grid cell on the test day. For N_2O_5, Figure 6.10(a) shows that both ICONET forecasts are as good as or better than the persistence model. Figure 6.10(b) shows that ICONET one-ts forecast has a very close results to the persistence model for N_2O, but ICONET iterative forecast does not align well with the other models.

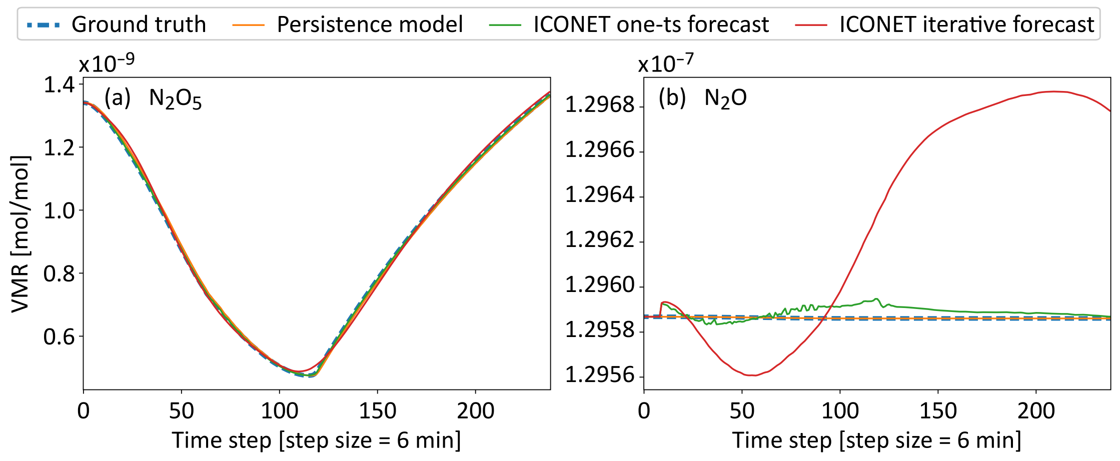


Figure 6.10: ICONET and persistence model forecasts vs. ground truth of two exemplary trace gases. Taken from Azmi et al. (2023).

In addition to the evaluation of ICONET in terms of performance, I quantify the computational complexity of the model with its run time speed-up over the ICON-ART simulation run time. I executed the atmospheric chemistry simulation of ICON-ART on one compute node without any parallelization, for forecasting of the same test case as for ICONET's forecasts. The test of forecasting all grid cells of one vertical model level during one day, resulted in the simulation run time of ~ 106 s, where the ICONET run time was ~ 34 s. In comparison, ICONET forecast showed 3.1 times speed-up over the simulation run time.

For evaluation of the stability of ICONET, I executed the ICONET forecast over one week (1670 time steps) in September 2014 and calculated the *RMSE* of all grid cells from the test case for each day separately. The daily mean of these *RMSE* values are presented in Figure 6.11. The presented *RMSE* values are calculated between the ground truth and forecast values on a standardized scale to facilitate a fair comparison between trace gases. To get an impression of the error on the physical units, I selected N_2O_5, which exhibited the highest *RMSE* value, as a reference for comparison. By comparing the ground truth values of N_2O_5 in a single time step on the test day depicted in Figure 6.7(a) and the mean *RMSE* value of the same day in physical units, as illustrated in Figure B.2(b) in Appendix B, it reveals an approximate error of 10%. This indicates the model's overall effectiveness. Figure 6.11 illustrates that the *RMSE* values of most trace gases remain roughly constant after the third day. This means that ICONET forecast remains stable for eight of twelve tested trace gases. However, *RMSE* values of NO_2, N_2O, N_2O_5, and O_3 increase linearly until the third day, followed by a gradual slowing down of the values' growth toward stability. For instance, Figure 6.12 depicts an

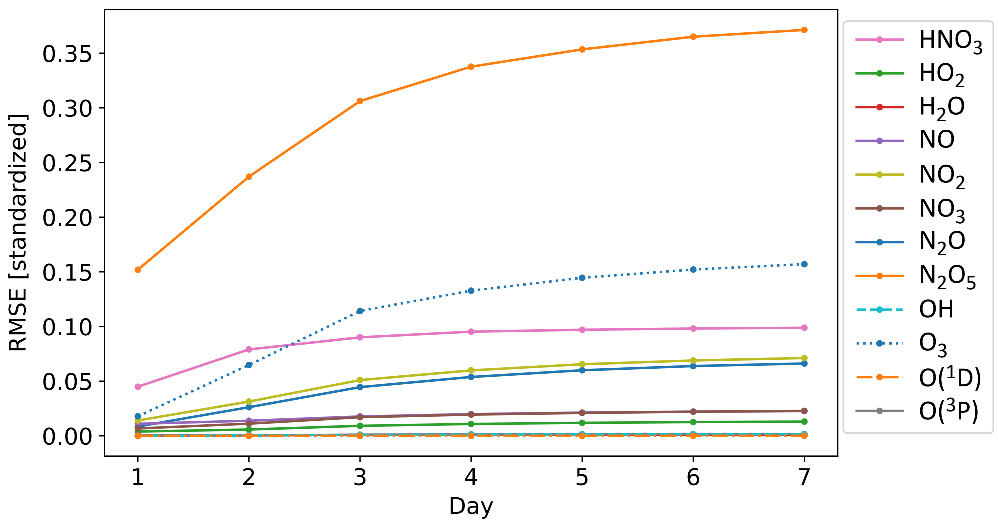


Figure 6.11: Evolution of mean *RMSE* value of all grid cells from the test case for all trace gases during one week. Taken from Azmi et al. (2023).

ICONET forecast over one week for the exemplary grid cell from the test case. The diagram shows a plausible fit and stable forecast of ICONET for most trace gases.

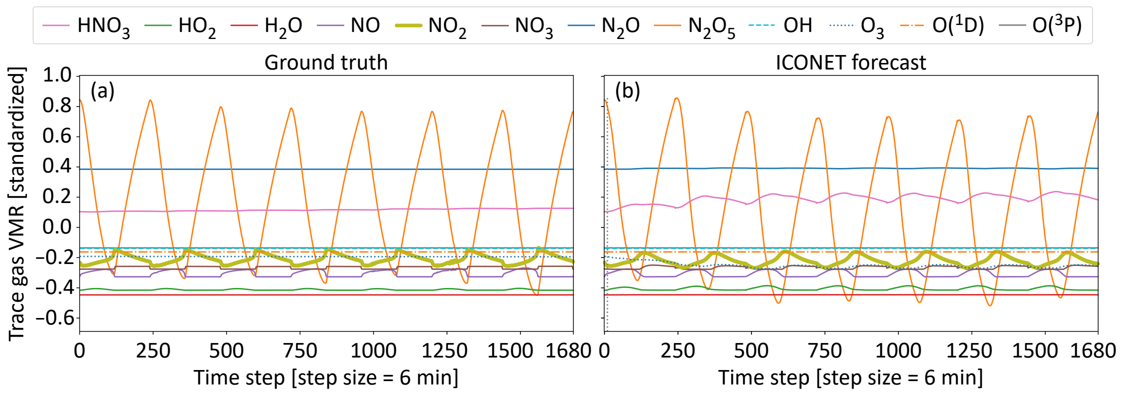


Figure 6.12: (a) ICON-ART simulation output (ground truth) and (b) ICONET forecast for VMR of all trace gases in the exemplary grid cell in one week. The vertical dotted line in (b) indicates the start time step of forecasting. Taken from Azmi et al. (2023).

6.3 GENERALIZATION

Neural networks offer immense potential for addressing a wide range of problems, however they may not always be the optimal choice for every use case. Particularly, they excel in areas like image and speech recognition, natural language processing, and pattern recognition, finding successful applications across various domains, including computer vision, finance, healthcare, climate research and more. Although neural networks offer versatile applications, establishing certain criteria is essential to determine the suitability of neural networks for a specific use case. It is important to evaluate various factors, including data availability, interpretability of the results, availability of computational resources, and the problem complexity. Keeping this in mind, I will discuss the generality of ICONET.

The first criterion to consider the applicability of ICONET in other use cases is data availability and their preprocessing. The data need to be in sequence format, such as time series. As the LSTM model used in the approach works with three-dimensional data, the intended use case data must have three dimensions or should be reasonably reshaped into the required dimensions. An advantage of this approach is that there is no need to label data manually or use any prepared feature set to train the model, instead the labels are extracted from the simulation data during the preprocessing steps.

The availability of computing resources is another factor to consider, before replacing an environmental simulation with a surrogate model. Training the model with more and various data helps to get more precise forecast results, necessitating more computing resources. However, the advantage of using ICONET is that HPC resources are required solely to train the model, and the forecasts could be executed without HPC resources.

The final important criterion is the problem complexity. While neural networks can address intricate problems, there are particular use cases where alternative methods are simpler and more suitable. As an instance, the ICON-ART chemistry simulation operates as a closed system without dynamics between grid cell, making the ICONET simply applicable to it. However, for environmental simulations with complex interaction between the model units, ICONET might need to be adjusted, e.g., the LSTM model should be replaced with another model such as convolutional LSTM to handle the units' neighborhood information. Additionally, the inherent complexity of the original model could make it challenging to fully capture all its details using a surrogate model. Consequently, ICONET might not perform as effectively in these situations.

6.4 IMPLEMENTATION ENVIRONMENT

I implemented ICONET in Python, mainly using PyTorch. In the implementation of LSTM, I used the RNN module in PyTorch (Paszke et al., 2019) wrapped by PyTorch Lightning (Falcon & The PyTorch Lightning team, 2019). This work was performed on the HoreKa supercomputer funded by the Ministry of Science, Research and the Arts Baden-Württemberg and by the Federal Ministry of Education and Research. It is also supported partially by the Helmholtz Association Initiative and Networking Fund on the HAICORE@KIT partition. For ICONET training, I used two Intel Xeon Platinum 8368 processors, 512 GB of main memory, four NVIDIA A100-40 GPUs of 40 GB memory with Red Hat Enterprise Linux (RHEL) 8.x. operating system. All the scripts and related data are available under the MIT license at `https://github.com/elnazazmi/iconet` (Azmi, 2023).

6.5 SUMMARY

High-resolution environmental simulations are compute-intensive and require a large amount of HPC resources. To approximate such simulations and reduce the computational complexity, and the resource demand, I developed a neural network-based approach (ICONET). This multi-feature LSTM model was developed on a use case for forecasting atmospheric chemistry. The model is applicable to simulations in any spatio-temporal resolution. ICONET consists of several steps namely, preprocessing, training, forecasting and post-processing.

The evaluation of ICONET, applied to a dataset generated from a meteorological simulation (ICON-ART), demonstrates an acceptable approximation of environmental simulations, enabling an accelerated forecasting. The forecasts of the use case exhibit low *RMSE* values and partially a plausible fit (high *KGE*) with the ground truth. The ICONET one-ts forecast displays improvements over the persistence model for certain trace gases. While the ICONET iterative forecast results in a plausible fit with the ground truth, it is naturally incomparable with the persistence model. Over a one-week forecast period, ICONET maintained an overall stability in the mean *RMSE* values, with the highest deviation of the first day by approximately 10%. Despite the iterative forecast causing cumulative deviations in the subsequent days, the short-lived trace gases consistently maintained stable *RMSE* values over the span of one week. This implies a plausible performance by ICONET.

In addition to the performance of the ICONET forecast, I evaluated the computational complexity of ICONET as its run time speed-up in comparison to the run time of the ICON-ART simulation. ICONET forecast showed a 3.1 times speed-up over the run time of the ICON-ART atmospheric chemistry simulation. Considering the need of ensemble simulations, even a small speed-up over the original simulation is a significant achievement.

It should be considered that significant modifications in the input feature set and the use case require redoing the training steps and tuning of the hyperparameters. Additionally, there is a potential for programmatically optimizing the preprocessing and training of ICONET, to be able to train a larger subset of grid cells and improve the accuracy of the forecast. The relative mass conservation error seems reasonable in the presented test case, but for longtime simulations, a more in-depth study should be done. This work is a proof of concept, and it has not been tested yet in an operational system.

The whole of science is nothing more than a refinement of everyday thinking.

Albert Einstein

7 Conclusion

This chapter summarizes the achievements of this thesis and its contributions to science, along with outlining a few directions of possible future research.

7.1 SUMMARY

The primary goal of this thesis was to develop an effective and computationally efficient approximation of large-scale and compute-intensive environmental simulations using machine learning methods that yields outputs acceptable for domain scientists.

Despite the availability of powerful computing resources and technologies, challenges persist as the resolution of the simulations and the number of observed features increase. The current favored approach, upscaling of these simulations through parallelization, faces its own set of hurdles. These include efficient parallelization of heterogeneous simulations, limitations in the availability of HPC resources, economical considerations, and notably, the rising energy consumption that raises environmental concerns.

This research integrates simulation, computational optimization, and machine learning to enhance the efficiency of simulations and facilitate compute-intensive simulations in high spatio-temporal resolutions. To address the forementioned challenges regarding compute-intensive environmental simulations in real-world scenarios, I proposed utilizing machine learning methods to identify patterns and similarities within model units of simulations, aiming to simplify the computational complexity of simulations, thereby reducing their resource demand.

Diverse objectives and intricacies of simulations underscore the need to establish a universal approach for evaluating them. I developed a general approach to evaluate simulations based on their performance and computational complexity, which can be applied across various domains. This approach evaluates the simulation's agreement with observations from related real-world systems or other simulations, and its resource consumption in terms of bits, employing principles from information theory.

To develop an approximation of compute-intensive simulations, firstly I investigated the application of unsupervised machine learning methods to identify similarities within simulation model units. By employing clustering methods on static properties of model units and simulating only representative model units of each cluster, the computational complexity of simulations is significantly reduced. Furthermore, based on the requirements of scientists, the extent of this approximation and optimization is adjustable by the number of clusters applied, and the corresponding computation time required for the selected representative from each cluster. This study led to the development of a K-determiner, which selects an appropriate number of clusters automatically. This method utilizes the balance between *K*, *RMSE* of each cluster member and a selected representative of that cluster, along with the computation time of representatives.

Secondly, I introduced an approximation approach, referred to as evolutionary approach, which is integrated into the original simulation. It utilizes K-means clustering to group similar model units and subsequently simulates the selected representatives from each cluster, scaling their output to the remaining cluster members. Throughout the simulation, these representatives are adjusted in accordance with the updated clusters, which are determined through redoing the clustering based on the dynamics imposed on the model units. Application of the evolutionary approach to a hydrological simulation resulted in achieving speed-up while maintaining low deviations from the original simulation.

To investigate the generality of the evolutionary approach across other environmental simulations, I applied it to a further hydrological and a meteorological simulation of atmospheric chemistry. In the evaluated hydrological scenario, the approach exhibits ample performance and speed-up when employing the clustering once with an appropriate number of clusters. However, in the atmospheric chemistry simulation, the clustering of millions of model units during the simulation hinders its optimization. Finally, this study revealed key criteria for determining appropriate simulations to employ the evolutionary approach efficiently. The simulation model units to be clustered should function individually without interacting with each other and exhibit similarities in properties or functionality. Furthermore, individual model units should be compute-intensive, ensuring that the simulation execution time of the representative model units, when combined with the overhead of the evolutionary approach, is shorter than that of the original simulation.

For simulations that do not meet the criteria for the evolutionary approach, I explored the use of supervised machine learning methods to approximate and optimize. I introduced a surrogate neural network model to replace compute-intensive simulations with forecasts generated by a trained model, utilizing the output data from a numerical simulation for training. The application of this approach to the atmospheric chemistry simulation resulted in achieving a speed-up while deviations from the original simulation were low and acceptable for domain scientists.

In conclusion, in this thesis, I demonstrated that utilizing similarities of model units to approximate complex simulations can be used reliably in practice through the proposed machine learning approaches. The proposed methods reduce the usage of computing resources, yielding overall acceptable results.

7.2 OUTLOOK

I think that the work presented in this thesis can provide a strong foundation for future research in several directions.

Utilizing the similarities within simulation units or properties has the potential to achieve a plausible approximation and optimize resource consumption. Introducing approximation approaches to find an approximate solution, rather than an exact one, can significantly reduce computational complexity while maintaining reasonable accuracy. Efficient clustering algorithms could be developed, or existing algorithms could be extended to reduce the computational complexity of clustering. Enhancing the computational efficiency of clustering algorithms without relying solely on parallelization is an active area of research for enabling efficient data processing, especially when parallel computing resources might not be readily available or applicable to the problem at hand. Within the evolutionary approach, this could help to reduce the overhead of clustering alongside simulations, resulting in a significant reduction in computational complexity of simulations.

There is extensive research potential in developing surrogate models to replace compute-intensive simulations. Future research should explore the utilization of other neural network models such as transformers, physics-informed and graph-based models, for approximation and optimization of similar simulations. Transformers have shown promise in capturing intricate relationships within data across various domains. Their attention mechanism holds potential for performant forecasting in time series through effective training of complex relationships in environmental simulations. Physics-informed neural networks incorporate established physical laws into the model architecture, thereby enhancing accuracy through imposing constraints derived from underlying physics principles. Graph-based models are adept at handling interconnected data by representing relationships between various components in a simulation, which makes them suitable for addressing problems involving complex interactions or networks.

The proposed approaches can be applied to various simulation scenarios and larger feature sets to create a robust forecasting approach. Expanding the training dataset allows for a more comprehensive evaluation of the approach's adaptability and accuracy across various conditions.

Finally, although the proposed approximation approach using supervised machine learning has proven promising results, it is not in operational use. Further comprehensive research and development is required to integrate the approach in an operational implementation. This could involve refining the model, training it with more diverse and representative data, improving its accuracy and reliability, and ensuring it meets the necessary performance standards for real-world application.

A
RMSE and KGE Tables

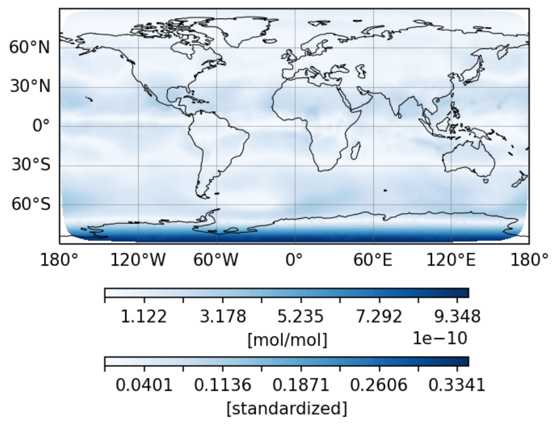
Table A.1: *RMSE* Distributions for all grid cells of the test case. The good values are highlighted.

Feature	Mean	Standard deviation	Min	Lower quartile	Median	Upper quartile	Max
HNO_3	0.045	0.032	0.003	0.022	0.038	0.061	0.339
HO_2	0.004	0.002	0.000	0.002	0.003	0.005	0.013
H_2O	0.000	0.000	0.000	0.000	0.000	0.001	0.001
NO	0.011	0.006	0.000	0.006	0.009	0.014	0.038
NO_2	0.014	0.005	0.006	0.011	0.013	0.015	0.049
NO_3	0.007	0.005	0.002	0.004	0.005	0.008	0.032
N_2O	0.008	0.005	0.000	0.004	0.009	0.012	0.034
N_2O_5	0.151	0.127	0.016	0.082	0.117	0.162	1.121
OH	0.000	0.000	0.000	0.000	0.000	0.000	0.000
O_3	0.018	0.012	0.001	0.010	0.015	0.023	0.107
$O(^1D)$	0.000	0.000	0.000	0.000	0.000	0.000	0.000
$O(^3P)$	0.000	0.000	0.000	0.000	0.000	0.000	0.001

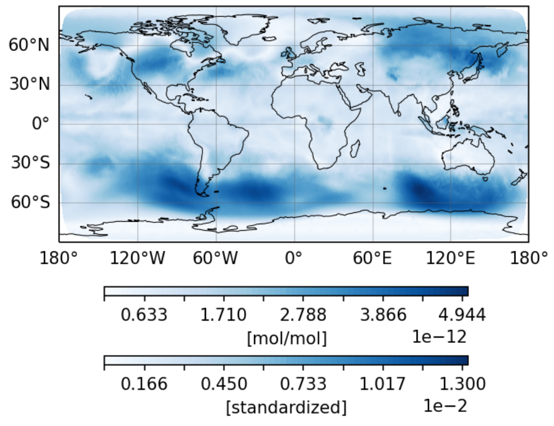
Table A.2: *KGE* Distributions for all grid cells of the test case. The best values are highlighted.

Feature	Mean	Standard deviation	Min	Lower quartile	Median	Upper quartile	Max
HNO_3	−21.66	46.79	−2539.39	−24.27	−14.41	−8.33	0.30
HO_2	0.23	3.90	−216.94	0.16	0.44	0.65	1.00
H_2O	−51.15	116.50	−3619.31	−49.60	−19.50	−7.50	0.88
NO	0.68	6.90	−474.50	0.76	0.85	0.91	0.99
NO_2	0.71	3.45	−271.70	0.76	0.85	0.91	0.98
NO_3	0.34	5.15	−266.79	0.48	0.71	0.82	0.95
N_2O	−395.56	551.02	−21 079.60	−431.87	−253.19	−171.98	−25.45
N_2O_5	−1.32	21.57	−1202.03	0.01	0.50	0.73	1.00
OH	−1.60	27.67	−1508.20	−1.00	−0.33	0.05	0.86
O_3	−187.52	2165.97	−121 549.99	−125.73	−69.37	−43.48	−6.10
$O(^1D)$	−6.32	8.42	−213.70	−7.93	−4.38	−2.98	0.48
$O(^3P)$	−3213.45	6181.22	−515 777.22	−3740.63	−2528.71	−1689.24	0.37

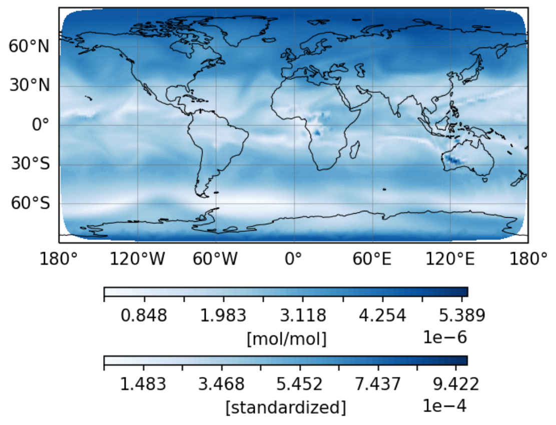
B
Global RMSE Maps



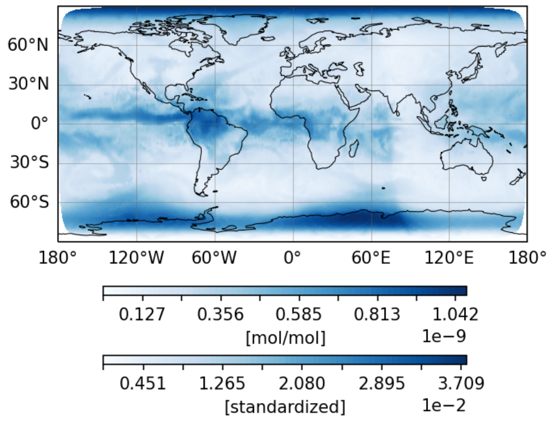
(a) HNO_3



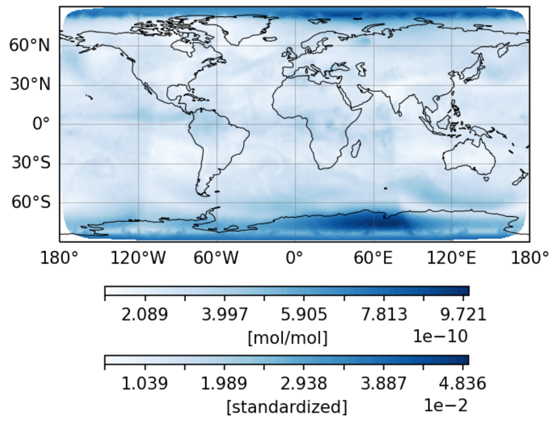
(b) HO_2



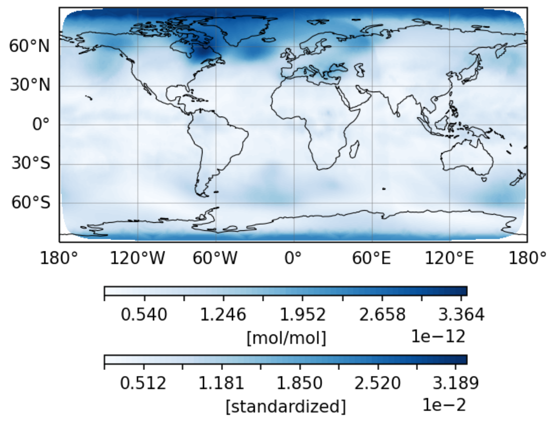
(c) H_2O



(d) NO

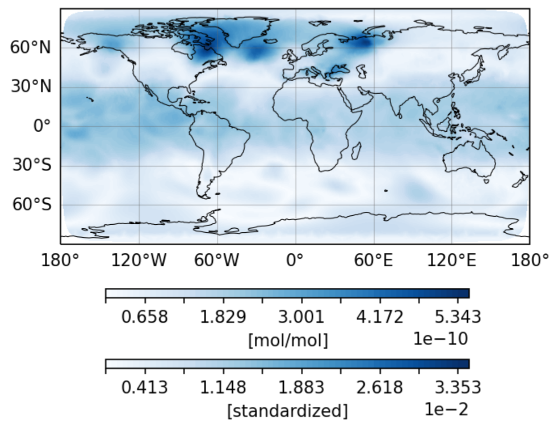


(e) NO_2

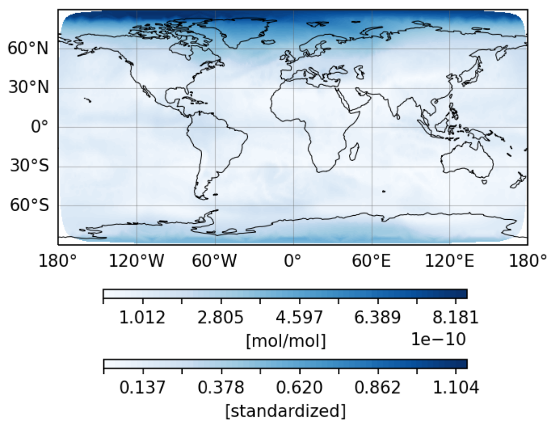


(f) NO_3

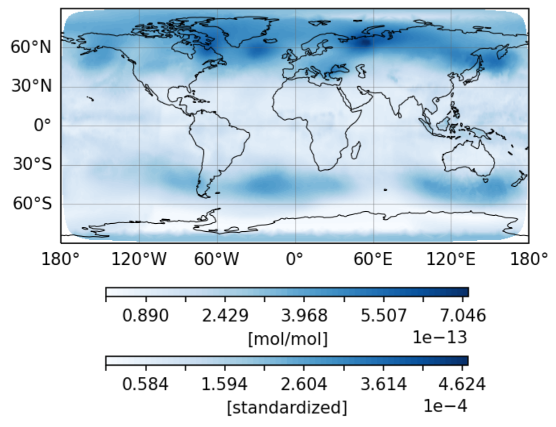
Figure B.1: *RMSE* between ground truth and ICONET forecast for VMR of trace gases in the test case (part 1).



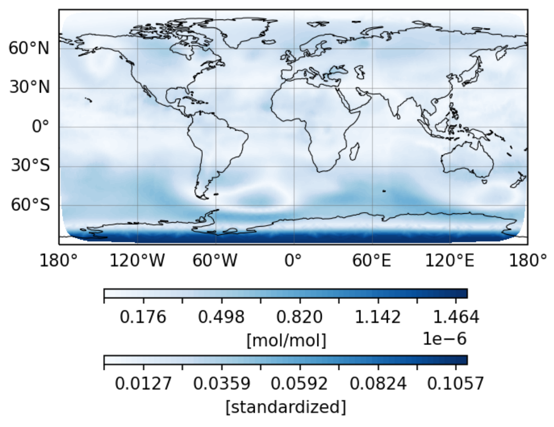
(a) N_2O



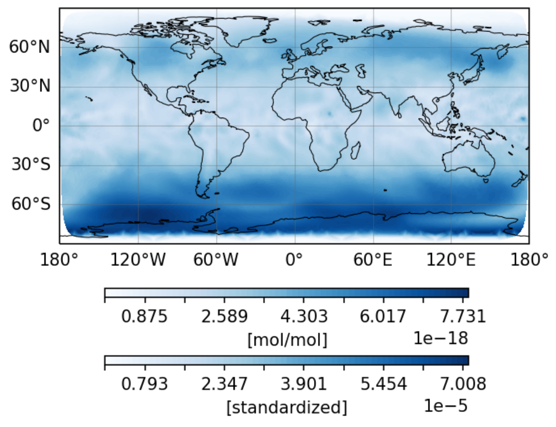
(b) N_2O_5



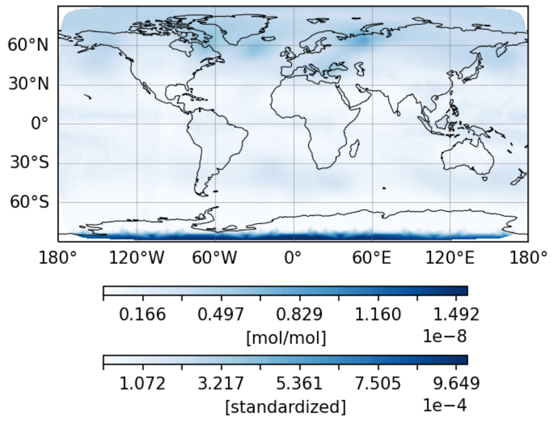
(c) OH



(d) O_3



(e) $O(^1D)$



(f) $O(^3P)$

Figure B.2: *RMSE* between ground truth and ICONET forecast for VMR of trace gases in the test case (part 2).

References

Abramowitz, M. & Stegun, I. A. (1972). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, volume 55. US Government printing office. https://digital.library.unt.edu/ark:/67531/metadc40302.

Agarwal, S., Tosi, N., Kessel, P., Breuer, D., & Montavon, G. (2021). Deep learning for surrogate modeling of two-dimensional mantle convection. *Physical Review Fluids*, 6. doi:10.1103/PhysRevFluids.6.113801.

Aggarwal, C. & Reddy, C. (2013). *Data clustering: algorithms and applications*. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. Taylor & Francis. ISBN: 9781466558212.

Aghabozorgi, S., Shirkhorshidi, A. S., & Wah, T. Y. (2015). Time-series clustering–A decade review. *Information Systems*, 53, 16–38. doi:10.1016/j.is.2015.04.007.

Aho, A. V., Hopcroft, J. E., & Ullman, J. D. (1975). The Design and Analysis of Algorithms.

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6), 716–723. doi:10.1109/TAC.1974.1100705.

Albrecht, F., Stiehler, F., Sinnhuber, B.-M., Versick, S., & Weigel, T. (2021). AI for Fast Atmospheric Chemistry. In *EGU General Assembly Conference Abstracts*. doi:10.5194/egusphere-egu21-4570.

Ali, G., Tetzlaff, D., Soulsby, C., McDonnell, J. J., & Capell, R. (2012). A comparison of similarity indices for catchment classification using a cross-regional dataset. *Advances in Water Resources*, 40, 11–22. doi:10.1016/j.advwatres.2012.01.008.

Alonso, G., Del Valle, E., & Ramirez, J. R. (2020). *Desalination in Nuclear Power Plants*. Woodhead Publishing. doi:10.1016/C2019-0-01164-8.

Alspaugh, S. (2018). k-Medoids Clustering. https://github.com/salspaugh/machine_learning/blob/master/clustering/kmedoids.py. (Accessed: March 4, 2024).

Álvarez-Farré, X., Gorobets, A., & Trias, F. X. (2021). A hierarchical parallel implementation for heterogeneous computing. Application to algebra-based CFD simulations on hybrid supercomputers. *Computers & Fluids*, 214, 104768. doi:10.1016/j.compfluid.2020.104768.

Amaran, S., Sahinidis, N. V., Sharda, B., & Bury, S. J. (2016). Simulation optimization: a review of algorithms and applications. *Annals of Operations Research*, 240, 351–380. doi:10.1007/s10479-015-2019-x.

Anderson, M. P., Woessner, W. W., & Hunt, R. J. (2015). *Applied Groundwater Modeling: Simulation of Flow and Advective Transport*. Academic press. doi:10.1016/C2009-0-21563-7.

Aral, M. M. (2010). *Principles of Environmental Modeling*, (pp. 37–61). Springer Netherlands: Dordrecht. doi:10.1007/978-90-481-8608-2_2.

Arkesteijn, L. & Pande, S. (2013). On hydrological model complexity, its geometrical interpretations and prediction uncertainty. *Water Resources Research*, 49(10), 7048–7063. doi:10.1002/wrcr.20529.

Arroyo, Á., Tricio, V., Corchado, E., & Herrero, Á. (2015). A Comparison of Clustering Techniques for Meteorological Analysis. In *10th International Conference on Soft Computing Models in Industrial and Environmental Applications* (pp. 117–130).: Springer. doi:10.1007/978-3-319-19719-7_11.

Arthur, D. & Vassilvitskii, S. (2007). K-Means++: The Advantages of Careful Seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07 (pp. 1027–1035).: Society for Industrial and Applied Mathematics. https://dl.acm.org/doi/10.5555/1283383.1283494.

Atkinson, S., Woods, R., & Sivapalan, M. (2002). Climate and landscape controls on water balance model complexity over changing timescales. *Water Resources Research*, 38(12), 50–1. doi:10.1029/2002WR001487.

Azmi, E. (2018a). Hydrological Data Analysis. https://github.com/elnazazmi/hyda. (Accessed: March 4, 2024).

Azmi, E. (2018b). On Using Clustering for the Optimization of Hydrological Simulations. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)* (pp. 1495–1496).: IEEE. doi:10.1109/ICDMW.2018.00215.

Azmi, E. (2020). Hydrological Data Analysis. https://github.com/elnazazmi/hyda. (Accessed: March 4, 2024).

Azmi, E. (2023). An ICON Neural Network based approach. https://github.com/elnazazmi/iconet. (Accessed: March 4, 2024).

Azmi, E. & Ehret, U. (2021). A Practical and General Approach for Model Evaluation. https://github.com/KIT-HYD/model-evaluation. (Accessed: March 4, 2024).

Azmi, E., Ehret, U., Meyer, J., van Pruijssen, R., Streit, A., & Strobl, M. (2019). Clustering as Approximation Method to Optimize Hydrological Simulations. In *European Conference on Parallel Processing* (pp. 256–269). doi:10.1007/978-3-030-29400-7_19.

Azmi, E., Ehret, U., Weijs, S. V., Ruddell, B. L., & Perdigão, R. A. (2021). Technical note: "Bit by bit": a practical and general approach for evaluating model computational complexity vs. model performance. *Hydrology and Earth System Sciences*, 25(2), 1103–1115. doi:10.5194/hess-25-1103-2021.

Azmi, E., Hassler, S., Mälicke, M., Meyer, J., Strobl, M., & Zehe, E. (2017). V-FOR-WaTer – Virtuelle Forschungsumgebung für die Wasser- und terrestrische Umweltforschung. In *E-Science-Tage 2017: Forschungsdaten managen*: heiDOK. doi:10.11588/heidok.00022889.

Azmi, E., Meyer, J., Strobl, M., Weimer, M., & Streit, A. (2023). Approximation and Optimization of Global Environmental Simulations with Neural Networks. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, PASC '23 New York, NY, USA: ACM. doi:10.1145/3592979.3593418.

Azmi, E., Strobl, M., van Pruijssen, R., Ehret, U., Meyer, J., & Streit, A. (2020). Evolutionary Approach of Clustering to Optimize Hydrological Simulations. In *International Conference on Computational Science and Its Applications* (pp. 617–633).: Springer. doi:10.1007/978-3-030-58799-4_45 [best paper award].

Balian, R. (2004). Entropy, a Protean Concept. In *Poincaré Seminar 2003 - Progress in Mathematical Physics*, volume 38 (pp. 119–144).: Birkhäuser Verlag, Basel. ISBN: 3-7643-7106-4.

Bărbulescu, A. (2016). *Studies on Time Series Applications in Environmental Sciences*, volume 103. Springer Cham. doi:10.1007/978-3-319-30436-6.

Barnston, A. G. (1992). Correspondence among the Correlation, RMSE, and Heidke Forecast Verification Measures; Refinement of the Heidke Score. *Weather and Forecasting*, 7(4), 699–709. doi:10.1175/1520-0434(1992)007%3C0699:CATCRA%3E2.0.CO;2.

Battiti, R. (1994). Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks*, 5(4), 537–550. doi:10.1109/72.298224.

Benner, P. & Faßbender, H. (2013). Model Order Reduction: Techniques and Tools. *Encyclopedia of Systems and Control*, (pp. 1–10). doi:10.1007/978-1-4471-5102-9_142-1.

Bennett, N. D., Croke, B. F., Guariso, G., Guillaume, J. H., Hamilton, S. H., Jakeman, A. J., Marsili-Libelli, S., Newham, L. T., Norton, J. P., Perrin, C., Pierce, S. A., Robson, B., Seppelt, R., Voinov, A. A., Fath, B. D., & Andreassian, V. (2013). Characterising performance of environmental models. *Environmental Modelling & Software*, 40, 1–20. doi:10.1016/j.envsoft.2012.09.011.

Bennett, N. D., Croke, B. F., Jakeman, A. J., Newham, L. T., & Norton, J. P. (2010). Performance evaluation of environmental models. In *International Congress on Environmental Modelling and Software*. https://scholarsarchive.byu.edu/iemssconference/2010/all/247.

Berger, M. J. & Colella, P. (1989). Local adaptive mesh refinement for shock hydrodynamics. *Journal of computational Physics*, 82(1), 64–84. doi:10.1016/0021-9991(89)90035-1.

Bergström, S. (1976). *Development and application of a conceptual runoff model for Scandinavian catchments*. Technical Report 7, Hydrology. ISSN: 0347-7827.

Bieniusa, A., Boehm, H.-J., Herlihy, M., & Petrank, E. (2018). New Challenges in Parallelism (Dagstuhl Seminar 17451). *Dagstuhl Reports*, 7(11), 1–27. doi:10.4230/DagRep.7.11.1.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford university press.

Bras, R. L. (2015). Complexity and organization in hydrology: A personal view. *Water Resources Research*, 51(8), 6532–6548. doi:10.1002/2015WR016958.

Carson, Y. & Maria, A. (1997). Simulation Optimization: Methods and Applications. In *Proceedings of the 29th conference on Winter simulation* (pp. 118–126). USA: IEEE Computer Society. doi:10.1145/268437.268460.

Castillo, A., Castelli, F., & Entekhabi, D. (2015). An entropy-based measure of hydrologic complexity and its applications. *Water resources research*, 51(7), 5145–5160. doi:10.1002/2014WR016035.

Chaitin, G. J. (1966). On the Length of Programs for Computing Finite Binary Sequences. *Journal of the ACM (JACM)*, 13(4), 547–569. doi:10.1145/321356.321363.

Chang, Y.-W., Hsieh, C.-J., Chang, K.-W., Ringgaard, M., & Lin, C.-J. (2010). Training and testing low-degree polynomial data mappings via linear svm. *Journal of Machine Learning Research*, 11(48), 1471–1490. http://jmlr.org/papers/v11/chang10a.html.

Cherkassky, V. & Mulier, F. M. (2007). *Learning from data: concepts, theory, and methods*. John Wiley & Sons. doi:10.1002/9780470140529.

Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20, 273–297. doi:10.1007/BF00994018.

Corzo, G. & Solomatine, D. (2007). Baseflow separation techniques for modular artificial neural network modelling in flow forecasting. *Hydrological Sciences Journal*, 52(3), 491–507. doi:10.1623/hysj.52.3.491.

Cover, T. M. & Thomas, J. A. (2006). *Entropy, Relative Entropy, and Mutual Information*, chapter 2, (pp. 13–55). John Wiley & Sons, Inc. doi:10.1002/047174882X.ch2.

Daetwyler, H. D., Villanueva, B., & Woolliams, J. A. (2008). Accuracy of Predicting the Genetic Risk of Disease Using a Genome-Wide Approach. *PLOS ONE*, 3(10), 1–8. doi:10.1371/journal.pone.0003395.

Daniel, T., Casenave, F., Akkari, N., & Ryckelynck, D. (2020). Model order reduction assisted by deep neural networks (ROM-net). *Advanced Modeling and Simulation in Engineering Sciences*, 7, 1–27. doi:10.1186/s40323-020-00153-6.

Darscheid, P., Guthke, A., & Ehret, U. (2018). A Maximum-Entropy Method to Estimate Discrete Distributions from Samples Ensuring Nonzero Probabilities. *Entropy*, 20(8). doi:10.3390/e20080601.

Daw, A., Karpatne, A., Watkins, W. D., Read, J. S., & Kumar, V. (2022). Physics-guided Neural Networks (PGNN): An Application in Lake Temperature Modeling. In *Knowledge Guided Machine Learning* (pp. 353–372). Chapman and Hall/CRC. doi:https://arxiv.org/abs/1710.11431.

Demirel, A., Niegemann, J., Busch, K., & Hochbruck, M. (2015). Efficient multiple time-stepping algorithms of higher order. *Journal of Computational Physics*, 285, 133–148. doi:10.1016/j.jcp.2015.01.018.

Ding, C. & He, X. (2004). K-means clustering via principal component analysis. In *Proceedings of the twenty-first international conference on Machine learning* (pp.2̃9).: ACM. https://icml.cc/Conferences/2004/proceedings.html.

Ding, S. & Yang, R. (2021). Reduced-order modelling of urban wind environment and gaseous pollutants dispersion in an urban-scale street canyon. *Journal of Safety Science and Resilience*, 2(4), 238–245. doi:10.1016/j.jnlssr.2021.09.001.

Dirac, P. A. M. & Fowler, R. H. (1927). The physical interpretation of the quantum dynamics. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 113(765), 621–641. doi:10.1098/rspa.1927.0012.

Dorier, M., Yildiz, O., Ibrahim, S., Orgerie, A.-C., & Antoniu, G. (2016). On the Energy Footprint of I/O Management in Exascale HPC Systems. *Future Generation Computer Systems*, 62, 17–28. doi:10.1016/j.future.2016.03.002.

Dubes, R. & Jain, A. K. (1976). Clustering techniques: the user's dilemma. *Pattern Recognition*, 8(4), 247–260. doi:10.1016/0031-3203(76)90045-5.

Dueben, P. D. & Bauer, P. (2018). Challenges and design choices for global weather and climate models based on machine learning. *Geoscientific Model Development*, 11(10), 3999–4009. doi:10.5194/gmd-11-3999-2018.

Ehret, U., van Pruijssen, R., Bortoli, M., Loritz, R., Azmi, E., & Zehe, E. (2020). Adaptive clustering: reducing the computational costs of distributed (hydrological) modelling by exploiting time-variable similarity among model elements. *Hydrology and Earth System Sciences*, 24(9), 4389–4411. doi:10.5194/hess-24-4389-2020.

Ehret, U., Zehe, E., Scherer, U., & Westhoff, M. (2014). Dynamical grouping and representative computation: a new approach to reduce computational efforts in distributed, physically based modeling on the lower mesoscale. In *AGU Chapman conference* (pp. 23–26).

Emmons, L. K., Walters, S., Hess, P. G., Lamarque, J. F., Pfister, G. G., Fillmore, D., et al. (2010). Description and evaluation of the Model for Ozone and Related chemical Tracers, version 4 (MOZART-4). *Geoscientific Model Development*, 3(1), 43–67. doi:10.5194/GMD-3-43-2010.

Engelhardt, S., Matyssek, R., & Huwe, B. (2009). Complexity and information propagation in hydrological time series of mountain forest catchments. *European journal of forest research*, 128, 621–631. doi:10.1007/s10342-009-0306-2.

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, volume 96 of *KDD'96* (pp. 226–231).: AAAI Press. https://dl.acm.org/doi/10.5555/3001460.3001507.

Eyring, V., Bony, S., Meehl, G. A., Senior, C. A., Stevens, B., Stouffer, R. J., & Taylor, K. E. (2016). Overview of the Coupled Model Intercomparison Project Phase 6 (CMIP6) experimental design and organization. *Geoscientific Model Development*, 9(5), 1937–1958. doi:10.5194/gmd-9-1937-2016.

Falcon, W. & The PyTorch Lightning team (2019). PyTorch Lightning. https://github.com/PyTorchLightning/pytorch-lightning. doi:10.5281/zenodo.3828935 (Accessed: March 4, 2024).

Fenicia, F., Kavetski, D., Savenije, H. H. G., Clark, M. P., Schoups, G., Pfister, L., & Freer, J. (2014). Catchment properties, function, and conceptual model representation: is there a correspondence? *Hydrological Processes*, 28(4), 2451–2467. doi:10.1002/hyp.9726.

Fenicia, F., Kavetski, D., Savenije, H. H. G., & Pfister, L. (2016). From spatially variable streamflow to distributed hydrological models: Analysis of key modeling decisions. *Water Resources Research*, 52(2), 954–989. doi:10.1002/2015WR017398.

Finger, D., Vis, M., Huss, M., & Seibert, J. (2015). The value of multiple data set calibration versus model complexity for improving the performance of hydrological models in mountain catchments. *Water Resources Research*, 51(4), 1939–1958. doi:10.1002/2014WR015712.

Fitzgerald, A. P., Kochunas, B., Stimpson, S., & Downar, T. (2019). Spatial decomposition of structured grids for nuclear reactor simulations. *Annals of Nuclear Energy*, 132, 686–701. doi:10.1016/j.anucene.2019.06.054.

Förster, K., Meon, G., Marke, T., & Strasser, U. (2014). Effect of meteorological forcing and snow model complexity on hydrological simulations in the Sieber catchment (Harz Mountains, Germany). *Hydrology and Earth System Sciences*, 18(11), 4703–4720. doi:10.5194/hess-18-4703-2014.

Fujimoto, R., Bock, C., Chen, W., Page, E., & Panchal, J. H. (2017). *Research Challenges in Modeling and Simulation for Engineering Complex Systems*. Springer. doi:10.1007/978-3-319-58544-4.

Fujimoto, R. M. (2016). Research Challenges in Parallel and Distributed Simulation. *ACM Trans. Model. Comput. Simul.*, 26. doi:10.1145/2866577.

Gan, T. Y., Dlamini, E. M., & Biftu, G. F. (1997). Effects of model complexity and structure, data quality, and objective functions on hydrologic modeling. *Journal of hydrology*, 192(1-4), 81–103. doi:10.1016/S0022-1694(96)03114-9.

Gell-Mann, M. (1995). What is complexity? Remarks on simplicity and complexity by the Nobel Prize-winning author of The Quark and the Jaguar. *Complexity*, 1(1), 16–19. doi:10.1002/cplx.6130010105.

Ghodsi, A. (2006). Dimensionality Reduction A Short Tutorial. *Department of Statistics and Actuarial Science, University of Waterloo, Ontario, Canada*, 37(38). https://people.computing.clemson.edu/~ekp/courses/cpsc9500/assets/pca2.pdf.

Gong, W., Yang, D., Gupta, H. V., & Nearing, G. (2014). Estimating information entropy for hydrological data: One-dimensional case. *Water Resources Research*, 50(6), 5003–5018. doi:10.1002/2014WR015874.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press. http://www.deeplearningbook.org.

Grayson, R. & Blöschl, G. (2001). *Spatial patterns in catchment hydrology: observations and modelling*. CUP Archive. ISBN: 9780521633161.

Grünwald, P. D. (2007). *The Minimum Description Length Principle*. MIT press. ISBN: 9780262529631.

Gupta, H. V., Kling, H., Yilmaz, K. K., & Martinez, G. F. (2009). Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling. *Journal of Hydrology*, 377(1-2), 80–91. doi:10.1016/j.jhydrol.2009.08.003.

Gupta, H. V., Wagener, T., & Liu, Y. (2008). Reconciling theory with observations: elements of a diagnostic approach to model evaluation. *Hydrological Processes: An International Journal*, 22(18), 3802–3813. doi:10.1002/hyp.6989.

Gupta, V., Mishra, V. K., Singhal, P., & Kumar, A. (2022). An Overview of Supervised Machine Learning Algorithm. In *2022 11th International Conference on System Modeling & Advancement in Research Trends (SMART)* (pp. 87–92).: IEEE. doi:10.1109/SMART55829.2022.10047618.

Hall, M. A. (1999). *Correlation-based feature selection for machine learning*. PhD thesis, The University of Waikato.

Han, J., Pei, J., & Kamber, M. (2012). *Data mining: concepts and techniques*. Elsevier. doi:10.1016/C2009-0-61819-5.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). Unsupervised learning. In *The elements of statistical learning* (pp. 485–585). Springer. doi:10.1007/978-0-387-84858-7_14.

Hautamaki, V., Nykanen, P., & Franti, P. (2008). Time-series clustering by approximate prototypes. In *19th International Conference on Pattern Recognition* (pp. 1–4).: IEEE. doi:10.1109/ICPR.2008.4761105.

Herzel, A., Ruzika, S., & Thielen, C. (2021). Approximation Methods for Multiobjective Optimization Problems: A Survey. *INFORMS Journal on Computing*, 33(4), 1284–1299. doi:10.1287/ijoc.2020.1028.

Hestenes, D. (1997). Modeling Methodology for Physics Teachers. In *International Conference on Undergraduate Physics Education (College Park, August 1996)*, volume 399 (pp. 935–958).: American Institute of Physics. doi:10.1063/1.53196.

Hochreiter, S. (1991). Untersuchungen zu dynamischen neuronalen Netzen. *Diploma, Technische Universität München*, 91(1).

Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780. doi:10.1162/neco.1997.9.8.1735.

Höge, M., Wöhling, T., & Nowak, W. (2018). A Primer for Model Selection: The Decisive Role of Model Complexity. *Water Resources Research*, 54(3), 1688–1715. doi:10.1002/2017WR021902.

Hong, Y.-Y. & Pula, R. (2020). Comparative Studies of Different Methods for Short-term Locational Marginal Price Forecasting. In *2020 5th International Conference on Green Technology and Sustainable Development (GTSD)* (pp. 527–532).: IEEE. doi:10.1109/GTSD50082.2020.9303121.

Hutton, C., Wagener, T., Freer, J., Han, D., Duffy, C., & Arheimer, B. (2016). Most computational hydrology is not reproducible, so is it really science? *Water Resources Research*, 52(10), 7548–7555. doi:10.1002/2016WR019285.

Hyndman, R. J. & Athanasopoulos, G. (2018). *Forecasting: principles and practice*. 2nd edtition, OTexts. https://otexts.com/fpp2/ Accessed: March 4, 2024.

Jacobson, M. Z. (1999). *Fundamentals of Atmospheric Modeling*. Cambridge university press. ISBN: 0-521-63717-1.

James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). *Statistical Learning*, (pp. 15–67). Springer International Publishing: Cham. doi:10.1007/978-3-031-38747-0_2.

Jenerette, G. D., Barron-Gafford, G. A., Guswa, A. J., McDonnell, J. J., & Villegas, J. C. (2012). Organization of complexity in water limited ecohydrology. *Ecohydrology*, 5(2), 184–199. doi:10.1002/eco.217.

Jones, J. E. & Woodward, C. S. (2001). Newton–Krylov-multigrid solvers for large-scale, highly heterogeneous, variably saturated flow problems. *Advances in Water Resources*, 24(7), 763–774. doi:10.1016/S0309-1708(00)00075-0.

Jovanovic, T., García, S., Gall, H., & Mejía, A. (2017). Complexity as a streamflow metric of hydrologic alteration. *Stochastic Environmental Research and Risk Assessment*, 31, 2107–2119. doi:10.1007/s00477-016-1315-6.

Kar, A. K., Goel, N., Lohani, A., & Roy, G. (2012). Application of clustering techniques using prioritized variables in regional flood frequency analysis-Case study of Mahanadi Basin. *Journal of Hydrologic Engineering*, 17(1), 213–223. doi:10.1061/(ASCE)HE.1943-5584.0000417.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., & Yang, L. (2021). Physics-informed machine learning. *Nature Reviews Physics*, 3(6), 422–440. doi:10.1038/s42254-021-00314-5.

Kashinath, K., Mustafa, M., Albert, A., Wu, J.-L., Jiang, C., Esmaeilzadeh, S., Azizzadenesheli, K., Wang, R., Chattopadhyay, A., Singh, A., Manepalli, A., Chirila, D., Yu, R., Walters, R., White, B., Xiao, H., Tchelepi, H. A., Marcus, P., Anandkumar, A., Hassanzadeh, P., & Prabhat, n. (2021). Physics-informed machine learning: case studies for weather and climate modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194), 20200093. doi:10.1098/rsta.2020.0093.

Kassambara, A. (2017). *Practical Guide to Cluster Analysis in R: Unsupervised Machine Learning*, volume 1. STHDA.

Kaufman, L. & Rousseeuw, P. J. (1987). Clustering by means of Medoids. In *Statistical data analysis based on the L1–norm and related methods, edited by Y. Dodge* (pp. 405–416).: North-Holland. ISBN: 0444702733.

Kaufman, L. & Rousseeuw, P. J. (2009). *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons.

Keogh, E. J. & Pazzani, M. J. (2000). A Simple Dimensionality Reduction Technique for Fast Similarity Search in Large Time Series Databases. In *Knowledge Discovery and Data Mining. Current Issues and New Applications* (pp. 122–133).: Springer. ISBN: 978-3-540-45571-4.

Kerr, L. A. & Goethel, D. R. (2014). Chapter Twenty One - Simulation Modeling as a Tool for Synthesis of Stock Identification Information. In S. X. Cadrin, L. A. Kerr, & S. Mariani (Eds.), *Stock Identification Methods* (pp. 501–533). Elsevier. doi:10.1016/B978-0-12-397003-9.00021-7.

Kiesling, T. (2005). Using Approximation with Time-Parallel Simulation. *SIMULATION*, 81(4), 255–266. doi:10.1177/0037549705051697.

Kirchner, J. W. (2006). Getting the right answers for the right reasons: Linking measurements, analyses, and models to advance the science of hydrology. *Water Resources Research*, 42(3), W03S04. doi:10.1029/2005WR004362.

Knuth, K. H. (2019). Optimal data-based binning for histograms and histogram-based probability density models. *Digital Signal Processing*, 95, 102581. doi:10.1016/j.dsp.2019.102581.

Kollet, S. J., Maxwell, R. M., Woodward, C. S., Smith, S., Vanderborght, J., Vereecken, H., & Simmer, C. (2010). Proof of concept of regional scale hydrologic simulations at hydrologic resolution utilizing massively parallel computer resources. *Water resources research*, 46(4), W04201. doi:10.1029/2009WR008730.

Kolmogorov, A. N. (1968). Three approaches to the quantitative definition of information. *International journal of computer mathematics*, 2(1-4), 157–168. doi:10.1080/00207166808803030.

Kranenburg, P., Lankester, B., & Sladkey, R. (2018). Strace - the linux syscall tracer. Version 4.24. Aug 14, 2018. https://github.com/strace/strace.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25. https://proceedings.neurips.cc/paper_files/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html.

Kullback, S. & Leibler, R. A. (1951). On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1), 79–86. doi:10.1214/aoms/1177729694.

Kumar, R. & Ezhilarasi, D. (2023). A state-of-the-art survey of model order reduction techniques for large-scale coupled dynamical systems. *International Journal of Dynamics and Control*, 11(2), 900–916. doi:10.1007/s40435-022-00985-7.

Kurth, T., Subramanian, S., Harrington, P., Pathak, J., Mardani, M., Hall, D., Miele, A., Kashinath, K., & Anandkumar, A. (2023). FourCastNet: Accelerating Global High-Resolution Weather Forecasting using Adaptive Fourier Neural Operators. In *Proceedings of the Platform for Advanced Scientific Computing Conference* (pp. 1–11). doi:10.1145/3592979.3593412.

Langbein, W. B. & Iseri, K. T. (1960). *General Introduction and Hydrologic Definitions, Manual of Hydrology: Part 1. General Surf ace-Water Techniques*. Geological Survey water-supply paper 1541-A. Washington: United States Government Printing Office. https://pubs.usgs.gov/wsp/1541a/report.pdf.

Lara-Benitez, P., Carranza-Garcia, M., & Riquelme, J. C. (2021). An Experimental Review on Deep Learning Architectures for Time Series Forecasting. *International Journal of Neural Systems*, 31(3). doi:10.1142/S0129065721300011.

Lassila, T., Manzoni, A., Quarteroni, A., & Rozza, G. (2014). *Model Order Reduction in Fluid Dynamics: Challenges and Perspectives*, (pp. 235–273). Springer International Publishing": Cham. doi:10.1007/978-3-319-02090-7_9.

Ley, R., Casper, M., Hellebrand, H., & Merz, R. (2011). Catchment classification by runoff behaviour with self-organizing maps (SOM). *Hydrology and Earth System Sciences*, 15(9), 2947–2962. doi:10.5194/hess-15-2947-2011.

Li, M. & Vitányi, P. (2008). *An Introduction to Kolmogorov Complexity and Its Applications*, volume 3. Springer. doi:10.1007/978-0-387-49820-1.

Liao, T. W. (2005). Clustering of time series data–a survey. *Pattern Recognition*, 38(11), 1857–1874. doi:10.1016/j.patcog.2005.01.025.

Linford, J. C. & Sandu, A. (2011). Scalable heterogeneous parallelism for atmospheric modeling and simulation. *The Journal of Supercomputing*, 56, 300–327. doi:10.1007/s11227-010-0380-8.

Lu, K., Zhang, K., Zhang, H., Gu, X., Jin, Y., Zhao, S., Fu, C., & Yang, Y. (2021). A Review of Model Order Reduction Methods for Large-Scale Structure Systems. *Shock and Vibration*, 2021, 1–19. doi:10.1155/2021/6631180.

MacKay, D. J. (2003). *Information Theory, Inference and Learning Algorithms*. Cambridge University Press. ISBN: 0521642981.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1 (pp. 281–297).: Oakland, CA, USA.

Magee, L. (1998). Nonlocal behavior in polynomial regressions. *The American Statistician*, 52(1), 20–22. doi:10.2307/2685560.

Maxwell, R., Condon, L., & Kollet, S. (2015). A high-resolution simulation of groundwater and surface water over most of the continental US with the integrated hydrologic model ParFlow v3. *Geoscientific model development*, 8(3), 923. doi:10.5194/gmd-8-923-2015.

McDonnell, J., Sivapalan, M., Vaché, K., Dunn, S., Grant, G., Haggerty, R., Hinz, C., Hooper, R., Kirchner, J., Roderick, M., et al. (2007). Moving beyond heterogeneity and process complexity: A new vision for watershed hydrology. *Water Resources Research*, 43(7). doi:10.1029/2006WR005467.

Merrill, C., Custer, R. L., Daugherty, J., Westrick, M., & Zeng, Y. (2008). Delivering Core Engineering Concepts to Secondary Level Students. *Journal of Technology Education*, 20(1), 48–64. doi:10.21061/jte.v20i1.a.4.

Meyer, J., Azmi, E., Hassler, S., Mälicke, M., Strobl, M., & Zehe, E. (2019). V-FOR-WaTer – the virtual research environment to discover and analyse environmental data. In *E-Science-Tage 2019: Data to Knowledge* (pp. 199–201).: heiBOOKS. doi:10.11588/heibooks.598.c8434.

Meyer, R. (2014). Efficient parallelization of molecular dynamics simulations with short-ranged forces. In *Journal of Physics: Conference Series*, volume 540 (pp. 012006).: IOP Publishing. doi:10.1088/1742-6596/540/1/012006.

Mitra, S., Kundu, P. P., & Pedrycz, W. (2012). Feature selection using structural similarity. *Information Sciences*, 198, 48–61. doi:10.1016/j.ins.2012.02.042.

Morán, M., Balladini, J., Rexachs, D., & Rucci, E. (2020). Towards Management of Energy Consumption in HPC Systems with Fault Tolerance. In *2020 IEEE Congreso Bienal de Argentina (ARGENCON)* (pp. 1–8).: IEEE. doi:10.1109/ARGENCON49523.2020.9505498.

Moriasi, D. N., Arnold, J. G., Van Liew, M. W., Bingner, R. L., Harmel, R. D., & Veith, T. L. (2007). Model Evaluation Guidelines for Systematic Quantification of Accuracy in Watershed Simulations. *Transactions of the ASABE*, 50(3), 885–900. doi:10.13031/2013.23153.

Murphy, K. P. (2012). *Machine learning : a probabilistic perspective*. Adaptive computation and machine learning series. Cambridge, Mass. [u.a.]: MIT Press. ISBN: 978-0-262-01802-9; 0-262-01802-0.

Nash, J. E. & Sutcliffe, J. V. (1970). River flow forecasting through conceptual models part I — A discussion of principles. *Journal of hydrology*, 10(3), 282–290. doi:10.1016/0022-1694(70)90255-6.

Nasteski, V. (2017). An overview of the supervised machine learning methods. *HORIZONS.B*, 4, 51–62. doi:10.20544/horizons.b.04.1.17.p05.

Netzel, P. & Stepinski, T. (2016). On using a clustering approach for global climate classification. *Journal of Climate*, 29(9), 3387–3401. doi:10.1175/JCLI-D-15-0640.1.

Orth, R., Staudinger, M., Seneviratne, S. I., Seibert, J., & Zappa, M. (2015). Does model performance improve with complexity? A case study with three hydrological models. *Journal of Hydrology*, 523, 147–159. doi:10.1016/j.jhydrol.2015.01.044.

Ossola, A., Hahs, A. K., & Livesley, S. J. (2015). Habitat complexity influences fine scale hydrological processes and the incidence of stormwater runoff in managed urban ecosystems. *Journal of Environmental Management*, 159, 1–10. doi:10.1016/j.jenvman.2015.05.002.

Pande, S. & Moayeri, M. (2018). Hydrological Interpretation of a Statistical Measure of Basin Complexity. *Water Resources Research*, 54(10), 7403–7416. doi:10.1029/2018WR022675.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., et al. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32* (pp. 8024–8035). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

Patterson, J. & Gibson, A. (2017). *Deep learning: A practitioner's approach*. O'Reilly Media, Inc.

Pearson, K. (1896). VII. Mathematical contributions to the theory of evolution.—III. Regression, heredity, and panmixia. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 187, 253–318. doi:10.1098/rsta.1896.0007.

Pechlivanidis, I. G., Jackson, B., Mcmillan, H., & Gupta, H. V. (2016). Robust informational entropy-based descriptors of flow in catchment hydrology. *Hydrological Sciences Journal*, 61(1), 1–18. doi:10.1080/02626667.2014.983516.

Perdigão, R., Pires, C., & Hall, J. (2019). Disentangling Nonlinear Spatiotemporal Controls on Precipitation: Dynamic Source Analysis and Predictability. doi:10.46337/mdsc.5273.

Perdigão, R. A. P. (2018). Synergistic Dynamic Theory of Complex Coevolutionary Systems. doi:10.46337/mdsc.5182.

Pfaffe, P. (2020). *Autotuning for Automatic Parallelization on Heterogeneous Systems*. PhD thesis, Karlsruher Institut für Technologie (KIT). doi:10.5445/IR/1000119646.

Prather, M. J. (2015). Photolysis rates in correlated overlapping cloud fields: Cloud-J 7.3c. *Geoscientific Model Development*, 8(8), 2587–2595. doi:10.5194/gmd-8-2587-2015.

Queipo, N. V., Haftka, R. T., Shyy, W., Goel, T., Vaidyanathan, R., & Tucker, P. K. (2005). Surrogate-based Analysis and Optimization. *Progress in Aerospace Sciences*, 41(1), 1–28. doi:10.1016/j.paerosci.2005.02.001.

Qureshi, M. S., Qureshi, M. B., Fayaz, M., Mashwani, W. K., Belhaouari, S. B., Hassan, S., & Shah, A. (2020). A comparative analysis of resource allocation schemes for real-time services in high-performance computing systems. *International Journal of Distributed Sensor Networks*, 16(8), 1550147720932750. doi:10.1177/1550147720932750.

Rasp, S. & Thuerey, N. (2021). Data-Driven Medium-Range Weather Prediction With a Resnet Pretrained on Climate Simulations: A New Model for WeatherBench. *Journal of Advances in Modeling Earth Systems*, 13(2). doi:10.1029/2020MS002405.

Reinert, D., Prill, F., Frank, H., Denhard, M., Baldauf, M., Schraff, C., Gebhardt, C., Marsigli, C., & Zängl, G. (2021). DWD Database Reference for the global and Regional ICON and ICON-EPS Forecasting System. *Technical report Version 2.1.7, Deutscher Wetterdienst*. doi:10.5676/DWD_pub/nwv/icon_2.1.7.

Reza, F. M. (1994 [1961]). *An Introduction to Information Theory*. Dover Publications, Inc., New York. ISBN: 0-486-68210-2.

Rieger, D., Bangert, M., Bischoff-Gauss, I., Förstner, J., Lundgren, K., Reinert, D., Schröter, J., Vogel, H., Zängl, G., Ruhnke, R., & Vogel, B. (2015). ICON–ART 1.0 – a new online-coupled model system from the global to regional scale. *Geoscientific Model Development*, 8(6), 1659–1676. doi:10.5194/gmd-8-1659-2015.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14(5), 465–471. doi:10.1016/0005-1098(78)90005-5.

Rissanen, J. et al. (2007). *Information and Complexity in Statistical Modeling*, volume 152. Springer New York, NY. doi:10.1007/978-0-387-68812-1.

Roper, D. C. (1979). The Method and Theory of Site Catchment Analysis: A Review. *Advances in Archaeological Method and Theory*, 2, 119–140. http://www.jstor.org/stable/20170144.

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20, 53–65.

Ruddell, B. L. & Kumar, P. (2009). Ecohydrologic process networks: 1. Identification. *Water Resources Research*, 45(3). doi:10.1029/2008WR007279.

Sander, R., Baumgaertner, A., Gromov, S., Harder, H., Jöckel, P., Kerkweg, A., et al. (2011). The atmospheric chemistry box model CAABA/MECCA-3.0. *Geoscientific Model Development*, 4(2), 373–380. doi:10.5194/gmd-4-373-2011.

Sang, Y.-F., Wang, D., Wu, J.-C., Zhu, Q.-P., & Wang, L. (2011). Wavelet-Based Analysis on the Complexity of Hydrologic Eeries Data under Multi-Temporal Scales. *Entropy*, 13(1), 195–210. doi:10.3390/e13010195.

Satopaa, V., Albrecht, J., Irwin, D., & Raghavan, B. (2011). Finding a "kneedle" in a haystack: Detecting knee points in system behavior. In *2011 31st international conference on distributed computing systems workshops* (pp. 166–171).: IEEE. doi:10.1109/ICDCSW.2011.20.

Saul, L. K. & Roweis, S. T. (2003). Think Globally, Fit Locally: Unsupervised Learning of Low Dimensional Manifolds. *Journal of Machine Learning Research*, 4(Jun), 119–155.

Sawicz, K., Kelleher, C., Wagener, T., Troch, P., Sivapalan, M., & Carrillo, G. (2014). Characterizing hydrologic change through catchment classification. *Hydrology and Earth System Sciences*, 18(1), 273–285. doi:10.5194/hess-18-273-2014.

Sawicz, K., Wagener, T., Sivapalan, M., Troch, P. A., & Carrillo, G. (2011). Catchment classification: empirical analysis of hydrologic similarity based on catchment function in the eastern USA. *Hydrology and Earth System Sciences*, 15(9), 2895–2911. doi:10.5194/hess-15-2895-2011.

Scher, S. (2018). Toward Data-Driven Weather and Climate Forecasting: Approximating a Simple General Circulation Model With Deep Learning. *Geophysical Research Letters*, 45(22), 12–616. doi:10.1029/2018GL080704.

Schilders, W. H., Van der Vorst, H. A., & Rommes, J. (2008). *Model Order Reduction: Theory, Research Aspects and Applications*, volume 13. Springer. doi:0.1007/978-3-540-78841-6.

Schöniger, A., Illman, W. A., Wöhling, T., & Nowak, W. (2015). Finding the right balance between groundwater model complexity and experimental effort via Bayesian model selection. *Journal of Hydrology*, 531, 96–110. doi:10.1016/j.jhydrol.2015.07.047.

Schoups, G., van de Giesen, N. C., & Savenije, H. H. G. (2008). Model complexity control for hydrologic prediction. *Water Resources Research*, 44(12). doi:10.1029/2008WR006836.

Schröter, J., Rieger, D., Stassen, C., Vogel, H., Weimer, M., Werchner, S., et al. (2018). ICON-ART 2.1: a flexible tracer framework and its application for composition studies in numerical weather forecasting and climate simulations. *Geoscientific Model Development*, 11(10), 4043–4068. doi:10.5194/gmd-11-4043-2018.

Schultz, M. G., Betancourt, C., Gong, B., Kleinert, F., Langguth, M., Leufen, L. H., et al. (2021). Can deep learning beat numerical weather prediction? *Philosophical Transactions of the Royal Society*, 379(2194). doi:10.1098/rsta.2020.0097.

Schulz, K., Seppelt, R., Zehe, E., Vogel, H.-J., & Attinger, S. (2006). Importance of spatial structures in advancing hydrological sciences. *Water Resources Research*, 42(3). doi:10.1029/2005WR004301.

Schwarz, G. (1978). Estimating the Dimension of a Model. *The Annals of Statistics*, 6(2), 461–464. doi:10.1214/aos/1176344136.

Sedgewick, R. & Flajolet, P. (2013). *An Introduction to the Analysis of Algorithms*. Pearson Education.

Shannon, C. E. (1948). A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27(3), 379–423. doi:10.1002/j.1538-7305.1948.tb01338.x.

Shobha, N. & Asha, T. (2017). Monitoring weather based meteorological data: Clustering approach for analysis. In *Innovative Mechanisms for Industry Applications (ICIMIA), 2017 International Conference on* (pp. 75–81).: IEEE.

Silvestrini, S. & Lavagna, M. (2022). Deep Learning and Artificial Neural Networks for Spacecraft Dynamics, Navigation and Control. *Drones*, 6(10), 270. doi:10.3390/drones6100270.

Sivakumar, B., Jayawardena, A., & Li, W. K. (2007). Hydrologic complexity and classification: a simple data reconstruction approach. *Hydrological Processes: An International Journal*, 21(20), 2713–2728. doi:10.1002/hyp.6362.

Sivakumar, B. & Singh, V. (2012). Hydrologic system complexity and nonlinear dynamic concepts for a catchment classification framework. *Hydrology and Earth System Sciences*, 16(11), 4119–4131. doi:10.5194/hess-16-4119-2012,2012.

Sivapalan, M. (2003). Process complexity at hillslope scale, process simplicity at watershed scale: Is there a connection? In *EGS-AGU-EUG Joint Assembly* (pp. 7973). doi:10.1002/hyp.5109.

Solomonoff, R. (1978). Complexity-based induction systems: Comparisons and convergence theorems. *IEEE transactions on Information Theory*, 24(4), 422–432. doi:10.1109/TIT.1978.1055913.

Solomonoff, R. J. (1964). A formal theory of inductive inference. Part I. *Information and control*, 7(1), 1–22. doi:10.1016/S0019-9958(64)90223-2.

Sønderby, C. K., Espeholt, L., Heek, J., Dehghani, M., Oliver, A., Salimans, T., Agrawal, S., Hickey, J., & Kalchbrenner, N. (2020). Metnet: A neural weather model for precipitation forecasting. *Computing Research Repository (CoRR)*, abs/2003.12140. doi:10.48550/arXiv.2003.12140.

Stigler, S. M. (1974). Gergonne's 1815 paper on the design and analysis of polynomial regression experiments. *Historia Mathematica*, 1(4), 431–439. doi:10.1016/0315-0860(74)90033-0.

Strobl, M., Azmi, E., Bischof, B., Dolich, A., Hassler, S. K., Mälicke, M., Manoj J, A., Meyer, J., Streit, A., & Zehe, E. (2023). Implementation of an InfraStructure for dAta-BasEd Learning in environmental sciences (ISABEL). In *E-Science-Tage 2023: Empower Your Research - Preserve Your Data* (pp. 295–300). doi:10.11588/heibooks.1288.c18091.

Strobl, M., Azmi, E., Hassler, S. K., Mälicke, M., Meyer, J., Streit, A., & Zehe, E. (2021). V-FOR-WaTer – a virtual research environment for environmental research. In *E-Science-Tage 2021: Share Your Research Data* (pp. 394–398). doi:10.11588/heibooks.979.c13755.

Sun, X., Li, B., Ma, X., Pan, Y., Yang, S., & Huang, W. (2020). Proper Orthogonal Decomposition-Based Method for Predicting Flow and Heat Transfer of Oil and Water in Reservoir. *Journal of Energy Resources Technology*, 142(1), 0124011–1240110. doi:10.1115/1.4044192.

Suslov, S., Schiek, M., Robens, M., Grewing, C., & van Waasen, S. (2020). Simulating Heterogeneous Models on Multi-Core Platforms using Julia's Computing Language Parallel Potential. In *2020 IEEE/ACM 24th International Symposium on Distributed*

Simulation and Real Time Applications (DS-RT) (pp. 1–4).: IEEE. doi:10.1109/DS-RT50469.2020.9213527.

Tibshirani, R., Walther, G., & Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2), 411–423.

Trenholme, R. (1994). Analog Simulation. *Philosophy of Science*, 61(1), 115–131. doi:10.1086/289783.

Türkeş, M. & Tatlı, H. (2011). Use of the spectral clustering to determine coherent precipitation regions in Turkey for the period 1929–2007. *International Journal of Climatology*, 31(14), 2055–2067.

Van der Maaten, L. & Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(11), 2579–2605.

Van Houdt, G., Mosquera, C., & Napoles, G. (2020). A review on the long short-term memory model. *Artificial Intelligence Review*, 53(8), 5929–5955. doi:10.1007/s10462-020-09838-1.

Vapnik, V. & Chervonenkis, A. (1979). Theory of Pattern Recognition (in Russian) Nauka, Moscow, 1974. *German translation: W.N. Wapnik, A.J. Tscherwonenkis Theorie der Zeichenerkennung, Akademie Verlag, Berlin*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., & Polosukhin, I. (2017). Attention Is All You Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems*, volume 30: Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Vidal, R., Ma, Y., & Sastry, S. S. (2016). Principal Component Analysis. *Generalized principal component analysis*, (pp. 25–62). doi:10.1007/978-0-387-87811-9_2.

Wagener, T., Sivapalan, M., Troch, P., & Woods, R. (2007). Catchment classification and hydrologic similarity. *Geography compass*, 1(4), 901–931. doi:10.1111/j.1749-8198.2007.00039.x.

Wallace, C. S. & Boulton, D. M. (1968). An Information Measure for Classification. *The Computer Journal*, 11(2), 185–194. doi:10.1093/comjnl/11.2.185.

Weijs, S. V. & Ruddell, B. L. (2020). Debates: Does Information Theory Provide a New Paradigm for Earth Science? Sharper Predictions Using Occam's Digital Razor. *Water resources research*, 56(2), e2019WR026471. doi:10.1029/2019WR026471.

Weijs, S. V., Schoups, G., & Van de Giesen, N. (2010). Why hydrological predictions should be evaluated using information theory. *Hydrology and Earth System Sciences*, 14, 2545–2558. doi:10.5194/hess-14-2545-2010.

Weimer, M., Buchmüller, J., Hoffmann, L., Kirner, O., Luo, B., Ruhnke, R., et al. (2021). Mountain-wave-induced polar stratospheric clouds and their representation in the global chemistry model ICON-ART. *Atmospheric Chemistry and Physics*, 21(12), 9515–9543. doi:10.5194/acp-21-9515-2021.

Weimer, M., Schröter, J., Eckstein, J., Deetz, K., Neumaier, M., Fischbeck, G., Hu, L., Millet, D. B., Rieger, D., Vogel, H., Vogel, B., Reddmann, T., Kirner, O., Ruhnke, R., & Braesicke, P. (2017). An emission module for ICON-ART 2.0: implementation and simulations of acetone. *Geoscientific Model Development*, 10(6), 2471–2494. doi:10.5194/gmd-10-2471-2017.

Winsberg, E. (2022). Computer Simulations in Science. In E. N. Zalta & U. Nodelman (Eds.), *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Winter 2022 edition.

Yan, J., Zhang, B., Liu, N., Yan, S., Cheng, Q., Fan, W., Yang, Q., Xi, W., & Chen, Z. (2006). Effective and efficient dimensionality reduction for large-scale and streaming data preprocessing. *IEEE transactions on Knowledge and Data Engineering*, 18(3), 320–333.

Yin, C. & McKay, A. (2018). Introduction to Modeling and Simulation Techniques. In *Proceedings of ISCIIA and ITCA*: University of Leeds White Rose Research Online. https://eprints.whiterose.ac.uk/135646.

Zaki, M. J., Meira Jr, W., & Meira, W. (2014). *Data mining and analysis: fundamental concepts and algorithms*. Cambridge University Press. doi:10.1017/CBO9780511810114.

Zängl, G., Reinert, D., & Prill, F. (2022). Grid Refinement in ICON v2.6.4. *Geoscientific Model Development*, 15, 7153–7176. doi:10.5194/gmd-15-7153-2022.

Zängl, G., Reinert, D., Rípodas, P., & Baldauf, M. (2015). The ICON (ICOsahedral Non-hydrostatic) modelling framework of DWD and MPI-M: Description of the non-hydrostatic dynamical core. *Quarterly Journal of the Royal Meteorological Society*, 141(687), 563–579. doi:10.1002/qj.2378.

Zarnani, A., Musilek, P., & Heckenbergerova, J. (2014). Clustering numerical weather forecasts to obtain statistical prediction intervals. *Meteorological Applications*, 21(3), 605–618.

Zehe, E., Ehret, U., Pfister, L., Blume, T., Schröder, B., Westhoff, M., Jackisch, C., Schymanski, S. J., Weiler, M., Schulz, K., Allroggen, N., Tronicke, J., van Schaik, L., Dietrich, P., Scherer, U., Eccard, J. A., Wulfmeyer, V., & Kleidon, A. (2014). HESS Opinions: From response units to functional units: a thermodynamic reinterpretation of the HRU concept to link spatial organization and functioning of intermediate scale catchments. *Hydrology and Earth System Sciences*, 18(11), 4635–4655. doi:10.5194/hess-18-4635-2014.

Zeng, A., Chen, M., Zhang, L., & Xu, Q. (2023). Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37 (pp. 11121–11128). https://ojs.aaai.org/index.php/AAAI/article/view/26317/26089.

Zhang, H., Ho, T. B., Zhang, Y., & Lin, M.-S. (2006). Unsupervised feature extraction for time series clustering using orthogonal wavelet transform. *Informatica*, 30(3).

Zhang, Y., Moges, S., & Block, P. (2016). Optimal cluster analysis for objective regionalization of seasonal precipitation in regions of high spatial-temporal variability: application to Western Ethiopia. *Journal of Climate*, 29(10), 3697–3717.

Zhou, Y., Zhang, Q., Li, K., & Chen, X. (2012). Hydrological effects of water reservoirs on hydrological processes in the East River (China) basin: complexity evaluations based on the multi-scale entropy analysis. *Hydrological Processes*, 26(21), 3253–3262. doi:10.1002/hyp.8406.