ADDIS ABABA UNIVERSITY

ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF CIVIL AND ENVIRONMENTAL ENGINEERING

Deep Learning Approach to Estimate Streamflow from Remote Sensing Data

Dissertation for the Degree of Doctor of Philosophy

By Eyob Betru Wegayehu

February 2024

Addis Ababa

Deep Learning Approach to Estimate Streamflow from Remote Sensing Data

Ph.D. Dissertation

by

Eyob Betru Wegayehu

Under the supervision of

Dr Fiseha Behulu

Assistance Professor, School of Civil and Environmental Engineering, Addis Ababa Institute of Technology, Addis Ababa University, Ethiopia

A Ph.D. Dissertation

Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy (Ph.D.) in Hydraulic Engineering.

Addis Ababa University (AAU)

Addis Ababa Institute of Technology (AAIT)

Doctoral Dissertation Approval Sheet

The undersigned have examined the dissertation entitled "**Deep Learning Approach to Estimate Streamflow from Remote Sensing Data**' presented by **Eyob Betru Wegayehu**, a candidate for the degree of **Doctor of Philosophy,** and hereby certify that it is worthy of acceptance.

Submitted by:

Eyob Betru Wegayehu		21/02/2024
Candidate (Ph.D.)	Signature	Date

Approved by:

Fiseha Behulu		21/02/2024
Advisor	Signature	Date
Dereje Hailu		21/02/2024
Internal Examiner	Signature	Date
Fasikaw A. Zimale		21/02/2024
External Examiner 1	Signature	Date
Assefa Melesse		21/02/2024
External Examiner 2	Signature	Date
Fiseha Behulu		21/02/2024
Chairperson	Signature	Date

ABSTRACT

Streamflow measurements that are reliable, precise, and continuous are essential for water resource analysis, planning, and management. However, there are relatively limited available hydrological data globally. The limited number of gauging stations worldwide is a common challenge, even worse in developing countries such as Ethiopia. As a result, novel research methodologies that can simulate river flow using multiple data sources are crucial. Remote sensing is one of the data sources that is considered a promising alternative. It has shown remarkable progress to date and in the future, such as the Surface Water and Ocean Topography (SWOT) mission, a satellite built to precisely observe nearly all the water on our planet's surface launched on December 16, 2022.

Artificial intelligence, analogically "the new electricity" as a tool, has enormous potential to support the big challenge we face in monitoring our water resources. Hence, this study aims to integrate remote sensing and deep learning approaches for continuous streamflow time series generation. This will be achieved by collecting remote sensing-based precipitation products, vegetation indices and ground-based hydrometeorological data. Various types of single, hybrid, and ensemble deep learning models with the conceptual hydrological model are applied as a tool to model these univariate and multivariate types of data in different agroclimatic conditions, including the upper Tiber River basin (Italy), Abay River basin (Gummera subcatchment, Ethiopia), Awash River basin (Borkena subcatchment, Ethiopia), and Baro-Akobo River basin (Sore and Masha subcatchment, Ethiopia).

Finally, by simulating streamflow with consistent top performance in all case study catchments, the results demonstrate the tremendous potential of the ensemble deep learning model. Although vegetation indices also showed much potential as an input for machine learning models, more study is needed to optimize performance with various data assimilation techniques. The findings of this research also provide further evidence for the significance of input data preprocessing, selection, length, and variability in conjunction with the kind of machine learning model we use to simulate streamflow accurately. The study also provides recommendations for further study and development in data-driven rainfall-runoff modelling.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

DECLARATION OF AUTHORSHIP

I, Eyob Betru Wegayehu, thus declare that the dissertation entitled " Advancing Streamflow Estimation: A Deep Learning Framework with Remote Sensing in Varied Agro-climatic Conditions " and the materials provided in this document are my original work. This study was completed entirely while pursuing a Ph.D. at Addis Ababa Institute of Technology (AAIT). The sources are appropriately cited when I have quoted from the work of others. Except for such references, this dissertation is my work, and I have recognized all sources of guidance, especially from my advisor Dr. Fiseha Behulu.

I. Wegayehu, E. B., & Muluneh, F. B. (2022). Short-Term Daily Univariate Streamflow Forecasting Using Deep Learning Models. Advances in Meteorology, 2022. https://doi.org/10.1155/2022/1860460

II. Wegayehu, E. B., & Muluneh, F. B. (2021). Multivariate streamflow simulation using hybrid deep learning models. Computational Intelligence and Neuroscience, 2021. https://doi.org/10.1155/2021/5172658

III. Wegayehu, E. B., & Muluneh, F. B. (2023). Superensemble based streamflow simulation using multi-source remote sensing and ground gauged rainfall data fusion. Heliyon, 9(7). https://doi.org/10.1016/j.heliyon.2023.e17982

IV. Wegayehu, E. B., & Muluneh, F. B. (2024). Comparing conceptual and super ensemble deep learning models for streamflow simulation in data-scarce catchments. Journal of Hydrology: Regional Studies, 52, 101694. https://doi.org/10.1016/j.ejrh.2024.101694.

Signed _____

Date February 27, 2024

ACKNOWLEDGEMENTS

God takes every single blessing in my life. You have given me more than I could have possibly dreamed. You have placed diligent individuals all around me. You have blessed me with family and friends who think and behave kindly towards me every day.

During my Ph.D. Journey, words cannot express how grateful I am to Dr. Admasu Gebeyehu for his compassion and support at the start of this journey. Our first-year coursework would be impossible without your significant contributions, which enabled my colleagues and me to focus on our research work in subsequent years. Dr. Admasu has given me not only scientific knowledge but also the entire image of heartfelt kindness and the joy of helping others, and I believe this is the human character knowledge pursued. Your seed in my heart will bear fruit for my future students. I also want to convey my heartfelt gratitude to my supervisor, Dr Fiseha Behulu. I was lucky to join his Ph.D. research team. Dr Fiseha is a young, dynamic-minded scholar I met in the school with outstanding research passion and commitment, which inspired me to unlock my full potential. His persistent guidance, encouragement, and patience assisted me in carrying out this research. Dr Fiseha, I am grateful for your efforts in stimulating and formulating this research. Your creative and energetic inspiration motivated me throughout this research period. You provided me with scientific support and social and personal directions, which enabled me to enter the industry in my field of expertise.

Debre Berhan University (DBU) and the Ethiopian Ministry of Education (MOE) have my sincere gratitude for awarding me this scholarship. The raw hydrological and meteorological data sets used for this study were collected from Ethiopia's Ministry of Water and Energy (MoWE) and the National Meteorological Agency of Ethiopia (NMA), respectively. The assigned personnel in each government office support are gratefully acknowledged. My sincere gratitude also goes to my Ph.D. colleagues at AAiT, with whom I shared my experiences, challenges, and successes. I will never forget sharing the office with you and discussing research ideas and the economy. There are many more to name and thank, but it may be preferable not to forget them. Finally, I am grateful to my family members and relatives, especially my wife, Dr. Kidist, for your unwavering love and patience throughout this journey. Honey, you have given me your constant support to complete this research successfully. You suffered more than I did. Without your support, encouragement, and scarification, this endeavor would not be possible.

ABBREVIATIONS

ARE	Absolute Relative Error
ANFIS	Adaptive Neuro-Fuzzy Inference System
ASTER	Advanced Spaceborne Thermal Emission and Reflection Radiometer
AVHRR	Advanced Very High Resolution Radiometer
AIC	Akaike Information Criterion
AI	Artificial Intelligence
ANN	Artificial Neural Networks
ARMAX	Auto Regressive Moving Average with eXogenous term
ARIMA	Auto-Regressive Integrated Moving Average
AARE	Average Absolute Relative Error
BP	Back Propagation
BIC	Bayesian Information Criterion
BMA	Bayesian Model Averaging
Bi-LSTM	Bidirectional Long Short Term Memory
CHIRPS	Climate Hazards group InfraRed Precipitation with Station data
CMORPH	Climate prediction center MORPHing
CERES	Clouds and Earth's Radiant Energy System
CNN	Convolutional Neural Network
CREST	Coupled Routing and Excess STorage
CN	Curve Number
DTR	Decision Tree Regressor
DL	Deep Learning
DPR	Dual-frequency Precipitation Radar
EDSC	Earth Data Search Capability
EO	Earth observation
ESDS	Earth Science Data Systems
ESN	Echo State Network
ERNN	Elman Recurrent Neural Network
ENN	Emotional Neural Network
EVI	Enhanced Vegetation Index

MOWE	Ethiopian Ministry of Water and Energy
EC	Evolutionary Computation
ETR	Extra Tree Regression
XGB	eXtreme Gradient Boosting
ELM	Extreme Learning Machine
FFNN	Feed Forward Neural Network
SPOT	French: Satellite Pour l'Observation de la Terre
FL	Fuzzy Logic
GRU	Gated Recurrent Units
GRNNs	Generalized Regression Neural Networks
GA	Genetic Algorithm
GP	Genetic Programming
GIS	Geographic Information System
GCOS	Global Climate Observing System's
GPM	Global Precipitation Measurement
GRDC	Global Runoff Data Centre's
GEE	Google Earth Engine
GMI	GPM Microwave Imager
GRACE	Gravity Recovery and Climate Experiment
GDP	Gross Domestic Product
HBV	Hydrologiska Byråns Vattenbalansavdelning
ICESat	Ice, Cloud and land Elevation Satellite
IR	InfraRed
IRS	Indian Remote Sensing
IHDM	Institute of Hydrology Distributed Model
IMERG	Integrated MultisatellitE Retrievals for GPM
IGARSS	International Geoscience and Remote Sensing Symposium
JAXA	Japan Aerospace and eXploration Agency
LASSO	Least Absolute Shrinkage and Selection Operator
LSSVM	Least-Squares Support Vector Machine
LIS	Lightning Imaging Sensor
LR	Linear Regression

LSTM	Long Short-Term Memory
LEO	Low Earth Orbit
ML	Machine Learning
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MSE	Mean Squared Error
MSAD	Mean Sum of Absolute Deviations
MEDAE	Median Absolute Error
MERIS	MEdium Resolution Imaging Spectrometer
MW	Micro Wave
MPMR	Minimax Probability Machine Regression
MOPEX	MOdel Parameter Estimation eXperiment
MODIS	MODerate resolution Imaging Spectroradiometer
MNDVI	Modified Normalized Difference Vegetation Index
MSI	Multi Spectral Instrument
MLP	Multi-Layer Perceptron
MLR	Multiple Linear Regression
MSWEP	Multi-Source Weighted-Ensemble Precipitation
MARS	Multivariate Adaptive Regression Splines
NSE	Nash Sutcliffe Efficiency
NASA	National Aeronautics and Space Administration
NMA	National Meteorological Agency of Ethiopia
NOAA	National Oceanic and Atmospheric Administration
NFS	Neuro-Fuzzy Systems
NDVI	Normalized Difference Vegetation Index
NDWI	Normalized Difference Water Index
NMBE	Normalized Mean Bias Error
NRMSE	Normalized Root Mean Square Error
OLCI	Ocean and Land Colour Instrument
PACF	Partial Auto Correlation Function
PM	Particular Matter
PMW	Passive Microwave

PCC	Pearson Correlation Coefficient
PET	Potential Evapotranspiration Estimation
PERSIANN	Precipitation Estimation from Remotely Sensed Information using Artificial Neural Networks
PE	Prediction Errors
PUB	Predictions in Ungauged Basins
PCA	Principal Component Analysis
R^2	Coefficient of determination
R_{eff}	Coefficient of efficiency
RBF	Radial Basis Function
RF	Random Forest
ReLU	Rectified Linear Units
RNN	Recurrent Neural Networks
RFE	Recursive Feature Elimination
RVM	Relevance Vector Machine
RS	Remote Sensing
RMSE	Root Mean Square Error
SAC-SMA	SACramento Soil Moisture Accounting
SARIMA	Seasonal Auto-Regressive Integrated Moving Average
SRTM	Shuttle Radar Topography Mission
SNR	Signal-to-Noise Ratio
SWAT	Soil & Water Assessment Tool
SCS	Soil Conservation Service
SMAP	Soil Moisture Active and Passive
SMOS	Soil Moisture and Ocean Salinity
S-LSTM	Stacked Long Short Term Memory
SWM	Stanford Watershed Model
SWMM	Storm Water Management Model
SSE	Sum of Squared Errors
NPP	National Polar-orbiting Partnership
SE	Super Ensemble
SVM	Support Vector Machine

SVR	Support Vector Regression
SWOT	Surface Water and Ocean Topography
SMHI	Swedish Meteorological and Hydrological Institute
SAR	Synthetic Aperture Radar
TIROS	Television InfraRed Observation Satellite
TCN	Temporal Convolutional Network
TS	Threshold Statistics
TRB	Tiber River Basin
TSAD	Total Sum of Absolute Deviations
TTPE	Training Time Per Epoches
TMI	TRMM Microwave Imager
TRMM	Tropical Rainfall Measuring Mission
USDA	United States Department of Agriculture
UTRB	Upper Tiber River Basin
VI	Vegetation Indices
VIRS	Visible and InfraRed Scanner
VIIRS	Visible Infrared Imaging Radiometer Suite
WCOM	Water Cycle Observation Mission
WRME	Water Resource Management and Evaluation
W-AI	Wavelet-Artificial Intelligence
WA	Weighted Average
XAJ	Xin'AnJiang

CHAPTER 1:

INTRODUCTION

1.1. Background

Streamflow time series modelling is crucial in the design and operation of hydraulic structures, real-time flood forecasting, estimation of flows in ungauged catchments, and prediction of the impact of land use and climate change on water resources. These models try to replicate the intricate hydrological processes that turn rainfall into runoff and are also crucial for river basin management and the planning of water resources (M L Tan, 2014).

The major concerns in implementing different models involve input parameters and model performance. Each model, with distinct characteristics, has consequently been implemented with varying degrees of success, from tiny basins to enormous catchments, encompassing both ungauged and gauged catchments (Devia et al., 2015). In addition, each model is connected with several downsides, including enormous data needs and a lack of user friendliness. With the growth of technology, researchers have highlighted the use of remote sensing, which can overcome the constraints related to the available data for model simulation, specifically concerning modelling in data-scarce catchments (Mushore et al., 2019).

In the age of artificial intelligence (AI), which has been contextualized as the new electricity, research areas in simulating streamflow with a novel approach integrating remote sensing and machine learning fields will contribute a step for the ongoing predictions in ungauged basins (PUB) research. This work provides futuristic insight to solve the complex problem in Ethiopia and around the world.

Most hydrological processes are often very nonlinear and may not be represented mathematically in simple or complicated forms. On the other hand, data-driven or AI-based modelling will have greater potential than process or physical-based models, owing to unknown parameters involved in the processes and spatiotemporal fluctuation of the processes and their driving factors. Artificial neural networks (ANNs), fuzzy logic (FL), recurrent neural networks (RNNs), support vector machines (SVMs), gated recurrent units (GRUs), and genetic programming (GP) are some of the AI algorithms used in data-driven hydrological modelling (Nourani, Molajou, Najafi, et al., 2019).

Over the last two decades, ANNs have become a popular data-driven modelling tool in river flow forecasting. This is due to its capacity to include complicated and nonlinear input-output linkages seen in hydrological processes within a river basin. Despite their popularity, ANNs are still susceptible to flaws such as overfitting and overparameterization, primarily when utilized with restricted data set availability. These issues frequently impact the predictive power of ANN-based models, resulting in erroneous and unreliable simulations (Adeyemo et al., 2018).

Furthermore, ANNs are accessible in various functionality and architectural shapes, from basic to complex. One of the most sophisticated ANN designs is the recurrent neural network (RNN). It is one of the particularly constructed deep learning networks for time series analysis that can readily adjust to temporal dynamics by leveraging prior time step information (Ni et al., 2020). Nevertheless, RNNs cannot capture long-term relationships and are prone to vanishing and exploding gradients. Couta et al. (2019) proposed improved RNN or long short-term memory (LSTM) as one of the most successful techniques to address these drawbacks.

An input gate, an output gate, and a forget gate make up the cell of the LSTM unit (Y. Bai et al., 2019). These gates allow the LSTM model to perform well in various applications, such as voice recognition, traffic flow forecasting, natural language processing, time series modelling, and handwriting recognition (Campos et al., 2019; Yuan et al., 2018). Previous studies by several researchers using various effective multilayered (ML) methods to study LSTM competency for streamflow prediction have shown that LSTM performed well (Sahoo et al., 2019; L.Yan et al., 2019; Yuan et al., 2018). The GRU capability for speech signal modelling and natural language processing was comparable to that of LSTM. However, there are arguments over the relative performance of these two architectures for streamflow prediction, which is not fully explored with diverse timescales and environments.

Different hybrid deep-learning models have recently received much interest from academia. C. Chen et al. (2020) predicted nitrogen oxide emissions using a convolutional neural network (CNN), LSTM, and hybrid CNN-LSTM models. They determined that CNN-LSTM forecasts periodic nitrogen oxide emissions from the refining sector accurately and consistently. Furthermore, Li et al. (2020) fed LSTM and CNN-LSTM models with univariate and multivariate time series data. Consequently, due to its low error and quick training time, the proposed multivariate CNN-LSTM model outperforms the competition for air quality analysis utilizing particulate matter (PM2.5) concentration prediction.

The combination of CNN and LSTM models improves time-series prediction models by allowing LSTM models to collect extended time sequences of pattern information more effectively. On the other hand, CNN models may filter out the noise in the input data and extract more relevant features, potentially increasing the prediction model's accuracy (Livieris et al., 2020). Furthermore, combining CNNs with GRUs may result in robust data preprocessing, giving a realistic approach to improving the model's accuracy (Ahmed et al., 2021).

Implementations of ensemble machine learning models in hydrology have significantly risen in recent years and are drawing greater attention, along with hybrid models (Zounemat-Kermani, Mahdavi-Meymand, et al., 2021). One machine learning (ML) model may perform better than others for comparable data sets, but the results will often vary for various data sets. The ensemble approach, which uses each model's (base learner) output as input and a significance level determined by an arbitrator, was developed to use each model's advantages without compromising the generic nature of the data (Nourani et al., 2021). Choosing the right time-series models from the various known deep-learning network architectures is difficult. The performance of single ML models, hybrid ML models, or ensemble ML models may not always be the decisive factors for time series estimation due to other parameters, such as time-series characteristics, temporal scale, and agro-climatic variation. As a result, choosing the best option for time-series analysis is greatly aided by this type of comparative research on network architectures.

The objective of the current study is to compare and evaluate the potential of various single, hybrid, and ensemble ML models with the conceptual hydrological model for streamflow simulation in the upper Tiber River basin (Italy), Abay River basin (Gummera subcatchment, Ethiopia), Awash River basin (Borkena subcatchment, Ethiopia), and Baro Akobo River basin (Sore and Mashaa subcatchment, Ethiopia). As a result, the performance of various deep learning architectures on various agro-climatic zones and the applicability of remote sensing (RS) indices and precipitation products for streamflow prediction are comprehensively examined throughout this research. The objectives will be achieved by collecting and estimating rainfall, potential evapotranspiration, land use/land cover, and RS indices using RS and ground-based data sources. These data will be used as input to deep learning and conceptual hydrological models for training, validation, testing, and generation of streamflow time-series data.

1.2. Statement of the Problem

River discharge has been classified as a fundamental physical variable and is included in the Global Climate Observing System's (GCOS) list of key climate variables. Even though river flow is one of the most monitored components of the hydrological cycle, its monitoring is still a work in progress. River discharge data collection, archiving, and sharing are globally limited (Tarpanelli et al., 2019).

Currently, streamflows are measured at river gauging stations. Various studies, however, have revealed that the availability of gauging station data is declining in most regions (A. Sichangi et al., 2018). Tourian et al. (2013) compiled a time series visualization of the number of stations having accessible discharge data from the Global Runoff Data Centre's (GRDC) publicly available data. Between 1970 and 2010, this data series shows a decrease in total measured yearly streamflows. Furthermore, with the bulk of stations no longer in service, poor discharge monitoring has become a serious issue in developing nations (A. W. Sichangi et al., 2016). As a result, a study into the robustness of discharge data estimates is an indisputably vital and futuristic topic, particularly for a nation such as Ethiopia.

The difficulties mentioned above restrict the projects that need river discharge data. In contrast to terrestrial observations, remote sensing allows continuous coverage of rivers and other bodies of water. Recent studies have shown that remote sensing offers strong spatial coverage and a lengthy monitoring period, resulting in rising interest in estimating discharge using remote sensing (Mushore et al., 2019).

Physical models have a long history of being used in the simulation, interpretation, and prediction of hydrological processes. Statistical models are another prominent set of modelling tools, although they have some limitations. The weakness in uncertainty analysis, accuracy, high processing cost, and the necessity for a large quantity of data have all been emphasized in the literature (Devia et al., 2015; Mushore et al., 2019).

The effectiveness of machine learning and deep learning techniques has been remarkable due to their effective intelligence and computation. These techniques have gained enormous popularity within the research community over the past few years (Faizollahzadeh et al., 2019). Without any previous understanding of the nature of the process, ANN is one of the most well-known and classic frameworks for connecting input and output variables in complicated hydrological systems.

ML modelling is a popular and effective approach in which the adoption of ANN models in hydrological simulations saw a significant increase. However, there are various drawbacks related to insufficient observed samples or when the time series has a high rate of nonstationary and seasonal fluctuation (Nourani, Molajou, Najafi, et al., 2019).

One of the most powerful ANN designs for time series analysis is the RNN, which can quickly adapt temporal dynamics using prior time step data (Ni et al., 2020). However, RNNs are prone to vanishing and exploding gradients and cannot capture long-term relationships. To fix these drawbacks, Couta et al. (2019) recommended the improved RNN or long short-term memory (LSTM) as one of the most practical solutions. GRU is the second architectural type that performs well in speech signal modelling and natural language processing. However, there are ongoing discussions over how well these two designs function for streamflow simulation in various environments and at various timescales.

Different ML models often perform differently on various types of data. The hybrid or ensemble approach, which uses each model's (base learner) output as input and a significance level determined by an arbitrator, was recently developed to make use of each model's advantages without losing the general nature of the data (Nourani et al., 2021).

The few available deep learning studies, both locally and internationally, attempt to model runoff from a single ground-based hydrometeorological data source. As a result, evaluating these architectures using multiple data sources, such as remote sensing-based meteorological data and alternative remote sensing-based indices, will provide a better solution to the challenge we face in data-scarce catchments.

1.3. Research Objectives

1.3.1. Main Objective

The study's main objective is to build a continuous streamflow time series utilizing ground and remote sensing data by applying several deep learning algorithms.

1.3.2. Specific Objective

The following particular objectives are planned to be achieved in this study under the umbrella of the main objective:

1. To assess the best single machine learning algorithms for univariate streamflow forecasting using ground-gauged time series.
2. To compare and evaluate several hybrid deep learning models in the simulation of streamflow time series employing ground multivariate hydrometeorological variables.
3. To investigate the efficiency of ensemble learning for continuous streamflow time series generation utilizing ground and remote sensing data fusion.
4. To compare the performance of a conceptual hydrological model for simulation of streamflow time series with single, hybrid, and ensemble deep learning models.

1.4. Research Questions

1. Which single machine learning algorithm is the most efficient for predicting univariate streamflow?
2. Do hybrid deep learning architectures outperform single model algorithms using ground-gauged data sets for estimating streamflow time series?
3. Which varieties of ensemble deep learning models are efficient in the literature, and how much do they influence streamflow generation performance more than single and hybrid models?
4. How much do data scarcity and agro-climatic variation impact the performance of deep learning architectures for estimating streamflow?
5. What kind of remote sensing-based precipitation products and vegetation indices are most suited for deep learning-based streamflow estimation?

1.5. Research Significance

Hydrologists rely on long-term streamflow data series for design, flood prediction, and water resource management. While the traditional practice of collecting in situ stream gauge data has been longstanding, it is important to note that such gauging has become increasingly scarce and is on the decline worldwide, especially in developing countries. Effective maintenance of stream gauge networks is not only time-consuming and expensive; institutional capacity and political constraints in certain countries also impede data access to users.

These facts inspire the research community to develop novel approaches that advance continuous streamflow data simulation with as little as time and money. As a result, the proposed research will contribute to this aim by predicting streamflow data utilizing remote sensing data and a deep learning approach with minimal ground data over a reasonably long time.

1.6. Scope of the Study

Four river basins are the focus of the study: Upper Tiber River Basin (Italy), Awash River Basin (Borkena subcatchment, Ethiopia), Abay River Basin (Gummera subcatchment, Ethiopia), and Baro Akobo River Basin (Sore and Mashaa subcatchment, Ethiopia). Several single, hybrid, and ensemble deep learning architectures are investigated using ground and RS-generated data products to estimate streamflow time series.

1.7. Outline of the Dissertation

This dissertation has eight chapters. A summary of the structure is briefly discussed in the following paragraphs.

Chapter 1 provides a general background for the dissertation, followed by a statement of the problem, general and specific objectives, research questions, research significance, and finally, the scope of the study.

Chapter 2 presents the theoretical basis for hydrological models, machine learning, and the use of remote sensing in hydrology. It begins with a brief explanation of the state-of-the-art rainfall-runoff models and their classification, followed by a discussion of the significance of remote sensing in hydrology. It also goes through the history of remote sensing, the many sensors aboard, and the techniques for estimating rainfall and streamflow using remote sensing data. Finally, the chapter discusses machine learning theories and several prominent deep learning architectures.

Chapter 3 describes the four study areas or river basins, including their topography and hydrometeorological characteristics, as well as the five subcatchments explored in this research.

In **Chapter 4**, we presented a study entitled "short-term daily univariate streamflow forecasting using deep learning models." This study compared stacked long short-term memory (S-LSTM), bidirectional long short-term memory (Bi-LSTM), GRU, and LSTM architectures with the traditional multilayer perceptron (MLP) model to forecast a single-step streamflow amount using daily data records.

Chapter 5 also discusses and compares numerous hybrid CNN-LSTM and CNN-GRU architectures to classic MLP, GRU, and LSTM networks to model single-step streamflow using two climatic zones, available precipitation, and minimum and maximum temperature data. The chapter has briefly addressed the performance variation of the given models when subjected to severe input variability, including climatic, input combination, input time window, and average rolling time window.

Chapter 6 describes a novel model averaging method for streamflow modelling that uses superensemble approaches to combine multisource remote sensing and ground-gauged rainfall data. This chapter also presented a novel technique that uses vegetation indices as input variables with ground gauged and satellite precipitation data and multiple bases and meta-learners. Furthermore, this chapter presents the results with five input fusion strategies.

Chapter 7 expands on the findings of the previous chapter by comparing superensemble deep learning models with a semidistributed conceptual hydrological model in data-scarce catchments. As a result, this chapter focuses solely on the Hydrologiska Byråns Vattenbalansavdelning (HBV) model data analysis, integrating and comparing the eight base and three super ensemble ML model outcomes with the HBV model result.

Chapter 8 summarizes the research's conclusions and recommendations based on the various case studies presented in this dissertation. Finally, we proposed various future study directions.

CHAPTER 2:

HYDROLOGICAL MODELS, MACHINE LEARNING, AND THE ROLE OF REMOTE SENSING IN HYDROLOGY

2.1. General Overview

Measurement of river discharge is necessary for hydrology and water resource management. Knowing the rate of a river's flow and how long it takes for flows to travel downstream is crucial for watershed modelling, reservoir operations, and flood predictions. Consequently, there is a vast demand for discharge data that are long-term, continuous, geographically consistent, and easily accessible (A. Sichangi et al., 2018).

However, extensive monitoring systems of land surface hydrologic fluxes in many regions of the world are unavailable because there may not be enough gathered gage information or lakes and rivers may be too far away or inaccessible for gage installation and maintenance. Given these difficulties, current networks provide less coverage (Bjerklie et al., 2018). Hence, innovative hydrologic simulation methods are needed. First, we must reduce our reliance on in situ measurements, such as the numerous gauging stations built along the modelled river (Gigi et al., 2019).

The mentioned concerns can be addressed by calculating streamflow using meteorological variables. Various models may be employed to estimate streamflow successfully for such applications. These models require meteorological variable data and catchment and model parameter values. However, most parts of the world, particularly developing countries, lack access to continuous meteorological time series variables. As a result, several researchers have tried to estimate these meteorological variables using remote sensing data.

Regardless of scale issues, integrating climate, weather, numerical modelling, and remote sensing can be a practical answer to the significant water conundrum human beings currently face (Peters-Lidard et al., 2019). As a result, satellite-based monitoring of surface storage change and river discharge has long been a focus of the remote sensing community (Bjerklie et al., 2018).

The machine learning discipline originated from the broader field of artificial intelligence, which has been increasingly active in recent years and tries to replicate the intellectual capacities of humans through computers, which gives many highly accurate and efficient simple algorithms for practitioners to utilize. It appears that discovering how and where machine learning may assist in generating forecasts for human worries to comprehend enormous volumes of data is profitable and becoming necessary for computer scientists and engineers.

ANN is a well-known machine learning architecture consisting of interconnected nodes, similar to neurons in the human brain, linked by weighted synaptic connections (Kumar et al., 2019). This algorithm, inspired by biological brain activity, is frequently used in water resource forecasting and hydrology. In newer and noisier environments, this algorithm is quicker, more flexible, more adaptable, and resilient and can handle a broad range of challenges.

Numerous studies using ANNs in engineering-related disciplines, such as rainfall-runoff modelling, time series prediction, and rule-based control, have been successfully completed. Engineers often use back propagation (BP) method network models in ANNs. The three-layer BP network model is adequate for solving all engineering issues in simulation and forecasting (Kumar et al., 2019). After the impressive results of ANNs, several other single machine learning (ML) algorithms were developed. Currently, hybridization and assembly of these algorithms are also becoming popular. The following paragraphs address literature relating to streamflow modelling, remote sensing, machine learning, and deep learning principles, in line with the objective of this research.

2.2. Streamflow Modelling

2.2.1. The Modelling Concept

Hydrology's rich history dates back millennia (Biswas, 1970). However, hydrologic modelling may date back to the 1850s, when Mulvany devised the rational approach for estimating peak discharge owing to a rainfall event of uniform intensity and length equal to or greater than the period of concentration (Singh, 2018). The approach was designed for small urban watersheds, which are also applied in urban drainage planning (Singh, 2018).

The streamflow estimation model is a simplified natural system model that includes rainfall, temperature, evapotranspiration, and groundwater. This model representation is constructed to imitate the water cycle or natural system by combining a set of mathematical expressions and logical arguments (Refsgaard & Knudsen, 1996). St. Venant, in 1871, developed equations for modelling surface flow, now known as St. Venant equations (Singh, 2018). Manning also devised an equation for calculating flow velocity in open channels two decades later, in 1895 (Manning, 1816). Imbeau established a relationship between the storm runoff peak and rainfall intensity in 1892. Sherman established the unit hydrograph idea in 1932, which set the foundation for linear systems hydrology. Horton developed a semiempirical formula for overland flow in 1939. Years later, in 1940, Barnes invented a method for hydrograph separation. Keulegan demonstrated the applicability of the simplified momentum equation for modelling overland flow in 1944 using hydraulic principles. In 1945, Clark created a unit hydrograph technique for calculating the rainfall-runoff hydrograph (Singh, 2018).

These works set the foundation for both conceptual and physically based rainfall-runoff modelling. Based on a large amount of data, the Soil Conservation Service (SCS) of the United States Department of Agriculture (USDA) developed the SCS-Curve Number (CN) method in 1956 for computing the amount of runoff generated by a rainfall event, taking abstractions, antecedent soil moisture conditions, hydrologic conditions of land use and land cover, and soil type into account. This approach is still widely used to calculate the amount of runoff or excess rainfall from small and medium agricultural watersheds. It has been expanded to urban and wooded watersheds. Nielsen and his colleagues also examined the contribution of the source area to runoff in 1959 (Singh, 2018).

Many revolutionary improvements in modelling various components of the hydrologic cycle were accomplished for over a century until the 1960s. Some of these breakthroughs were founded on mathematical physics rules, while others were based on laboratory and/or field investigations. The present status of hydrologic research and engineering owes a significant lot to breakthroughs made before 1960. Chow's handbook of applied hydrology, published in 1964, gave an up-to-date summary of hydrologic developments until the 1960s (Singh, 2018).

The 1960s witnessed the start of a computer revolution, and hydrologic modelling advanced significantly. The computer gave the ability to perform calculations that were previously unavailable. Consequently, a new hydrological field, digital or numerical hydrology, emerged. Another area that also evolved was statistical or stochastic hydrology, which often demanded the analysis of vast amounts of data (Burges, 2004). Then, other significant developments took place, such as the simulation of the entire hydrologic cycle. The first model was the Stanford Watershed Model (SWM). This model considered the physical processes associated with the hydrologic water cycle at the catchment scale (Burges, 2004).

Spatial analytic approaches were presented on a geographic information system (GIS) platform in the late 1960s and early 1970s. This invention enabled the development of streamflow estimate models in a GIS platform, allowing for the spatial variability of hydrometeorological factors in hydrological processes (Lillesand et al., 2015). The development of improved infrastructure, such as relevant institutions, updated technology to assess model input variables, and increased human capability, propelled hydrological modelling to a new level, forming various models. The prominent models include the Storm Water Management Model (SWMM), TOPMODEL, MIKE SHE, and Soil & Water Assessment Tool (SWAT) (Singh Vijay P. & Woolhiser David A., 2002). These models differ depending on their internal processes, meteorological parameter requirements, and catchment and model variables to predict streamflow.

Given the scarcity of data, selecting a specific model for an application might be a major challenge. However, the categorization of streamflow estimating models informs the applicability of models for diverse contexts.

2.2.2. Classification of Streamflow Estimation Models

There are many classification schemes for rainfall-runoff models. However, the most basic distinction is generally made between lumped and distributed models, which take model parameters as a function of space and time (Keith J Beven, 2005).

According to Moradkhani and Sorooshian, in lumped models, the entire river basin is treated as a single unit with no regard for spatial variability. Thus, the outputs are generated without taking into account the spatial processes. In contrast, a distributed model can make spatially distributed predictions by dividing the entire catchment into small units, usually square cells or triangulated irregular networks, so that the parameters, inputs, and outputs can vary spatially (Moradkhani &

Sorooshian, 2008). In addition, the other classification is deterministic and stochastic. The deterministic model will create the same outcome for a single set of input values, but stochastic models might produce varied output values for the same set of inputs.

We may group as static and dynamic models based on the time factor. The static model does not contain time, but the dynamic model does. Moradkhani and Sorooshian classed the models as either event-based or continuous. The former generates output exclusively at certain times, while the latter generates continuous production (Moradkhani & Sorooshian, 2008). The popular classifications include empirical, conceptual, and physically based models (Devia et al., 2015).

- **Empirical Models (Metric Model)**

These models are mainly based on hydrometeorological observations and attempt to define the resulting rainfall-runoff interactions. Engineers and hydrologists have employed these modelling approaches for over a century to estimate streamflow. During this period, the methodology changed due to improved processing power, advancements in available analytical tools, and a continually rising number and duration of records; hence, these models are also known as data-driven models (Devia et al., 2015).

It employs mathematical equations generated from continuous input and output time series rather than physical catchment dynamics. These models are only valid inside the boundaries. A unit hydrograph shows this approach. Statistical approaches use regression and correlation models to determine the functional connection between inputs and outputs. Machine learning approaches such as ANN and fuzzy regression are also employed in data-driven methodologies (Devia et al., 2015).

- **Conceptual Models (Parametric Models)**

Conceptual models understand runoff processes by linking simplified components in the entire hydrological process. They are based on reservoir storage and simplified equations of physical hydrological processes that provide a conceptual understanding of catchment behavior (Devia et al., 2015). This approach employs semiempirical equations, and model parameters are determined from field data and calibration.

Calibration requires a significant number of meteorological and hydrological data. The calibration process requires curve fitting, which makes interpretation challenging; hence, the impact of land-use change cannot be anticipated with great certainty. Many conceptual models with various complexities have been created. Crawford and Linsley created the first major conceptual model, Stanford Watershed Model IV (SWM), in 1966, with 16 to 20 parameters (Burges, 2004).

- **Physically Based Model**

A physically based model is an idealized mathematical description of the underlying phenomena. These are also known as mechanistic models since they involve physical process concepts. It employs measurable state variables that are functions of both time and space. Finite difference equations reflect the hydrological processes of water transport. It does not need large hydrological and meteorological data for calibration, but it necessitates assessing many factors reflecting the watershed's physical features. This approach needs massive data, including soil moisture content, initial water depth, terrain, topology, and river network dimensions (Devia et al., 2015).

Because of the usage of parameters with physical interpretation, a physical model may overcome many of the shortcomings of the above two models. It may supply a massive quantity of information even when it is beyond the border and can be used in various circumstances. The Soil and Water Assessment Tool (SWAT) is an example of a physically based model. Additional process models include the Institute of Hydrology Distributed Model (IHDM) and the SHE/MIKE SHE models, which are also commonly used to simulate runoff (Devia et al., 2015).

2.3. Role of Remote Sensing in Hydrology

The science and art of acquiring information about an item, region, or phenomenon via the analysis of data obtained by a device that is not in touch with the object, area, or phenomenon under examination is the definition given for remote sensing in its broadest sense (Lillesand et al., 2015). You are using remote sensing to read these words. The light reflected from this page is sensed by your eyes, functioning as sensors. Your eyes get impulses that are correlated to the quantity of light reflected from the dark and bright portions of the page as "data." Your brain computer analyses or interprets these facts to help you understand the dark regions of the page as groups of letters creating words. Beyond this, you understand that words form sentences, and you interpret each phrase's meaning (Lillesand et al., 2015). In recent years, academics have become more interested in remote sensing and its applications. The 2016 International Geoscience and Remote Sensing Symposium (IGARSS), which received over 3,000 papers, is one obvious sign. This field is rapidly increasing and changing (Kwan et al., 2016).

2.3.1. History of Remote Sensing

The birth of the camera marks the beginning of the history of remote sensing. Evelyn L. Pruitt of the United States office of naval research coined the phrase "remote sensing" in 1960. The first aerial image was taken in 1858, 162 years before the phrase "remote sensing" was coined (Cazenave et al., 2016).

In 1038 AD, Al Hazen, an Arabian mathematician, explained the notion of a camera obscure to see a solar eclipse. Cardano was the first to use an optic on the camera lens to create a higher-quality picture in 1550. In 1666, Sir Isaac Newton discovered that light could be dispersed into a spectrum of red, orange, yellow, green, blue indigo, and violet when working with prisms. He discovered that by combining a second prism, light could be recombined into white light (Khan, 2000).

Almost two centuries later, in 1858, French photographer and balloonist Gaspard Felix, known as "NADAR," took the first aerial shot from a balloon suspended several hundred feet above the Bievre Valley. His preliminary work, however, was lost. In 1859, he contacted the French military to take "military images" during the French army's battle in Italy and prepare maps from aerial photographs. During the 1860 Civil War, aerial surveillance and perhaps photography for military reasons were obtained (Khorram et al., 2016). NADAR's varied efforts to develop and promote aerial photography continue. In 1868, he ascended several hundred feet in a tethered balloon to picture Paris from above (Khorram et al., 2016).

M. Arthur Batut used a kite to take the first aerial pictures in the 1880s. It was captured in Labruguiere, France. Years later, in 1887, Germans started experiments with aerial photography and photogrammetric methods for quantifying features and areas in the forest. Julius Neubranner, a photographer, created and patented a breast-mounted airborne camera for carrier pigeons in 1903. The birds first appeared in the Dresden International Photographic Exhibition in 1909. The Wright brothers' first successful flight of a heavier-than-air aircraft occurred in 1903. It was another sort of aerial platform accessible (Khan, 2000).

The use of aerial photography increased dramatically during World War I in 1914. Photogrammetric engineering was initially published in the United States of America in 1934. Albert W. Stevens captured the first photograph of the actual curvature of the earth from a free balloon at an altitude of 72000 feet in 1936. World War II, which began in 1940, witnessed significant development and recognition in aerial photography, which continues to this day. The British Secret Intelligence Service was alerted of a new rocket being built at Peenemunde in late 1942. On June 23, 1943, an aerial photoreconnaissance aircraft was sent and captured the first photograph of the V-2 rocket (Khan, 2000).

During the 1950s, aerial photography evolved from efforts begun during World War II and the Korean War. Color-infrared technology has also helped distinguish various kinds of vegetation and detect unhealthy and damaged plants (Khan, 2000). The Television Infrared Observation Satellite Program (TIROS-I) was the first meteorological satellite launched in 1960. The Nimbus weather satellite program began with the launch of Nimbus-I in 1964 (Khorram et al., 2016).

The Landsat satellite series has the longest historical records in RS data collection, beginning in the early 1970s. The National Aeronautics and Space Administration (NASA) has successfully operated the Landsat series (Landsat 1, 2, 3, 4, 5, 6, 7, and 8) for the last 50 years, and sensor capabilities have been continuously upgraded from Landsat 1 to Landsat 8. (Khan, 2000). These are only a handful of the numerous satellites deployed by various governments throughout the globe. In 1986, "French: Satellite Pour l'Observation de la Terre" (SPOT-I) was launched; in 1990, SPOT-II was launched; in 1993, SPOT-III was launched; in 1995, India launched Indian Remote Sensing (IRS); in 2004, China launched Roc-Synthetic Aperture Radar (SAR)-2; and in 2011, Pakistan launched Bader-B, Bader-2 (Khorram et al., 2016).

2.3.2. Remote Sensing Sensors

There are two kinds of remote sensing systems/sensors: passive and active. The sun is the primary source of energy for remote sensing. Solar energy reflected or emitted from the Earth's surface is recorded by remote sensors. A passive sensor obtains its energy from some distance other than the sensor itself. Photographic cameras and electro-optical, thermal infrared, and antenna sensors are passive sensors (Qihao Weng, 2016).

Electro-optical sensors are used to measure satellite pictures. Each sensor may have a different number of detectors. The detectors collect and measure reflected or emitted light from the ground as electrical signals, which are then converted into numerical values that are either kept onboard or communicated to a receiving station on the ground. Incoming light is often directed toward various detectors, so illumination within a specified wavelength range may be captured as a spectral band. The most common kinds of scanning systems used to gather multispectral picture data are across-track and along-track scanners, and these remote sensors have four distinct resolutions:

- **Spatial resolution:** The amount of spatial detail portrayed in a picture, commonly described as the smallest potential feature detectable from an image. As a result, the smaller the item that can be detected, the greater the spatial resolution.
- **Spectral resolution:** The spectral bandwidth used to acquire data.
- **Radiometric resolution:** Refers to a sensor's sensitivity to incoming radiance or how much radiance must change on the sensor before a change in the recorded brightness value occurs.

- **Temporal resolution (revisit time):** The length of time it takes for a sensor to return to a previously photographed region, also known as the repeat cycle or the time gap between two subsequent picture acquisitions.

The platform is the vehicle on which the remote sensor is installed (i.e., satellite, aeroplane balloon). Finally, data from the RS system are transferred to the ground station/s. These data are captured on hardware and made available to data analysts for analysis and interpretation. The RS data processing and interpretation degree is closely related to the applications (Qihao Weng, 2016).

- **Satellites and Sensors on Board**

Sensor technology advancements have resulted in significant gains in Earth observation (EO) over recent decades. Throughout the globe, new satellites, aircraft, and ground-based remote sensing technologies are being developed. These incredible accomplishments have resulted in an explosion of Earth observation data, ranging from low to high spatial, temporal, and radiometric resolution, overgrowing in bulk and diversity. For example, when you read this paragraph, NASA might have acquired 1.73 terabytes of data from approximately 100 active missions. During the fiscal year ending September 2016, NASA's Earth Science Data Systems (ESDS) delivered approximately 1.5 billion files of data (approximately 14.6 petabytes) to users all around the globe through the Earth Data Search Capability (EDSC) (Ramapriyan & Murphy, 2017). Earth observation has entered a new age, the era of big data (L. Chen & Wang, 2018).

These data sets provide information on the land surface, energy, climatic elements, and water vapour. Some observation systems are purpose-built for hydrological study. Eight of NASA's current 19 earth science missions, such as Soil Moisture Active and Passive (SMAP), Global Precipitation Measurement (GPM), and Gravity Recovery and Climate Experiment (GRACE), are extremely important to hydrology (L. Chen & Wang, 2018). Emerging satellite missions have generated large amounts of data for hydrology. Every day, NASA SMAP sends back 458 GB of data on soil moisture. It is time for hydrology to embrace data-intensive research, the fourth paradigm (L. Chen & Wang, 2018).

Table 2. 1 Summary of the ongoing and future missions highly relevant to the hydrological cycle (L. Chen & Wang, 2018).

Hydrological Cycle Component	Mission/Sensor	Spatial Resolution(km)	Temporal Resolution(days)	Launch Year
Precipitation	GPM	5	0.125	2014
Evapotranspiration	Terra/MODIS	0.5	1	1999
	Aqua/MODIS			2002
	Landsat 8			2013
	Landsat 9			2023
Snow and Ice Cover	ICESat-2	0.01	33	2018
	CryoSat-2	0.25	369	2010
Evapotranspiration	Terra/MODIS	0.5	1	1999
Soil Moisture	SMOS	36	3	2011
	SMAP	36	3	2015
Streamflow	SWOT	0.1	11	2022
Ground Water	GRACE	220	30	2002
Water Cycle	WCOM	2–5	-	2020

2.3.3. Streamflow Monitoring Satellites

Satellite remote sensing data have been employed in various engineering applications in the past. However, no past mission can monitor streamflow discharge on a global or regional scale. Many future missions, such as the SWOT mission, launched recently on December 16, 2022, will estimate global and regional water discharge, which will help to alleviate this dilemma (L. Chen & Wang, 2018).

- **Streamflow Estimation Using Remote Sensing Data**

The roles of remote sensing in obtaining streamflow information are mainly grouped as follows: -

1. **Streamflow modelling:-** Remotely sensed data as "input" for a hydrological model (Birkinshaw et al., 2014; Capolongo et al., 2019; Fereidoon et al., 2019; W. Sun et al., 2015; W. C. Sun et al., 2010; Xiong & Zeng, 2019; Ymeti, 2007).
2. **Streamflow estimation:-** Estimating streamflow by remote sensing data alone without using any hydrological model. (Gleason et al., 2014; A. Sichangi et al., 2018; Tarpanelli et al., 2019; Zaji et al., 2019).

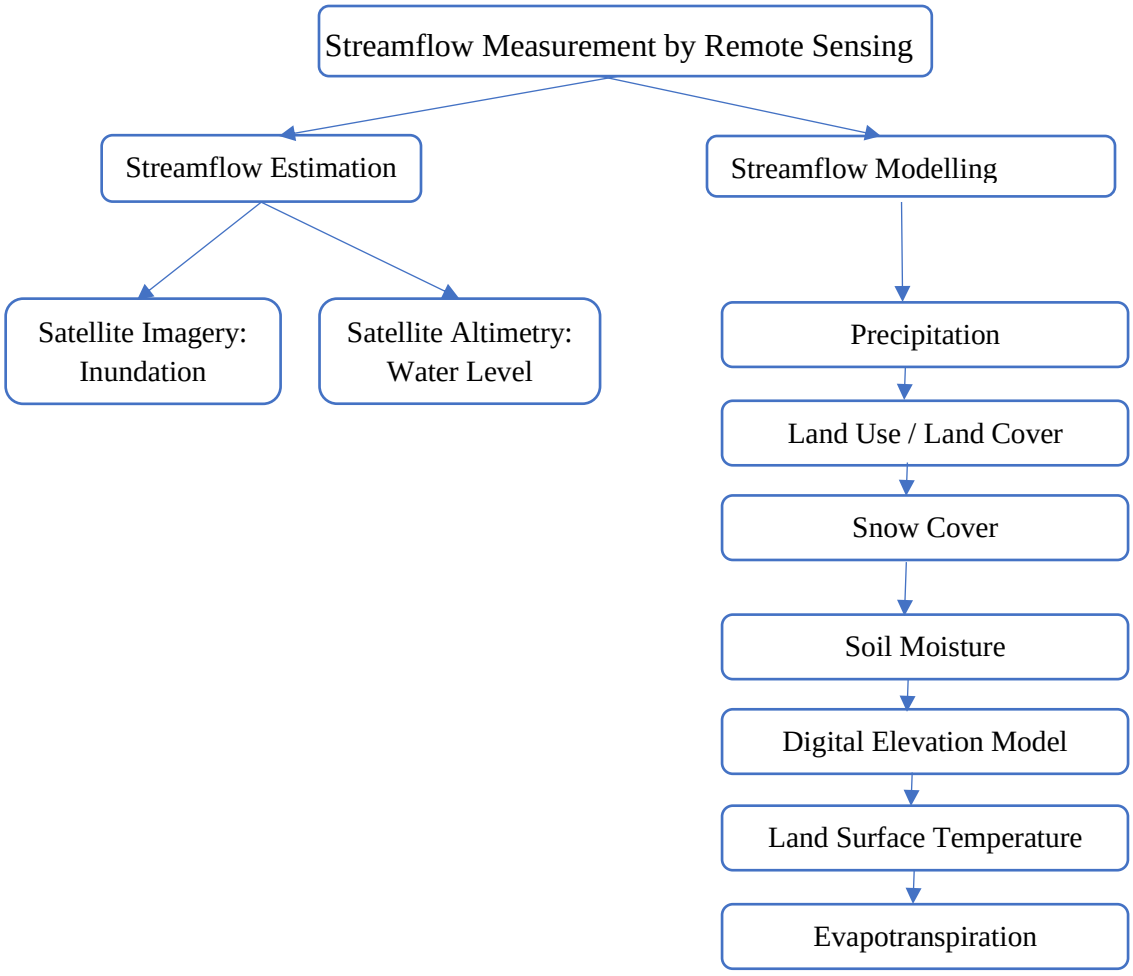


Figure 2. 1 Role of remote sensing in streamflow measurement (M L Tan, 2014).

Table 2. 2 Commonly used space-borne remote sensors for surface water detection listed by group [a] (Huang et al., 2018).

Sensor Group	Satellite / Sensor	Number of Bands	Spatial (m)	Temporal Resolution (day)	Maximum Swath at Nadir (km)	Scale of Application [b]	Data Distribution policy (costs)	Data Availability
Coarse-resolution sensor	NOAA/ AVHRR	5	1100	0.5	2800	R-G	no	1978-
	MODIS	36	250-1000	0.5	2330	R-G	no	1999--
	Suomi NPP-VIIRS	22	375-750	0.5	3040	R-G	no	2012--
	MERIS	15	300	3	1150	R-G	no	2002-2012
	Sentinel-3 OLCI	21	300	2	1270	R-G	no	2016--
Medium resolution sensor	Landsat	4-9	15-80	16	185	L-G	no	1972--
	SPOT	4-5	2.5-20	26	120	L-R	yes	1986--
	Aster	14	15-90	16	60	L-G	no	1999--
	Sentinel-2 MSI	13	10-60	5	290	L-R	no	2015--
High-resolution sensor	IKONOS	5	1-4	1.5-3	11.3	L-R	yes	1999--
	Quick Bird	5	0.61-2.24	2.7	16.5	L	yes	2001--
	World View	4-17	0.31-2.4	1-4	17.6	L	yes	2007--
	Rapid Eye	5	5	1-5.5	77	L-R	yes	2008--
	ZY-3	4	2.1-5.8	5	50	L-R	yes	2012--
	GF-1/GF-2	5	1-16	4-5	800	L-R	yes	2013--

[a] *The spatial resolution, temporal resolution, and spectral resolution (number of bands) vary among sensors. The area coverage (swath) determines the scale of application and varies among sensors.* [b] *L, landscape; R, regional; G, global; L-R, landscape to regional; L-G, landscape to global; R-G, regional to global.*

2.3.4. Rainfall Estimation Using Remote Sensing Data

If all of the world's rain gauges were assembled in one spot, they would roughly cover the size of two basketball courts. Collecting worldwide precipitation data from the ground is almost impossible since rain gauges and sensors cannot be installed anywhere rain may fall (Blumenfeld, 2015). Satellite observations alleviate these shortcomings by providing more spatially homogenous and temporally full coverage over vast portions of the globe using Improved InfraRed (IR) and Micro Wave (MW) types of equipment (Serrat-Capdevila et al., 2016).

In April 1960, the first Television and Infrared Observation Satellite (TIROS) was launched, generating imagery of clouds that could be compared to simultaneous meteorological measurements. Since then, the number of satellite sensors for studying the atmosphere has increased dramatically. Satellite sensors are the only equipment capable of providing homogenous precipitation readings worldwide. Visible infrared (VIS/IR) sensors on GEOstationary (GEO) and low earth orbit (LEO) satellites, passive microwave (PMW) sensors on LEO spacecraft, and active MW sensors on LEO satellites are the three types of sensors. Corresponding precipitation derivation approaches have been developed, including VIS/IR-based methods, active and passive MW techniques, and a combined VIS/IR and MW approach (Q. Sun et al., 2018). The climate prediction center MORPHing (CMORPH), Tropical Rainfall Measuring Mission (TRMM), Precipitation Estimation from Remotely Sensed Information using Artificial Neural Networks (PERSIAN), Global Precipitation Measurement (GPM), Integrated MultisatellitE Retrievals for GPM (IMERG), Climate Hazards group InfraRed Precipitation with Station data (CHIRPS), and Multi-Source Weighted-Ensemble Precipitation (MSWEP) are the primary satellite precipitation data sets presently accessible (Q. Sun et al., 2018).

- **Tropical Rainfall Measuring Mission (TRMM)**

The TRMM contribution to the science and technology of remote sensing precipitation is much beyond the mission's primary goals. TRMM's outstanding success is undeniably due to the excellent quality and lifespan of its sensors, which have produced a continuous record of high-quality infrared and microwave radiometer and precipitation radar observations for more than 22 years (Andronache, 2018).

TRMM's ultimate goal was to quantify tropical rainfall using active sensors (a precipitation radar, the first of its type in space) and passive sensors (the Visible and InfraRed Scanner (VIRS) and the TRMM Microwave Imager (TMI)) (Blumenfeld, 2015). TRMM also carried a Lightning Imaging Sensor (LIS) to investigate the distribution and variability of lightning, as well as the Clouds and Earth's Radiant Energy System (CERES), a broadband scanning radiometer to measure emitted and reflected radiative energy from the Earth's surface and the atmosphere (Blumenfeld, 2015). TRMM data were used to obtain multiyear sets of tropical and subtropical rainfall observations; to gain a better understanding of the interactions between sea, air, and landmasses and their impact on global rainfall and climate; to improve tropical rainfall modelling; and to improve satellite rainfall measurement techniques (Blumenfeld, 2015).

- **Terra and Aqua Moderate-Resolution Imaging Spectroradiometer (MODIS)**

This study uses MODIS data, one of the most often used missions for estimating vegetation indices. The Terra satellite was launched in 1999, and the Aqua satellite was launched in 2002. Every 1 to 2 days, the two satellites scan the whole Earth's surface, collecting data in 36 spectral bands (groups of wavelengths) at three spatial resolutions: 250, 500, and 1000 meters. MODIS bands have a very high signal-to-noise ratio (SNR), making them ideal for usage in small- and medium-scale catchments (L. Chen & Wang, 2018). MODIS Terra and Aqua level 1 data are applied for this investigation.

- **Global Precipitation Measurement (GPM)**

As a result of the TRMM project's success, scientists began planning the GPM mission, which started in the early 2000s. GPM was created to enhance the TRMM data and ensure data continuity (Blumenfeld, 2015). The National Aeronautics and Space Administration (NASA) and the Japan Aerospace and eXploration Agency (JAXA) launched the GPM project as an international satellite effort to standardize and enhance global precipitation observations from space. The GPM Core Observatory (GPM-CO, launched in February 2014) will be used to carry out the mission. It will be equipped with a Ka/Ku-band Dual-frequency Precipitation Radar (DPR) a multifrequency (10-183 GHz) microwave radiometer, and a GPM Microwave Imager (GMI) (Skofronick-Jackson et al., 2018).

GPM enhances TRMM's capabilities in a variety of ways. Even though GPM only has two sensors compared to TRMM's five, these two instruments (a Dual-frequency Precipitation Radar (DPR) and a radiometer termed GPM Microwave Imager (GMI)) are among the most powerful yet created to monitor precipitation from space. The DPR is space's first dual-frequency radar, capable of constructing 3-D profiles and estimating precipitation intensity ranging from rain to snow and ice. The GMI has a broader frequency range than the TRMM (13 channels vs 9 channels), allowing GPM to assess precipitation intensity and type across all cloud levels utilizing a larger data swath (Blumenfeld, 2015).

GPMs have goals as a science-discovery mission (Blumenfeld, 2015), which are as follows:

1. Improving hydrological modelling and prediction of high-impact natural hazard occurrences (e.g., floods, droughts, landslides, and hurricanes) via enhanced temporal sampling and model-downscaled precipitation outputs.
2. Development of new reference standards for space-based precipitation measurements utilizing active and passive remote sensing methods.
3. Improving understanding of precipitation systems, water cycle variability, and freshwater supply by improved observations of worldwide precipitation spacetime distribution.
4. Improving climate modelling and prediction capabilities by accurately measuring latent heating, precipitation microphysics, and surface water fluxes.
5. Improving weather forecasting abilities via more precise estimations of instantaneous precipitation data and error characterizations.

- **Integrated Multi-Satellite Retrievals for GPM (IMERG-final)**

The IMERG method estimates global precipitation using data from the GPM satellite constellation. This method is helpful in locations where there are no precipitation-measuring sensors. The IMERG algorithm mixes early TRMM satellite precipitation estimates (2000-2015) with more recent GPM satellite precipitation projections (2014 - present). To fulfil the demands of data consumers, IMERG data are accessible in early, late, and final run formats and processing types. The data are also available on the NASA website (pmm.nasa.gov/data-access/downloads/gpm) and may be downloaded at half-hourly, 3-hourly, and daily intervals. We chose the daily IMERG-final precipitation data product since it is typically suggested for research purposes. The spatial resolution of this product is 0.1°/10 km, and it was extracted using Python code for every case study's individual meteorological station data points and locations.

- **Climate Hazards Group Infrared Precipitation with Stations (CHIRPS)**

The satellite-based precipitation product CHIRPS provides a quasiglobal rainfall data set over approximately three decades. These gridded rainfall time series data, with a spatial precision of 0.05°/5 km, have been accessible since 1981 (Sulugodu & Deka, 2019). We used the Google Earth Engine (GEE) code editor for each meteorological station and case study location to extract daily precipitation data.

- **Multisource weighted-ensemble precipitation (MSWEP-V2)**

The MSWEP-V2 data set, which spans the years 1979 to the present and has a 0.1° (11 km at the equator) resolution, was created to optimally integrate a variety of gauge, satellite, and reanalysis estimates. Since the release of MSWEP version 1 (0.25° spatial resolution) in May 2016, it has been successfully used globally for various research projects (Beck et al., 2019). The upgraded MSWEP-V2 was employed in this work, and Python code was used to extract the precipitation time series for each meteorological station.

2.4. Machine Learning

The broad field of artificial intelligence, which strives to have computers imitate human intelligence, gave rise to the field of machine learning. A critical topic in the science of machine learning is how to enable computers to "learn" (Gunnar, 2004). Without taking the formal mathematical structure of the model into account, machine learning investigates the relationship between the output and its pertinent variables using one or more algorithms. Nearest neighbors, naive Bayes, decision trees (DTs), support vector machines (SVMs), and artificial neural networks (ANNs) are a few examples of machine learning methods (J. Yan et al., 2019).

ANN is a data-driven method that aims to provide the lowest variance solution to a given issue. ANNs can recreate practically any kind of input-output relationship when adequately trained. In particular, ANNs have been widely used in remote sensing applications because they provide a simple but effective way of merging input data from diverse sources into the same retrieval algorithm (Tarpanelli et al., 2019).

2.4.1. Deep Learning Models

Deep learning models are a subset of a broader family of machine-learning algorithms, including deep belief networks (DBNs), recurrent neural networks (RNNs), and convolutional neural networks (CNNs). These models have been used in a variety of research areas, such as time series analysis, computer vision, and voice recognition (Cho, van Merriënboer, et al., 2014; Livieris et al., 2020; Sezer et al., 2020; Y. Wang et al., 2018; Zou et al., 2020).

The following section will briefly discuss machine learning or deep learning architectures employed for this study:

- **Linear Regression**

Linear regression is the most basic base learner used in this investigation. A linear connection exists between the dependent and independent variables in linear regression (Dabhade et al., 2021). A wide variety of recent modelling tools are built on linear regression. When the sample size is small or the information is limited, linear regression is often an appropriate approximation of the underlying regression function (Su et al., 2012).

- **Least Absolute Shrinkage and Selection Operator (LASSO)**

In statistics and machine learning, LASSO is a regression analysis approach named after Robert Tibshirani (Tibshirani, 1996). LASSO combines variable selection with regularization to increase the predicted accuracy and interpretability of the resulting statistical model. Furthermore, it applies the LASSO penalty (L1 shrinkage) to the least-squares technique, intending to shrink its coefficients while allowing the elimination of noninfluential predictor variables through coefficient nullification (Kukreja et al., 2006; Tyralis et al., 2019).

- **Support Vector Regression (SVR)**

SVR is a regression variant of the support vector machine (SVM) proposed by Cortes and Vapnik in 1995 (Cortes & Vapnik, 1995) that has been widely used in hydrological simulations. In SVR, the independent features are separated by a hyperplane. Support vectors, or data points near the hyperplane, are used to build the boundary line. Unlike other regression models, which seek to minimize the difference between actual and predicted values, SVR seeks to fit the ideal line within a certain threshold value or distance between the hyperplane and the boundary line (Dabhade et al., 2021).

One of the key benefits of SVR is that its computational complexity is independent of the input space size rather than the number of support vectors (a small set of training data samples). SVR creates a model that can reflect a variable's relevance in explaining the relationship between input and output (F. Zhang & O'Donnell, 2020). Furthermore, as indicated by its great prediction accuracy, it has extraordinary potential for generalization (Awad & Khanna, 2015).

- **eXtreme Gradient Boosting (XGB)**

Tianqi & Carlos (2016) introduced XGB, a version of the gradient-boosted decision tree approach. Predictions are created using weak learners who constantly improve due to the errors of their predecessors. The basic concept behind XGB is to accelerate the training process by using all samples and modifying their weights (Zounemat-Kermani, Batelaan, et al., 2021). When the residual error is small enough or reaches a certain number of iterations, the prediction result is the weighted average of the prediction results from each round. XGB is the gradient boosting framework's fast and scalable version. Furthermore, it increases performance by reducing overfitting with a more regularized model formalization. It has recently gained prominence as the

algorithm of choice for many winning teams in machine learning competitions (Carmona et al., 2019).

- **Artificial Neural Network (ANN)**

Streamflow estimation has used ANN, a prominent data-driven modelling approach, over the last 20 years. Its capacity to capture the complex and nonlinear input-output relationships seen in hydrological processes within a river basin is vital. For nonlinear hydrological time series modelling, the conventional feed-forward neural network (FFNN) with three layers of input-output and hidden layers trained using the backpropagation (BP) algorithm is becoming increasingly popular.

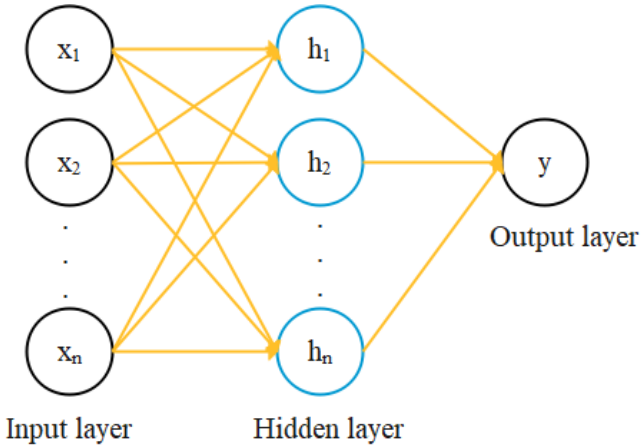


Figure 2. 2 Typical architecture of ANN Multi-Layer Perceptron (MLP) neural network (After: Oyebode and Stretch, 2019).

$$\hat{y}_j = f_j\left[\sum_{h=1}^{m} w_{jh} * f_h\left(\sum_{i=1}^{n} w_{hi}x_i + w_{hb}\right) + w_{jb}\right] \quad (1)$$

where i, h, j, b, and w indicate, respectively, neurons of the input, hidden, and output layers and bias and applied weight (or bias) by the neuron; f_h and f_j show activation functions of hidden layer and output layer, respectively; x_i, n, and m represent, respectively, input value, input, and hidden neuron numbers; and y and $Y^{\wedge}{}_j$ denote the observed and calculated target values, respectively.

The values of the hidden and output layers and accompanying weights might be changed and calibrated during the model's calibration phase (Nourani, Molajou, Najafi, et al., 2019). The usage of ANN models in hydrological simulations has increased dramatically due to the capacity of ANNs to connect input and output variables in complicated hydrological systems without the requirement for preliminary information about the nature of the process (Nourani et al., 2019).

- **Long Short-Term Memory (LSTM)**

The long short-term memory network (LSTM) differs from the standard ANN (MLP) network in that its layers of neurons contain recurrent connections, allowing the state from earlier activation time steps to be utilized to create an output. LSTM is a recurrent neural network (RNN) that can learn long-term dependencies. RNN is a circular network with an extra input representing the state of the neuron in the hidden layer at earlier time steps (Zhu et al., 2020). In the hidden layer, the LSTM replaces the typical neuron with a memory cell and three gates: an input gate, a forget gate, and an output gate (L. Yan et al., 2019). Overcoming vanishing and exploding gradients and holding memory to capture long-term temporal dependence with input sequences are two significant advantages of LSTM over RNN.

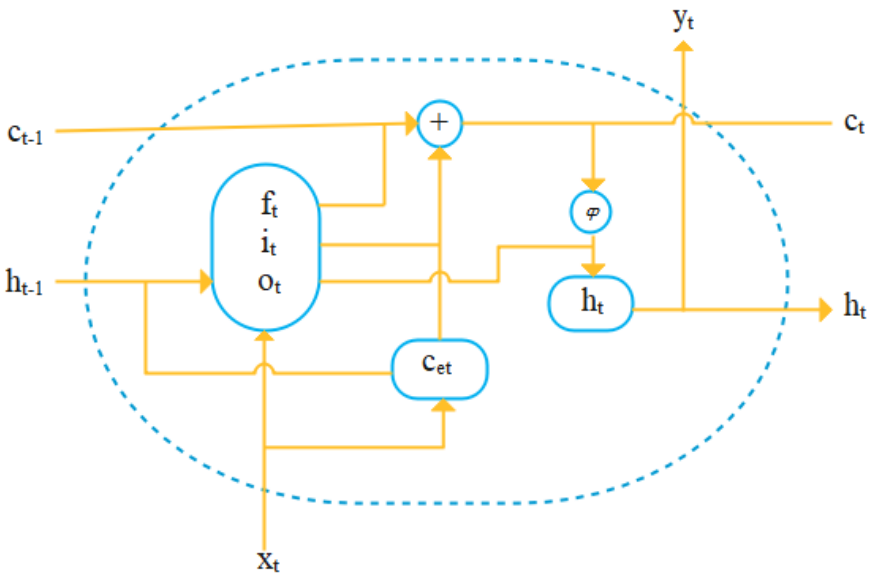


Figure 2. 3 LSTM memory cell with three gated layers: forget gate f_t, input gate i_t, and output gate o_t, controlling the activation of cells c_{t-1} and c_t (After: Sahoo et al., 2019).

Forget Gate: - Decides what information to discard from the cell

$$f_t = \sigma(u_f x_t + w_f h_{t-1} + b_f) \quad (2)$$

Input Gate: - Decides what values from the input to update the memory state

$$i_t = \sigma(u_i x_t + w_i h_{t-1} + b_i) \quad (3)$$

Output Gate: - Determines what to output depending on the input and the cell's long-term memory

$$o_t = \sigma(u_o x_t + w_o h_{t-1} + b_o) \quad (4)$$

Cell and Hidden State

$$C_{et} = \tanh (W_c X_t + U_c h_{t-1} + b_c) \quad (5)$$

$$C_t = f_t * C_{t-1} + i_t * C_{et} \quad (6)$$

$$h_t = \tanh (C_t) * O_t \quad (7)$$

where W_i, W_f, W_o, and W_c are weights that map the hidden layer input to the three gates of input, forget, and output, respectively, while the U_i, U_f, U_o, and U_c weight matrices map the hidden layer output to gates; b_i, b_f, b_o, and b_c are vectors. Moreover, C_t and h_t are the outcome of the cell and the outcome of the layer, respectively (Apaydin et al., 2020).

- **Bidirectional LSTM (Bi-LSTM)**

Bi-LSTM is another strategy for maximizing RNN performance by stepping through input time steps in both the forward and backward directions. Despite Bi-LSTMs being designed for voice recognition, the usage of bidirectional input sequences is currently one of the primary possibilities for sequence prediction. The hidden layer of the Bi-LSTM model must store two values: h_t in the forward computation and h'_t in the backward calculation. The ultimate output value Y_t is generated by merging the forward and backwards layer outputs (Yin et al., 2020).

Each point in the output layer's input sequence receives entire past and future contextual information. Since there is no information movement between the forward and backwards hidden layers, the extended graph is acyclic (Zou et al., 2020).

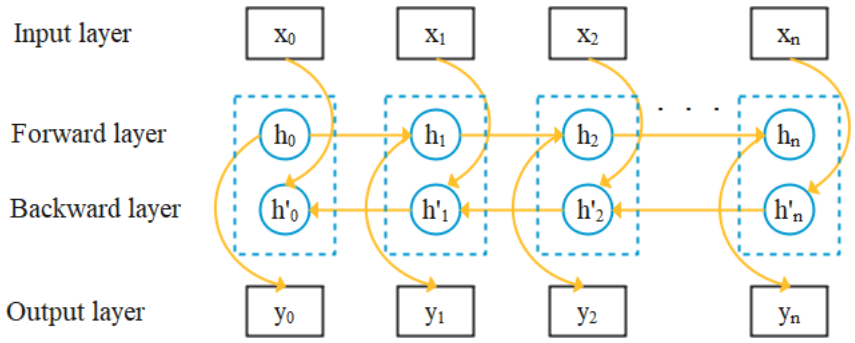


Figure 2. 4 Bidirectional LSTM Architecture (After: Yin et al., 2020).

- **Gated Recurrent Unit (GRU)**

GRU is a type of LSTM architecture that combines the input and forget gates and turns them into an update gate, resulting in fewer parameters and faster training. At any one epoch, there are two input features: the input vector x_t and the preceding output vector h_{t-1}. Each gate's output may be achieved by logical operation and nonlinear processing of input (Y. Wang et al., 2018). The mathematical formulation between the input, output, and various parameters is provided below.

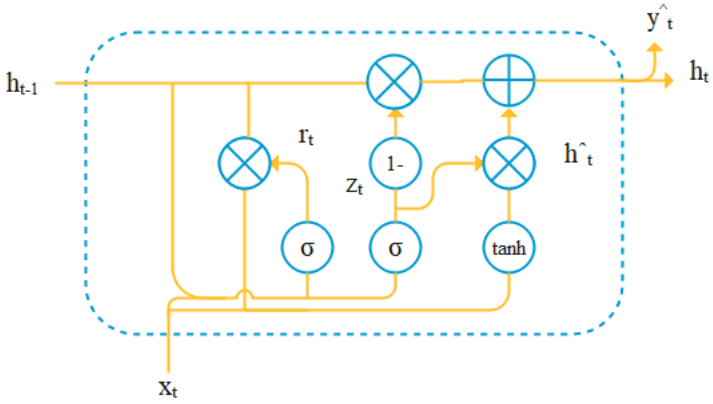


Figure 2. 5 The structure of the gated recurrent unit (GRU) network (Y. Wang et al., 2018)

$$Z_t = \sigma\left(W_z X_t + U z h_{t-1} + b_z\right) \quad (8)$$

$$r_t = \sigma\left(W_r X_t + U_r h_{t-1} + b_r\right) \quad (9)$$

$$\hat{h}_t = \tanh\left(W_h X_t + (r_t * h_{t-1})\, U_h + b_h\right) \quad (10)$$

$$h_t = (1 - Z_t) * h_{t-1} + Z_t * \hat{h}_t \quad (11)$$

where Z_t is the update gate vector, r_t is the reset gate vector, *W* and *U* are parameter matrices, σ is a sigmoid function, and *tanh* is a hyperbolic tangent.

- **Convolutional Neural Network (CNN)**

The convolutional neural network (CNN) is one of the best deep learning models for feature extraction and has network architectures with 1D, 2D, and 3D CNNs (T. Li et al., 2020). A convolution layer, a pooling layer, and a full-connection layer form the typical CNN structure (Y. Liu et al., 2021). According to Osama et al. (2016), 1D CNN is primarily utilized for processing sequence data, 2D CNN is typically employed for text and image identification, and 3D CNN is typically recognized for modelling medical image and video data identification (Shin et al., 2016). As a result, we applied a 1D CNN since the current study's goal is time series analysis. Figure 2.6 details the 1D CNN architecture in depth.

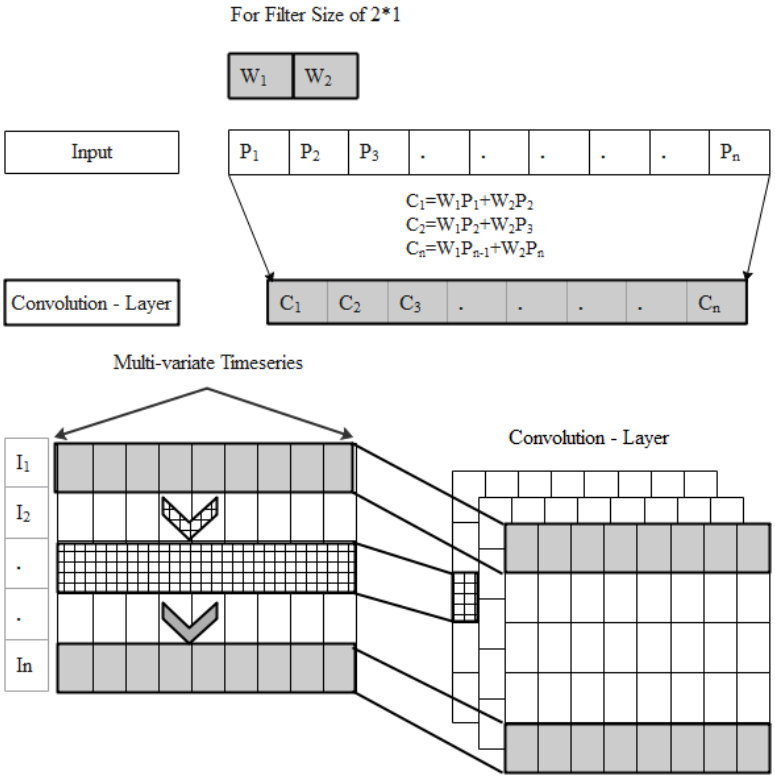


Figure 2. 6 The process of 1D CNN (After T. Li et al., 2020).

The convolution layer size depends on the quantity of input data dimensions, the size of the filter, and the convolution step length, as shown in Figure 2.6. The input series is convoluted to the convolution layer from top to bottom (shown by the arrows), and the gray or mesh colors represent various filters.

2.4.2. Model Development

A variety of methods must be followed to construct machine learning models. Data collection, data splitting, data handling and preprocessing, input selection, model selection (in terms of architecture, activation function, and learning algorithm), and network training and testing are the primary requirements for model construction (Oyebode & Stretch, 2019).

Maier, Dandy, and their colleagues examined 210 journal publications published between 1999 and 2007 that focused on the ANN development approach for predicting water resource variables in river systems and proposed several model-building alternatives. Greater emphasis, for example, should be placed on the ANN model construction processes, input variable selection and data division processes. Prior system knowledge techniques are often used in each of these domains, and other model development procedures that are not adequately examined may have the potential to dramatically damage model performance and hence must be replaced with state-of-the-art methodologies (Maier et al., 2010).

- **Determination of Model Inputs**

Selecting an adequate set of inputs is a vital phase in constructing the machine learning model. However, this job generally receives little attention, and most inputs are set using a priori system knowledge. This choice affects the insertion of either too few or too many model inputs; both are undesirable. Excluding one or more significant inputs may prevent the model from developing the best possible input-output connection given the available data (Maier et al., 2010).

Input redundancy, in which some of the chosen inputs give important information but are connected and hence supply duplicate information, is a typical cause of the inclusion of too many inputs. This may lead to numerous issues. First, duplicated inputs raise the possibility of overfitting (overtraining). Second, duplicate model inputs add more local minima to the error surface in weight space (Maier et al., 2010).

The five methods listed below represent the primary methodologies used for input determination in machine learning modelling for water resources (Bowden et al., 2005):

I. Methods based on a priori knowledge of the modelled System

II. Correlation analyses

III. Heuristic approaches

IV. Knowledge extraction

V. Composite methods

Methods based on previous knowledge of the simulated process have been extensively used in various water-related modelling investigations. Adopting the prior technique to choose the best input variables depends on the modeller's understanding of the simulated process. This means that the strategy's efficiency is contingent on expert knowledge, making it a subjective and case-dependent approach (Oyebode & Stretch, 2019).

When the connection to be simulated is poorly known, an analytical method is used, particularly when the system to be modelled is poorly understood. It helps to understand the links that exist between system processes. Linear and nonlinear correlation analysis are the two main correlation-based methodologies utilized for input variable selection (Bowden et al., 2005).

Cross-correlation, autocorrelation, and partial autocorrelation are all methods based on linear correlation analysis. Since linear correlation analysis cannot capture nonlinear interactions, nonlinear correlation analysis is occasionally utilized. Average and partial mutual information are two nonlinear correlation approaches utilized in hydrological studies (Oyebode & Stretch, 2019).

In the literature, heuristic approaches are also often utilized. Various ANN models are trained using different subsets of inputs in this approach. A sequential selection of inputs is another typical heuristic strategy. This approach is often used to avoid enumerating all possible input combinations. Individual networks are trained for each input variable in the stepwise selection strategy (Bowden et al., 2005).

As the name implies, the knowledge extraction approach involves extracting information from a trained ML model. Sensitivity analysis is the most frequent way of obtaining information from a trained ML model. In this procedure, graphical representations of the responsiveness to inputs are scrutinized, and relevant explanatory inputs are chosen based on human judgment. The challenge with this strategy is determining an acceptable value by which to disrupt the input and determine the appropriate cut-off point for input relevance (Bowden et al., 2005).

The employment of any of the methodologies mentioned above in combination is referred to as composite methods. Other methods for selecting model inputs not listed in the preceding classes include principal component analysis (PCA), which is frequently used when there are many inputs. Evolutionary optimization methods such as the genetic algorithm (GA) and multiobjective GA are also applied to optimize the model inputs and network architecture simultaneously (Oyebode & Stretch, 2019).

- **Data Division**

The given data are separated into training, validation, and testing subsets as a common approach in the ML model generation process. The training data set estimates the unknown connection between neurons or weights, the validation data set determines when to terminate training to prevent overfitting and/or which network topology is ideal, and the testing data set evaluates the trained model's generalization capacity (Maier et al., 2010).

Like other empirical models, ML models work best when they are not used to extrapolate outside the training data range. As a result, the training, testing, and validation sets should all have the same statistical features to construct the best feasible model given the available data. Lebaronand and Weigend investigated the variability of ANN performance owing to data variability. In their work, the authors randomly created 2523 instances of training choices, test, and validation data sets for a specific data set and trained an ANN model for each instance. A histogram of the model performance distribution was employed to investigate the variability that data splitting might introduce into the ANN model generation process. The authors discovered that model performance variability owing to data splitting is more significant than model structural variability (Wu et al., 2012). According to Maier, Dandy, and colleagues, data splitting is broadly categorized into supervised and unsupervised strategies (Maier et al., 2010).

Unsupervised methods do not explicitly account for the statistical qualities of data subsets, and only stratified unsupervised approaches seek to guarantee that the statistical properties of the subsets are identical. The data are randomly split into their appropriate subgroups in the random unsupervised technique. In the physics-based method, the data are classified into several groups based on understanding the underlying physical processes or domain knowledge. Data may be randomly partitioned in the ad hoc method (Maier et al., 2010).

The statistical parameters of the different subsets are ensured via supervised data division techniques. This can be accomplished through a trial-and-error approach in which manual adjustments to the composition of the various subsets are made until an arbitrarily satisfactory level of agreement between the statistical properties of the various data subsets is achieved or through a formal optimization approach in which a measure of the difference between the statistical properties of the data subsets is minimized (Maier et al., 2010).

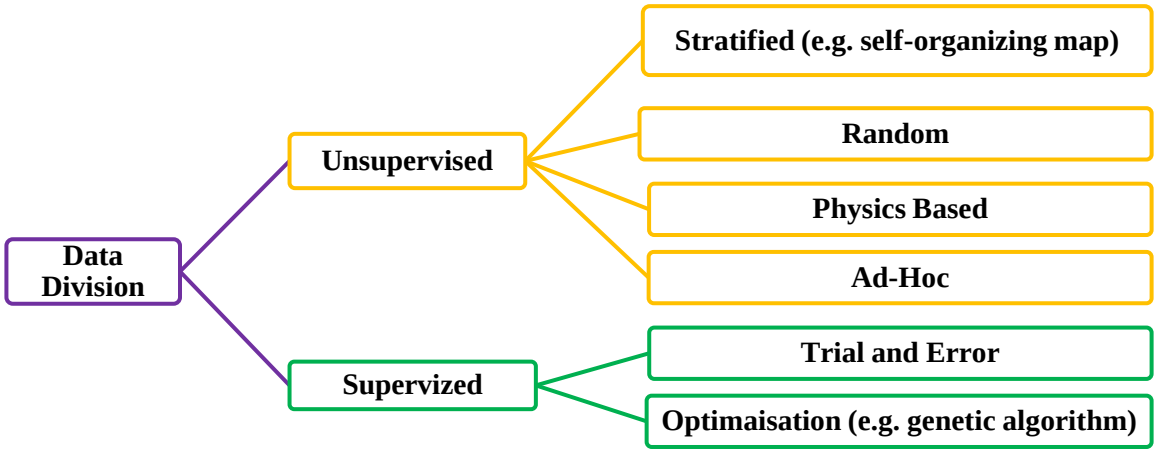


Figure 2. 7 Classification approaches of data division on ANN (Maier et al., 2010).

- Model Architecture

Feedforward and recurrent networks are the two types of traditional ANN designs. In feed-forward networks, information is only propagated in one way, from the input layer to the output layer. The most prevalent feed-forward model architecture is multilayer perceptions (MLPs). Generalized regression neural networks (GRNNs), radial basis function (RBF) networks, neuro-fuzzy systems (NFS), and support vector machines (SVMs) are some other feed-forward neural network (FFNN) designs in use (Maier et al., 2010).

Currently, recurrent neural networks are gaining popularity among ML researchers. Through a feedback loop, architectural information is sent forward from the input layer to the output layer and backward from the output layer to the input and/or hidden layer in this paradigm. The information received from the output layers may be utilized to adjust the model's weights or structure, allowing it to represent the intricacies of highly dynamic systems. Combining ANNs with other models (e.g., process-based models) or methods (e.g., regression or fuzzy logic) to produce a hybrid model that can harness the benefits of any current system information and diverse modelling approaches might be advantageous at times. This has the potential to improve model performance, particularly in exceedingly complex environmental and hydrological systems (Wu et al., 2014).

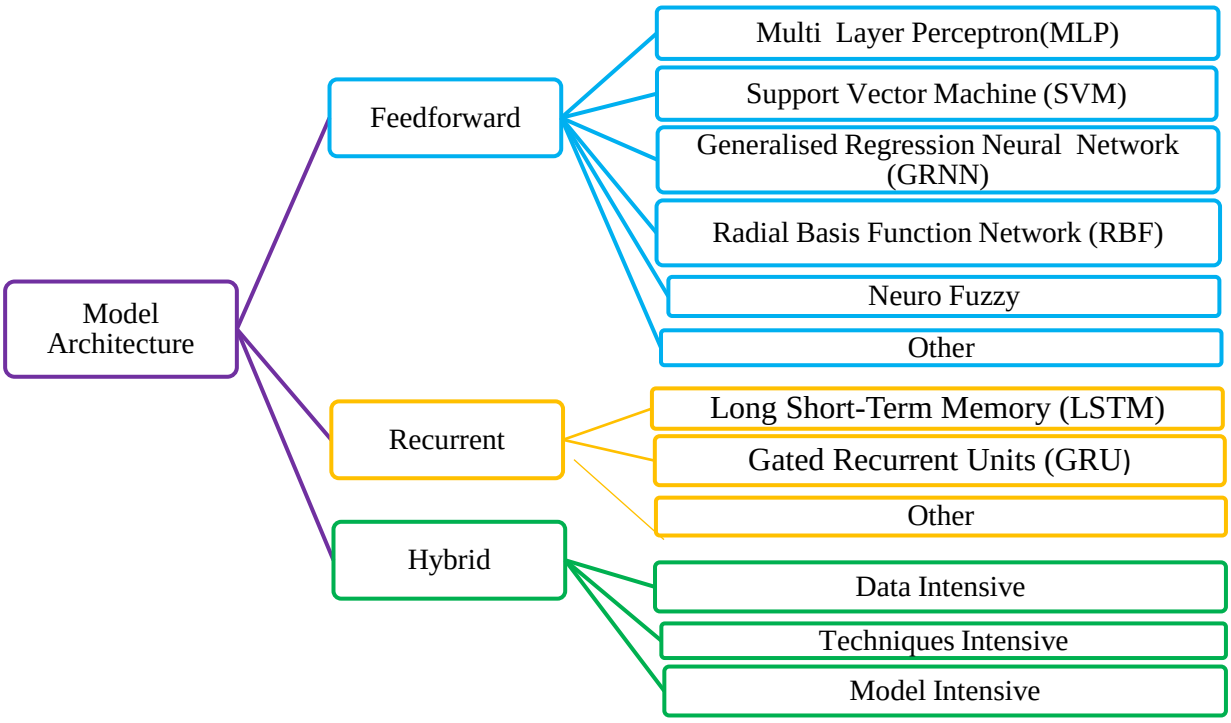


Figure 2. 8 Classification of model architectures (Maier et al., 2010).

- **Model Structure**

Setting the number of layers, the number of nodes in each layer, and how they are linked is a typical ANN model construction. An optimum network structure generally reduces forecasting/prediction errors while optimizing model performance (Wu et al., 2014). The universal optimum ANN structure classification approach is global, stepwise, and ad hoc.

- **Global**

The structure of an ML model in terms of hidden layers and/or hidden neurons is determined utilizing global approaches based on competitive evolution observed in nature, such as genetic algorithms, particle swarm optimization, and simulated annealing. This method allows you to improve network parameters (such as network weights) and structure simultaneously (e.g., the number of hidden layer nodes). Global approaches, when utilized correctly, are likely to provide the optimal ML structure and/or parameters; nevertheless, they are computationally costly (Oyebode & Stretch, 2019).

- **Step Wise**

This trial-and-error approach may be employed by the initial assumption of a basic ML structure, which is updated with each trial to obtain a structure that is neither too complicated nor simple. This technique is further classified into two types: those that use pruning algorithms and those that use constructive methods. A pruning method begins with a suitably complicated ML structure capable of capturing the intricacies of the physical system under consideration. The connection weights and related neurons are then eliminated one at a time, depending on a grading system of their magnitude, until model performance deteriorates considerably. On the other hand, a constructive algorithm begins with the simplest ML structure and gradually becomes more complicated by adding hidden neurons/layers one at a time and calibrating the resultant model. This method is continued until no substantial increase in model performance is seen. Pruning and constructive methods may also be computationally demanding since ML models with many structures must frequently be trained and manually reviewed before determining the ideal structure (Maier et al., 2010).

- **Ad-Hoc**

This way of establishing the structure of ANN models was by far the most popular since it is a means to choose a network structure based on experience and/or intuition (Wu et al., 2014).

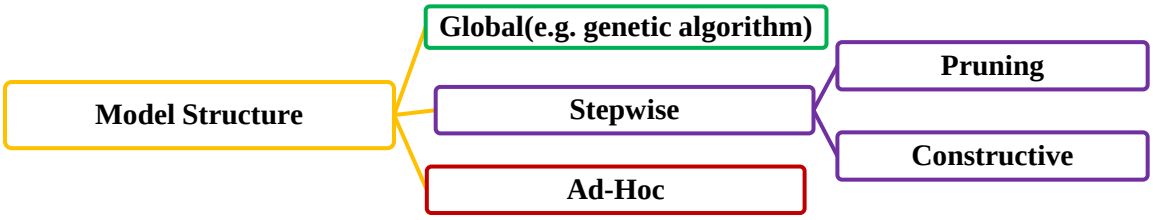


Figure 2. 9 Classification of methods for optimizing model structure (Maier et al., 2010).

- **Model Calibration**

Calibration determines a set of model parameter values or weights that allow the model to map the connection between the inputs and outputs of a given data set. It often necessitates optimizing methods (Wu et al., 2014). There are two types of broad training methods: deterministic and stochastic.

- **Deterministic**

This method employs a set of model parameter values to minimize the difference between predicted and observed outputs. Local optimization techniques, such as the traditional back-propagation approach created by Rumelhart and his colleagues in 1986, and global optimization algorithms, such as genetic algorithms, are two types of optimization algorithms (Wu et al., 2014).

Local optimization methods are often computationally efficient; nevertheless, since they operate on gradient information, they are prone to becoming stuck in local optima when the error surface is rough. Global optimization techniques may enhance the likelihood of discovering global optima in a rough error surface in a single run; nevertheless, they are more computationally costly and often need the determination of extra parameters to improve their performance (Hamm et al., 2007). Despite their popularity, deterministic calibration approaches disregard uncertainty in model inputs and outputs, model structure, and model parameters. Stochastic calibration techniques may be used to account for parameter uncertainty during the calibration process (Maier et al., 2010).

- **Stochastic**

Instead of looking for single parameter values, stochastic techniques such as Bayesian methods have been employed to produce the distribution of model parameters. The main advantage of these techniques is that prediction intervals for model predictions may be derived automatically (Oyebode & Stretch, 2019).

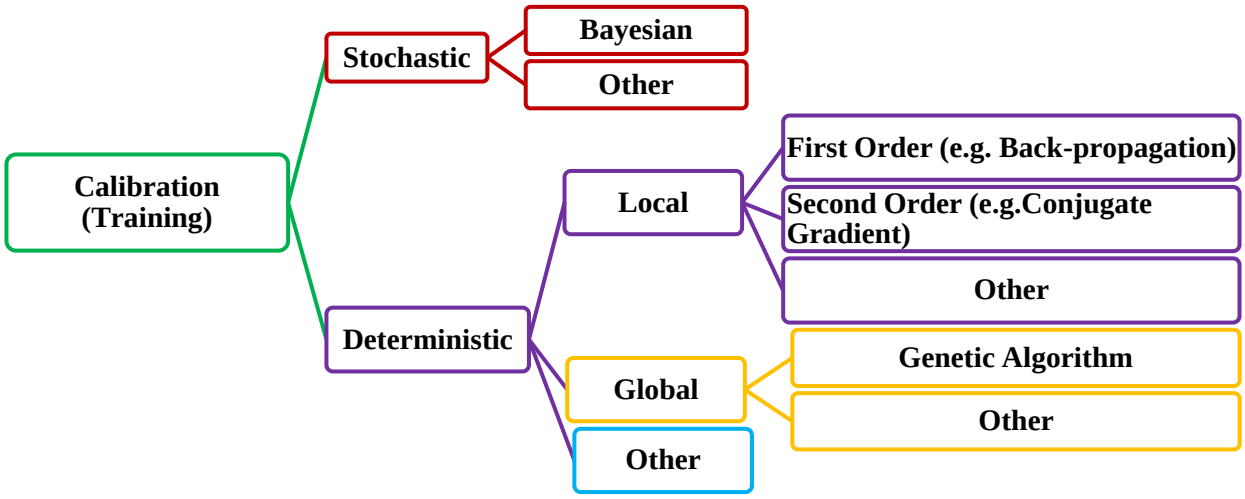


Figure 2. 10 Classification of calibration (training) methods (Maier et al., 2010).

- **Model Validation**

Model validation ensures that a trained ML model has no known or identifiable flaws that might prohibit the model from performing optimally. On the other hand, the ubiquitous, complicated error surfaces of environmental and water systems in ML models make this a challenging endeavor since there are often several permutations of ML structures and/or weights that lead to a comparable model. As a result, robust model validation that goes beyond model prediction performance is essential to guarantee that the produced ML models are physically credible.

Model validation methodologies may be divided into three categories based on three components of model assessment (Gass, 1983):

- **Replicative Validity**

Suppose the model matches the data obtained from the existing system and utilized in the previous phases of the ML model creation process (i.e., data processing, input selection, data splitting, ML architecture, or structure selection and training). In that case, it is regarded as replicative valid. As with conventional statistical models, its validity may be verified using typical statistical procedures such as means and variances, analysis of variance, goodness-of-fit testing, regression and correlation analysis, and confidence interval generation (Wu et al., 2014).

- **Predictive Validity**

The input-output connection in the data that will be gathered from the existing system in the future is used to assess predictive validity. In other words, predictive validity assesses the model's capacity to generalize across the range of data used for calibration. This may be accomplished by using an independent validation data set. ML models' prediction ability is often evaluated using quantitative error metrics acquired from the validation data set (Wu et al., 2014). Predictive validity metrics are usually classified into five categories: squared errors, absolute errors, relative errors, product differences, and information criteria (Maier et al., 2010).

The squares of the variations between the actual and predicted output values are used to calculate squared errors. This category includes measurements such as the sum of squared errors (SSE), root mean square error (RMSE), and Nash Sutcliffe efficiency (NSE). Squared error metrics are distinguished by the fact that large-scale errors dominate them. Absolute errors, which are based on the absolute differences between actual and modelled outputs and include measurements such as the total sum of absolute deviations (TSAD) and the mean sum of absolute deviations (MSAD), may also be utilized. While absolute errors give information on the amount of the error, they do not provide information on the model's overall under- or overprediction performance. This issue may be solved by calculating the total or mean sum of the differences without using absolute values, yielding total bias (Tbias) and mean bias (Mbias) statistics.

To make it easier to assess the performance of models with varied magnitudes of outputs, relative error measures such as the average absolute relative error (AARE), the normalized root mean square error (NRMSE), and the normalized mean bias error (NMBE) may be employed (Maier et al., 2010).

Finally, utilizing product difference moment error statistics, such as the Pearson correlation coefficient, a measure of the empirical error between actual and modelled outputs may be produced. Model complexity and model error are considered by information criteria such as the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC). As a consequence, they can produce more efficient models. Aside from these statistics, numerous more may be used to assess model performance. For example, threshold statistics (TS) may provide the distribution of the number of data points predicted by an ML model with certain degrees of absolute relative error (ARE). Furthermore, the accuracy of forecasting certain time series features, such as errors in simulating peak flow, peak timing, and total volume, may be used to evaluate the performance of ML models. According to the reviewed literature, squared error metrics were the most often used measures based on absolute and relative errors, and correlation was also regularly utilized (Maier et al., 2010).

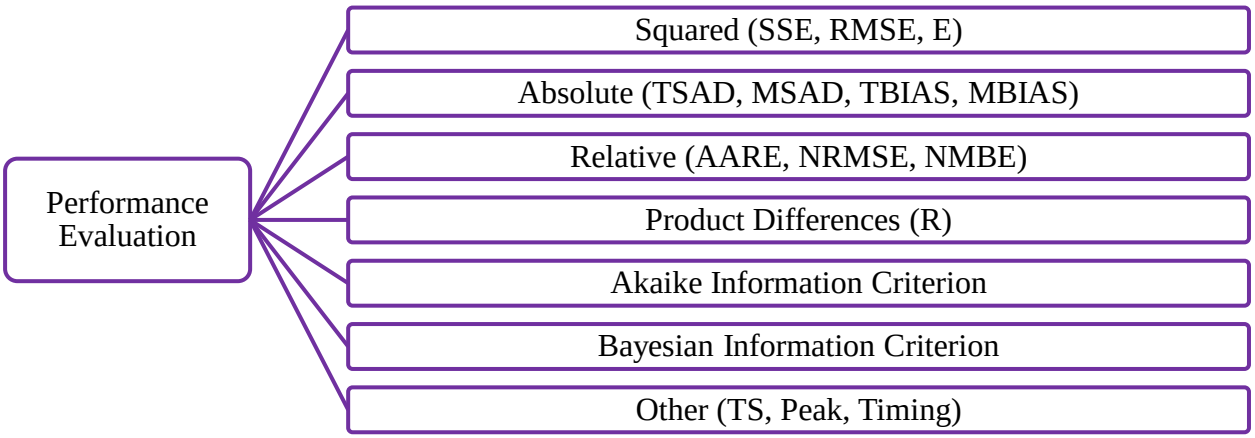


Figure 2. 11 Classification of performance evaluation (Maier et al., 2010).

- **Structural Validity**

Structural validity is conducted to establish model credibility when contrasted to a priori knowledge of the system behavior that the model is supposed to portray. This is only possible if a model reproduces observable real-world behavior and accurately depicts how the real-world system is thought to work to create that behavior. In this manner, structural validity analysis may be used to examine the uncertainty in ML outputs. These uncertainties are often connected to the limited information in available data or even flaws in sampling processes and cannot be addressed only via predictive validity evaluation. Structural validity is one of the most often neglected validation criteria for ML models. One of the structural validity measurement methodologies is sensitivity analysis (Wu et al., 2014).

CHAPTER 3:

STUDY AREA DESCRIPTION

This study was conducted in four study areas. The selection of these study areas is conducted mainly by considering the availability of data and their location in different climatic zones. First, an appropriate research region with a relatively good data set having optimum ground-observed meteorological and streamflow data is selected to allow the subsequent analysis of streamflow estimation using RS data. The various strategies are evaluated with relatively good and poor availability of data sets.

Streamflow data sets are the main inputs in our research. Given this condition, the Upper Tiber River Basin (UTRB-Italy) was selected as the first case study area and a relatively data-rich watershed. Ethiopia's hydrological network for streams and lakes consists of 560 gauging stations, 454 functioning nationwide. Data availability per square area within Ethiopia's 12 river basins is generated for selecting the second, third, and fourth case study areas. Next, three river basins were chosen as case study regions based on the number of operable gauging stations in each basin (Table 3.1), cross-checking rough data availability and basin economic relevance. The second case study region was chosen in the Abbay River basin, which contains relatively large gauged stations next to the Rift Valley. The third and fourth case study areas will be the Awash and Baro Akobo river basins. Table 3.1 shows the basin-wide description of these stations (MoWR,2011).

Table 3. 1 Number of hydrological stations in Ethiopian river basins (MoWR,2011).

No.	Basin	Established	Operational	Area (km^2)	Average area covered by one operational station (km^2)
1	Awash	97	72	112,696	1565.222
2	Abbay	160	131	204,100	1558.015
3	Wabi-Shebelle	50	30	205,697	6856.567
4	Genale Dawa	38	36	171,042	4751.167
5	Rift Valley	70	54	54,900	1016.667
6	Omo-Gibe	57	46	75,912	1650.261
7	Baro Akobo	31	32	75,912	2372.25
8	Tekeze	40	39	90,001	2307.718
9	Dankile	12	11	62,882	5716.545
10	Ogaden	-	-	77,121	-
11	Aysha	-	-	2223	-
12	Mereb	5	3	5,700	1900
	Total	**560**	**454**		

3.1. Tiber River Basin

The first case study is conducted in the Tiber River basin, Italy's second-largest catchment area (Annis & Nardi, 2019). Geographically, the basin is situated between 40.5°N and 43°N latitudes and 10.5°E to 13°E longitudes, spanning an area of approximately 17,500 km^2 and accounting for approximately 5% of the Italian territory. The basin crosses six administrative areas and twelve provinces. Almost 90% of the basin is located in the regions of Umbria and Lazio, with the remaining 10% in Emilia-Romagna, Tuscany, Marche, and Abruzzo. Figure 3.1 depicts the Tiber River Basin area. The basin has major towns such as Perugia, Terni, and Rieti, as well as Italy's ancient city of Rome.

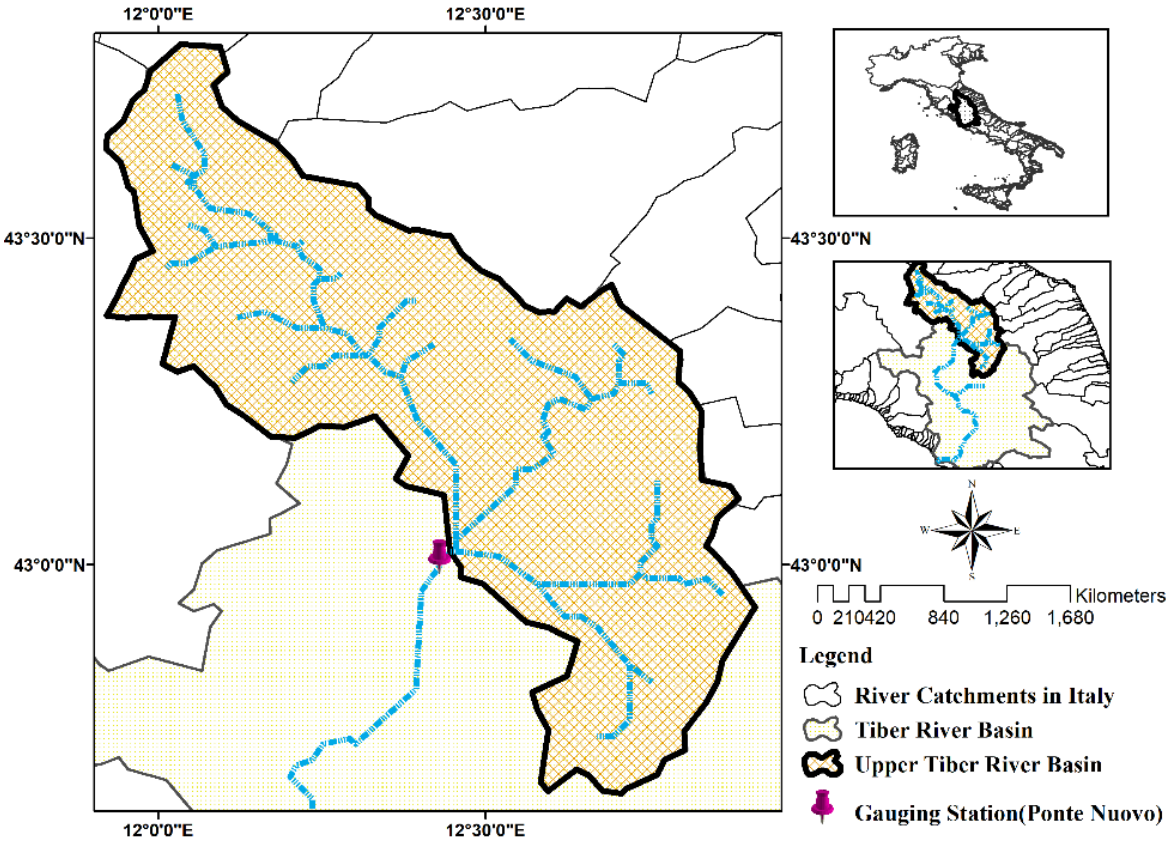


Figure 3. 1 The Tiber River basin and river catchments in Italy. The black boundary shows the first case study area.

3.2. Abay River Basin

The second case study area was chosen due to its economic relevance and a catchment with a relatively strong data set. The selected watershed should also be large enough to work with RS data and have little streamflow regulation. As a result, the Gummera subcatchment in the Abay basin, Lake Tana catchment, is chosen as a study area. The Abay basin is the largest and most important watershed in Ethiopia. It directly impacts the Ethiopian economy and downstream countries, as it provides 84% of the Nile River flow during peak flows (Awulachew et al., 2009).

The Blue Nile River originates at Lake Tana (1,786 m.a.s.l.), which has a drainage area of over 15,000 km^2, of which the lake occupies approximately 3,000 km^2. The lake receives runoff from more than 40 rivers and is situated in the northwestern highlands between 12°00'N and 37°15'E. Gilgel Abay from the south, Ribb and Gumara from the east, and Megech River from the north are the principal rivers supplying the lake. Only minor river systems drain into the lake from its western side.

The Ethiopian government has designated the Lake Tana Basin as a region for irrigation and hydropower development, which are essential for food security and Ethiopia's economic development. The Lake Tana Basin has numerous irrigation and hydroelectric projects planned for the near future (Tegegne et al., 2017).

Table 3. 2 Rivers discharging into Lake Tana (Alemayehu et al., 2010)

	Gilgel Abbay	Ribb	Gumara	Megech	Other	Total
Catchment area (km^2)	1664	1592	1394	462	7050	12165
Mean annual flow (Mm^3)	1810	430	938	189	1619	4986

Criteria for the selection of subcatchments

- Least or no streamflow regulation
- Appropriate catchment size for remote sensing study (>1000 km^2)
- The availability of key ground-measured data for testing. Main streamflow time series.

At a height of approximately 3250 m.a.s.l., the Guna Mountains to the south and east of Debre Tabor are the source of the Gumara River. The river travels 132.5 kilometres before arriving at Lake Tana, and it covers a total catchment area of approximately 1592 km^2. In the basin, numerous minor intermittent and perennial rivers and springs flow into the major Gumara River.

The catchment is made up of a rough and undulating landscape, with elevations ranging from 1788 m to 3750 m.a.s.l. The region contains steep slopes (sometimes > 25%) in the high mountainous region to the east and gentle slopes (3%) near Lake Tana, which is situated at an elevation of 1788 m.a.s.l (Worqlul et al., 2018).

3.3. Baro-Akobo River Basin

In the third case study area, the performance of the suggested methodology for streamflow estimation employing RS data in the deep learning approach will be evaluated. The third case study area was chosen because it is a basin of significant economic and human importance and has fewer ground monitoring stations. Even though this catchment contains some ground-measured data, it was classified as a catchment with limited ground-measured data.

The Baro-Akobo River basin is part of the southwestern Ethiopian Plateau with a peak elevation of 3240 masl. Towards the west, the rolling and hilly morphology abruptly changes into a dominant long chain of sharp escarpments and lowland plain with the lowest elevation of 395 masl. The basin is the second smallest, with an area of 75,912 km^2 (6.9% of the country) among Ethiopia's eight major river basins. However, it contains the second highest runoff of 23.6 Bm^3, 27% of the country's potential irrigable land, and 8.9% of the hydropower potential (Awulachew et al., 2007).

The Baro Akobo wetland is one of the country's wettest areas. The wetlands are fed by a combination of local precipitation, the torrents originating from the western Ethiopian Plateau, and spillover from the Baro, Akobo, Gilo, and Alwero rivers. While other rivers spill during periods of high flows into the adjoining wetlands, the Baro River spills north across the Ethiopia-Sudan border towards the Machar Marshes. Table 3.3 shows the Baro-Akobo River Basin surface water resources compared to other Ethiopian river basins (Wabi Shebele River Basin Integrated Development Master Plan Study Project – Phase Iii – Master Plan Main Report, 2005).

The Sore and Mashaa River catchments are selected in this river basin. The Sore subcatchment is located in the highlands of Ethiopia's Baro-Akobo River basin, covering an area of 1711 km^2. The Sore watershed is situated between 8°21′ and 7°49′ latitudes and 35°31′ and 35°57′ longitudes, with the highest elevation ranges found in the east and south. Its elevation ranges from 2658 to 1578 m.a.s.l. The elevation drops to the north and northwest. The range of annual precipitation is 1804 to 2020 mm. With monthly maximum temperatures ranging from 24°C to 28°C and monthly minimum temperatures ranging from 12°C to 14°C, aridity rises from east to west throughout the Baro Akobo basin and the sore watershed. With a surface area of 1668 km^2, the Mashaa subcatchment borders the Sore catchment. With elevations ranging from 2783 to 1669 m.a.s.l., the Mashaa watershed is located between latitudes 7°30′ and 8°02′ and longitudes 35°26′

and 35°54′. The range of annual precipitation is 1800 to 2387 mm. The temperature ranges from 10 to 13 °C for the monthly low and 20 to 26 °C for the monthly high.

Table 3. 3 Estimated annual water yields of selected Ethiopian river basins (*Wabi Shebele River Basin Integrated Development Master Plan Study Project – Phase III – Master Plan Main Report*, 2005)

Basin	Awash	Abbay	Baro Akobo
Annual Rainfall (mm)	557	1 224	1 419
Annual Runoff (Bm^3)	4.60	54.4	23.32
Basin Area (km^2)	114, 919	199,812	75,912
Annual Runoff Coefficient	0.07	0.21	0.11
Annual Specific Yield ($l/s/km^2$)	1.26	8.63	9.74

3.4. Awash River Basin

The Awash River Basin is one of the major river basins of Ethiopia, with a total land area of 114,919 km^2. The river originates in Ethiopia's high plateau at Ginchi Town, west of Addis Abeba, runs down the rift valley into the Afar triangle, and finally drains into the salty Lake Abbe, an endorheic basin on the border with Djibouti. The main course is over 1,200 km long overall. The Awash Basin is divided into the upper valley (lands above 1500 masl), middle (1000 to 1500 masl), lower valley (500 to 1000 masl), and eastern catchment (closed subbasin between 1000 and 2500 masl). The upper, middle, and lower valleys are a part of the Great Rift Valley systems. The deltaic alluvial plains in the Tendaho, Assaita, Dit Behri, and terminal lake regions make up the lower Awash Valley (Taddese et al., 2009).

The annual rainfall in the basin ranges from approximately 1600 mm near the origin northeast of Addis Abeba to 160 mm closer to the basin's northern edge in the Afar region, with a mean of 850 mm. The warmest months are May and June, with the basin's temperature ranging from 19 to 23°C.

The mean annual evaporation varies from 1800 mm in the upper valley to 2348 mm in the lowland area. Additionally, the basin is home to the majority of the nation's large-scale mechanized public and private irrigated fields, which significantly contribute to the gross domestic product (GDP) (Abera et al., 2020).

One of the Awash River tributaries, the Borkena River catchment, is selected as a case study area. It is situated in the highlands of Wollo and receives substantial runoff from the Upper Awash Basin. Dessie, Kombolcha, and Kemissie are the largest cities in the Amhara regional state. These cities are a portion of this catchment and are home to several industrial and massive municipal irrigation projects. The major rainy season, which occurs from July to September and is locally known as “Kiremt”, delivers approximately 84% of the watershed's total annual precipitation. The majority of the year is primarily dry (Bega: October to January), except for occasional wet seasons from March to May (locally known as Belg) (Abera et al., 2020).

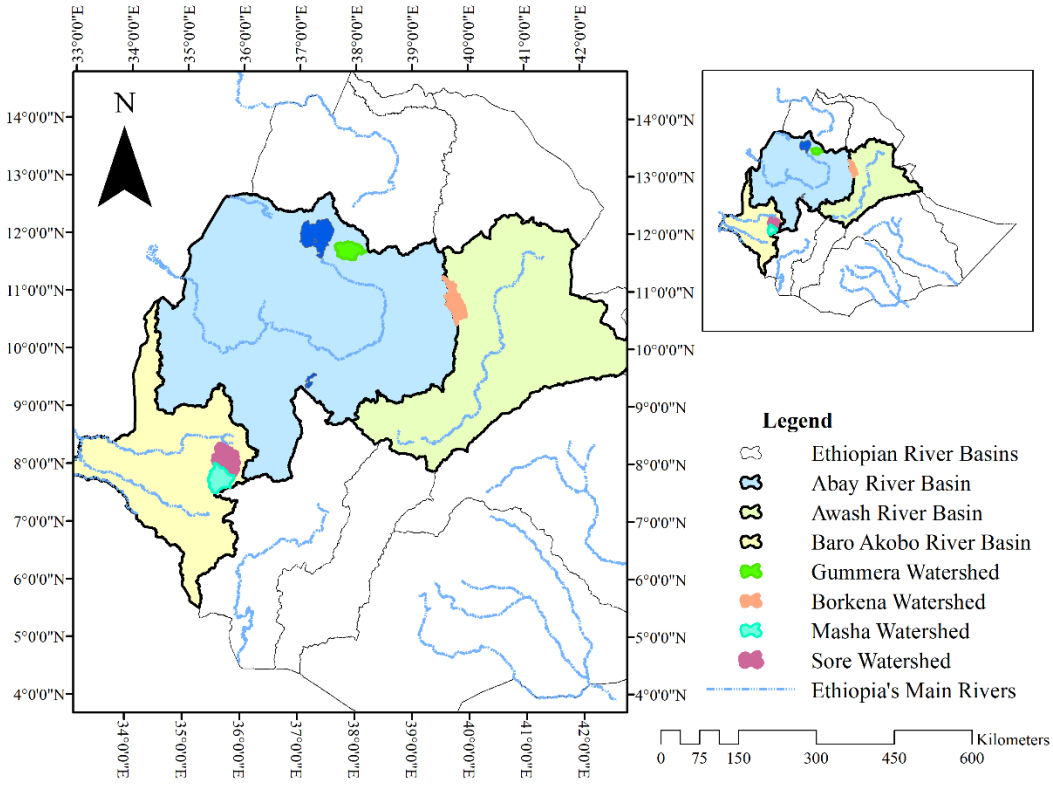


Figure 3. 2 The four case study areas and river catchments in Ethiopian river basins (Gumera, Borkena, Mashaa and Sore subwatersheds).

CHAPTER 4:

SHORT-TERM DAILY UNIVARIATE STREAMFLOW FORECASTING USING DEEP LEARNING MODELS

Abstract[1]

One of the most important fields of study in hydrology is hydrological forecasting. Innovative forecasting technologies will be needed to overhaul water resource management systems, flood early warning systems, and agricultural and hydropower management plans. As a result, in this research, we compared stacked long short-term memory (S-LSTM), bidirectional long short-term memory (Bi-LSTM), and gated recurrent unit (GRU) networks for one-step daily streamflow forecasting to the standard multilayer perceptron (MLP) network. The study employed daily time series data acquired from streamflow stations in Borkena (in the Awash River basin) and Gummera (in the Abay River basin). All data sets were subjected to strict quality assurance procedures, and null values were filled using linear interpolation. A partial autocorrelation was also used to determine the suitable time lag for producing the input time series. The data are then divided into training and testing data sets at a ratio of 80:20. The performance of the developed models was evaluated using root mean square error (RMSE), mean absolute error (MAE), mean absolute percentage error (MAPE), and coefficient of determination (R^2). Finally, the results are organized into three categories: model variability, lag time variability, and time series characteristic themes. In conclusion, time series features (climatic variability) had a more significant impact than input-lagged time steps and deep learning model architectural changes on streamflow forecasting performance. Borkena's river catchment forecasting result is thus more accurate than Gummera's catchment forecasting result, with RMSE, MAE, MAPE, and R^2 values ranging between (0.81 to 1.53, 0.29 to 0.96, 0.16 to 1.72, 0.96 to 0.99) and (17.43 to 17.99, 7.76 to 10.54, 0.16 to 1.03, 0.89 to 0.90) for both catchments, respectively. MLP and GRU outperformed S-LSTM and Bi-LSTM on a roughly equal basis, even though performance relies on lag time differences.

[1] *This chapter is based on the publication*- https://www.hindawi.com/journals/amete/2022/1860460/

4.1. Introduction

Streamflow forecasting research is still a vital area of study in hydrology. For water resource planning, management, and disaster mitigation response agencies, accurate and reliable streamflow forecasting is critical (Sharma & Machiwal, 2021). There are two main types of streamflow forecasting. First, short-term or real-time forecasting includes daily and hourly timestamps, which are extensively used in flood management systems. Second, long-term forecasting includes weekly, monthly, and yearly streamflow forecasting, which is critical for reservoir operation, irrigation system management, and hydropower production (Yaseen et al., 2015).

Streamflow forecasting methods are further classified into conceptual, physical-based, and data-driven models (Yaseen, Mohtar, et al., 2019). In nature, conceptual models are lumped and often depend on empirical correlations between numerous hydrological factors. Because of its dependence on observable data, the model is seldom applicable to data-scarce catchments. Physical models of hydrological processes may also be represented using mass, momentum, and energy conservation equations. These models can account for geographical variability, but since they are distributed in nature, a substantial quantity of data on land use, slope, soil, and climate is needed (Jaiswal et al., 2020). Finally, without physical catchment knowledge and minimal data needs, data-driven models produce a nonlinear input-output connection. As a result, with the progress of computing capabilities and data set availability, the popularity of data-driven models is growing (Ghaith et al., 2020).

Zhang et al. (2018) categorized data-driven methods into conventional and AI-based models. Conventional data-driven models such as ARMAX (autoregressive moving average with exogenous term), MLR (multiple linear regressive), and ARIMA (autoregressive integrated moving average) are simple to apply (Z. Zhang et al., 2018). However, the nonlinearity of hydrological processes was left out of the equations. AI-based data-driven models, on the other hand, can identify nonlinearity and perform better in streamflow predictions. As a consequence, machine learning models are now a prominent field of study.

AI-based data-driven streamflow forecasting models can be univariate when the model's input and output are created using a single time series variable. Univariate forecasting models are simple to train with sparse data and enable clear inference for assessing prediction performance.

Due to the complexity of agro-meteorological data, forecasting the variables separately is easier and more economical (Suradhaniwar et al., 2021). Multivariate models, on the other hand, are created using different variables such as precipitation, temperature, evaporation, and other factors as input and a streamflow variable as output (Z. Zhang et al., 2018). Thus, univariate modelling is more practical in data-scarce catchments with limited data and has attracted much attention in recent years (Danandeh Mehr & Safari, 2021; Essien & Giannetti, 2019; Yaseen, Mohtar, et al., 2019; Z. Zhang et al., 2018).

For time series forecasting, a wide range of classical and deep learning models are available in the literature, including artificial neural networks (ANNs), support vector machines (SVMs), fuzzy logic (FL), recurrent neural networks (RNNs), long short-term memory (LSTM), gated recurrent units (GRUs), adaptive neuro-fuzzy inference systems (ANFISs), and genetic programming (GP). However, due to the nonlinearity of streamflow time series, the forecasting efficacy of these models is often debated (Yaseen, Mohtar, et al., 2019; Yaseen, Sulaiman, et al., 2019). Suradhaniwar et al. (2021) examined the performance of machine learning (ML) and deep learning (DL) based time series forecasting algorithms under one-step and multistep-ahead prediction scenarios. They also used feed-forward validation to assess recursive one-step forward predictions. Finally, with defined prediction horizons, seasonal autoregressive integrated moving average (SARIMA) and support vector regression (SVR) models dominate their DL-based counterparts: neural network (NN), recurrent neural network (RNN), and long short-term memory (LSTM).

The most often used classical machine learning architecture in hydrology is ANN (MLP) (Oyebode & Stretch, 2019). Cui et al. (2020) demonstrated that the new generation of ANN or Emotional Neural Network (ENN) models outperformed the Multivariate Adaptive Regression Splines (MARS), Minimax Probability Machine Regression (MPMR), and Relevance Vector Machine (RVM) models when used for hourly river flow forecasting. Yaseen et al. (2015) also conducted a detailed review of literature from high-impact journals from 2000 to 2015 on the state-of-the-art application of artificial neural networks (ANNs), support vector machines (SVMs), fuzzy logic (FL), evolutionary computation (EC), and wavelet-artificial intelligence (W-AI) for streamflow forecasting. According to the same research, time series preprocessing, input variable selections, and time scale selections are essential characteristics for high-performing forecasting models.

RNN is a common deep learning architecture suited for time series analysis. However, it has certain downsides, such as vanishing and exploding gradients. Hochreiter & Schmidhuber (1997) enhanced the RNN version with LSTM, a well-known time series model for long-term step forecasting. Various disciplines of research have recently experimented with these models (Y. Bai et al., 2021; Couta et al., 2019; Din et al., 2019; Jain et al., 2020, 2020; Rahimzad et al., 2021a; Sahoo et al., 2019; L. Yan et al., 2019). Furthermore, Cho, Van Merrienboer, et al. (2014) were the first to present GRU as a simplified version of the LSTM model. GRU combines short-term and long-term memory cells into a single gate with high performance and a quick running time (Hussain et al., 2021). Lara-Benítez et al. (2021) assessed the accuracy and efficiency of seven popular deep learning architectures: multilayer perceptron (MLP), Elman recurrent neural network (ERNN), long short-term memory (LSTM), gated recurrent unit (GRU), echo state network (ESN), convolutional neural network (CNN), and temporal convolutional network (TCN). Furthermore, they built over 38000 different models to find the best architectural configuration and training hyperparameters, with LSTM attaining the lowest weighted absolute percentage error, followed by GRU.

Although we discovered several excellent forecasting models using GRU and LSTM in various domains, particularly hydrology, the accuracy of these models must be fine-tuned using different data processing approaches and data input changes (Althelaya et al., 2018; Kumar et al., 2018; Shahid et al., 2020, 2020; Yamak et al., 2019). For univariate time series forecasting, we may reorganize the previous time step data series as a predictor input variable and the present or future step as the output variable. However, deciding on the number of input time steps is challenging without previous information. To produce reliable models, it is necessary to investigate the influence of lagged time selection in streamflow forecasting. Lagged variables in univariate streamflow forecasting are major characteristics that affect the model's performance favorably or adversely and serve as predictor variables for temporal dependence information (Papacharalampous et al., 2018b). Tyralis & Papacharalampous (2017) discovered that employing a small number of recent predictor variables results in greater accuracy for time series forecasting using the random forest (RF) method. Applying this discovery to other popular deep learning models and diverse environmental circumstances is critical.

Papacharalampous et al. (2018) used vast time series data to assess 20 one-step forward univariate time series forecasting algorithms. They concluded by stating the impact of time series length on the performance of different forecasting techniques and the most and least accurate methodologies for one-step forward forecasting. Furthermore, the same research found that the machine learning model's optimization strongly relies on hyperparameter optimization and delayed variable selection. Torres et al. (2021) highlighted research needs in various domains by assessing the most successful deep learning architectures for accurately predicting time series and emphasizing MLP, RNN, GRU, and Bi-LSTM architectures.

In this research, we evaluated several types of LSTM architectures, S-LSTM, Bi-LSTM, and GRU, with the conventional MLP network to forecast a single-step streamflow amount using daily streamflow data records. To the best of our knowledge, the LSTM has been mostly investigated for monthly multivariate time series forecasting rather than daily univariate streamflow forecasting. Although machine learning may effectively model hydrological forecasting, researchers should carefully investigate the effects of appropriate input variables and model parameter selection on model accuracy (Niu & Feng, 2021).

4.2. Data and Methods

For this research objective, two river basin subcatchments in Ethiopia are chosen: (a) the Gummera subcatchment in the Abay River basin (Figure 4.2) and (b) the Borkena subcatchment in the Awash River basin (Figure 4.1).

4.2.1 Borkena Catchment (Awash River Basin/Ethiopia)

The Borkena River rises in Kutaber woreda, near the confluence of the Abay and Awash River basins (Ethiopia). Lower Awash, Middle Awash and Upper Awash are the three primary catchments of the Awash River basin. The Borkena River and its several tributaries are found in Lower Awash, including the Berbera River, Arawutie River, Wuranie River, Abba Abdela/Desso River, Worka River, and Leyole River. This river's overall length is considered to be approximately 165 kilometres. Major cities in the Borkena River watershed include Kombolcha, Dessie, and Kemissie. The streamflow outlet for the case study region is at Kombolch station, and the catchment area is 1709 km^2, delimited by 39° 30' E to 40° 0' E to 10° 15' N to 11° 30' N. Furthermore, the region's elevation ranges from 1775 m to 2638 m above sea level. This catchment's rainfall pattern is unimodal, with 84% falling between July and September (Abera et al., 2020).

4.2.2 Gummera Catchment (Abay River Basin/Ethiopia)

The Gummera subbasin, one of Lake Tana's primary tributaries in the Abay River basin, is the second case study region. The lake, situated at 12°00' N and 37°15' E in the northwestern highlands, gathers water from over 40 rivers. The lake is fed by many large rivers, including the Gilgel Abay in the south, the Megech River in the north, and the Ribb and Gummera in the east. Small river streams run into the lake from the lake's western side. The Gummera River rises in the Guna Mountains southeast of Debre Tabor at an elevation of approximately 3250 meters above sea level. The Gummera watershed area is approximately 1592 km^2. Many small intermittent and perennial rivers and springs feed the Gummera River. The catchment's geography is uneven, ranging from 1788 to 3750 meters above sea level.

4.2.3 Data

Ethiopia's Ministry of Water and Energy (MoWE) provided daily streamflow time series data from two hydrometeorological stations. The data were then used to forecast single-step streamflow. A total of 6575 accessible streamflow data series were recorded at Borkena station between January 1, 1972, and December 31, 1989. Similarly, from January 1, 1981, to December 31, 2006, 9496 streamflow time series measurements were recorded at the Gummera station. The time series includes 866 null values (i.e., 658 at Borkena and 208 at Gummera).

Simple interpolation to complicated statistical approaches is available to fill these gaps (Mwale et al., 2012). The approach is determined by the length and season of missing data, the availability of hydrometeorological data, the climatic area, and the duration of prior observations. The sample mean or subgroup mean might be employed to fill up missing values in daily streamflow data. Replacing missing data with sample means, on the other hand, may result in underestimating variance and inaccurate subgroup identification (Mesta et al., 2021). Instead, the linear interpolation approach is rapid and straightforward, which may be enough for data with a small gap (Hamzah et al., 2021). Thus, in this investigation, we used linear interpolation, and since the bulk of the null values were stacked at low flows, the interpolation is acceptable (Fleig et al., 2011; Gao et al., 2020; Gnauck, 2004; Yawson et al., 2005).

Following these strict quality control procedures, the raw data were separated into training and testing data sets using a ratio of 80:20. As a result, varied widths of single overlapping step sliding windows were employed to reconstruct the input time series into a supervised learning format. The subsets were then normalized using a standard scalar technique. Figure 4.3 shows descriptive statistics and split data plots for both catchments.

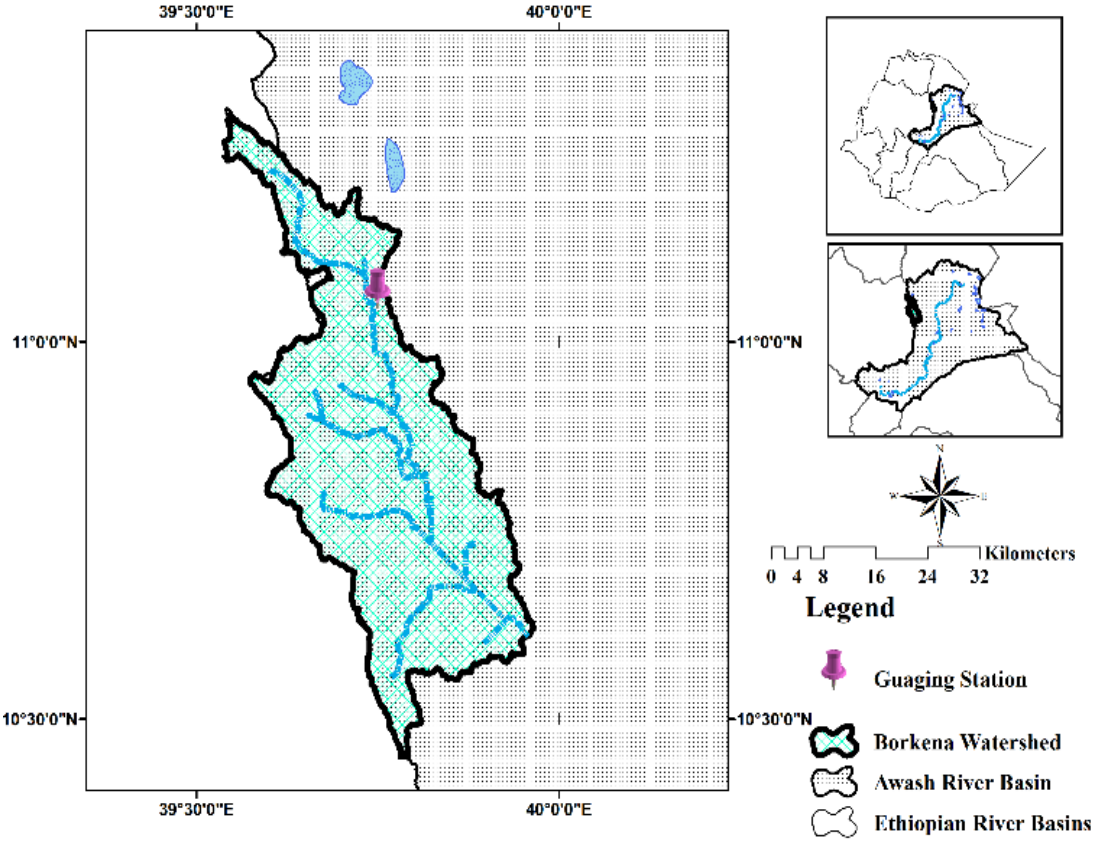


Figure 4. 1 Location of the Borkena catchment.

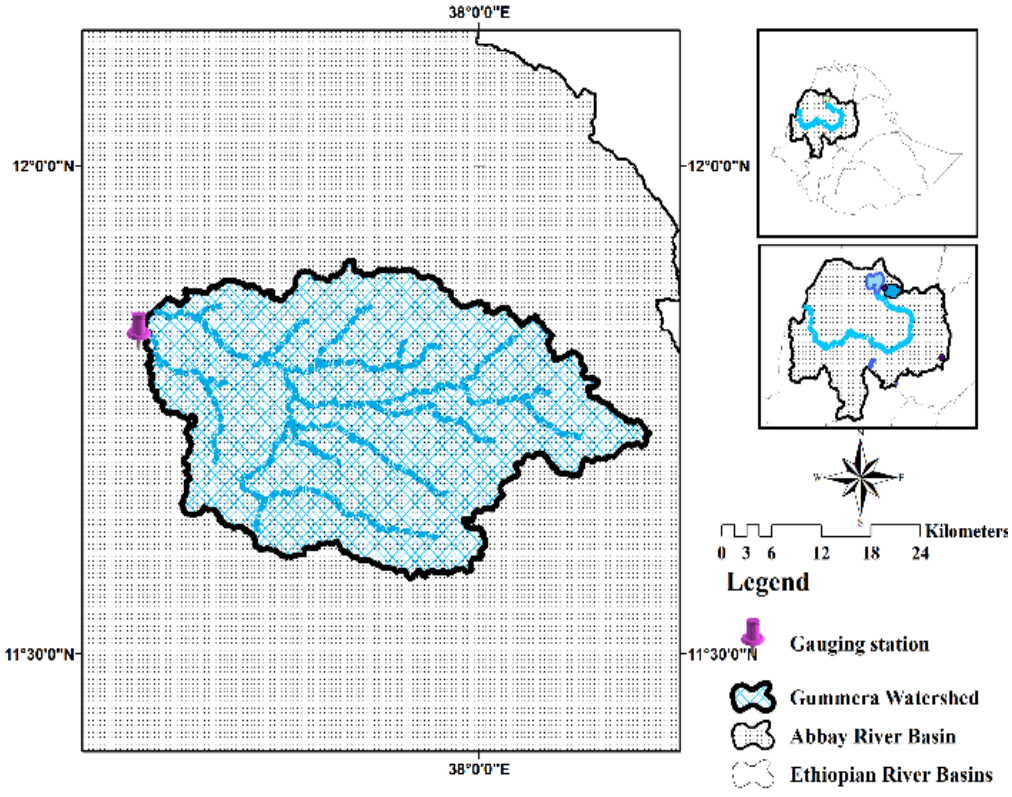
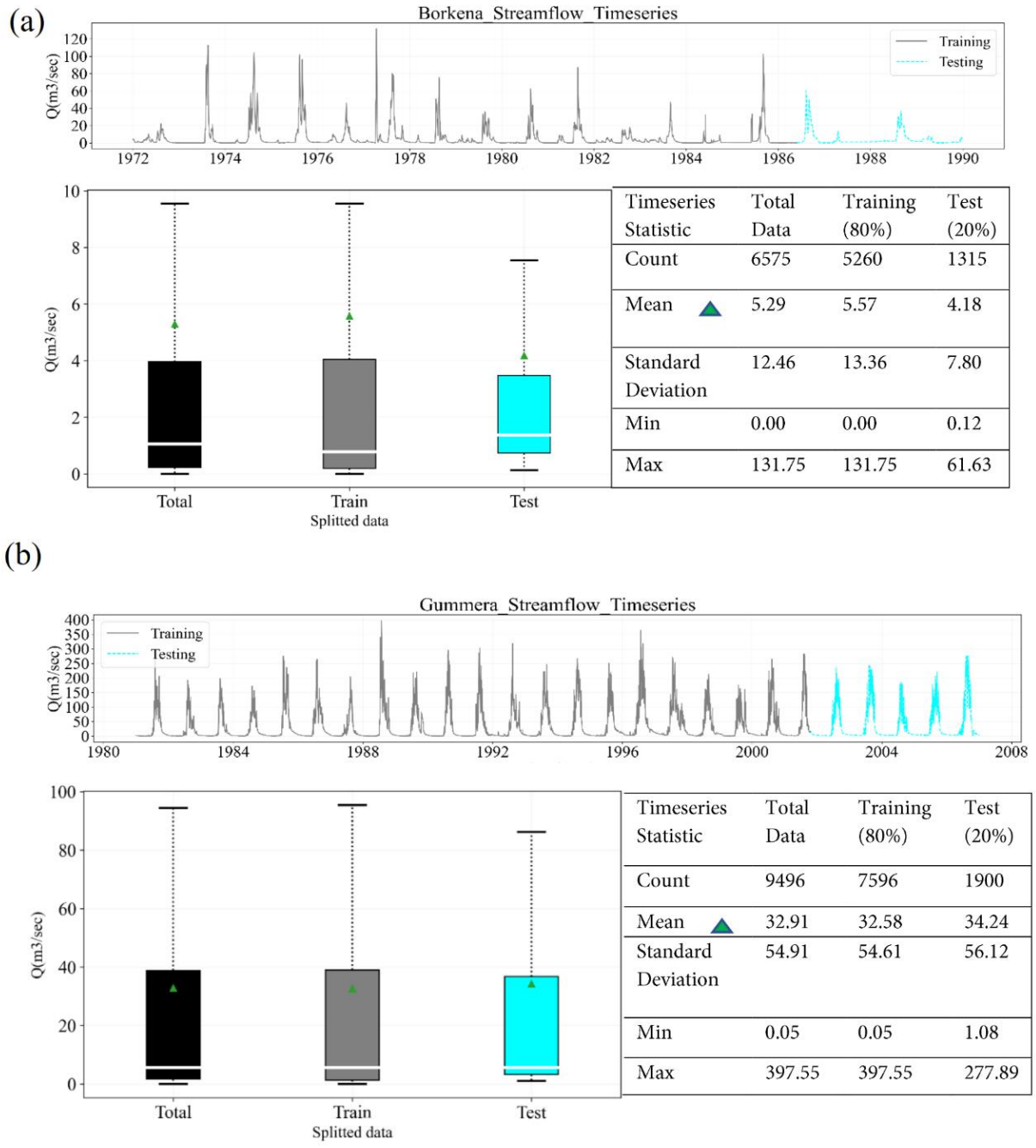


Figure 4. 2 Location of the Gummera catchment.



Timeseries Statistic	Total Data	Training (80%)	Test (20%)
Count	6575	5260	1315
Mean	5.29	5.57	4.18
Standard Deviation	12.46	13.36	7.80
Min	0.00	0.00	0.12
Max	131.75	131.75	61.63

Timeseries Statistic	Total Data	Training (80%)	Test (20%)
Count	9496	7596	1900
Mean	32.91	32.58	34.24
Standard Deviation	54.91	54.61	56.12
Min	0.05	0.05	1.08
Max	397.55	397.55	277.89

Figure 4. 3 Descriptive statistics and the corresponding box plots of split data. (a) Borkena station, (b) Gummera station

4.2.4 Methods

Deep learning model optimization requires judgments on a variety of major hyperparameters, such as the number of layers, number of units, batch size, epochs, and learning rate (Torres et al., 2019). From the four basic hyperparameter optimization techniques: trial-and-error, grid, random, and probabilistic methods, a random search may create an endless number of hyperparameter combinations with a median cost (Torres et al., 2021). As a result, in this work, we employ Keras Tuner, a computationally fast randomized search approach created by the Google team, to search for random combinations of parameters that improve performance. Figure 4.4 depicts a complete flow chart of the suggested technique.

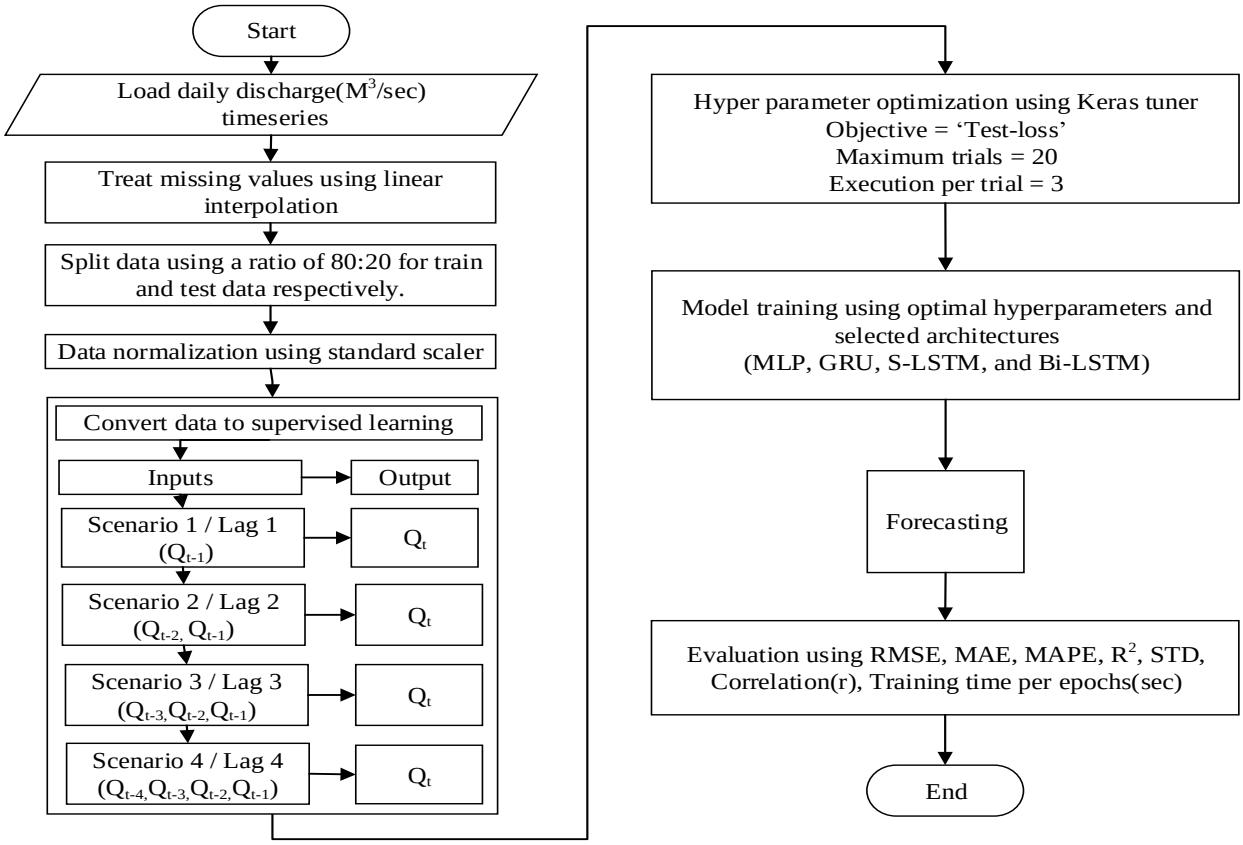


Figure 4. 4 Data analysis flow chart proposed for the study.

A double fully connected hidden layer was used in the suggested models. With experience, the minimum and maximum number of neurons at each hidden layer were determined. The output layer is then connected to a dense layer that has a single output neuron with a linear activation function.

The network is built using the Adam optimizer and the mean squared error loss function, and the hyperparameter value ranges and alternatives are detailed in Table 1. Furthermore, the following paragraph describes each hyperparameter that was tuned using the Keras tuner.

- **Activation function**

The activation function in deep learning models determines the output from the inputs that each node receives. Rectified linear units (ReLU) were used in all layers except the output layer in our study.

- **Learning rate**

In deep learning, one of the hyperparameters determining the step size when the model moves to a minimal loss function is the learning rate. As a result, it is critical to tune the learning rate appropriately; otherwise, the model may converge slowly with a low learning rate or diverge from the optimal error values with a high learning rate (Konar et al., 2020). Using Keras Tunner, this research assigned three alternative values (1e-2, 1e-3, or 1e-4).

- **Number of epochs**

Another hyperparameter determines how long the deep learning algorithm will learn using the complete training set. When we increase the number of epochs, the weight in the model has more opportunities to update itself. The loss curve progresses through underfitting, optimal state, and overfitting. However, there are no rigid restrictions for these hyperparameter settings, and we use Keras Tunner to select the lowest (10) and maximum (100) values for optimization.

- **Amount of batch sizes**

The batch size is a sample size cluster handled before the model changes.

- **Drop out**

A hyperparameter that minimizes overfitting and improves training performance. As a result, the dropout freezes a portion of the neuron from training at each iteration, and it is specified on a scale of 0 to 1.

Table 4. 1 Model hyperparameter choices or value ranges for optimization by Keras tuner.

Nº	Hyperparameters	Value Ranges**			Choices	Default
		Min	Max	Step		
1	MLP, LSTM, Bi-GRU+LSTM, or GRU Layer 1 units	5	40	5	*	*
2	Dropout 1	0.0	0.3	0.1	*	0.2
3	MLP, LSTM, Bi-LSTM, or GRU Layer 2 units	5	40	5	*	*
4	Learning rate	*	*	*	1e-2, 1e-3 or 1e-4	*
5	Number of epochs	10	100	10	*	*
6	Number of batch sizes	10	100	10	*	*

*** Value ranges or choices for optimization by Keras tuner: (Objective = "Test Loss", Max Trials = 20, Executions Per Trial = 3)*

** Not applicable*

TensorFlow, Keras, Scikit-Learn, Matplotlib for visualization, and Statsmodels for performance assessment were among the open-source Python libraries used for model building. Furthermore, the simulation was carried out using a computer equipped with a Processor: Intel(R) Core (TM) i7-6500U CPU 2.50 GHz and RAM: 8 Gigabytes memory.

- **Input time lag selection**

The autoregressive model with the partial autocorrelation function (PACF) was used to reprocess the highly correlated time series delay that was deconstructed and utilized as a single input to a deep learning neural network. The autoregressive model is shown in Equation 1. (p).

$$x_t = \varphi_1 x_{(t-1)} + \varphi_2 x_{(t-2)} + \cdots + \varphi_p x_{(t-p)} + \varepsilon_t \quad (1)$$

where φ is the autoregressive parameter, x is the observation at time t, and ε is the weighted noise at time t. The autoregression explores the correlation between current and past values (Aljahdali et al., 2020). Figure 4.5 shows the partial autocorrelation of daily streamflow time series with a 95% confidence interval and a time delay of 4 days was examined in this research for both case study locations.

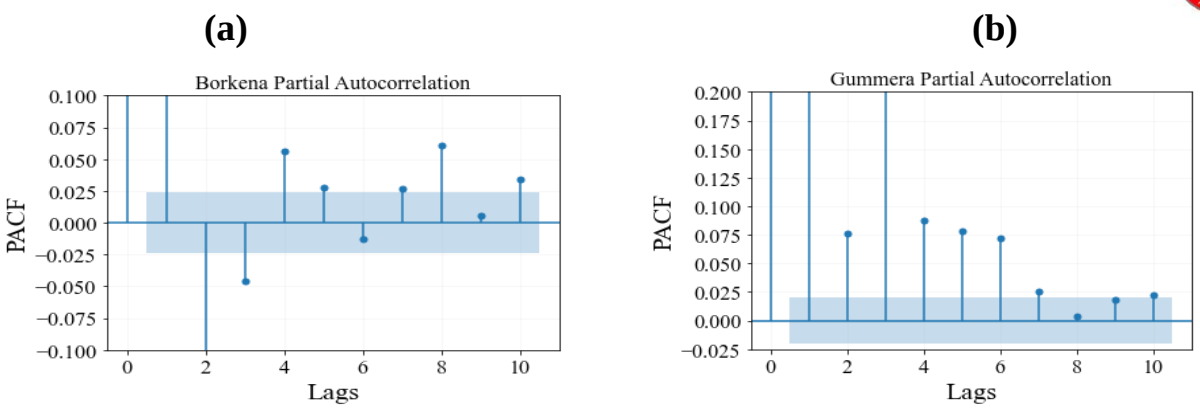


Figure 4. 5 Partial autocorrelations of the daily time series for both catchments (Lag units are in days, and the blue shadow lines indicate 95% confidence intervals). (a) Borkena, (b) Gummera

4.3. Results and Discussion

Previous research yielded various outcomes, depending on the kind of deep learning architecture utilized, the degree of climate variability, and the timeframe used. On monthly univariate streamflow data from the Shangjingyou and Fenhe reservoir stations in the upper reaches of the Fenhe River, the GRU model beats both the extreme learning machine (ELM) and the least squares support vector machine (LSSVM) (Zhao et al., 2021). On univariate daily discharge data from the Basantapur gauging station in India's Mahanadi River basin, Sahoo et al. (2019) proved the superiority of LSTM over RNN. Suradhaniwar et al. (2021) revealed that when hourly averaged univariate time series data are employed, the SARIMA and SVR models outperform the NN, LSTM, and RNN models.

Despite the complexity of best model generalization, case-based analysis is the most effective way to decide which model best suits a specific circumstance (Gunathilake et al., 2021). In this work, we sought for the first time to arrange the results around three themes: model variability, lag time variability, and time series features (climatic variability).

The following section analyses the performance of four chosen models under four distinct input time lag situations; 16 experimental findings are reported. Tables 4.2 and 4.3 show the optimum hyperparameter settings using the Keras Tuner for both stations and all circumstances. Tables 4.4 and 4.5 also provide performance indicators such as RMSE, MAE, MAPE, R^2, and Training Time Per Epoches (TTPE) (sec). During the testing phase, single-step streamflow forecasting findings for both case study locations revealed highly satisfactory performance.

4.3.1 Model Variability

We investigated the influence of model variability on the accuracy of single-step streamflow forecasting using four evaluation matrices. The distribution of forecast error (m^3/s) was then shown using box plots. We also created a bar chart (Figure 4.8) with multiple prediction error classes to determine the class boundary with the most significant error concentration. As a consequence, as shown in Table 4.4, the GRU model outperforms the MLP and S-LSTM models for the Borkena station's lag time (T+2 and T+3). MLP outperformed GRU and S-LSTM in terms of performance increase in lag time (T+1 and T+3) for the Gummera catchment (Table 4.5). Predicting error box plots and bar chart graphs (Figures 4.6 & 4.7) were utilized to study these high-performing models further. As a result, the GRU prediction error is often concentrated in narrow ranges (0 to 0.5 m^3/s) for the Borkena watershed and (0 to 2.5 m^3/s) for the Gummera catchment. Furthermore, when evaluating computational performance, MLP displayed the shortest training time per epoch, followed by S-LSTM, GRU, and Bi-LSTM, as shown in Tables 4.4 and 4.5.

Table 4. 2 Keras tuner optimized hyperparameter values for Borkena station with its MSE score.

Hyperparameters	MLP				S-LSTM				Bi-LSTM				GRU			
	T+1	T+2	T+3	T+4	T+1	T+2	T+3	T+4	T+1	T+2	T+3	T+4	T+1	T+2	T+3	T+4
Layer_1_units	35	10	40	35	25	35	40	40	35	40	25	20	25	30	15	10
Dropout_1	0.0	0.0	0.0	0.1	0.1	0.0	0.2	0.2	0.1	0.2	0.0	0.1	0.1	0.0	0.1	0.1
Layer_2_units	10	20	25	15	35	25	30	10	20	30	20	30	20	25	40	25
Learning rate	0.0001	0.0001	0.0001	0.001	0.0001	0.0001	0.001	0.001	0.0001	0.001	0.001	0.001	0.0001	0.001	0.0001	0.001
Number of epochs	90	100	100	40	60	80	20	30	70	20	50	70	50	60	40	50
Number of batch sizes	20	20	30	30	30	40	20	10	40	20	20	30	50	30	10	90
Score (MSE)	0.099	0.099	0.096	0.097	0.099	0.099	0.098	0.096	0.099	0.099	0.097	0.096	0.099	0.099	0.099	0.097

Table 4. 3 Keras tuner optimized hyperparameter values for Gummera station with its MSE score.

Hyperparameters	MLP				S-LSTM				Bi-LSTM				GRU			
	T+1	T+2	T+3	T+4	T+1	T+2	T+3	T+4	T+1	T+2	T+3	T+4	T+1	T+2	T+3	T+4
Layer_1_units	25	25	10	25	20	30	15	25	10	25	15	15	5	40	25	15
Dropout_1	0.1	0.0	0.0	0.0	0.0	0.0	0.2	0.2	0.0	0.0	0.1	0.1	0.0	0.0	0.0	0.2
Layer_2_units	15	40	30	25	30	10	35	10	5	20	10	20	30	20	25	15
Learning rate	0.0001	0.0001	0.0001	0.001	0.01	0.01	0.1	0.01	0.01	0.01	0.1	0.01	0.1	0.1	0.001	0.01
Number of epochs	70	90	80	50	60	50	50	80	60	90	50	30	80	40	50	30
Number of batch sizes	60	60	70	60	80	40	30	50	50	40	20	10	20	30	50	40
Score (MSE)	0.019	0.011	0.012	0.015	0.019	0.011	0.013	0.015	0.018	0.011	0.013	0.013	0.018	0.012	0.013	0.016

4.3.2 Time Series Characteristics (Climatic Variability)

Time series features are another significant factor influencing deep learning model performance. As a result, the four evaluation metrics displayed in this study show that Borkena's river catchment forecasting result is more accurate than Gummera's catchment, with RMSE, MAE, MAPE, and R^2 ranging between (0.81 and 1.53, 0.29 and 0.96, 0.16 and 1.72, 0.96 and 0.99) and (17.43 and 17.99, 7.76 and 10.54, 0.16 and 1.03, 0.89 and 0.90) for Borkena and Gummera catchments respectively. The significant natural streamflow time series variability in the Borkena watershed might be the reason for this performance discrepancy between the two catchments. Furthermore, the distribution of prediction error (m^3/sec) in Figures 4.6 and 4.7 demonstrates that the error class limit is lower in the Borkena station than in the Gummera station in most circumstances.

Table 4. 4 Performance comparison of the proposed models for different input time lags at the Borkena station.

Borkena	T+1					T+2					T+3					T+4				
	RMSE	MAE	MAPE	R^2	TTPE * (sec)	RMSE	MAE	MAPE	R^2	TTPE* (sec)	RMSE	MAE	MAPE	R^2	TTPE * (sec)	RMSE	MAE	MAPE	R^2	TTPE* (sec)
MLP	1.09	0.47	0.85	0.98	0.24	0.89	0.38	0.76	0.99	0.16	1.06	0.35	0.19	0.98	0.12	**0.81**	**0.29**	**0.34**	**0.99**	0.29
GRU	1.15	0.64	0.84	0.98	1.29	**0.85**	**0.31**	**0.16**	**0.99**	0.79	**0.91**	**0.35**	**0.18**	**0.98**	0.99	1.35	0.69	1.41	0.97	1.31
S-LSTM	1.07	**0.38**	**0.29**	**0.98**	0.30	0.87	0.36	0.26	0.99	0.54	1.02	0.64	0.99	0.98	0.81	1.53	0.96	1.72	0.96	0.58
Bi-LSTM	1.06	0.39	0.31	0.98	0.88	0.90	0.46	0.76	0.99	1.55	1.12	0.51	0.95	0.98	1.55	0.98	0.44	0.53	0.98	5.3

*TTPE(Training Time per Epochs). The bold values indicate the best performance score for each time lag.

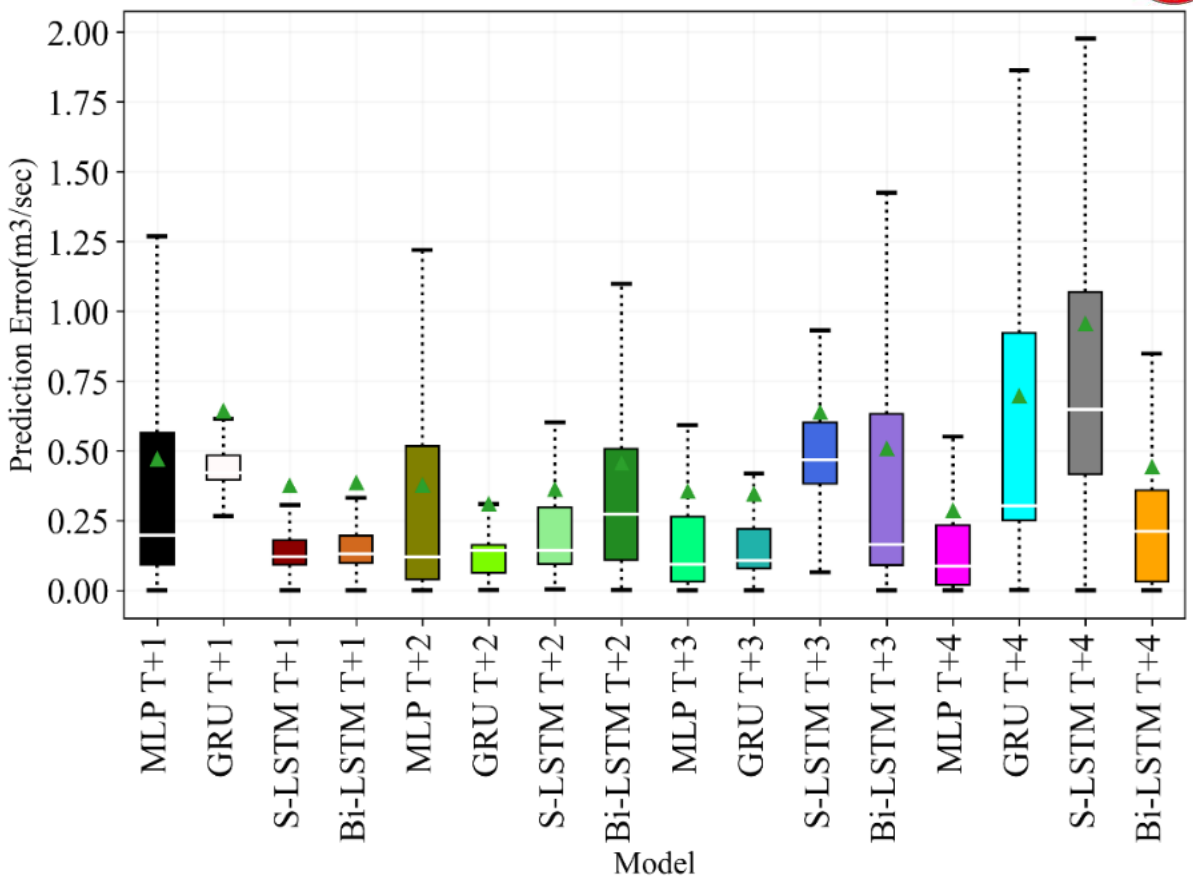


Figure 4. 6 Spread of prediction error (m^3/s) or box plot for the proposed models during the test period (Borkena station).

Table 4. 5 Performance comparison of the proposed models for different input time lags at Gummera station.

Gummera	T+1					T+2					T+3					T+4				
	RMSE	MAE	MAPE	R^2	TTPE* (sec)	RMSE	MAE	MAPE	R^2	TTPE* (sec)	RMSE	MAE	MAPE	R^2	TTPE* (sec)	RMSE	MAE	MAPE	R^2	TTPE* (sec)
MLP	17.67	7.84	0.18	0.90	0.88	17.68	7.87	0.19	0.90	0.75	17.43	7.76	0.18	0.90	0.61	17.56	9.73	0.78	0.90	0.95
GRU	17.66	8.39	0.41	0.90	0.51	17.76	7.82	0.16	0.90	1.43	17.47	8.29	0.35	0.90	1.24	17.99	10.54	1.03	0.89	0.53
S-LSTM	17.66	8.29	0.39	0.90	1.54	17.71	8.04	0.22	0.90	0.75	17.69	8.51	0.32	0.90	2.99	17.69	8.41	0.35	0.90	3.31
Bi-LSTM	17.63	7.98	0.27	0.90	0.92	17.93	8.39	0.29	0.89	2.34	17.83	9.18	0.56	0.89	2.77	17.56	8.56	0.39	0.90	1.84

**TTPE(Training Time per Epochs).The bold values indicate the best performance score for each time lag.*

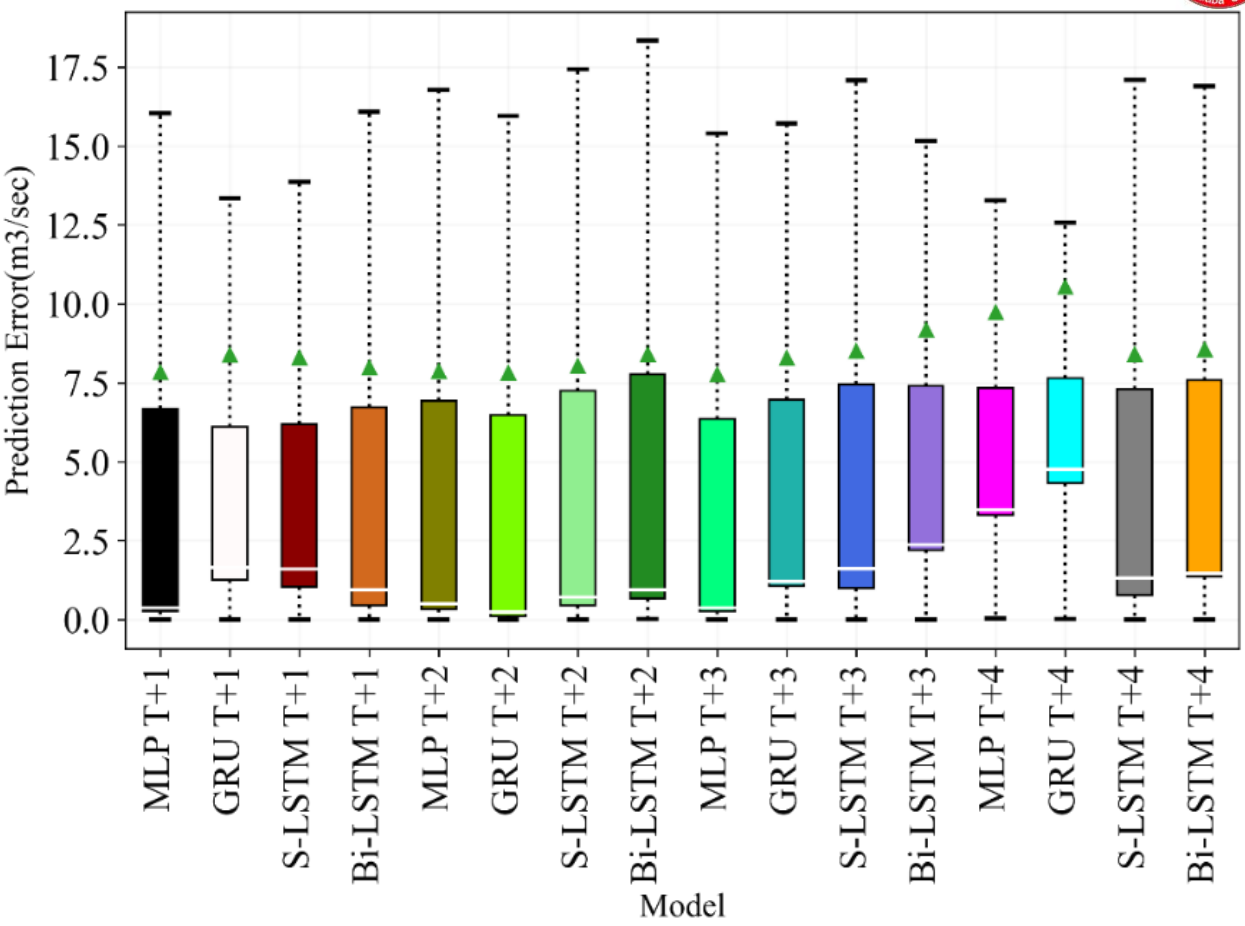


Figure 4. 7 Spread of prediction error (m^3/s) or box plot for the proposed models during the test period (Gummera station).

4.3.3 Lagged Time Variability

Other major parameters that store temporal information and impact model performance in univariate streamflow forecasting are lagged variables. Figure 4.9 depicts the predicting capabilities of the proposed models with observed test data for both case study locations using a Taylor diagram. The figure is shown on a two-dimensional scale, with the standard deviation on the polar axis, the root mean square error on the radial axis, and the correlation coefficient on the radial axis. This demonstrates that, regardless of deep learning models, forecasting with a four-day lag produces a time series that is closest to the standard deviation of the actual test data. Furthermore, Figures 4.10 and 4.11 also show the high optimized score deep learning model's actual and predicted values for each time lag and both case study catchments.

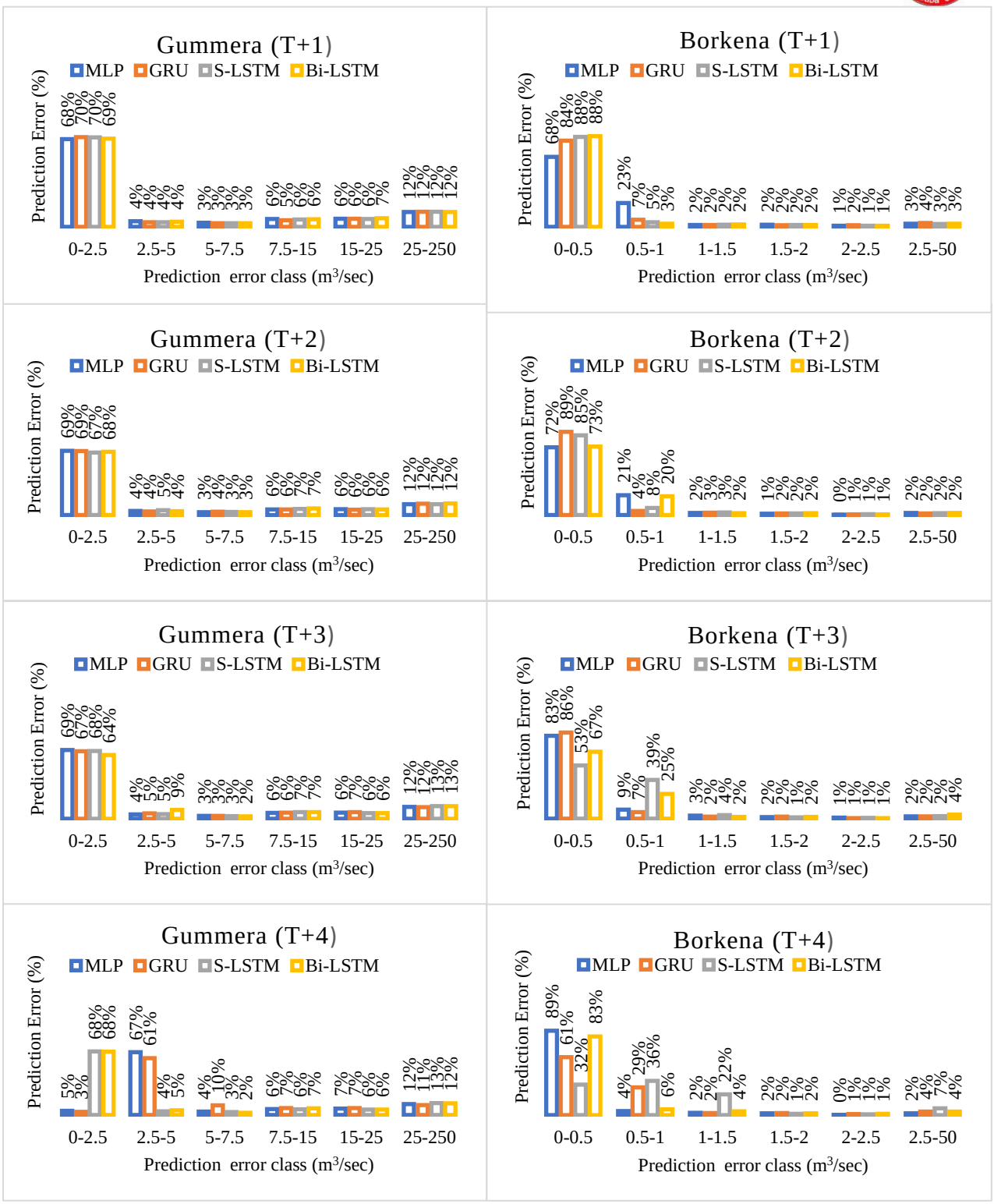


Figure 4. 8 Bar chart showing the frequency (%) of absolute prediction errors | PE | (m^3/sec) with different class limits for the proposed four models and input lagged time variables in both catchments.

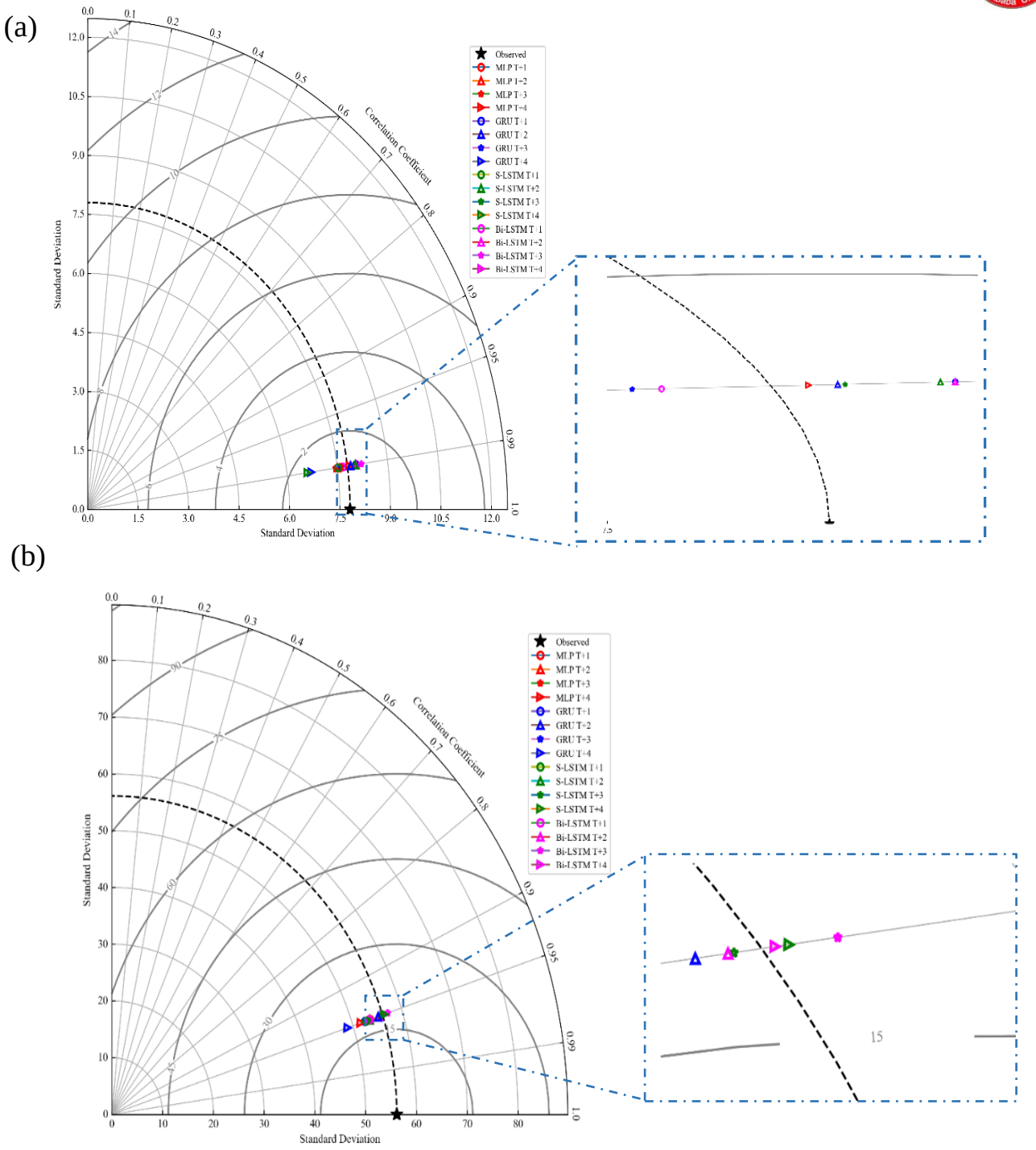


Figure 4. 9 Taylor diagram displays the standard deviations, root mean square error, and correlation coefficient between observed and predicted streamflow for the proposed four models and lagged time variables (Q, m^3/s). (a) Borkena, (b) Gummera.

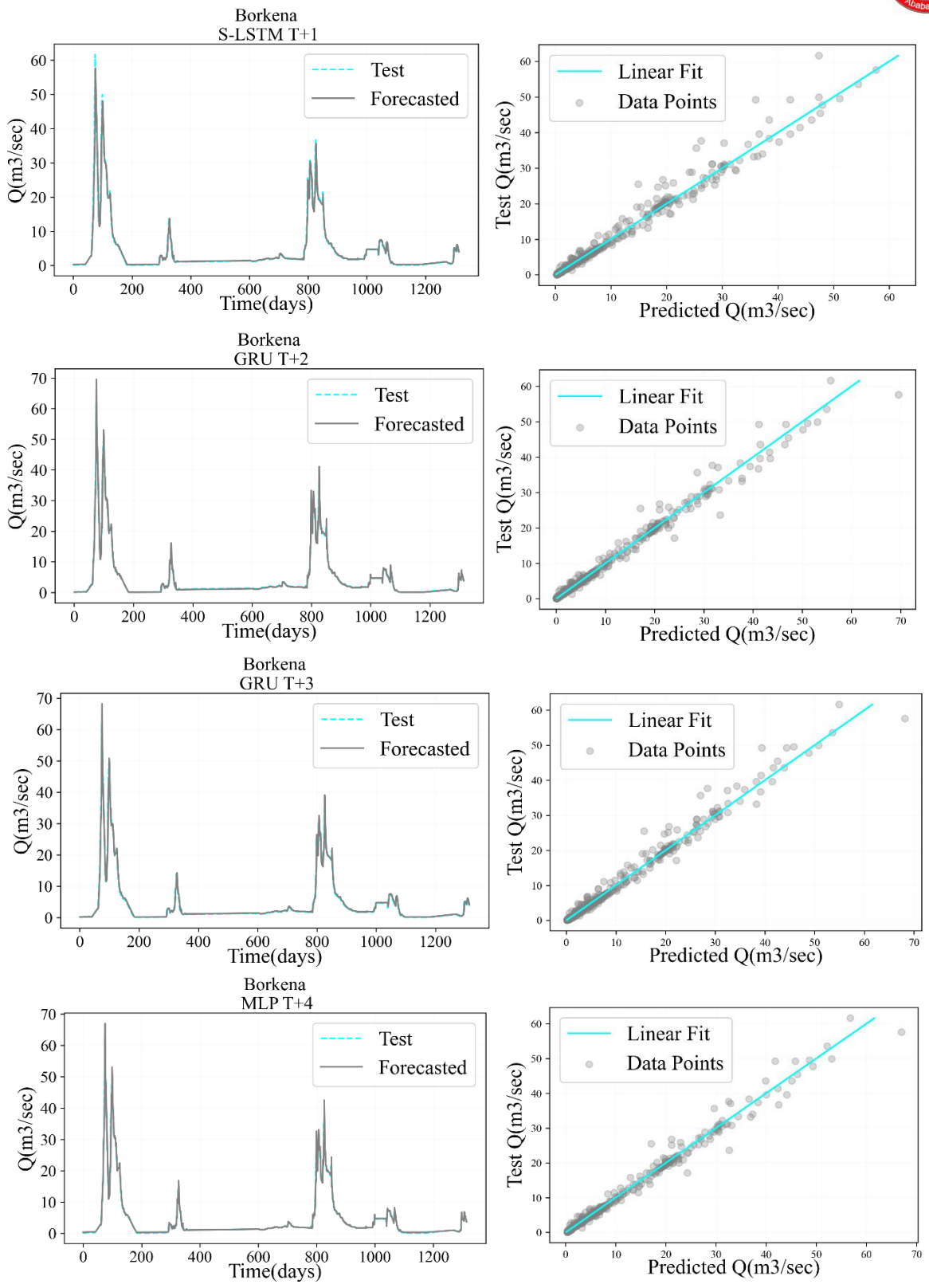


Figure 4. 10 Comparison of true and predicted values of the highly optimized score deep learning model for each time lag (Borkena station).

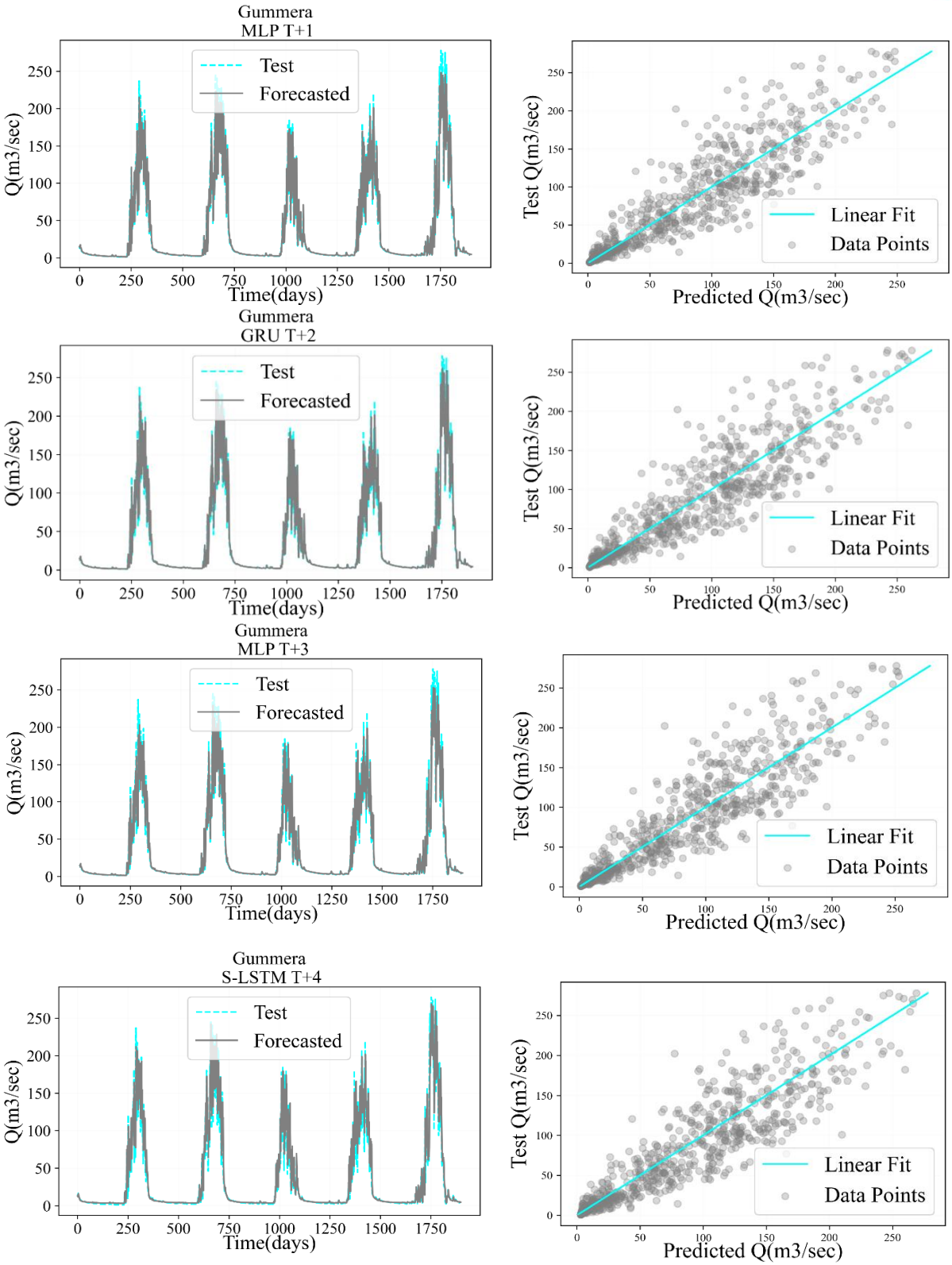


Figure 4. 11 Comparison of true and predicted values of the highly optimized score deep learning model for each time lag (Gummera station).

4.4. Conclusions

A comparison of several deep learning algorithms for one-step daily streamflow forecasting at two subcatchment streamflow outlets was presented in this work. The four algorithms utilized in this work are MLP, S-LSTM, Bi-LSTM, and GRU. The research convincingly demonstrated the effects of climate (time series characteristics) and delayed temporal variability on the performance of several proposed deep-learning models. The following significant points will summarize the findings of this study.

I. Deep learning models have a high potential for predicting short-term daily streamflow in various time series properties.

II. Deep learning model performance in short-term streamflow forecasting changes with time series features and input time lag variations. Furthermore, the Borkena station has smaller time series variability than the Gummera station, which is reflected in the model findings. As a result, our research demonstrated that catchment response variability influences deep learning model performance.

III. In both case study regions, the MLP and GRU outperform the S-LSTM and Bi-LSTM on a nearly equal basis for single-step short-term streamflow forecasting. The performance, however, is related to delayed temporal variations.

IV. Catchment properties significantly influenced streamflow forecasting performance more than deep learning model structures and lagged time variations.

V. The research also found that classical MLP performed almost as well as S-LSTM and GRU deep learning networks on a limited quantity of streamflow time series data.

Future research may extend the results of this work to additional climatic locations, hybrid deep-learning model architectures, hyperparameter modifications, and delayed time selection approaches. We also need to look at the consequences of high input variability on deep learning models for univariate streamflow forecasting. As part of our future study, we intend to employ hybrid deep learning models and multivariate input data to simulate streamflow.

CHAPTER 5:

MULTIVARIATE STREAMFLOW SIMULATION USING HYBRID DEEP LEARNING MODELS

Abstract[2]

Streamflow modelling is critical in developing water resources, particularly in agriculture, the environment, household water supply, hydropower production, flood management, and early warning systems. Deep learning algorithms have recently received much interest in this area because of their high-performance simulation capability. We examined a multilayer perceptron (MLP), long short-term memory (LSTM), and gated recurrent unit (GRU) with the suggested hybrid models, CNN-LSTM and CNN-GRU. As a result, to simulate one-step daily streamflow in various agro-climatic conditions, a rolling time frame and various variable input combinations were used. Daily multivariate and multisite time series data were obtained from the study's Awash River basin (Borkena watershed, Ethiopia) and Tiber River basin (Upper Tiber River basin, Italy) stations. The datasets were subjected to strict quality control procedures. As a result, it rolled to a different time lag to eliminate noise in the time series before splitting it into training and testing datasets in an 80:20 ratio. Finally, the findings demonstrated that combining the GRU layer with the CNN layer and employing monthly rolling average daily input time series may significantly enhance streamflow time series simulation.

[2] *This chapter is based on the publication*- https://www.hindawi.com/journals/cin/2021/5172658/

5.1. Introduction

A hydrological simulation is an emerging study topic in hydrology (Z. Zhang et al., 2018). Catchment responses are analysed in terms of meteorological forcing factors. Flood protection, water supply distribution, hydraulic construction design, and reservoir operation require hydrological modelling (Ni et al., 2020; Yuan et al., 2018). On the other hand, river flow modelling is a challenging task since river flow time series are often unpredictable, dynamic, and chaotic. The link between streamflow formation and other hydrologic processes is nonlinear, with external climatic influences and global warming influencing it and physical catchment features.

The majority of streamflows are measured at river gauging stations. Various studies, however, reveal that the availability of gauging station data is declining in most regions of the globe (A. Sichangi et al., 2018). Tourian et al. (2013) obtained a time series graph of the number of stations having accessible discharge data from the Global Runoff Data Centre (GRDC). Between 1970 and 2010, this data series shows a decrease in total measured yearly streamflows. Furthermore, insufficient discharge surveillance and malfunctioning gauging stations are exacerbated in underdeveloped nations (A. W. Sichangi et al., 2016). Ethiopia's sparsely spread rain gauge stations further hinder the effectiveness of physical hydrological models. As a result, research into the resilience of novel discharge data estimation techniques is significant.

In the literature, streamflow simulation models are often classified into two types: (1) process- or physical-based models derived from catchment features and (2) data-driven models based on previously acquired data (Aljahdali et al., 2020; Ni et al., 2020; Yuan et al., 2018). Process-based models often use an experimental formula, which provides insight into physical properties and has demanding data requirements. On the other hand, data-driven models are appropriate and may be implemented without considering the watershed system's fundamental physical mechanism (Aljahdali et al., 2020; Ni et al., 2020; Yuan et al., 2018).

The most widely used and researched "black-box" or machine learning models are artificial neural networks (ANNs). They are used in a wide variety of scientific and technological fields than the list of available black-box algorithms, including support vector machines (SVMs), genetic programming (GP), fuzzy logic (FL), recurrent neural networks (RNNs), and long short-term memory (LSTM) (Aljahdali et al., 2020; Couta et al., 2019). ANNs are accessible in various functionalities and architectural shapes, from basic to complex.

One of the most sophisticated ANN designs is the recurrent neural network (RNN). It is said to be a primarily built deep learning network for time series analysis that efficiently adjusts to temporal dynamics utilizing previous time step data (Ni et al., 2020). RNNs, on the other hand, cannot capture long-term dependencies and are vulnerable to vanishing and exploding gradients.

One of the most successful methods for time series analysis, according to Sepp Hochreiter and Jürgen Schmidhuber, is advanced RNN or long short-term memory (LSTM) (Couta et al., 2019). The LSTM unit has a cell with an input gate, an output gate, and a forget gate (Y. Bai et al., 2019). The LSTM model has demonstrated promising results in various applications, including voice recognition, time series modelling, natural language processing, handwriting recognition, and traffic flow simulation (Campos et al., 2019; Yuan et al., 2018). LSTM has also been demonstrated in studies to outperform stronger multilayered (ML) methods for streamflow modelling (Sahoo et al., 2019; Yuan et al., 2018). Campos et al. (2019) predicted floods at four Para'iba do Sul River stations in Brazil using an autoregressive integrated moving average (ARIMA) and an LSTM network. Aljahdali et al. (2020) compared the LSTM network and layered RNN to forecast streamflow in the Black and Gila rivers in the United States. A recent article by Rahimzad et al. (2021b) employed time-lag Q_{t-1}, Q_{t-2}, and other climatic factors to predict Q_t in the future and concluded that the LSTM network outperforms linear regression (LR), multilayer perceptron (MLP), and support vector machine (SVM) in forecasting daily streamflow.

Cho et al. (2014) proposed gated recurrent units (GRUs) a few years ago, which are comparable to LSTM with a forget gate but have fewer parameters than LSTM since they lack an output gate. The capabilities of GRU in speech signal modelling and natural language processing were comparable to those of LSTM. However, there are debates over the relative performance of these two systems for streamflow and reservoir inflow modelling, which has not been well researched across different timelines and environments.

Identifying acceptable time series models from numerous known deep-learning network designs is challenging, regardless of their performance differences. LSTMs and GRUs are not always the best solutions for sequence prediction. However, additional study is needed to develop simulations with higher prediction accuracy, shorter run times, and simpler models. As a result, this comparative examination of network designs aids in determining the best option for time-

series analysis.

Different hybrid deep-learning models have recently received much interest from academics. Chen et al. (2020) estimated nitrogen oxide emissions using a convolutional neural network (CNN), LSTM, and hybrid CNN-LSTM models. They conclude that CNN-LSTM forecasts periodic nitrogen oxide emissions from the refining sector accurately and consistently. Furthermore, Li et al. (2020) feed univariate and multivariate time series data into LSTM and CNN-LSTM models. Consequently, due to its low error and quick training time, the proposed multivariate CNN-LSTM model outperforms the competition for air quality analysis utilizing particulate matter ($PM_{2.5}$) concentration prediction.

The combination of CNN and LSTM models helps time-series prediction models by allowing the LSTM model to collect long-time sequences of pattern information more effectively. On the other hand, CNN models may filter out the noise in the input data and extract more relevant features, potentially increasing the prediction model's accuracy (Livieris et al., 2020). Furthermore, combining CNNs with GRUs may result in robust data preprocessing, giving a realistic alternative for improving the model's accuracy (Ahmed et al., 2021). Although combining CNN and LSTM produced impressive results in several studies, its applicability in hydrological sectors requires more investigation (Y. Liu et al., 2021). Muhammad et al. (2019) simulated streamflow using LSTM, GRU, and Hybrid CNN-GRU models based on 35 years of Model Parameter Estimation Experiment (MOPEX) data from 10 river basins in the United States. They show that the suggested hybrid model beats the traditional LSTM, yet the performance is almost the same as GRU. Barzegar et al. (2020) recently investigated short-term water quality variable prediction using a hybrid CNN-LSTM model and successfully identified low and high water quality variables, namely, dissolved oxygen concentrations.

Screening input variables for various model architectures is another difficult challenge for researchers. Even though rainfall, evaporation, and temperature are all causative factors for streamflow modelling, data availability and research goals restrict the flexibility of choice (Van et al., 2020). According to Van et al. (2020), including temperature and evapotranspiration input nodes in the model increases network complexity and leads to overfitting. Parisouj et al. (2020), on the other hand, believe that employing easily accessible input variables such as temperature and precipitation for data-driven streamflow modelling will provide a credible outcome. As a result, this study will add to the discussion by examining the performance of the suggested models

with diverse input combinations from various climatic locations.

To the best of our knowledge, we have identified scant literature demonstrating the performance variation of different hybrid models for streamflow simulation under many input variable circumstances simultaneously. As a result, we evaluated several hybrid CNN-LSTM and CNN-GRU architectures to the traditional MLP, GRU, and LSTM networks to simulate single-step streamflow utilizing two climatic zones, accessible precipitation, and minimum and maximum temperature data. Furthermore, the research evaluates hybrid models with various layer layouts and employs a Keras tuner to optimize model hyperparameters. The primary goal of this research will be to investigate the performance variation of the suggested models under various input variability situations, including climatic, input combination, input time window, and average rolling time window variability.

The following open-source software and machine learning libraries were utilized in this study: Python 3.6 for programming, Numpy, Pandas, Scikit-Learn, Hydroeval, Statsmodels, and Matplotlib libraries. They were utilized for data preparation, assessment, and graphical representation. Furthermore, the TensorFlow and Keras deep-learning frameworks were used to design deep-learning architectures.

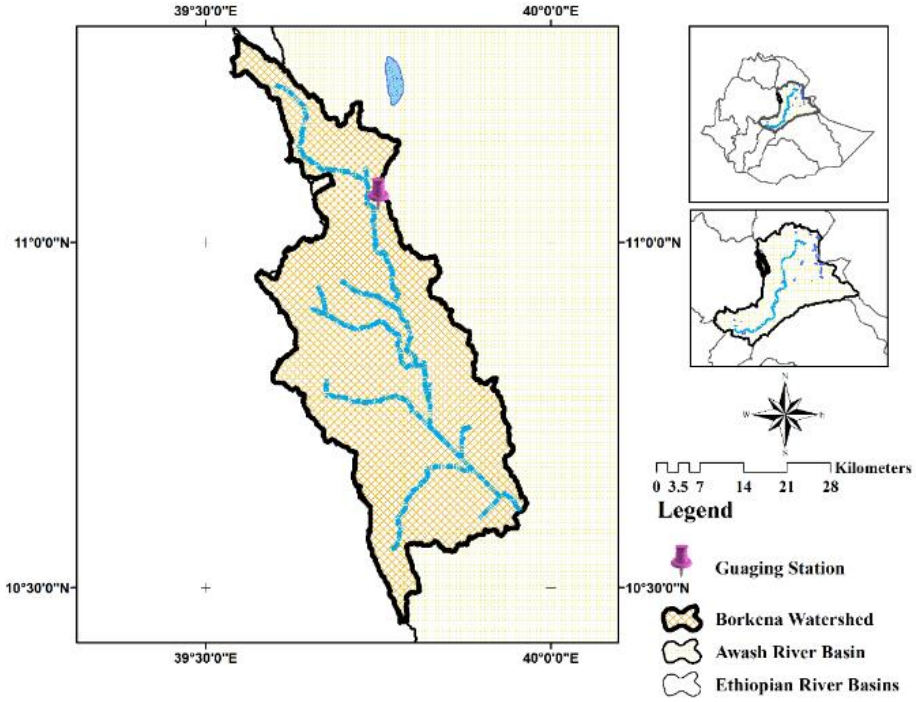
5.2. Data and Methods

Two river subcatchments in two climatic areas were chosen for this study: the Awash River Basin, the Borkena subcatchment in Ethiopia (Figure 5.1 (a)), and the Upper Tiber River Basin in Italy (Figure 5.1 (b)).

5.2.1 Borkena Watershed (Ethiopia)

The first case study location is in the Borkena watershed, situated in northern Ethiopia's upper Awash River basin. The watershed's main streamflows are from Tosa Mountain located near Dessie town. The elevation in the region varies from 1,775 meters at Kombolcha to 2,638 meters upstream of Dessie. This watershed's primary rainy season lasts from July through September.

(a)



(b)

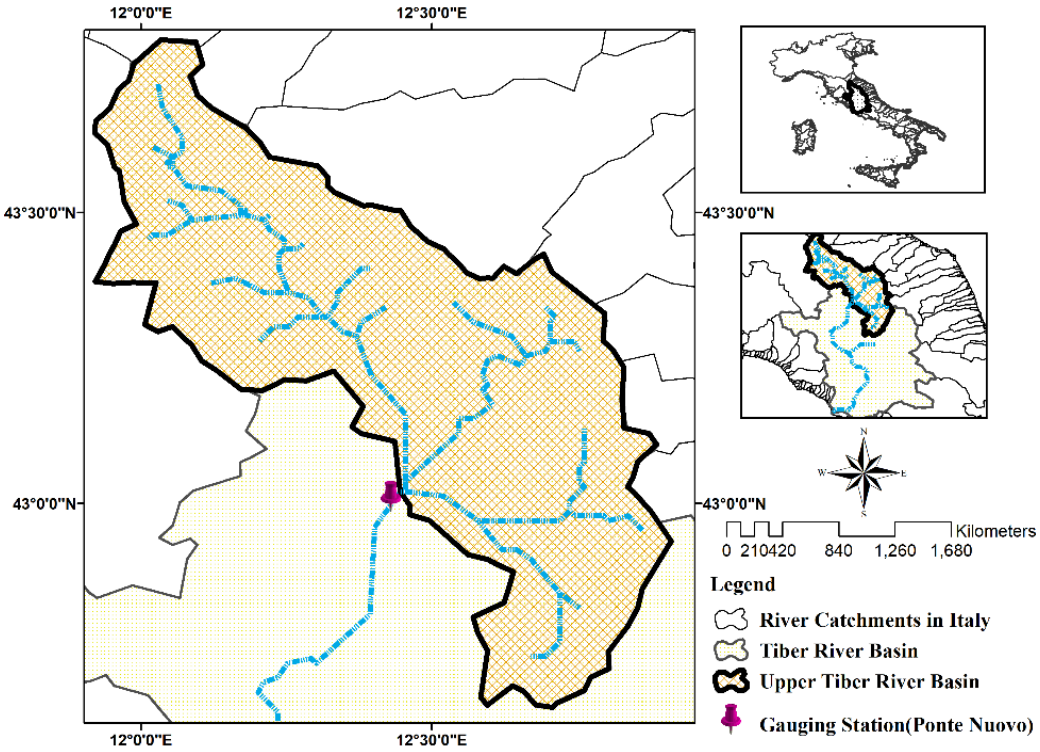


Figure 5. 1 Location of case study areas. (a) Borkena (b) UTRB

5.2.2 Upper Tiber River Basin (Italy)

The second case study is in Italy's Upper Tiber River Basin (UTRB). The TRB is Italy's second-largest watershed (Annis & Nardi, 2019). Geographically, the basin is situated between 40.5° N and 43° N latitudes and 10.5° E to 13° E longitudes, spanning approximately 17,500 km^2 and accounting for approximately 5% of the Italian land. The TRB includes the Upper Tiber River Basin (UTRB), which covers 4145 km^2 (20% of the TRB) and has its outflow at Ponte Nuovo. The catchment's elevation varies from 148 to 1561 meters above sea level. The climate in the region is Mediterranean, with precipitation mainly falling from autumn (March to May) until spring (September to November). Intense rainfall significantly impacts the hydrology of the upstream basin, causing frequent flooding in the downstream districts (Fiseha et al., 2014).

5.2.3 Data

Borkena's needed hydrological and meteorological datasets were obtained from Ethiopia's Ministry of Water and Energy (MoWE) and the National Meteorological Agency (NMA). The datasets from the UTRB are gathered from the Italian National Research Center (Centro National di Riccerca CNR) and stored for public access through the Water Resource Management and Evaluation (WRME) platform at http://hydrogate.unipg.it/wrme/. We gathered 5844 accessible data series from the Borkena watershed between January 1, 1999, and December 31, 2014. Similarly, 7670 data series were collected for UTRB from January 1, 1958, to December 31, 1978. The datasets for both case studies are multivariate and multisite. Although we are very concerned and picked a series of time frames with a minor data gap for both stations, the datasets include multiple missing values for various reasons. As a result, the initial step for this study was to fill in the missing numbers using the Monte Carlo method.

The research used linear correlation statistics to assess the degree of dependence between various input variables (Rania et al., 2000). Even though Mehr and Gandomi (Mehr & Gandomi, 2021) said that linear correlation might mislead or supply an abundance of inputs, our research does not have an extensive feature size that necessitates intense feature selection criteria. As a result, we chose a linear correlation coefficient.

Furthermore, Kun et al. (2020) indicated that the Pearson correlation coefficient (PCC) is the best fit for multiple linear regression (MLR), while Oyebode (2019) said that inputs chosen using PCC demonstrated higher model accuracy. As a result, the Pearson linear correlation coefficient was used in this investigation (Dehghani et al., 2021; Yuvaraj et al., 2021). It has a value between "+1" and "-1," with "+1" indicating a positive linear correlation, "0" indicating no linear correlation, and "-1" indicating a negative linear correlation (Rania et al., 2000). The PCC is calculated using Equation 1, and the results are shown in Tables 5.1 and 5.2. Positive (0 and 0.3) or negative (0 and 0.3) correlation values indicate a weak linear connection between variables (Ratner, 2009). However, due to the short number of variables and data size, we chose to exclude Borkena station (T_{max}) values, which had r values ranging from (-0.129) to (+0.107), from this analysis. The details are shown in Table 5.1.

$$r = \frac{N\sum XY - (\sum X \sum Y)}{\sqrt{[N\sum x^2 - (\sum X)^2][N\sum Y^2 - (\sum Y)^2]}} \quad (1)$$

After passing strict quality control techniques, the raw data were separated chronologically into training and testing datasets at an 80:20 ratio. Figure 5.2 shows the time series graph and the associated box plot of split data for both stations. There are many options in the literature for removing noise from time series. A sliding window is the first option for temporarily approximating the true value of time series data (Tanbeer et al., 2009). Rolling windows (moving average) is the second option that smooths time series data by computing the average, maximum, minimum, or total over a defined time (Zivot & Wang, 2003). As a result, for this analysis, we used average rolling windows to smooth and eliminate noise from the time series while keeping the data length constant.

The input and output time series was then rebuilt into a supervised learning framework using daily, weekly, and monthly average rolling sliding windows. As a result, for single-step streamflow simulation at the Borkena and UTRB stations, the rolling time series data were produced with a time lag window of 30 or 45. Furthermore, split time series data variable scaling was conducted using Standard Scaler for computational ease and numerical stability of the modelling process.

Table 5. 1 Descriptive statistics of split time series data for the Borkena watershed.

Stations	Data type	Pearson Correlation with streamflow	Training Data (80%)				Testing Data (20%)			
			Mean	Max	Min	SD	Mean	Max	Min	SD
Kombolch	Streamflow (m^3/sec)	1.000	10.9	216.9	0.00	23.2	10.1	94.8	0.0	20.2
	P(mm/day)	0.321	3.1	73.2	0.0	7.5	2.9	60.4	0.0	7.2
	T_{min}(°c)	0.271	12.5	20.9	1.5	3.3	12.5	20.6	2.6	3.4
	T_{max}(°c)	-0.099	27.2	33.6	16.4	2.5	27.3	33.0	19.6	2.1
Chefa	P(mm/day)	0.344	3.5	81.6	0.0	8.6	3.4	64.3	0.0	8.1
	T_{min}(°c)	0.266	13.3	21.5	0.1	3.7	14.1	22.2	3.9	3.5
	T_{max}(°c)	-0.069	29.9	38.0	18.5	2.8	30.3	38.0	22.2	2.5
Dessie	P(mm/day)	0.335	3.5	80.6	0.0	8.6	2.9	67.0	0.0	7.3
	T_{min}(°c)	0.319	8.5	15.5	0.1	2.5	7.8	15.5	0.0	3.1
	T_{max}(°c)	0.107	23.8	30.0	16.0	1.9	24.1	30.0	15.0	2.1
Kemise	P(mm/day)	0.372	3.1	81.9	0.0	8.3	2.9	72.1	0.0	7.5
	T_{min}(°c)	0.282	13.8	22.0	3.0	3.4	13.5	20.1	4.5	3.6
	T_{max}(°c)	-0.129	31.0	38.3	14.0	2.7	31.9	37.8	23.5	2.4
Majete	P(mm/day)	0.347	3.3	80.7	0.0	8.6	3.3	81.3	0.0	8.6
	T_{min}(°c)	0.202	14.7	23.0	1.4	2.9	14.6	21.5	6.7	2.9
	T_{max}(°c)	-0.057	28.6	37.8	17.2	2.8	29.1	38.0	20.8	2.4

Table 5. 2 Descriptive statistics of split time series data for the UTRB.

Stations	Data type	Pearson Correlation with streamflow	Training Data (80%)				Testing Data (20%)			
			Mean	Max	Min	SD	Mean	Max	Min	SD
Ponte Nuovo	Streamflow (m^3/sec)	1.000	50.6	939.0	1.9	75.5	50.6	737.0	3.7	68.6
Castelrigone	P(mm/day)	0.384	2.6	72.8	0.0	6.6	2.7	67.7	0.0	6.9
Montecoronaro	P(mm/day)	0.339	3.9	229.0	0.0	10.7	4.0	110.0	0.0	10.5
Perugia (ISA)	P(mm/day)	0.379	2.4	120.4	0.0	6.6	2.5	61.8	0.0	6.3
	T_{min}(°c)	-0.353	9.7	30.4	-9.0	6.3	9.3	25.2	-5.0	5.6
	T_{max}(°c)	-0.379	17.4	37.4	-4.5	8.1	16.3	33.0	0.6	7.2
Petrelle	P(mm/day)	0.345	2.51	90.0	0.0	6.9	2.7	117.1	0.0	7.4
Pietralunga	P(mm/day)	0.428	3.22	150.0	0.0	8.1	3.1	73.1	0.0	7.3
Spoleto	P(mm/day)	0.412	2.9	113.6	0.0	7.9	2.9	94.2	0.0	7.8
	T_{min}(°c)	-0.265	7.5	23.0	-12.6	6.4	8.8	21.7	-5.4	5.8
	T_{max}(°c)	-0.383	18.8	38.7	-3.5	8.6	18.7	36.8	2.0	7.8
Torgiano	P(mm/day)	0.364	2.4	141.2	0.0	7.1	2.5	62.0	0.0	6.9
Gubbio	T_{min}(°c)	-0.315	8.7	26.0	-12.0	5.9	6.1	19.3	-11.3	5.4
	T_{max}(°c)	-0.377	18.1	39.0	-8.0	8.1	17.4	34.1	-0.9	7.5
Assisi	T_{min}(°c)	-0.325	9.2	25.6	-11.6	6.2	8.2	21.5	-8.0	5.6
	T_{max}(°c)	-0.378	18.2	37.8	-5.0	8.3	18.1	35.8	0.0	7.8

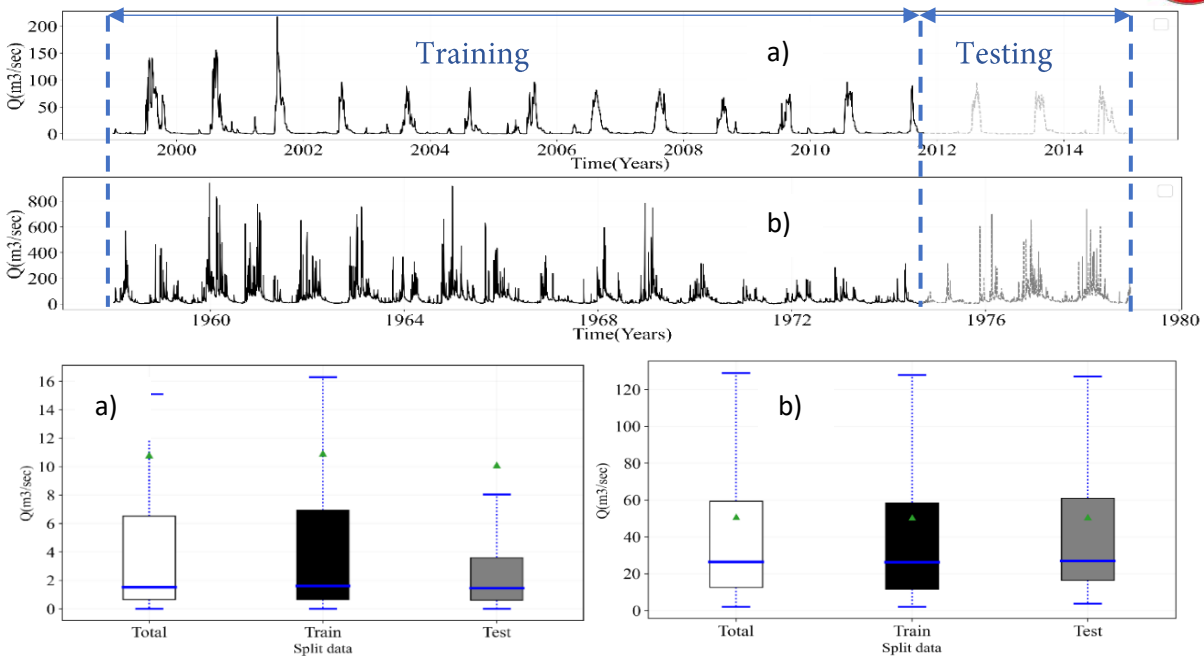


Figure 5. 2 Streamflow time-series graph and the corresponding box plot of split data. (a) Borkena (b) UTRB

5.2.4 Methods

In this study, three network architectures, MLP, GRU, and LSTM, were compared with the proposed hybrid deep neural network architectures CNN-LSTM and CNN-GRU for the simulation of single-step streamflow using various input combinations, such as precipitation (P), minimum temperature (Tmin), and maximum temperature (Tmax). Figure 5.3 depicts a flowchart of the suggested simulation model designs and their input and output variables.

- **Hybrid CNN-LSTM and CNN-GRU Models**

Hybrid models were created in this work by combining CNN with LSTM or GRU layers. As a result, the CNN layer's feature sequence was used as input for the LSTM or GRU layer, and the short and long-term dependences were retrieved. The suggested CNN-LSTM or CNN-GRU models are made up of two major components: single or double convolutional and average pooling layers in one dimension. Furthermore, a flattened layer is attached to transform the data further into the format needed by the LSTM or GRU.

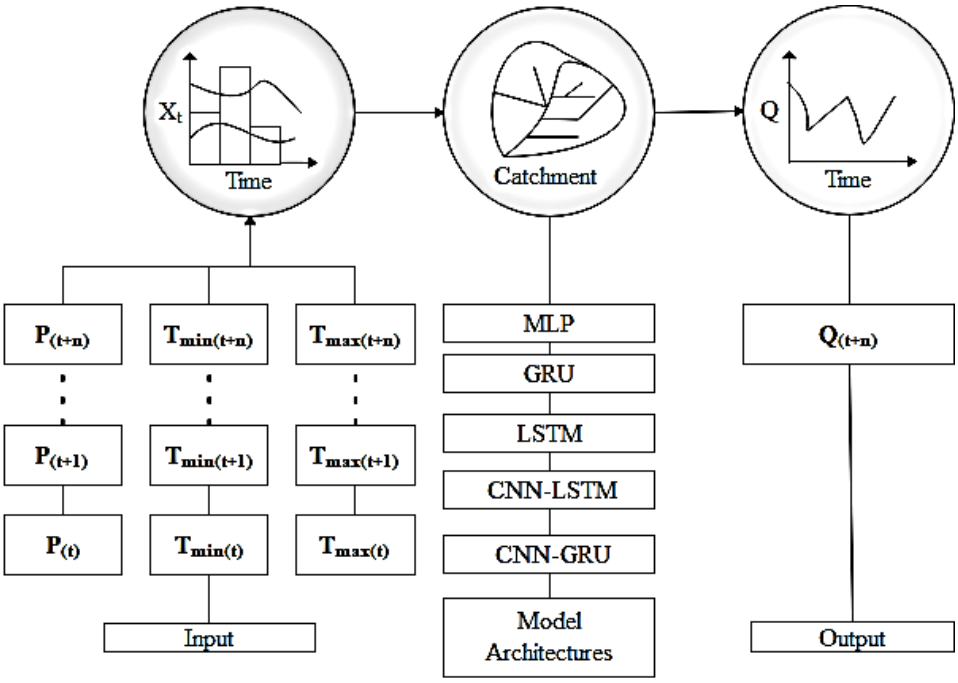


Figure 5. 3 A simple architecture of the proposed models.

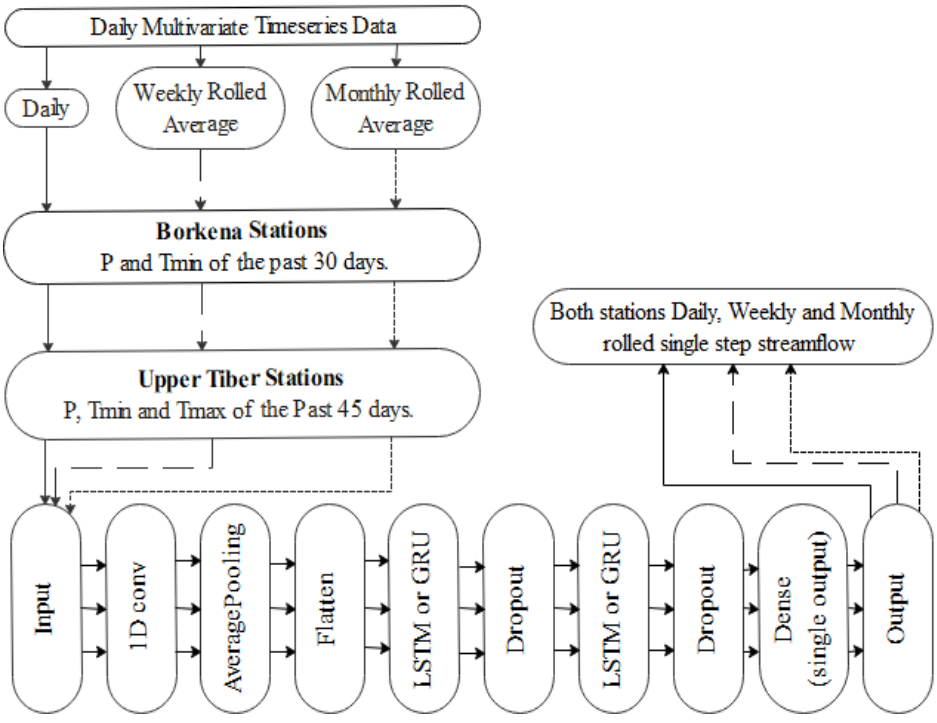


Figure 5. 4 The basic architecture of the proposed CNN-LSTM or CNN-GRU models.

The generated features are processed using LSTM, GRU, and dense layers in the second component. Dropouts are also included to avoid overfitting. Figure 8 depicts the suggested model inputs and outputs, as well as a brief explanation of the proposed convolutional, pooling, and LSTM or GRU layers.

Deep learning simulation necessitates selecting a suitable set of hyperparameters, including batch size, epochs, number of layers, and number of units for each layer (Couta et al., 2019). Since there are no clear rules to follow, optimizing hyperparameters is not always consistent. "It is more of an art than a science" (Essien & Giannetti, 2019). As a result, we selected the Keras Tuner optimizer built by the Google team for this research and included it in the Keras open library (Abdelminaam et al., 2021; Kazakov & Mikheenko, 2020).

- **Optimization of Hyperparameters**

It is vital to fine-tune machine learning model hyperparameters. Changing hyperparameter settings often results in models that behave considerably differently (B. Wang & Gong, 2018). The models used in this research primarily comprise two hyperparameters: constant hyperparameters that do not change throughout the optimization process and variable hyperparameters that change during the optimization process. The Adam optimizer is implemented as a constant hyperparameter because of its efficiency and ease of implementation, which uses little memory and is often used to solve various issues (Boyraz & Engin, 2018). The rectified linear unit (ReLU) is also utilized as an activation function in this category, while the mean squared error (MSE) is employed as a loss function.

On the other hand, the second form of shifting hyperparameters is tuned using a Keras tuner, and hyperparameter selections or value ranges for optimization are determined using various trials. We also assessed the computational capability of our computer (Processor: Intel(R) Core (TM) i7-6500U CPU 2.50 GHz and RAM: 8 Gigabytes) with Windows 10 operating systems. Since deep learning networks have specific training and validation plots for each run, we optimize the model in 20 trials by repeating each iteration three times.

Table 5.3 lists all hyperparameter ranges or options. The CNN-LSTM$_1$ and CNN-GRU$_1$ models were optimized with hyperparameter values ranging from 1 to 13, while the CNN-LSTM$_2$ and CNN-GRU$_2$ optimization omitted values ranging from 4 to 6. The remaining deep learning models, MLP, LSTM, and GRU, used a set of hyperparameters numbered 7 through 13. Finally, each improved hyperparameter is employed for each training and testing trial. Furthermore, each run's training and test traces may be plotted to provide a more robust picture of the model's behavior and to check overfitting and underfitting concerns.

Table 5. 3 Model hyperparameter choices or value ranges for optimization by the Keras tuner.

Nº	Hyperparameters	Value Ranges**			Choices	Default
		Min	Max	Step		
1	Conv_1_filter	8	32	8	*	*
2	Conv_1_kernal	*	*	*	2 or 3	*
3	Conv_1_pool_size	*	*	*	2 or 3	*
4	Conv_2_filter	8	32	8	*	*
5	Conv_2_kernal	*	*	*	2 or 3	*
6	Conv_2_pool_size	*	*	*	2 or 3	*
7	CNN-LSTM$_1$, CNN-LSTM$_2$, CNN-GRU$_1$, CNN-GRU$_2$, LSTM, GRU, or MLP Layer 1 units	5	30	5	*	*
8	Dropout 1	0.0	0.3	0.1	*	0.2
9	CNN-LSTM$_1$, CNN-LSTM$_2$, CNN-GRU$_1$, CNN-GRU$_2$, LSTM, GRU or MLP Layer 2 units	5	30	5	*	*
10	Dropout 2	0.0	0.3	0.1	*	0.2
11	Learning rate	*	*	*	1e-2, 1e-3 or 1e-4	*
12	Number of epochs	10	100	10	*	*
13	Number of batch sizes	10	100	10	*	*

*** Value ranges or choices for optimization by Keras tuner: (Objective = "Validation Loss", Max Trials = 20, Executions Per Trial = 3)*
** Not applicable*

5.3. Results and Discussion

Tables 5.4, 5.5, and 5.6 show the results of streamflow simulations using the suggested seven deep learning architectures, different input time window series, two climatic areas, two input combinations, and three average rolling time windows. Regardless of how these circumstances were combined, the CNN-GRU model produced promising outcomes in most instances (Tables 5.4 and 5.6). The top scores are shown here.

1. In daily streamflow simulation, CNN-GRU_1 scored 7.94, 3.66, 0.85, and 0.63 for Borkena station and 45.61, 21.79, 0.57, and 0.64 for UTRB station for RMSE, MAE, R^2, and training time per epoch, respectively.
2. In the weekly rolled streamflow simulation, CNN-$LSTM_2$ scored 7.33, 3.86, 0.87, and 0.52 for the Borkena station, and GRU scored 25.21, 14.83, 0.77, and 5.56 in the UTRB catchment for RMSE, MAE, R^2, and training time per epoch, respectively.
3. In monthly rolled streamflow simulation, the CNN-GRU_2 model performed well, with RMSE, MAE, R^2, and training time per epoch scores of 5.15, 3.18, 0.92, 0.78 for Borkena station and 17.98, 12.99, 0.83, 0.71 for UTRB station, respectively.

Table 5. 4 Daily streamflow simulation and performance comparison of the proposed models for different input variables and climatic conditions.

MODEL 1	Borkena								UTRB							
	P+Tmin				P				P+Tmin+Tmax				P			
	RMSE	MAE	R^2	TTPE * (sec)	RMSE	MAE	R^2	TTPE* (sec)	RMSE	MAE	R^2	TTPE * (sec)	RMSE	MAE	R^2	TTPE* (sec)
MLP	9.91	5.01	0.77	0.89	9.38	4.63	0.79	0.63	49.11	22.74	0.49	0.78	56.57	28.14	0.33	0.41
GRU	8.78	4.37	0.82	3.61	7.94	3.64	0.85	3.32	46.63	20.89	0.55	2.61	51.09	26.74	0.45	3.39
LSTM	8.41	4.09	0.83	2.35	9.65	4.87	0.78	2.92	48.64	22.79	0.51	3.86	48.59	25.00	0.51	5.98
CNN-$LSTM_1$	8.09	4.07	0.84	0.46	8.57	4.67	0.82	0.41	51.20	22.95	0.45	1.19	56.16	26.55	0.34	0.57
CNN-$LSTM_2$	7.99	4.09	0.85	0.72	9.14	4.50	0.80	0.45	45.38	21.85	0.57	0.82	51.57	25.84	0.44	1.85
CNN-GRU_1	7.94	3.66	0.85	0.63	8.32	4.09	0.83	0.86	55.06	23.49	0.37	1.16	52.42	24.98	0.43	0.83
CNN-GRU_2	9.07	4.19	0.80	1.01	8.43	4.26	0.83	0.28	45.61	21.79	0.57	0.64	49.96	25.38	0.48	0.68

**TTPE(Training Time per Epochs)* *The grey cell indicates the highest performance score*

Table 5. 5 Weekly rolled streamflow simulation and performance comparison of the proposed models for different input variables and climatic conditions.

MODEL 2	Borkena								UTRB							
	P+Tmin				P				P+Tmin+Tmax				P			
	RMSE	MAE	R^2	TTPE * (sec)	RMSE	MAE	R^2	TTPE* (sec)	RMSE	MAE	R^2	TTPE * (sec)	RMSE	MAE	R^2	TTPE* (sec)
MLP	8.11	4.29	0.84	0.23	7.33	4.19	0.87	0.22	33.01	20.01	0.60	0.74	38.03	25.17	0.47	0.79
GRU	7.59	3.71	0.86	2.04	7.15	4.13	0.87	2.43	25.21	14.83	0.77	5.56	31.39	19.26	0.64	16.79
LSTM	8.41	4.01	0.82	2.98	7.93	3.91	0.84	1.27	31.07	18.87	0.65	3.55	31.07	19.49	0.65	2.69
CNN-$LSTM_1$	7.90	4.09	0.85	0.78	7.72	4.25	0.85	0.63	28.04	17.33	0.71	0.93	34.57	21.92	0.57	0.62
CNN-$LSTM_2$	7.33	3.86	0.87	0.52	7.63	4.25	0.86	0.55	28.45	16.66	0.71	1.14	35.04	21.77	0.55	1.56
CNN-GRU_1	7.83	3.94	0.85	0.44	7.91	4.31	0.85	0.50	30.57	18.01	0.66	2.32	35.14	22.58	0.55	0.63
CNN-GRU_2	8.73	4.61	0.81	0.43	8.43	4.35	0.82	0.97	27.81	16.99	0.72	4.37	33.76	23.01	0.59	1.01

**TTPE(Training Time per Epochs)* *The grey cell indicates the highest performance score*

Table 5. 6 Monthly rolled streamflow simulation, performance comparison of the proposed models for different input variables and climatic conditions.

MODEL 3	Borkena								UTRB							
	P+Tmin				P				P+Tmin+Tmax				P			
	RMSE	MAE	R^2	TTPE * (sec)	RMSE	MAE	R^2	TTPE* (sec)	RMSE	MAE	R^2	TTPE * (sec)	RMSE	MAE	R^2	TTPE* (sec)
MLP	6.68	4.37	0.87	0.58	5.57	3.80	0.91	0.41	20.24	13.84	0.78	0.44	28.79	21.05	0.56	0.41
GRU	5.15	3.52	0.91	1.62	5.22	3.06	0.92	3.31	20.79	14.30	0.77	16.63	26.47	20.08	0.63	4.70
LSTM	5.55	3.49	0.91	2.75	5.76	3.51	0.90	2.51	21.49	15.11	0.76	4.15	32.29	24.47	0.45	5.09
CNN-$LSTM_1$	6.05	4.42	0.89	0.98	5.58	3.40	0.91	0.58	21.53	14.87	0.76	1.29	27.48	21.19	0.60	0.42
CNN-$LSTM_2$	5.36	3.17	0.92	1.41	6.87	4.05	0.86	1.44	19.07	13.53	0.81	0.70	27.79	20.90	0.59	0.42
CNN-GRU_1	5.76	3.62	0.90	0.52	5.77	3.56	0.90	0.69	19.31	13.78	0.80	4.87	28.67	21.07	0.57	3.08
CNN-GRU_2	5.36	3.25	0.92	0.62	5.15	3.18	0.92	0.78	17.98	12.99	0.83	0.71	27.77	20.36	0.59	1.22

**TTPE(Training Time per Epochs)* *The grey cell indicates the highest performance score*

Furthermore, among the four suggested hybrid models, CNN-GRU_2, or the model constructed by a single 1D CNN layer, yielded the most promising results on trial models 1 (UTRB) and 3, as shown in Tables 5.4 and 5.6. In contrast, the second-most promising result was shared by GRU on model 2 (UTRB), CNN-$LSTM_2$ on model 2 (Borkena), and CNN-GRU_1 on model 1 (Borkena). Streamflow simulation using the CNN-GRU_2 model outperformed the other studied hybrid deep learning and state-of-the-art LSTM, GRU, and MLP models. Following our aims, the outcome is detailed in the following paragraphs under various variable situations.

Variability in Climatic Regions:- Testing models in various climatic circumstances with historical data will almost certainly result in robust deep-learning models for streamflow modelling in the future (P. Bai et al., 2021). As a result, this study investigated multiple models in two climatic zones. The CNN-GRU model performed best on the tested case study locations regardless of climatic and time frame fluctuations.

Variability in input combination: - In the Borkena station, the input combination of minimum temperature (Tmin) and precipitation (P) does not show a substantial performance improvement (Tables 5.4 and 5.5). Sometimes, using simply P as an input improves the model's performance (Table 5.6). For UTRB, however, streamflow simulation using all input variables or Tmin, Tmax, and P demonstrated considerable performance improvements (Table 5.6).

Variability of the average rolling time frame: Streamflow modelling without rolling daily time series data performed poorly compared to monthly rolled average time series. This might be because the time series noise in UTRB is more noticeable than in Borkena station. Consequently, the performance increase from daily to monthly rolled window models in UTRB is substantially greater than that in Borkena station.

The monthly rolling time frame with the CNN-GRU_2 model generally produced the best results at both stations (Table 5.6). Figure 5.7 depicts our improved high-score hybrid model's associated training and test loss functions for both stations. As a result, Figure 5.8 contrasts this model's actual and forecasted values. The optimized hybrid model outperforms the GRU and LSTM models in terms of performance and training time per epoch. Tables 5.7 and 5.8 show the model, input feature, and Keras tuner optimal hyperparameter settings for each station, along with the MSE score. Figures 5.5 and 5.6 also depict the internal network topologies of these models, as well as the model input and output parameter matrices for each layer.

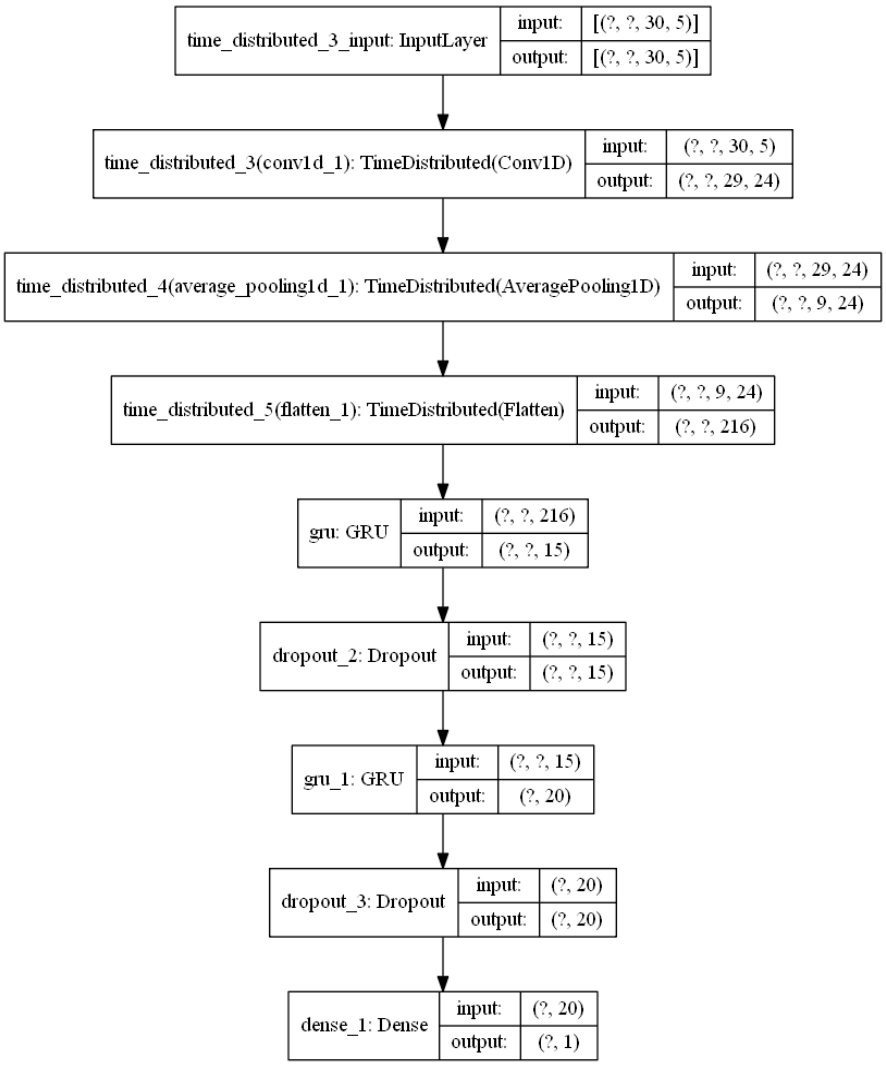


Figure 5. 5 Internal network structure of the optimized high score hybrid CNN-GRU_2 model for Borkena Station.

Table 5. 7 Best hybrid model type, input feature, and Keras tuner optimized hyperparameter values for Borkena station with its MSE score.

	CNN-GRU_2
Hyperparameters:	Monthly Rolled P
Conv_1_filter:	24
Conv_1_kernal:	2
Conv_1_pool_size:	3
GRU_l1_units:	15
Dropout1:	0.1
GRU_l2_units:	20
Dropout2:	0.2
Learning rate:	0.0001
Number of epochs:	80
Number of batch sizes:	20
Score (MSE):	0.083

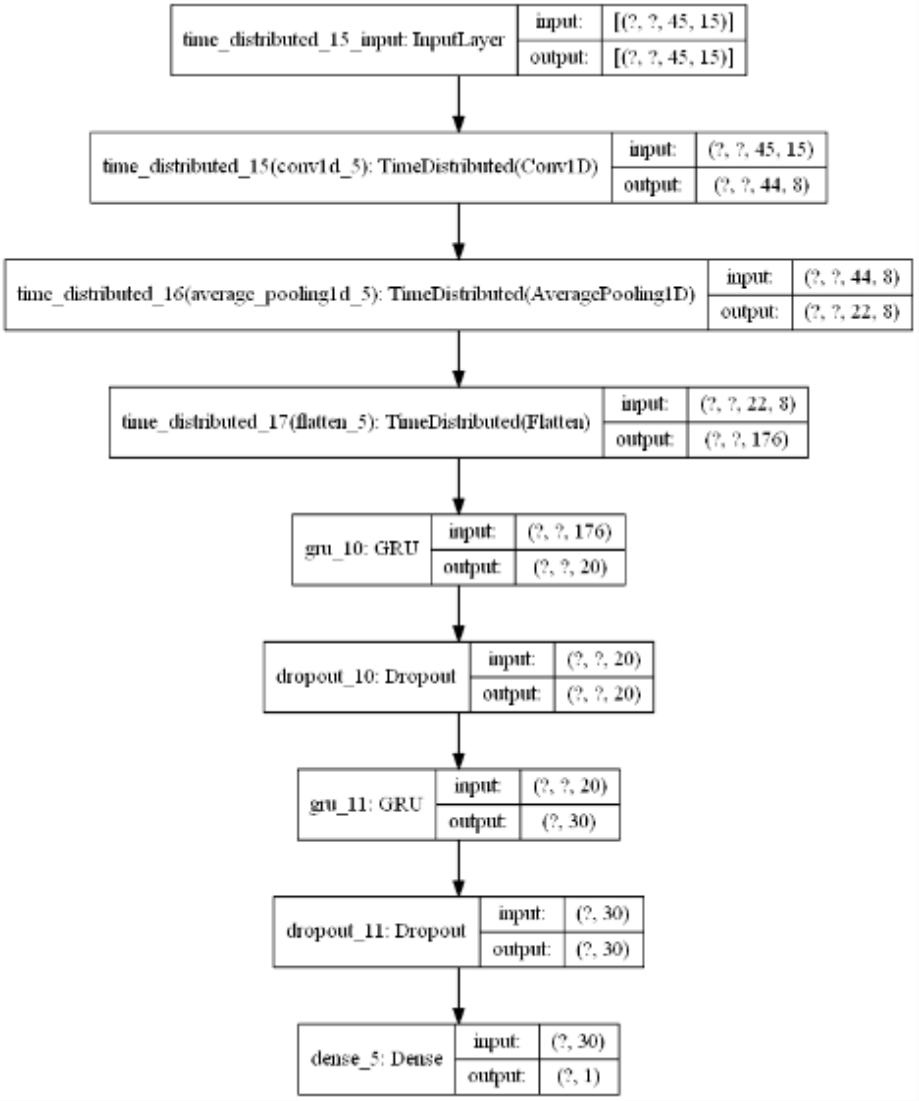


Figure 5. 6 Internal network structure of the optimized high score hybrid CNN-GRU_2 model for UTRB Station.

Table 5. 8 Best hybrid model type, input features, and Keras tuner optimized hyperparameter values for the UTRB station with its MSE score.

	CNN-GRU_2
Hyperparameters:	Monthly Rolled P, T_{min} and T_{max}
Conv_1_filter:	8
Conv_1_kernal:	2
Conv_1_pool_size:	2
GRU_l1_units:	20
Dropout1:	0.3
GRU_l2_units:	30
Dropout2:	0.2
Learning rate:	0.0001
Number of epochs:	60
Number of batch sizes:	40
Score (MSE):	0.193

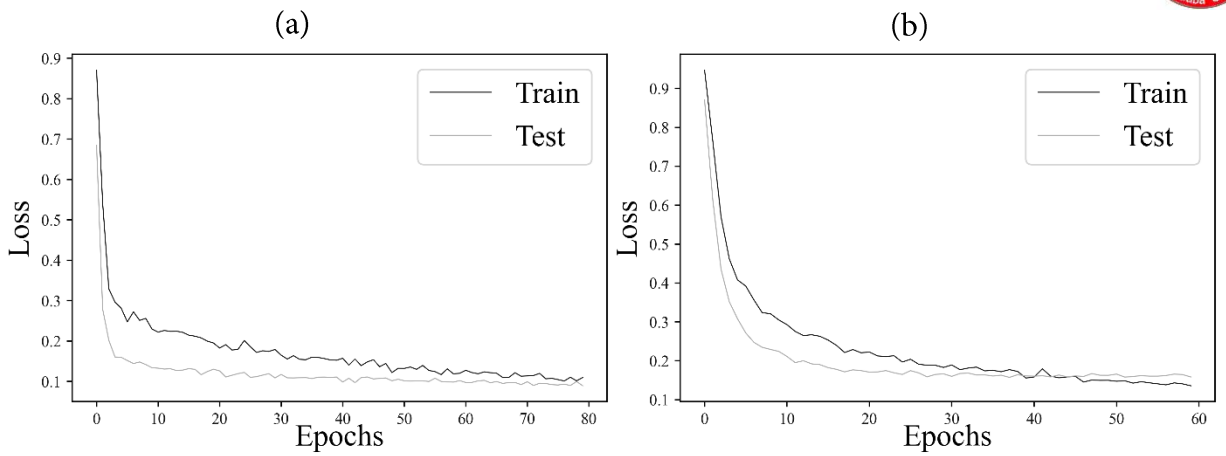
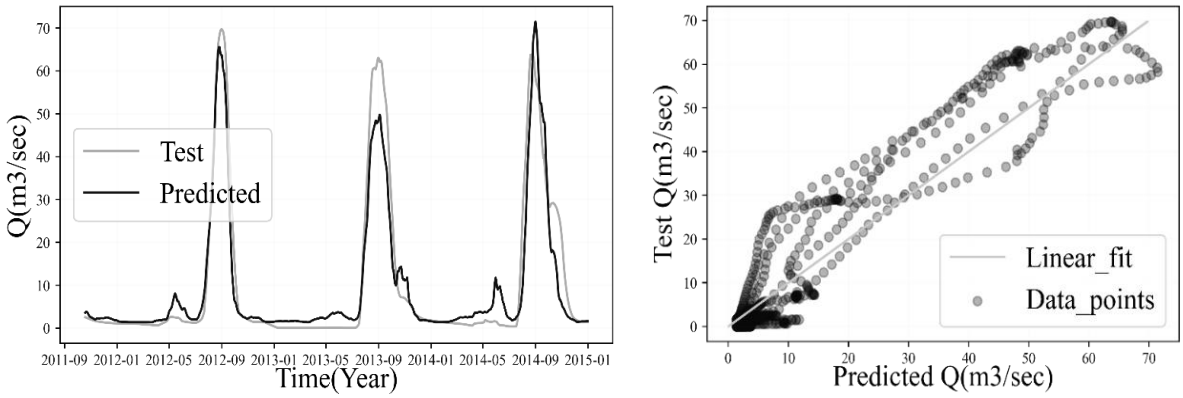


Figure 5. 7 Training and test loss function of the optimized high score hybrid model. (a) CNN-GRU_2 model for Borkena Station. (b) CNN-GRU_2 model for UTRB Station.

(a)



(b)

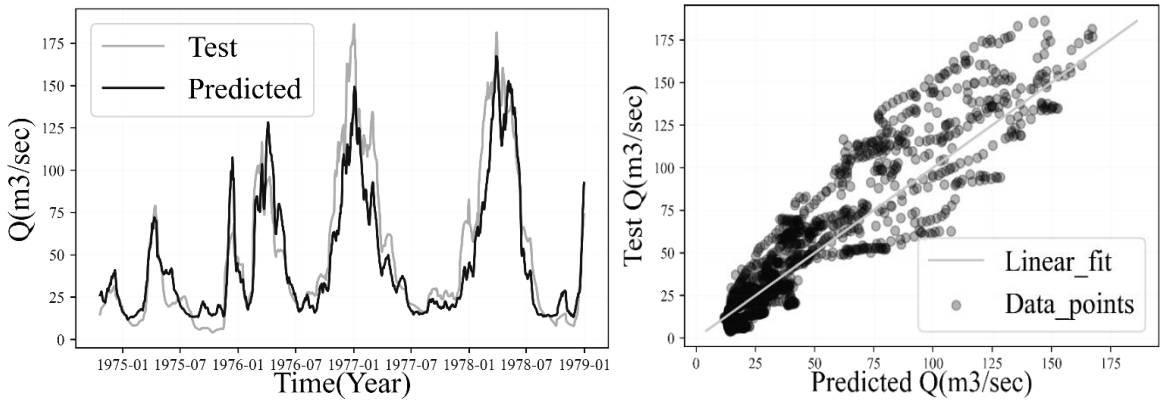


Figure 5. 8 Comparison of true values and predicted values of the optimized high score hybrid model. (a) CNN-GRU_2 model for Borkena Station. (b) CNN-GRU_2 model for UTRB Station.

5.4. Conclusions

This research compared several hybrid deep learning algorithms with cutting-edge machine learning models for one-step daily streamflow modelling at two river basins or subcatchment streamflow outlets. The CNN-LSTM and CNN-GRU hybrid deep learning models suggested in this study are designed with one or two 1D CNN layers, including the standard MLP, LSTM, and GRU models. This work performed a series of tests to examine the performance variation of the suggested models for streamflow simulation by adding various input combinations, rolling time frames, and climatic variables. The following list of points will describe the study's key results.

I. CNN-GRU_2 with one 1D CNN layer outperformed all other models in both case study regions, having the lowest RMSE, MAE, and R^2. Such findings imply that the performance of the chosen topologies is independent of the basins' meteorological features.

II. Combining temperature data with precipitation as input and entering it into the suggested models resulted in a lower performance increase in Borkena than in the UTRB case study region, indicating that temperature data scarcity has a more significant impact on performance loss at the UTRB station. The Borkena station, on the other hand, has more natural streamflow variability than UTRB, which is reflected in the model findings. This indicates that catchment response should be considered before using deep learning models.

III. Using the proposed models, the rolled input time series window significantly improves streamflow simulation performance in the UTRB compared to the Borkena station.

IV. The analysis results also show that the training time per epoch for the hybrid deep learning models is much lower than that for the GRU and LSTM models.

Deep learning models often need enormous datasets, and their performance suffers when provided with small to medium-sized datasets. However, based on this case study's acceptable findings and hybrid models' hyperparameter sensitivity and complexity, future research may build better configurations. Furthermore, they may put these hybrid models to the test for long-term streamflow modelling in ephemeral, seasonal, and perennial river systems, as well as in other research areas. As part of our ongoing research, we want to use a mix of neural networks, decision trees, and boosting algorithms to simulate streamflow utilizing a remote sensing-derived data product (precipitation and vegetation indices).

CHAPTER 6:

SUPER ENSEMBLE BASED STREAMFLOW SIMULATION USING MULTI-SOURCE REMOTE SENSING AND GROUND GAUGED RAINFALL DATA FUSION

Abstract[3]

Traditionally, data-driven streamflow estimates use a single model that performs inconsistently under varied variability situations. Model ensembles, or combining the advantages of many models without compromising the overall nature of the data, are increasingly popular in hydrology. This research compared three super ensemble learners to eight base models. Twelve years of monthly rolled daily time series data from three Ethiopian rivers (Borkena watershed: Awash River basin, Gummera watershed: Abay River basin, and Sore watershed: Baro Akobo River basin) are utilized for single-step daily streamflow simulation utilizing prior thirty-day input timesteps. Three vegetation indices, three remote sensing-based precipitation products, ground-gauged rainfall, all fused inputs, and chosen inputs using the recursive feature elimination (RFE) technique are deployed in five input scenarios. The time series is then separated into 80:20 training and testing datasets. The root mean squared error (RMSE), mean absolute error (MAE), median absolute error (MEDAE), and coefficient of determination (R^2) were used to assess the performance of the suggested models. Finally, the outcome is shown along with the five input scenarios. The average performance of the catchment and input scenarios revealed that the three superensemble learners outperformed the eight base models with generally steady performance. The top-ranked WASE model outperformed the linear regression baseline by 13.3%. The base models with the best performance were XGB, CNN-GRU, and LSTM. This research also demonstrated that the main disadvantage of LSTM is its poor performance in the absence of feature selection criteria. In contrast, XGB outperformed after managing redundant inputs internally. Furthermore, this work is the first to demonstrate the use of remote sensing-based vegetation indices in the science of data-driven streamflow modelling for nongauged catchments with no meteorological time series.

[3] *This chapter is based on the publication*- https://www.sciencedirect.com/science/article/pii/S2405844023051903

6.1. Introduction

Water resources are becoming more vital due to population growth, industrialization, and climate change-related floods and droughts. Each country develops a national water resource management plan to guarantee the proper management of water resources. Understanding changes in discharge data is critical for implementing various water resource management programs, such as integrated water resource management (D. Y. Kim & Song, 2020).

Hydrological time series are essential for planning and designing water resource infrastructure. The lack of such data, especially ground streamflow monitoring, jeopardizes the effectiveness of development activities. Currently, streamflows are monitored at river gauge stations. However, numerous studies show that most of the world's gauging station records are becoming rare (A. Sichangi et al., 2018). Tourian et al. (2013) created a time series depiction of the number of stations with accessible discharge data using publicly available data from the Global Runoff Data Centre (GRDC). This dataset shows a decrease in total annual streamflows observed between 1970 and 2010. Furthermore, since most stations have been decommissioned, poor discharge monitoring has become a severe problem in developing countries (A. W. Sichangi et al., 2016). Thus, a study into the robustness of discharge data estimates is vital and forward-thinking, especially for Ethiopia.

Simulating streamflow using ground meteorological datasets is another option. However, this is not always possible due to a lack of necessary meteorological data. Given these limits, the research suggests that remote sensing (RS) data might be feasible (Gamage et al., 2011). Rainfall estimation using remote sensing with various data sources (satellite, gauge, radar, analysis, or reanalysis), broad spatial coverage (from continental to fully global), spatial resolution (from 0.05° to 2.5°), repeatable temporal coverage (from 30 minutes to monthly), temporal span (from 1 to 115 years), and latency (from 3 hours to several years) has emerged as an adequate response to global data scarcity over the last two decades (Beck et al., 2017; Bui et al., 2019).

Numerous studies have been undertaken to determine the advantages and disadvantages of these P datasets (Maggioni & Massari, 2018; Q. Sun et al., 2018). Several studies also used independent gauge observations (C. Li et al., 2018; Macharia et al., 2020), and others compared their spatiotemporal patterns (Zhu et al., 2021; Yu et al., 2020).

There were also comparisons between predicted and observed river discharges using remote-sensing precipitation products (Meresa, 2019; Alquraish & Khadr, 2021). Beck et al. (2017) conducted the most thorough examination of a global-scale P dataset. They compared 13 non-gauge-corrected P datasets to 76086 daily P gauge observations from throughout the globe. By calibrating a hydrological model for 9053 catchments (50,000 km^2) spread across the globe, nine more gauge-corrected P datasets were evaluated. According to the same research, MSWEP V2.0 is an excellent option since it has a long track record (1979-2020), global coverage, high temporal and spatial precision (0.3 hours and 0.1 km), daily gauge adjustments, and top-ranked performance in all climate types. Assume that a daily temporal resolution is sufficient for tropical environments. In such an instance, CHIRPS V2.0 may be a suitable alternative as long as the peak magnitude is underestimated and spurious drizzle is not as significant as it used to be (Beck et al., 2017). Pradhan et al. (2022) further proved IMERG's greater capacity to recreate spatial and temporal patterns and severe precipitation variability compared to other satellite products.

Using remote sensing spectral data, scientists have also evaluated indices for quantitatively and subjectively measuring vegetation cover, stamina, and growth dynamics. Vegetation Indices(VI) were derived using a range of aircraft and satellite platforms; over 100 VIs are now in use (Tahsin et al., 2018). The normalized difference vegetation index (NDVI), the normalized difference water index (NDWI), the modified normalized difference vegetation index (MNDVI), and the enhanced vegetation index (EVI) have all made significant contributions to hydrological research as input data (Mushore et al., 2019). However, there has been little research into the potential use of RS indices to appropriately represent various hydrometeorological factors and utilize them as input for various data-driven or machine learning-based streamflow estimations.

Due to extrinsic factors such as precipitation, underlying surfaces, and evaporation, streamflow time series are significantly nonstationary, nonlinear, and intricate. For trustworthy engineering systems, precise and reliable streamflow modelling is essential. Streamflow simulation techniques are often classified as process-driven or data-driven (Zhao et al., 2021). Process-driven model simulation accuracy depends on a large amount of physical process data, making modelling difficult.

Data-driven models need less data than process-driven models and are relatively more efficient in data-scarce places (Yaseen et al., 2015). Data-driven solutions are preferred when exact estimates predominate physical interpretations (Nourani, Molajou, Uzelaltinbulat, et al., 2019). Two kinds of data-driven models exist: traditional and artificial intelligence (AI) (Sharma & Machiwal, 2021). Conventional models, such as multiple linear regression (MLR), linear regression (LR), autoregressive moving average with the eXogenous term (ARMAX), and autoregressive integrated moving average (ARIMA), have a fundamental structure and are frequently employed in hydrological forecasts (Z. Zhang et al., 2018). Traditional models for time series presuppose a linear correlation structure, which streamflow data rarely exhibit. As a result, streamflow simulation necessitates a robust nonlinear and optimized modelling approach (Zhao et al., 2021).

Regularities and patterns are used in AI-based data-driven models to develop models with excellent performance and minimal complexity (Mosavi et al., 2018). The advancement of machine learning (ML) techniques over the past two decades has shown their suitability for streamflow simulations (Xu & Liang, 2021). In terms of prediction accuracy, ML models outperform traditional statistical models (Prodhan et al., 2022). Artificial Neural Networks (ANNs), Support Vector Machine (SVM), Fuzzy sets, Evolutionary Computation (EC), and Wavelet-Artificial Intelligence (W-AI) models are the most often employed AI approaches in hydrologic research (Yaseen et al., 2015). Zounemat-Kermani et al. (2020) thoroughly evaluated research advances over the previous two decades, the current development, and the potential for employing machine learning in many aspects of hydrological sciences in the future. The same research indicates that neurocomputing simulation models should be hybridized and merged with other soft computing approaches to increase performance and overcome constraints.

ML performance was improved by combining ML methodology with soft computing techniques, numerical simulations, and physical models. These applications resulted in more durable and efficient models for dealing with complex flood systems (Mosavi et al., 2018). According to a recent study, hybrid neurocomputing models perform better on various tasks (Wegayehu & Muluneh, 2021; Zounemat-Kermani, Mahdavi-Meymand, et al., 2021). There is also room for advancement in the science of machine learning, with a focus on future hybrid AI modelling, which will make hydrological research even more fascinating, demanding, and rewarding for academics (Ibrahim et al., 2022).

Along with hybrid models, ensemble machine-learning models in hydrology have grown dramatically in recent years and are gaining traction (Zounemat-Kermani, Batelaan, et al., 2021). One ML model may outperform another for comparable datasets, but the results will generally vary for various datasets. The ensemble approach was designed to take advantage of each model without losing the data's general nature, taking each model's (base learner) output as input, with an arbitrator assigning a priority level (Nourani et al., 2021).

Nourani et al. (2021) used numerous source satellite and ground-gauged rainfall datasets to research ensemble rainfall-runoff modelling for Gilgel-Abay, Ethiopia. The researchers conclude that input fusion from several satellite rainfall products is a feasible alternative for accurate rainfall-runoff modelling in unmeasured or weakly gauged catchments. Laan et al. (2007) proposed a super learning ensemble strategy that optimizes the weights of the base learners by minimizing a loss function given the cross-validated output of the learners. Superlearning determines the ideal weight matrix for the learners and guarantees that their performance is at least as excellent as that of the best individual learner (S. Young et al., 2018). A diversified set of base learners is essential for optimal performance and generalization in an ensemble. Because the component weights are set for the task, the super learner adapts to various problems given a wide range of base learners. Depending on the task or computer resources, the base learner set may also be selected (S. Young et al., 2018).

Tyralis et al. (2019) used superensemble learning to estimate one step ahead of streamflow by merging ten machine learning methods. They developed the model using a large ground dataset consisting of a 10-year time series of daily streamflow, precipitation, and temperature from 511 basins. The superlearner outperforms other regression approaches in terms of performance. However, further research is needed to evaluate superensemble learning using multiple data assimilation strategies, combinations of base models, and customized meta-learners. Many satellite-based sensors with high spatial and temporal resolutions are available for wall-to-wall runoff and erosion imaging, but their effectiveness in runoff modelling is uncertain. Researchers should also design integrative methodologies that may be employed in every situation and provide maximum accuracy (Mushore et al., 2019).

Following a thorough review of previous research, we intended to run a single-step streamflow simulation using a super ensemble and fivefold cross-validation. This research uses various remote sensing datasets, including precipitation (IMERG-final, CHIRPS, and MSWEP-V2), vegetation indices (NDVI, NDWI, and EVI), and ground gauge rainfall data from three Ethiopian river basins. Due to the computational expense of training models and the declining rewards in performance, the number of base models in the ensemble is often kept to a minimum. Ensembles of three, five, or ten trained models are typically used. Consequently, we used neural networks, hybrid models, decision trees, and boosting algorithms to develop a set of base learner algorithms (GRU, LSTM, MLP, CNN-GRU, SVR, Lasso, XGB, LR). We tested three different meta-learners (extra tree regression (ETR), Bayesian model averaging (BMA), and weighted average (WA)) to learn from the output of the base models.

To the best of the authors' knowledge, this is new research analysing a combination of vegetation indices as input variables with gauged and satellite precipitation data, using a hybrid model (CNN-GRU) as a base learner and integrating a range of different base learners. Furthermore, the research investigated streamflow estimation using five input fusion approaches for the first time.

6.2. Data and Methods

This research concentrated on three Ethiopian river basin subcatchments, the Borkena subcatchment in the Awash River basin (Figure 6.1), the Gummera subcatchment in the Abay River basin (Figure 6.2 (a)), and the Sore subcatchment in the Baro Akobo River basin (Figure 6.2 (b)).

6.2.1 Borkena Catchment (Awash River Basin, Ethiopia)

One of Ethiopia's 12 main river basins is the Awash River Basin. The basin originates in the central Ethiopian highlands and flows northeast for 1200 kilometres until it meets Lake Abe on the Djibouti-Ethiopia border. Mountains in the highlands reach 4195 meters and 210 meters in the plains. The yearly rainfall varies from 1600 mm at the source northeast of Addis Abeba to 160 mm towards the northern border. The temperature fluctuates from 19 to 23°C, with the warmest months being May and June. The Borkena River basin in the Wollo highlands receives substantial runoff from the Upper Awash Basin, and the rainfall in this watershed is unimodal. The major rainy season (Kiremt) lasts from July to September, accounting for more than 80% of the total rainfall, whereas the rest of the year (Bega: October-January) is primarily dry.

6.2.2 Gummera Catchment (Abay River Basin/Ethiopia)

The second case study region is Ethiopia's Gummera subbasin, one of Lake Tana's main tributaries in the Abay River basin. The Lake Tana subbasin encompasses approximately 15,114 km^2. On the Earth's surface, it covers a latitude range of 10.95° N to 12.78° N and a longitude range of 36.89° E to 38.25° E. Its elevations vary from 914 to 4096 masl. The lake is situated in the northwest highlands and receives drainage from approximately 40 rivers. The primary watersheds in this subbasin are Gummera, Gilgelabay, Megech, and Ribb. The Gummera watershed is situated southeast of Lake Tana and has a drainage area of approximately 1,592 km^2, with elevations ranging from 1,788 to 3,750 masl. The Gummera River originates in the Guna highlands southeast of Debre Tabor, at a height of approximately 3250 meters above sea level. The land is primarily agricultural, with hilly, rugged, dissected topography and steep slopes in the upper and central parts. Rainfall is unimodal, with a single peak quantity occurring from July through August.

6.2.3 Sore Catchment (Ethiopia's Baro Akobo River Basin)

The Baro-Akobo River basin is situated on the southwestern Ethiopian Plateau, with a maximum height of 3240 masl. The undulating and mountainous geography quickly transitions to a long succession of abrupt escarpments and lowland plains to the west, with the lowest elevation of 395 masl. The basin has the second largest runoff (23.6 Bm^3) among the 12 basins in Ethiopia, with 75,912 km^2 (6.9 % of the country). The principal rivers are the Baro, Alwero, Gilo, and Akobo. These rivers originate in the eastern highlands, flow west to the Gambela plain, and drain north into the Machar Marshes across the Ethiopia-Sudan border.

The Sore watershed is a subcatchment of the Baro Akobo River basin. It covers an area of 1711 km^2 and has elevations ranging from 2661 to 1547 masl, with the highest elevations being to the east and south. The annual rainfall ranges from 1804 mm to 2020 mm. The monthly high temperature varies from 24°C to 28°C, while the monthly low temperature ranges from 12°C to 14°C. A hydrological gauging station near Mettu was employed in this investigation.

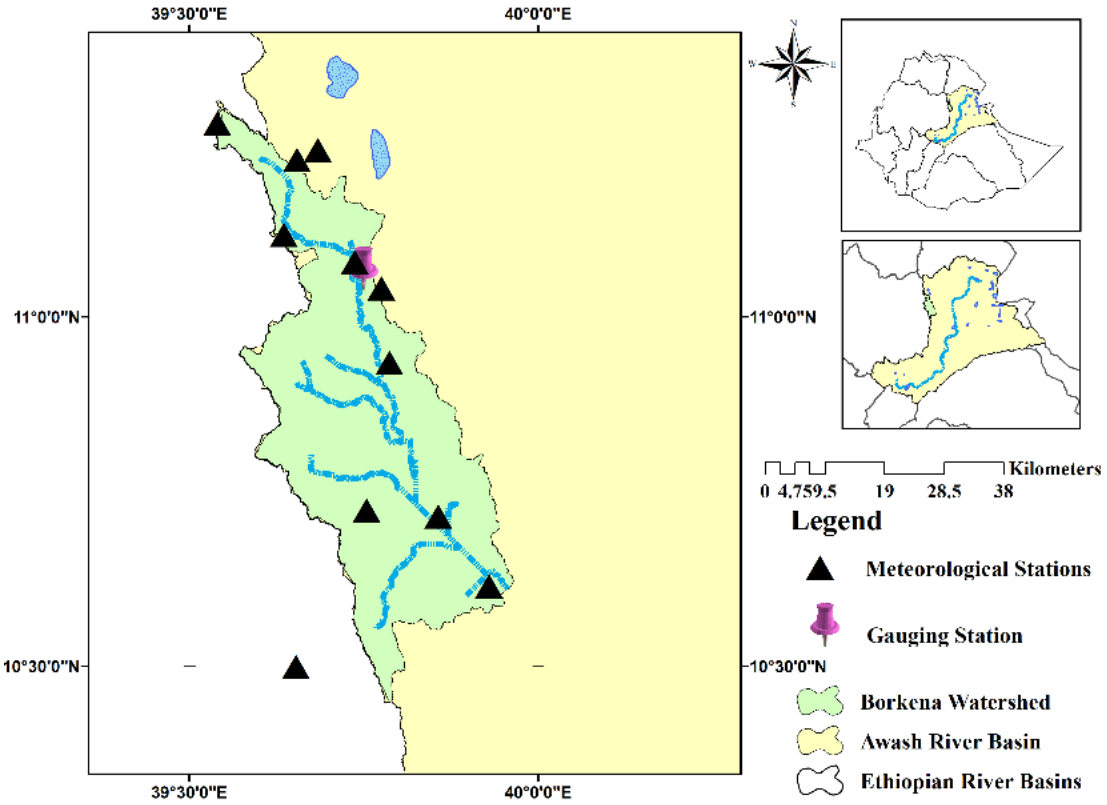


Figure 6. 1 The location of case study area one (Borkena: Awash River Basin).

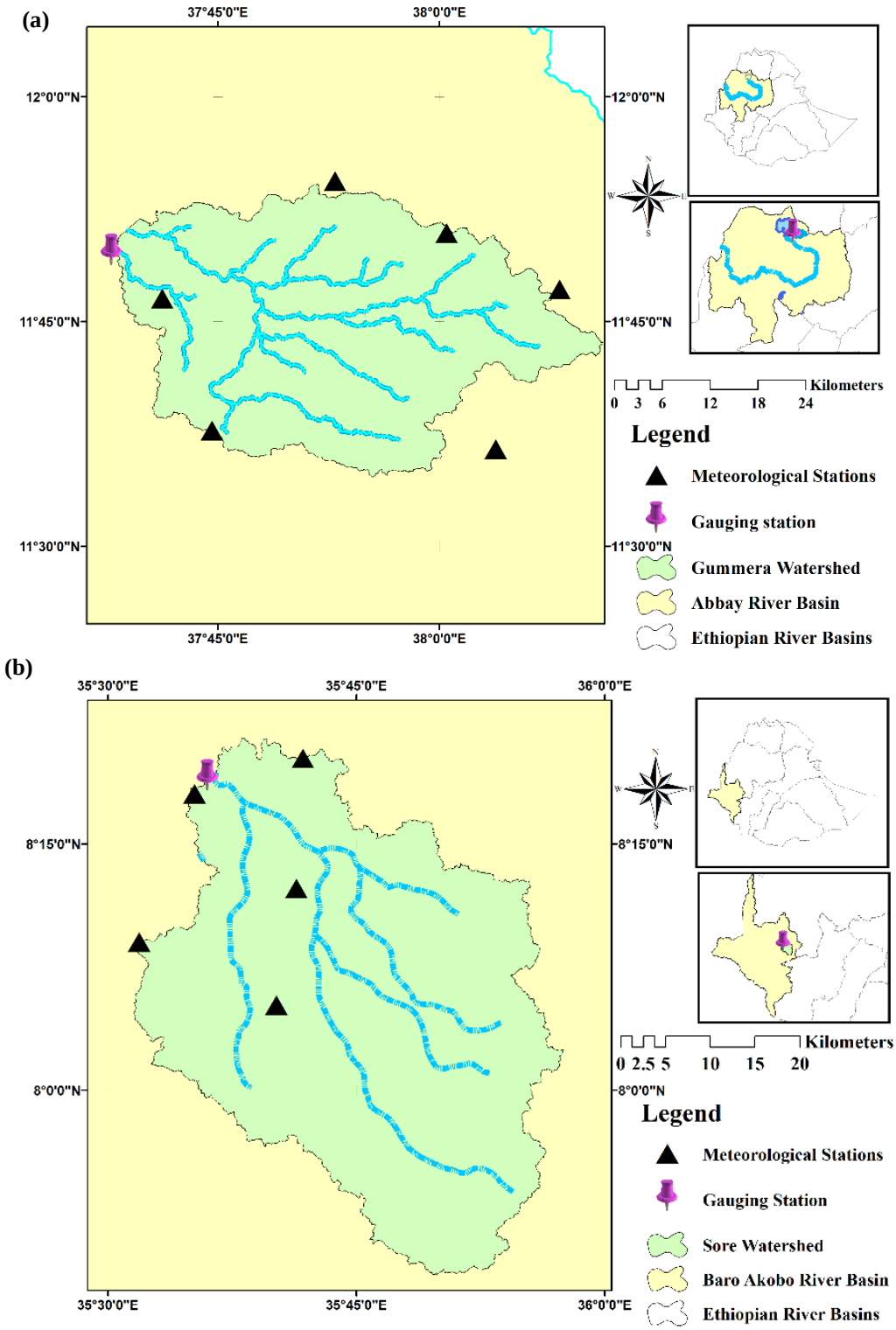


Figure 6. 2 The location of case study areas two and three. (a) Gummera: Abay River Basin, (b) Sore: Baro Akobo River Basin.

6.2.4 Data

- **Ground Datasets**

This study's ground data collection contains daily rainfall and streamflow data from 2003 to 2014. The first ten years of data were utilized for training, and the latter two years were utilized for model testing. The Ethiopian Ministry of Water and Energy (MOWE) provided the three gauging station streamflow time records utilized in this investigation. The Ethiopian National Metrological Agency (NMA) rainfall datasets were collected from all accessible meteorological stations in the three catchments. Table 6.1 displays the available meteorological and hydrological stations and the time series statistical features for each watershed.

Table 6. 1 The available meteorological and hydrological stations and their time series statistical properties for each case study area.

Catchment	Station	Maximum	Minimum	Mean	Standard deviation
Borkena	Chefa (mm)	81.6	0.00	3.52	8.51
	Desi (mm)	80.6	0.00	3.29	8.13
	Kemise (mm)	81.9	0.00	3.11	7.88
	Kombolcha (mm)	73.2	0.00	3.14	7.71
	Majeti (mm)	81.3	0.00	3.23	8.31
	Flow at gauging station (m^3/sec)	95.98	0.00	9.47	18.68
Gummera	Amed Ber (mm)	91	0.00	3.79	8.49
	Debre Tabor (mm)	115	0.00	4.06	8.41
	Wanzaye (mm)	134.2	0.00	3.96	9.82
	Arb Gebey (mm)	70	0.00	2.89	6.61
	Mekan Eyesus (mm)	79.6	0.00	3.57	7.37
	Flow at gauging station (m^3/sec)	306.73	1.77	43.65	67.53
Sore	Gore (mm)	71.8	0.00	4.63	8.16
	Flow at gauging station (m^3/sec)	267.99	0.97	51.15	52.79

- **Remote Sensing Based Precipitation Data (RSP)**

This research uses three remote sensing-based precipitation data products: IMERG-final, CHIRPS, and MSWEP-V2. Table 6.2 displays the meteorological stations in each watershed that were utilized to construct all RS-based precipitation time series products using Python code and the time series statistical properties.

Table 6. 2 The meteorological stations used for remote sensing precipitation data generated for each of the three case study areas and their time series statistical properties.

Catchment	Station	Maximum			Mean			Standard deviation		
		IMERG	CHIRPS	MSWEP	IMERG	CHIRPS	MSWEP	IMERG	CHIRPS	MSWEP
Borkena	Harbu	76.19	67.93	50.44	2.74	2.55	2.67	7.38	6.77	5.84
	Boru	73.32	86.52	63.94	2.13	2.93	2.49	6.2	8.01	5.73
	Kutaber	75.76	82.43	70.19	2.14	2.99	2.44	6.39	8.19	5.86
	Sulula	73.32	86.52	63.94	2.13	2.93	2.49	6.2	8.01	5.73
	Mekoy	107.78	95.87	77.31	3.2	2.67	2.69	8.4	7.09	5.93
	Jimate	134.94	53.18	69.93	3.26	2.41	2.02	8.91	6.46	5.51
	Ancharo	83.33	67.29	51.56	2.47	2.74	2.74	6.79	7.39	6.15
	Kombolcha	83.33	67.2	51.56	2.47	2.57	2.74	6.79	6.86	6.15
	Desie	68.56	75.83	62.56	2.27	2.99	2.56	6.35	8.16	5.93
	Rabel	85.22	74.2	129	2.84	3.09	3.03	7.47	8.49	7.01
	Kemise	127.3	58.51	63.38	3.17	2.5	2.52	8.53	6.65	5.27
Gummera	Amed Ber	107.41	77.41	49.31	2.86	3.51	3.07	6.65	8.39	5.29
	Arb Gebey	67.68	88.14	43.19	2.82	2.69	3.59	6.53	7.3	5.41
	Debre Tabor	81.14	128.83	43.25	3.28	4.22	3.39	6.8	10.24	5.29
	Gasay	81.33	110.77	67.13	3.04	3.96	3.78	6.49	10.01	7.54
	Mekan Eyesus	92.31	110.14	46.63	3.03	3.57	3.53	6.69	8.96	5.49
	Wanzaye	62.04	71.29	40.69	2.62	3.82	3.54	6.38	7.76	5.4
Sore	Becho	86.77	57.91	75.5	3.99	4.94	4.67	8.06	8.13	6.01
	Gore	81.19	99.76	91.62	3.94	5.46	4.32	8.24	8.68	5.77
	Hurumu	88.31	57.48	93.06	3.95	4.78	4.58	8.25	8.06	5.89
	Leka	176.76	83.98	89.94	4.26	5.15	5.07	8.85	8.55	6.47
	Metu	74.76	73.82	79.75	3.73	4.73	4.21	7.69	8.19	5.63

N.B. The minimum precipitation at all stations and data products is zero

- **Remote Sensing-Based Vegetation Indices (VI)**

We carefully analyse the essential RS indices from the different possibilities available in the literature for hydrological investigations. Consequently, we recommended three widely utilized VIs (NDVI, NDWI, and EVI). The NDVI is computed from each near-IR and red band scene as (NIR - red)/(NIR + red). The NDWI is sensitive to changes in the liquid water content of plant canopies and employs two infrared bands (approximately 840-860 nm (NIR) and 1630-1660 nm (IR)) in a formula very similar to the NDVI or (NIR - IR)/(NIR + IR). Furthermore, EVI is a vegetation index that uses MODIS near-IR, red, and blue (B) surface reflectance to improve the vegetation signal in regions with high biomass (Solymosi et al., 2019). The VI values range from -1.0 to 1.0. The average catchment VI time series for each three-case study region was derived using MODIS/MYD09GA surface reflectance composites and the Google Earth Engine (GEE) code editor.

6.2.5 Methods

This is the first study to investigate super ensemble learning using three meta-learners (extra tree regression (ETR), Bayesian model averaging (BMA), and weighted average (WA)) and eight base models (GRU, LSTM, MLP, CNN-GRU, SVR, Lasso, XGB, LR) for one-step daily streamflow simulation using three remote sensing-based vegetation indices (NDVI, NDWI, EVI) and precipitation products (IMERG-final, CHIRPS, and MSWEP-V2). Figure 6.3 depicts the basic flow chart of the proposed five input situations, eight base models, and three meta-learners. Consequently, we comprehensively analyse 360 scenarios for three Ethiopian study locations. Furthermore, we applied a monthly rolling average to all input time series, and the streamflow time series was simulated in a single time step using data from the previous 30 days.

- **Ensemble Learning**

The concept of ensemble learning stretches back to the eighteenth century. During a livestock show, Sir Francis Galton (1822-1911), an English philosopher and statistician, devised a weight-guessing challenge. The participants had to predict the weight of an ox. Hundreds of people participated in this challenge, but no one correctly predicted the weight: 1,198 pounds. Surprisingly, all estimates' averages came near the exact weight: 1,198 pounds. Galton used multiple guesses in this experiment to obtain an accurate forecast (Sagi & Rokach, 2018). The ensemble learning theory, explored in the literature by Bates & Granger (1969) and Wallis (2011), lends even more credence to using forecast combinations. In particular, Zounemat-Kermani, Batelaan, et al. (2021), in hydrology, analysed over 160 peer-reviewed scientific works published during the previous two decades. This study demonstrates ensemble learning in various hydrological domains, including surface hydrology, hydrogeology, and extreme hydrological events. The same research concludes by demonstrating the ability of ensemble machine learning models to effectively tackle all hydrological issues and promote them as the main choice for challenging hydrological problems. Ensemble learning makes use of a variety of techniques, each with its own framework. This includes stacking, averaging, bagging, and boosting (Zounemat-Kermani, Batelaan, et al., 2021). Stacking generalization is used in this investigation.

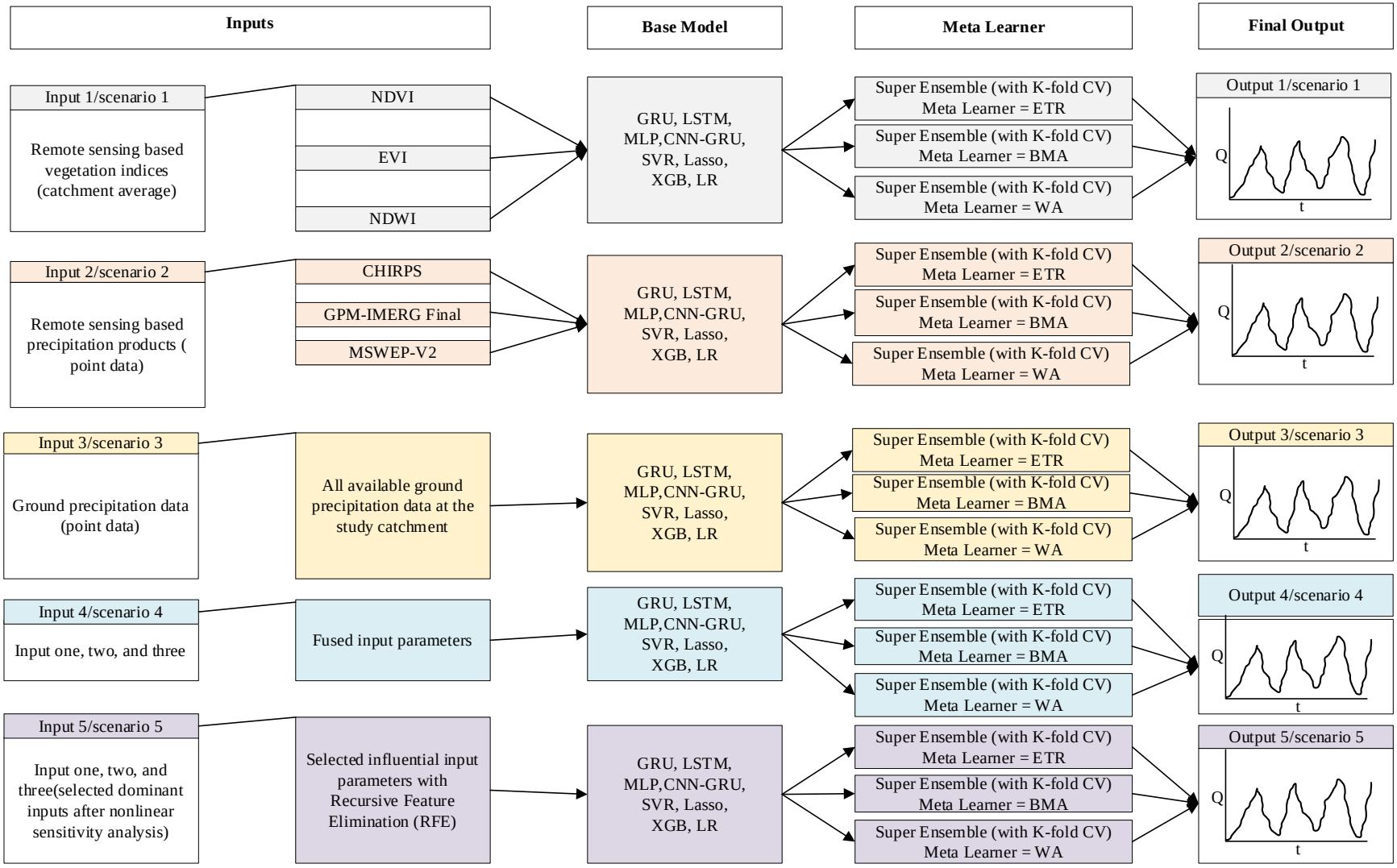


Figure 6. 3 The basic flow chart of the proposed five input scenarios, eight base models, and three meta-learners.

- **Base learners**

The ensemble's strategic components or individual learners are known as base learners. Most ensemble techniques generate homogeneous base learners with a single learning algorithm, while some generate heterogeneous learners with numerous learning algorithms. As there is no standard basis learning method, different researchers mix base learners based on their goals and computing power (Zhou, 2009). Consequently, we combine many different algorithms in our research, such as a decision tree, a boosting approach, and a neural network. In chapter two, we review all the base learners used in this research, including LR, LASSO, SVR, GRU, LSTM, XGB, MLP, and CNN-GRU.

- **Superensemble**

Laan et al. (2007), from the University of California, Berkeley, introduced the superlearner algorithm in their work "Superlearner," published in a biology journal. Other disciplines, notably hydrology, have not yet wholly accepted this notion (Baćak & Kennedy, 2019; Sinisi et al., 2007; Tyralis et al., 2019). Superlearner is a cross-validation-based ensemble strategy for merging base learners that yields predictions as least as excellent as the best single base learner. The superlearner approach exemplifies "stacked generalization," also known as "stacking." The method starts by predefining the training data's k-fold split, which is fitted to the chosen base learners. All base model simulations are saved and utilized to train the meta-model and optimally combine the simulations (Baćak & Kennedy, 2019; Naimi & Balzer, 2018; E. C. Polley et al., 2011; E. Polley & Laan, 2010). The meta-learner is usually a linear model; however, in this work, we employed extra tree regression (ETR), Bayesian model averaging (BMA), and weighted average (WA) as meta-learners. These meta-learners are discussed in the following paragraphs. Figure 6.4 depicts the super ensemble process in further detail.

- **Extra Tree Regression (ETR)**

Geurts et al. (2006) developed the extra tree regression (ETR) approach, which is based on the random forest (RF) model. This approach uses a meta-estimator to fit numerous randomized decision trees (also known as extra trees) on distinct subsamples of the dataset. It uses averaging to improve the projected accuracy and control overfitting. It can typically outperform the random forest technique; however, it employs a simpler algorithm to create the ensemble's decision trees.

- **Bayesian model averaging (BMA)**

When we pick one model over another, we may reach conclusions that are overly confident of ourselves and make riskier actions because we neglect the model's uncertainty in favor of certain distributions and assumptions about that model. As a result, it would be beneficial to model this source of uncertainty so that appropriate models may be selected or merged. Bayesian inference has been proposed as a framework for achieving these objectives, and Bayesian model averaging (BMA) is an extension of traditional Bayesian inference techniques (Hinne et al., 2020). It is feasible to obtain posterior distributions for model parameters and the model itself using Bayes' theorem, allowing for direct model selection, combined estimation, and prediction.

- **Weighted average (WA)**

When producing ensemble predictions, average-weighted ensembles presume that some models in the ensemble perform better than others and give them more significant credit. This meta-learner is superior to voting ensembles, which presume that all models are equally excellent and contribute equally to the ensemble's predictions.

In this strategy, each model is given a set weight multiplied by the base learner prediction value and used to examine the total or average prediction. Even if it is difficult to determine how to compute, assign, or seek model weights that contribute to improved performance in this ensemble learning, we suggest base learner prediction R^2 values as a weight parameter in our work.

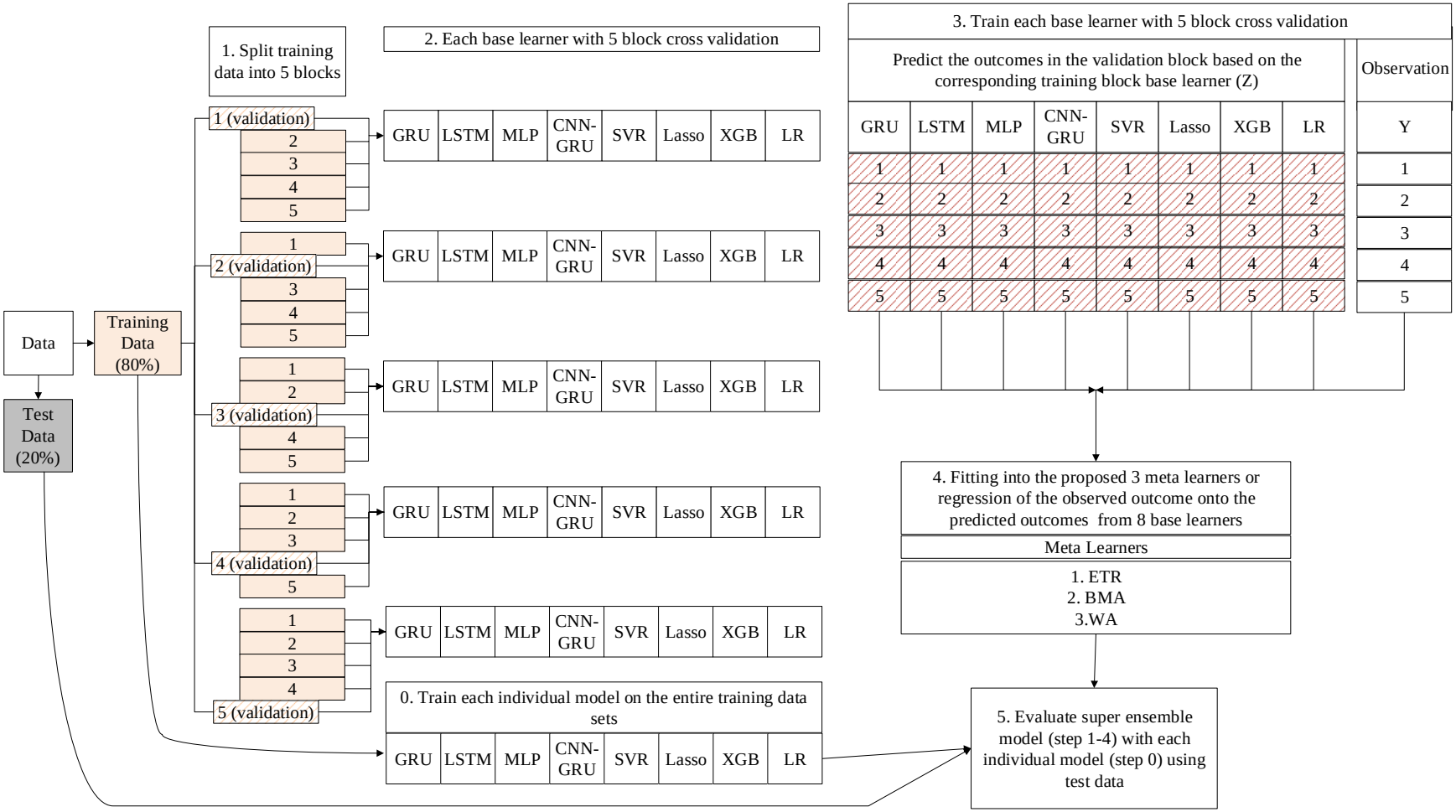


Figure 6. 4 Data analysis flow diagram of the modified super ensemble learner with three meta-learners (Laan et al., 2007).

- **Model Development**

In model development, the hyperparameter values are the model's key, impacting its accuracy; they are also the model's external configuration, which should be specified before model training. To properly train the base models in ensemble learning, hyperparameters such as the number of layers, batch size, number of epochs, and learning rate must be suitably tuned. After considering all options, we selected Keras Tuner, a computationally efficient random search technique developed by the Google team, from among the different optimization methodologies for hyperparameter tuning in the literature: grid, random, trial-and-error, and probabilistic approaches. The hyperparameter value range is fixed and described in Table 6.3 below to optimize the Keras tuner effectively.

Table 6. 3 Model hyperparameter choices or value ranges for optimization by the Keras tuner.

Nº	Hyperparameters	Value Ranges**			Choices	Default
		Min	Max	Step		
1	Conv_1_filter	8	32	8	*	*
2	Conv_1_kernal	*	*	*	2 or 3	*
3	Conv_1_pool_size	*	*	*	2 or 3	*
4	GRU, LSTM, MLP, CNN-GRU, Layer 1 units	5	40	5	*	*
5	Dropout 1	0.0	0.3	0.1	*	0.2
6	GRU, LSTM, MLP, CNN-GRU, Layer 2 units	5	30	5	*	*
7	Learning rate	*	*	*	1e-2, 1e-3 or 1e-4	*
8	Number of epochs	10	100	10	*	*
9	Number of batch sizes	10	100	10	*	*

N.B. For XGB, we used five hyperparameters, the ranges of which are as follows: - Base score (0.25,0.5,0.75,1), Estimator (100, 500, 900, 1100, 1500), Maximum depth (2, 3, 5, 10, 15), Booster (gbtree), Learning rate (0.05,0.1,0.2,0.3), and minimal child weight (1,2,3,4). While for SVR, the values are C (0.1,1, 10, 100), gamma (1,0.1,0.01,0.001), kernel (rbf), and for LASSO: alphas (0, 1, 0.01), Number of alphas (200).

*** Value ranges or choices for keras tuner: (Objective = "Validation Loss", Max Trials = 30, Executions Per Trial = 1)*

** Not applicable*

6.3. Results and Discussion

6.3.1 Modelling with solely remote sensing-based indices

Remote sensing technologies opened up new possibilities for studying surface water dynamics. In contrast to conventional in situ observations, it can continually monitor the Earth's surface at several scales and record track changes at regular and frequent intervals (Huang et al., 2018).

The importance of remote sensing in continuous streamflow dataset generation has been well established and broadly recognized. M L Tan (2014) classifies the function of remote sensing in streamflow time series estimation into two categories. Remote sensing data may be used as "input" for a hydrological model, or they can be utilized directly to predict streamflow without needing a hydrological model (M L Tan, 2014). Precipitation, evapotranspiration, temperature, and soil moisture are all popular independent variables for data-driven hydrological modelling using deep learning. Previous research has demonstrated that vegetation is vital in runoff modelling. However, research proving the use of vegetation indices in streamflow modelling, especially in deep learning, is limited.

This work used three common MODIS vegetation indices (NDVI, EVI, and NDWI) to predict single-step streamflow in three Ethiopian river basin subcatchments using eight base machine learning models and three super ensemble meta-learners. Table 6.4 shows the model's performance indices, with R^2 as the significant criterion and the other indices as the secondary criterion. The table also shows the average performance on the three catchments, with BMASE as the best model and WASE and LSTM in second and third place, respectively. The weakest two models (LASSO and LR) performed 16% worse than the top ensemble models, whereas LSTM outperformed the base models. Furthermore, the agro-climatic variability of the catchments showed considerable performance variations, with Gummera having the greatest performance score and Borkena having the lowest. The results reveal a link between streamflow generation and vegetation indices. Figure 6.5 also shows the mean dispersion of prediction residuals for eleven models over the test period using a box plot and a time series graph of predicted versus test values for the optimized, high-score BMASE model. The red circles show that the vegetation index input data did not capture the high flows.

Table 6. 4 The individual and average performance of eleven algorithms in three subcatchments using vegetation indices as input and displayed with a heatmap.

	Borkena				Gummera				Sore				Average				
	RMSE	MAE	MEDAE	R^2	RMSE	MAE	MEDAE	R^2	RMSE	MAE	MEDAE	R^2	RMSE	MAE	MEDAE	R^2	RANK
BMASE	13.35	6.63	2.36	0.6	37.34	24.65	14.76	0.76	30.97	20.72	11.39	0.69	27.22	17.33	9.5	0.683	1
WASE	13.71	6.74	2.31	0.57	37.37	24.27	13.89	0.76	30.38	21.76	14.71	0.71	27.15	17.59	10.3	0.68	2
LSTM	13.31	7.79	4.32	0.59	38.77	23.28	9.71	0.74	32.07	19.79	8.85	0.67	28.05	16.95	7.63	0.667	3
ETRSE	14.1	6.52	1.45	0.55	38.51	22.25	9.53	0.74	31.66	19.34	7.72	0.68	28.09	16.04	6.23	0.657	4
SVR	13.74	9.33	6.27	0.57	39.98	30.41	24.57	0.72	32.73	22.81	15.75	0.66	28.82	20.85	15.53	0.65	5
CNN-GRU	15.54	7.27	2.13	0.45	38.03	22.29	8.59	0.75	30.23	19.09	8.52	0.71	27.93	16.22	6.41	0.637	6
XGB	14.6	6.57	1.52	0.52	41.45	25.11	10.96	0.7	31.97	20.66	10.19	0.67	29.34	17.45	7.56	0.63	7
MLP	15.39	8.45	3.64	0.46	39.81	23.78	8.76	0.72	31.31	20.04	9.64	0.69	28.84	17.42	7.35	0.623	8
GRU	14.6	6.48	1.23	0.52	37.1	24.31	12.91	0.76	36.55	24.15	12.63	0.57	29.42	18.31	8.92	0.617	9
LR	13.89	9.73	6.04	0.56	45.36	34.77	29.35	0.64	44.41	36.74	34.58	0.37	34.55	27.08	23.32	0.523	10
LASSO	13.99	9.86	6.16	0.56	45.24	34.68	28.92	0.64	44.44	36.8	34.51	0.37	34.56	27.11	23.2	0.523	11

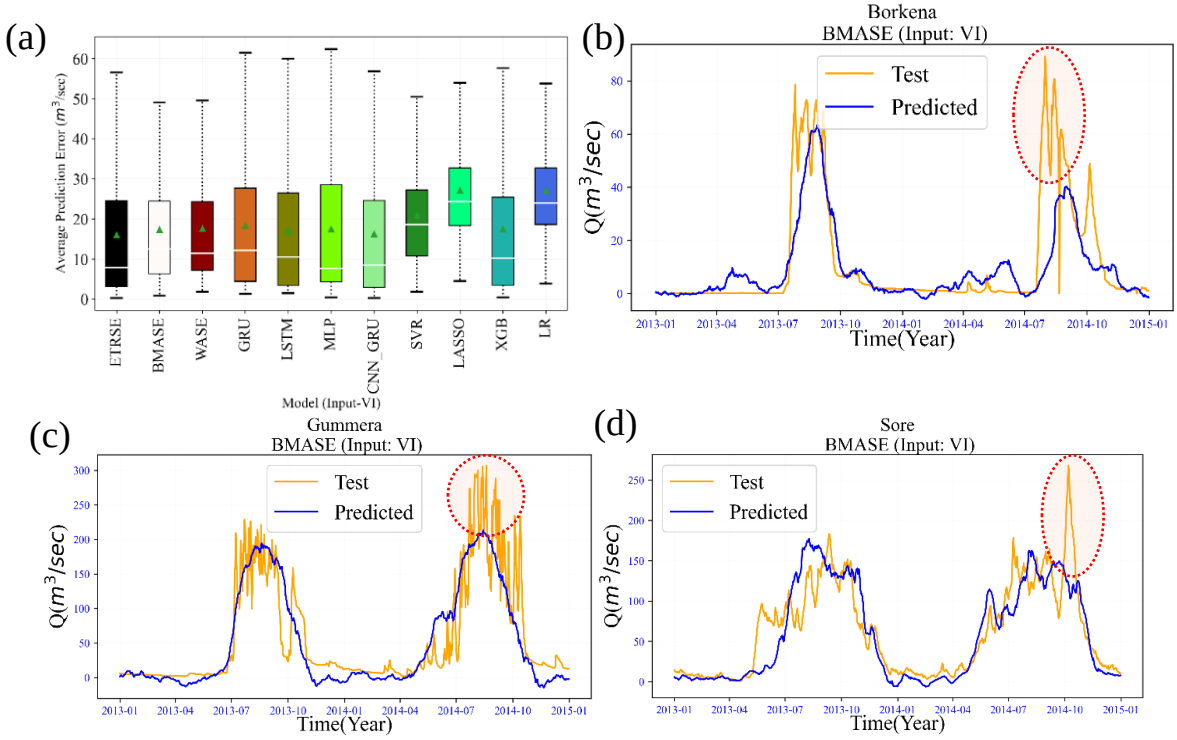


Figure 6. 5 Mean spread of prediction error (m^3/s) or box plot for the 11 models during the test period (a) and time series graph of actual values and predicted values of the optimized BMASE model (b) Borkena, (c) Gummera, and (d) Sore catchments.

6.3.2 Modelling with Solely Remote Sensing-Based Precipitation Products

The fundamental driver of the hydrologic cycle and the most crucial input for hydrological models is precipitation (Beck et al., 2017). Reliable and consistent precipitation measurements are necessary for accurate hydrological models (Jiang & Wang, 2019). Due to the poor spread of rain gauges and the fact that many weather stations have inadequate or inconsistent historical records of observations globally, high-quality ground weather data are challenging to obtain (Macharia et al., 2020; Nourani et al., 2021). Remote sensing has the potential to augment meteorological data. In this work, we selected the best-performing remote-sensing precipitation products from the literature, considering a wide range of accessible gridded P datasets (Beck et al., 2017). Consequently, we investigated three fused high-performance remote sensing P products (IMERG-final, CHIRPS, and MSWEP-V2) as inputs to the eleven algorithms investigated in this work.

Remote sensing precipitation products demonstrated a few mean performance increments compared to the prior vegetation index input scenario, as shown in Table 6.5, with ETRSE earning the highest score, followed by LSTM and WASE.

Table 6. 5 The individual and average performance of eleven algorithms in three subcatchments using remote sensing precipitation products as input and displayed with a heatmap.

	Borkena				Gummera				Sore				Average				
	RMSE	MAE	MEDAE	R^2	RMSE	MAE	MEDAE	R^2	RMSE	MAE	MEDAE	R^2	RMSE	MAE	MEDAE	R^2	RANK
ETRSE	9.44	4.51	1.23	0.8	37.07	21.72	9.18	0.76	36.00	23.92	11.89	0.58	27.5	16.72	7.43	0.713	1
LSTM	11.12	5.05	0.83	0.72	39.92	23.99	11.26	0.72	30.53	21.45	14.94	0.70	27.19	16.83	9.01	0.713	2
WASE	10.21	5.23	2.08	0.76	43.82	26.72	13.84	0.67	33.11	23.36	14.12	0.65	29.05	18.44	10.01	0.693	3
XGB	9.84	4.49	1.21	0.78	39.55	22.4	9.37	0.73	37.81	26.24	15.99	0.54	29.07	17.71	8.86	0.683	4
BMASE	8.95	4.33	1.61	0.82	42.16	25.33	12.98	0.69	38.02	26.00	13.71	0.54	29.71	18.55	9.43	0.683	5
SVR	10.61	7.28	5.43	0.74	42.81	27.60	15.53	0.68	34.88	23.31	11.47	0.61	29.43	19.40	10.81	0.677	6
GRU	12.53	5.72	0.81	0.64	37.27	22.09	8.57	0.76	34.20	23.69	12.93	0.62	28.00	17.17	7.44	0.673	7
MLP	11.02	5.79	2.52	0.72	44.09	25.45	9.36	0.66	33.83	23.21	11.89	0.63	29.65	18.15	7.92	0.67	8
CNN-GRU	11.29	5.23	0.97	0.71	44.81	26.82	11.05	0.65	35.99	23.90	11.21	0.58	30.7	18.65	7.74	0.647	9
LASSO	12.07	7.73	4.06	0.67	51.36	33.86	18.28	0.54	36.43	26.57	18.15	0.57	33.29	22.72	13.5	0.593	10
LR	12.29	7.99	4.48	0.65	51.47	33.93	18.59	0.54	36.46	26.64	18.51	0.57	33.41	22.85	13.86	0.587	11

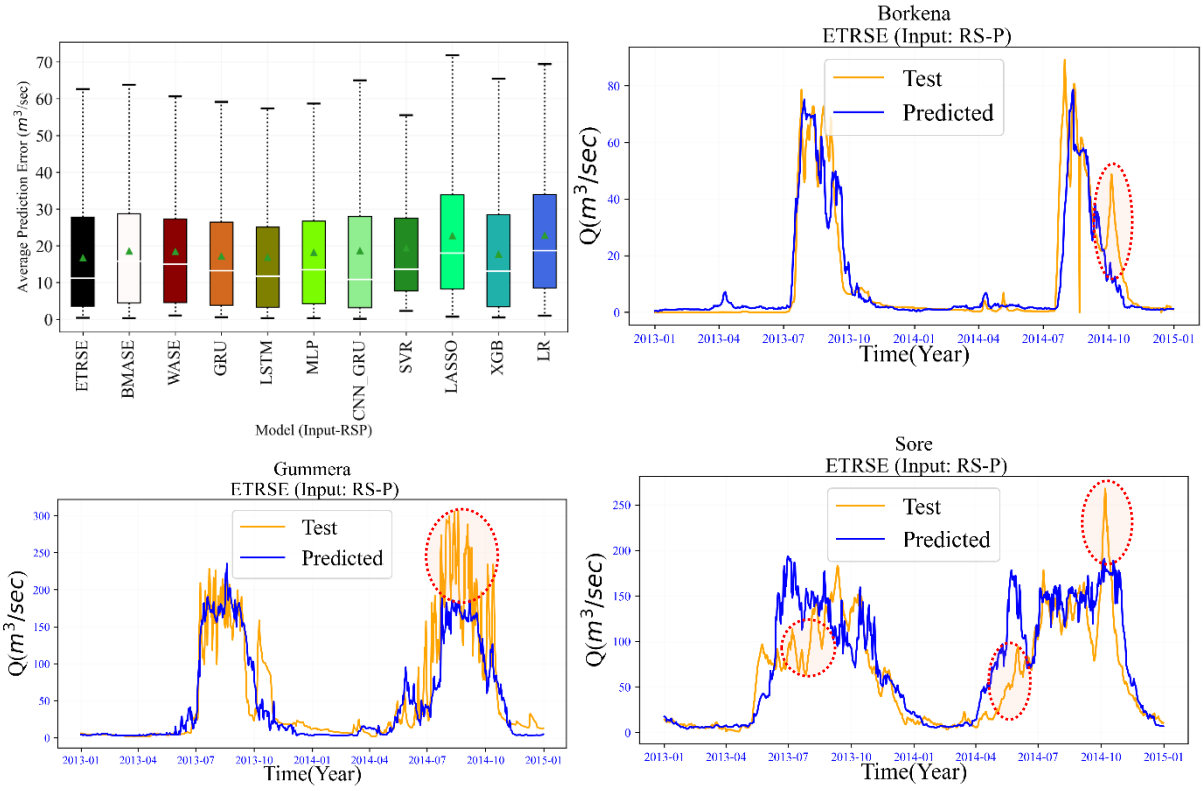


Figure 6. 6 Mean spread of prediction error (m^3/s) or box plot for the 11 models during the test period (a) and time series graph of actual values and predicted values of the optimized ETRSE model (b) Borkena, (c) Gummera, and (d) Sore catchments.

Among the eight base models, LSTM's performance is almost equal to that of the super ensemble model (ETRSE), which had a 13% better R^2 value than the lowest-scoring LASSO and LR models. Figure 6.6 depicts the super ensemble model's actual and estimated time series, which has a high mean R^2 score over all three catchments. Borkena had the highest results, followed by Gummera and Sore. The red circles in the figure also depict the high-performing ETRSE algorithm deficiencies in creating the time series; however, this model typically displayed the mean high performance in the three case study catchments. Figure 6.6 (a) also depicts the mean distribution of residual errors for each of the eleven models using a box plot.

6.3.3 Modelling with Solely Ground-Based Rainfall Data

Ground rain gauges are the principal way of monitoring precipitation, particularly in developing nations (T. et al., 2019). However, the accuracy and stability of precipitation data cannot be assured owing to a lack of precipitation station placements, unequal geographical distributions, limited periods, and vulnerability to environmental and human influences. It is also customary worldwide to utilize a particular gauge to represent precipitation over large regions, such as tens or hundreds of square kilometres (N. Wang et al., 2020). These trends are often the major source of uncertainty in hydrological modelling, and combining ground and satellite-based precipitation measurements now gives an alternate option for hydrological modelling performance accuracy (Beck et al., 2017; Hu et al., 2019). In this research, we will first analyse the existing ground meteorological rainfall data as input to the eleven suggested algorithms. Then, we will combine ground observations with remote sensing and analyse the results in the next section.

The Gummera and Borkena catchments use five available rainfall stations each, and the streamflow simulation results are very good; nevertheless, since the Sore watershed only has one rainfall station with an acceptable continuous time series, the estimated streamflow time series using these station data is unacceptable. Consequently, we generated and published the average performance indices in Table 6.6 below using only the two catchments.

Table 6. 6 The individual and average performance of eleven algorithms in three subcatchments using ground data as input and displayed with a heatmap.

	Borkena				Gummera				Sore				Average				
	RMSE	MAE	MEDAE	R^2	RMSE	MAE	MEDAE	R^2	RMSE	MAE	MEDAE	R^2	RMSE	MAE	MEDAE	R^2	RANK
WASE	9.44	5.04	2.28	0.797	37.25	24.07	14.03	0.76	52.08	35.65	19.06	0.13	23.35	14.56	8.16	0.779	1
BMASE	9.26	4.98	2.11	0.805	37.66	23.23	12.64	0.75	49.37	34.72	18.64	0.22	23.46	14.11	7.38	0.778	2
LSTM	9.19	4.68	1.39	0.808	38.32	22.74	11.89	0.74	46.96	33.17	18.09	0.29	23.76	13.71	6.64	0.774	3
GRU	11.22	5.08	1.27	0.71	32.52	19.45	7.04	0.82	49.83	34.01	18.46	0.2	21.87	12.27	4.16	0.765	4
ETRSE	9.54	4.74	1.43	0.793	39.74	23.57	9.69	0.73	53.86	37.42	23	0.07	24.64	14.16	5.56	0.762	5
XGB	9.11	4.81	1.74	0.812	42.63	24.18	9.19	0.68	53.06	37.29	26.48	0.09	25.87	14.5	5.47	0.746	6
CNN-GRU	9.47	4.68	1.63	0.796	42.63	23.54	7.44	0.68	45.43	32.25	19.78	0.34	26.05	14.11	4.54	0.738	7
LASSO	11.52	7.96	5.61	0.699	43.69	30.04	18.13	0.67	59.77	41.81	21.1	-0.14	27.61	19	11.87	0.685	8
LR	11.59	8.02	5.65	0.695	43.32	29.89	17.98	0.67	59.77	41.81	21.11	-0.14	27.46	18.96	11.82	0.683	9
SVR	13.02	8.84	5.81	0.62	41.54	27.71	17.72	0.7	54.26	38.86	21.89	0.06	27.28	18.28	11.77	0.66	10
MLP	12.05	6.58	2.91	0.671	48.18	26.61	11.57	0.6	42.13	31.04	24.55	0.43	30.12	16.6	7.24	0.636	11

While XGB in Borkena and GRU in Gummera achieved the highest R^2 scores, this model's performance is variable in other catchments, with WASE, BMASE, and LSTM ranking one to three on average, respectively. Furthermore, the MLP and SVR models perform surprisingly poorly for this input situation, performing worse than LR, highlighting the necessity of carefully selecting machine learning models based on the inputs we will employ.

Figures 6.7 (b), (c), and (d) show the accuracy between the actual and estimated values. The red circles in the figure indicate that the input scenario has more underestimated and overestimated points than the remote sensing-based precipitation input scenario. Moreover, Figure 6.7 (a) shows the mean prediction residuals for all algorithms in a box plot.

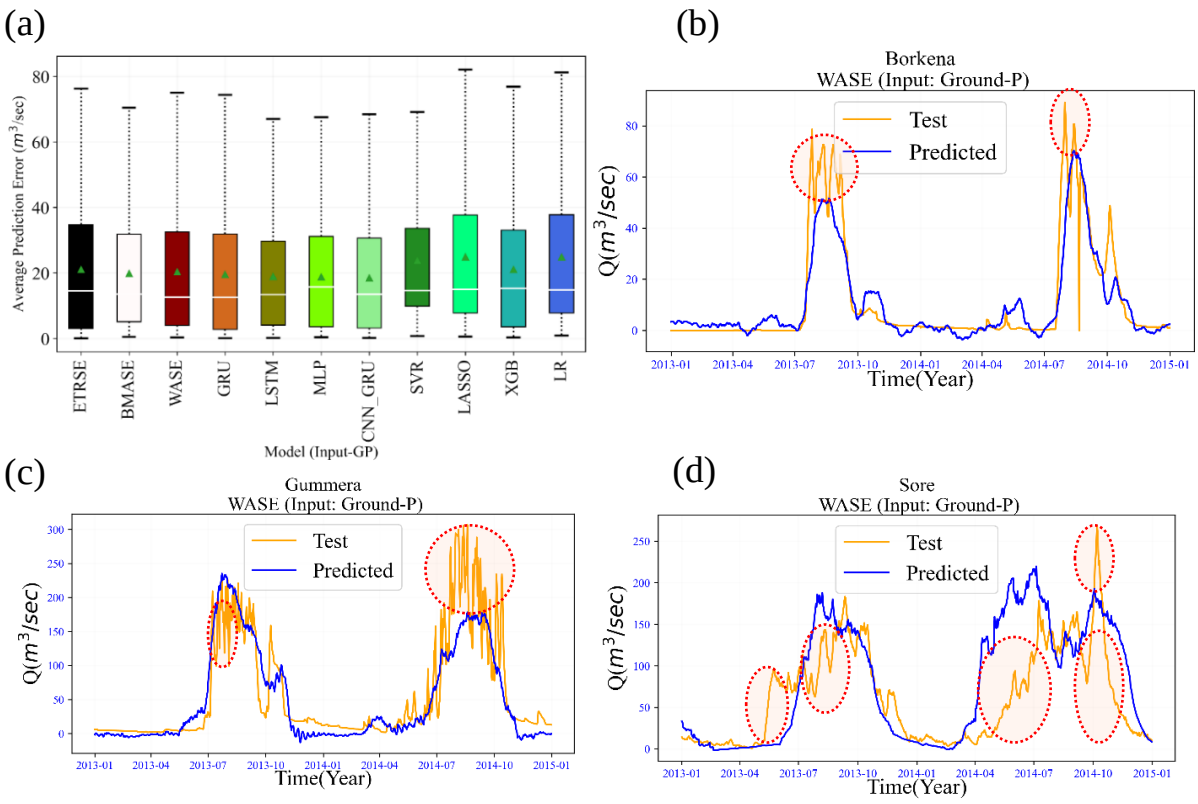


Figure 6. 7 Mean spread of prediction error (m^3/s) or box plot for the 11 models during the test period (a) and time series graph of actual values and predicted values of the optimized WASE model (b) Borkena, (c) Gummera, and (d) Sore catchments.

6.3.4 Modelling with Fused Input Data

Ensemble streamflow modelling appears to be the way ahead in streamflow modelling when uncertainty is a significant problem. This case further highlights the need to use ensemble precipitation products (Sorooshian et al., 2011; Zounemat-Kermani, Batelaan, et al., 2021). Furthermore, Nourani et al. (2021) found that integrating diverse sources of satellite and gauged rainfall products is a viable option for increasing rainfall-runoff model accuracy in data-scarce catchments. As a result, for an accurate streamflow simulation in catchments such as the Sore Watershed, this research advocated the fusion of three variables: three satellite precipitation products, ground-gauged rainfall data, and three satellite-based vegetation indices.

Table 6.7 displays the findings. The simulation performance accuracy in the data sparse sore watershed has increased compared to previous input conditions, with XGB scoring the greatest and LSTM earning the lowest in terms of R^2 value. This output reveals that decision tree model ensembles, such as XGB, are intended to successfully optimize the internal relevance of a prediction model's input features. The LSTM model result, on the other hand, demonstrated that this model is particularly vulnerable to redundant features. However, LSTM ranked in the top three algorithms for each of the other three input circumstances. It scored the lowest out of the eleven algorithms in this instance.

In addition to XGB, BMASE and WASE yielded reliable outcomes. Figures 6.8 (b), (c), and (d) show the time series difference between actual and estimated values, with red circles highlighting places with high flows that were incorrectly estimated. Figure 6.8 (a) also shows the average spread of prediction error for all models as a box plot.

Table 6. 7 The individual and average performance of eleven algorithms in three subcatchments using all fused inputs and displayed with a heatmap.

	Borkena				Gummera				Sore				Average				
	RMSE	MAE	MEDAE	R^2	RMSE	MAE	MEDAE	R^2	RMSE	MAE	MEDAE	R^2	RMSE	MAE	MEDAE	R^2	RANK
XGB	8.88	4.26	1.27	0.82	38.51	24.5	11.4	0.74	30.62	20.88	10.81	0.70	26	16.55	7.83	0.753	1
BMASE	9.49	4.69	1.58	0.8	41.43	24.54	11.7	0.7	28.85	20.25	12.74	0.73	26.59	16.49	8.67	0.743	2
WASE	9.65	4.57	1.65	0.79	43.54	26.91	13.94	0.67	27	18.6	10.47	0.77	26.73	16.69	8.69	0.743	3
CNN-GRU	10.95	5.12	1.56	0.73	39.3	21.39	5.2	0.73	29.02	18.95	7.61	0.73	26.42	15.15	4.79	0.73	4
ETRSE	10.36	4.54	1.22	0.76	40.37	22.9	8.24	0.72	31.88	22.33	15.13	0.67	27.54	16.59	8.2	0.717	5
MLP	8.87	4.74	2.61	0.82	49.45	31.71	15.62	0.57	29.95	21.74	14.87	0.71	29.42	19.4	11.03	0.7	6
SVR	10.26	6.08	4.04	0.76	40.91	25.63	13.89	0.71	36.84	27.65	19.08	0.56	29.34	19.79	12.34	0.677	7
GRU	12.18	5.55	0.95	0.66	42.67	24.16	9.21	0.68	34.65	21.9	9.83	0.62	29.83	17.2	6.66	0.653	8
LASSO	15.27	11.33	8.61	0.47	54.67	36.11	17.93	0.48	32.56	25.41	19.31	0.66	34.17	24.28	15.28	0.537	9
LR	15.79	11.62	8.8	0.43	55.24	36.89	21.13	0.47	32.59	25.39	19.94	0.66	34.54	24.63	16.62	0.52	10
LSTM	10.44	4.65	1.15	0.75	56.8	47.18	38.21	0.44	44.91	30.18	16.56	0.35	37.38	27.34	18.64	0.513	11

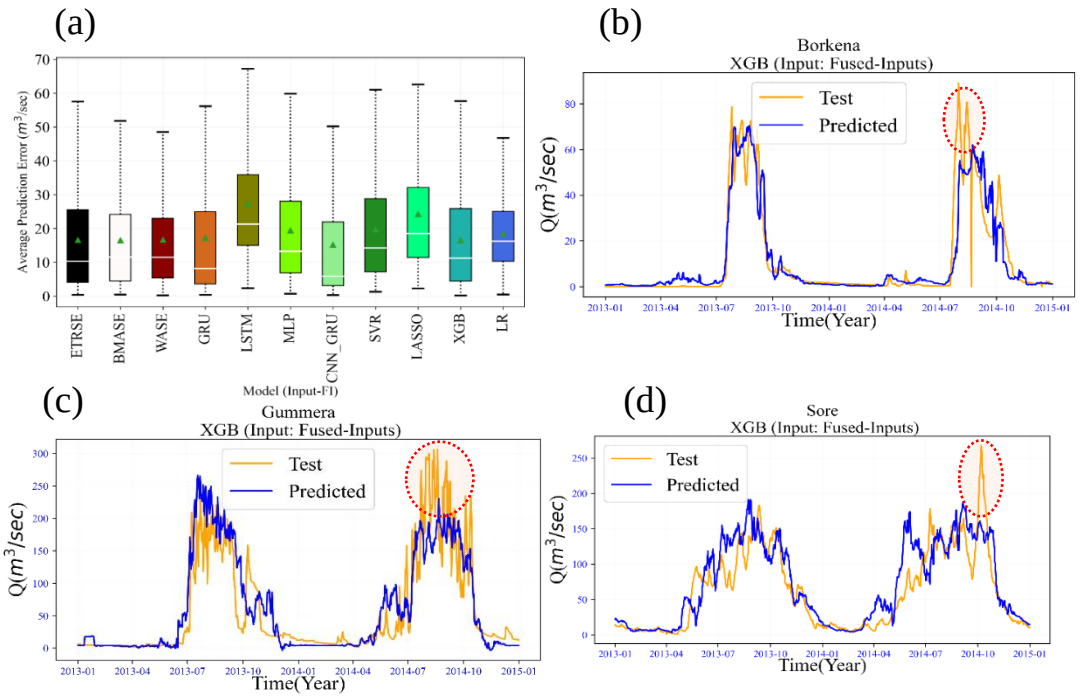


Figure 6. 8 Mean spread of prediction error (m^3/s) or box plot for the 11 models during the test period (a) and time series graph of actual values and predicted values of the optimized XGB model (b) Borkena, (c) Gummera, and (d) Sore catchments.

6.3.5 Modelling with Selected Input Data

After removing irrelevant and duplicated information, feature selection determines the valuable input from the subset of original datasets. Consequently, the training process is accelerated, and accuracy is improved; feature selection can be classified as supervised, unsupervised, or semisupervised (Cai et al., 2018). The popular supervised wrapper recursive feature elimination (RFE) approach was carefully chosen for this study. This feature selection technique is straightforward to create and use, while implementing RFE, the two configuration choices are the number of features to pick and the optimizing algorithm for feature selection (Lian et al., 2020). Following early testing, we decided on ten variables and the decision tree regressor (DTR) as the optimization algorithm. RFE then tunes all original data features in the training time series until the desired number is attained.

The analysis result showed WASE as the top-rated model, followed by LSTM, which enhanced its performance owing to feature selection, while CNN-GRU and ETRSE performed almost equally, as shown in Table 6.8. Figures 6.9 (b), (c), and (d) also show the gap between the top-ranked model's actual and estimated time series. The graph also shows that the estimated time series is more stable than the noise in the preceding time series graph with a fused input condition. Figure 6.9 (a) also shows the average prediction error spread using the box plot.

Table 6. 8 The individual and average performance of eleven algorithms in three subcatchments using selected inputs and displayed with a heatmap.

	Borkena				Gummera				Sore				Average				
	RMSE	MAE	MEDAE	R^2	RMSE	MAE	MEDAE	R^2	RMSE	MAE	MEDAE	R^2	RMSE	MAE	MEDAE	R^2	RANK
WASE	8.99	4.37	1.66	0.82	40.56	24.19	12.07	0.72	27.42	18.72	10.32	0.76	25.66	15.76	8.02	0.767	1
LSTM	10.09	4.58	1.2	0.77	33.6	18.85	6.49	0.8	29.59	19.98	9.35	0.72	24.43	14.47	5.68	0.763	2
CNN-GRU	9.63	4.18	0.78	0.79	40.21	23.81	10.93	0.72	27.69	18.01	7.97	0.76	25.84	15.33	6.56	0.757	3
ETRSE	9.15	4.74	1.82	0.81	38.96	21.93	7.53	0.74	29.57	19.6	8.44	0.72	25.89	15.42	5.93	0.757	4
BMASE	9.15	4.67	2.02	0.81	43.37	26.26	13.63	0.67	28.61	19.44	10.79	0.74	27.04	16.79	8.81	0.74	5
XGB	9.03	4.55	1.5	0.81	40.38	22.74	7.29	0.72	31.62	21.05	11.19	0.68	27.01	16.11	6.66	0.737	6
MLP	11.29	6.07	2.03	0.71	40.54	25.73	11.57	0.71	28.06	17.67	7.59	0.75	26.63	16.49	7.06	0.723	7
SVR	10.94	6.92	4.65	0.73	40.47	28.72	20.83	0.72	33.53	24.75	18.01	0.64	28.31	20.13	14.5	0.697	8
LASSO	11.15	7.19	4.43	0.72	42.58	27.11	14.92	0.68	32.21	25.31	19.49	0.67	28.65	19.87	12.95	0.69	9
LR	11.17	7.16	4.49	0.71	42.62	27.16	15.15	0.68	32.26	25.41	19.82	0.66	28.68	19.91	13.15	0.683	10
GRU	9.57	4.87	2.38	0.79	41.7	24.09	11.23	0.7	39.14	30.51	22.92	0.51	30.14	19.82	12.18	0.667	11

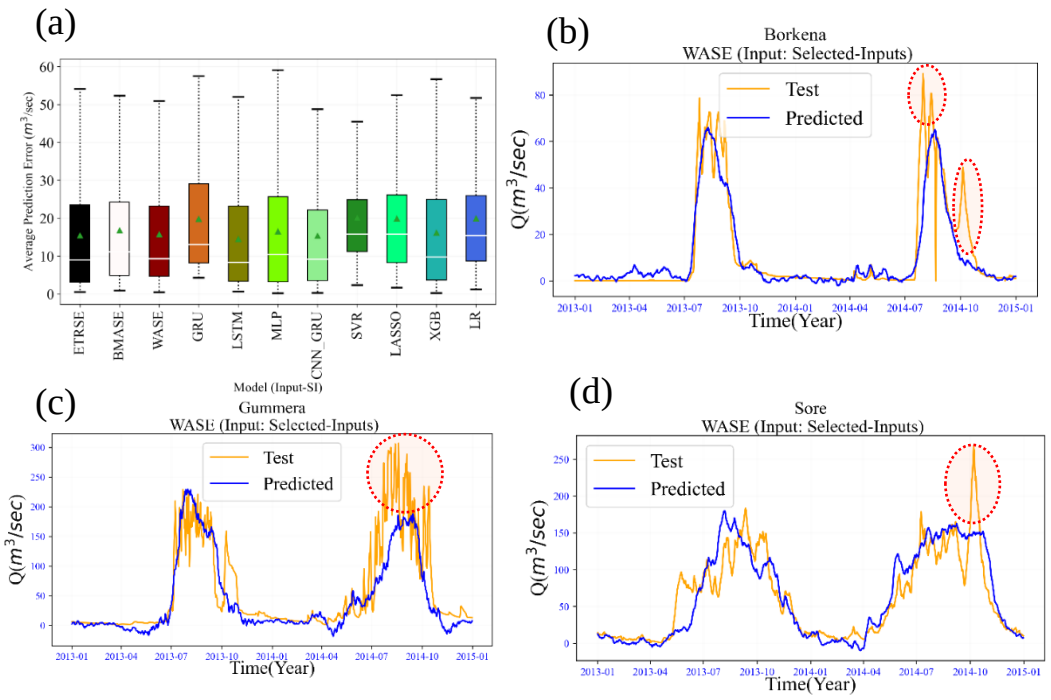


Figure 6. 9 Mean spread of prediction error (m^3/s) or box plot for the 11 models during the test period (a) and time series graph of actual values and predicted values of the optimized WASE model (b) Borkena, (c) Gummera, and (d) Sore catchments.

6.4. Conclusions

This investigation presents a novel strategy for predicting daily streamflow by combining a remote sensing-based vegetation index, precipitation product, and ground gauge rainfall data. To effectively comprehend the performance enhancements caused by these inputs, five input scenarios are designed: only vegetation index, only remote sensing-based precipitation, only ground gauged precipitation, all fused inputs, and chosen input variables. To train the input scenarios, we used super ensemble learning with three separate meta-learners (BMASE, ETRSE, WASE) and eight base models (GRU, LSTM, MLP, CNN-GRU, SVR, LASSO, XGB, LR). We identified three Ethiopian river basin subcatchments with reasonably good datasets (Gummera and Borkena) and the Sore watershed with limited meteorological data as case studies. We employed 12 years of time series (2003-2014) for all investigations, rolled the predictor variables monthly, and forecasted one-step streamflow values using data from the preceding thirty days.

The following important findings summarize the overall research outputs: -

I. As shown in Table 6.9, the average performance of eleven algorithms in three subcatchments and five input situations for the three suggested superensemble models was practically identical. Even though WASE, BMASE, and ETRSE displayed the highest R^2 performance, respectively. Moreover, according to the same performance index, R^2, the top-ranked WASE model outperformed the linear regression baseline by 13.3%.

II. The vegetation indices input scenario consistently performed well in all catchments, with a top-ranked BMASE model average R^2 performance of 0.68%. This finding implies that VI has the potential to be used for data-driven streamflow modelling, particularly in nongauged catchments with no meteorological time series. However, continued research is needed on this outcome using additional novel data assimilation methodologies.

III. XGB, CNN-GRU, and LSTM performed the best among the eight evaluated base models. This research also demonstrates that one of LSTM's key weaknesses is that its performance suffers when feature selection criteria are not employed. XGB, on the other hand, demonstrated higher performance after controlling redundant inputs internally. Furthermore, even if specific base models outperformed others in a given catchment, this performance is not transferrable to other catchments. On the other hand, superensemble models displayed consistent performance across all catchments and input circumstances.

IV. Following modelling with five input scenarios, the ground data input scenario with WASE (0.779% R^2) had the highest performance of all scenarios, excluding the sore watershed, which had only one ground station continuous rainfall time series, followed by the selected input scenario (WASE, 0.77% R^2), fused inputs (XGB, 0.75% R^2), remote sensing-based precipitation (ETRSE and LSTM, 0.71% R^2), and vegetation indices input scenario (BMASE, 0.683% R^2) in order of the highest average catchment performance score. The input fusion scenario significantly improves R^2 performance for a nongauged catchment with a few meteorological datasets or the Sore catchment, from 0.44% R^2 (MLP model, ground input data) to 0.77% R^2 (WASE model, fused inputs). These findings exemplify the potential for incorporating remote sensing data into streamflow simulations and are especially advantageous to catchments such as the Sore watershed.

Figure 6.10 shows the performance of the eleven models in three subcatchments (Borkena (a), Gummera (b), and Sore (c)) on a two-dimensional scale, with the correlation coefficient and root mean square error on the radial axis and the standard deviation on the polar axis. That is the degree of statistical agreement between the actual and estimated time series. The model with the best performance will be near the observed streamflow with a sign (★) in this graph. Additionally, as shown in the graph, the symbol with the red symbols (mostly superensemble models) performed the best, specifically in Figure 6.10 (d), which displays the average performance for all catchments.

The study's insightful results should lead future research to other ensemble learning methodologies, including alternative remote sensing-based input data assimilation strategies. Furthermore, the scientific community has to pay greater attention to including vegetation indice variables in data-driven streamflow modelling. Finally, we urge the research community to improve the presented approaches by using numerous feature selection criteria, extra meta and base learner models, and other agro-climatic catchment situations.

Table 6. 9 The average performance of eleven algorithms in three subcatchments and five input scenarios are displayed with a heatmap.

Model	RMSE	MAE	MEDAE	Correlations (r)	R^2	RANK
WASE	26.39	16.61	9.04	0.88	0.732	1
BMASE	26.80	16.66	8.76	0.88	0.726	2
ETRSE	26.73	15.78	6.67	0.87	0.721	3
XGB	27.46	16.46	7.27	0.86	0.710	4
CNN-GRU	27.39	15.89	6.01	0.85	0.702	5
LSTM	28.16	17.86	9.52	0.87	0.686	6
GRU	27.85	16.95	7.87	0.86	0.675	7
SVR	28.64	19.69	12.99	0.85	0.672	8
MLP	28.93	17.61	8.12	0.84	0.670	9
LASSO	31.65	22.60	15.36	0.83	0.606	10
LR	31.73	22.69	15.76	0.83	0.599	11

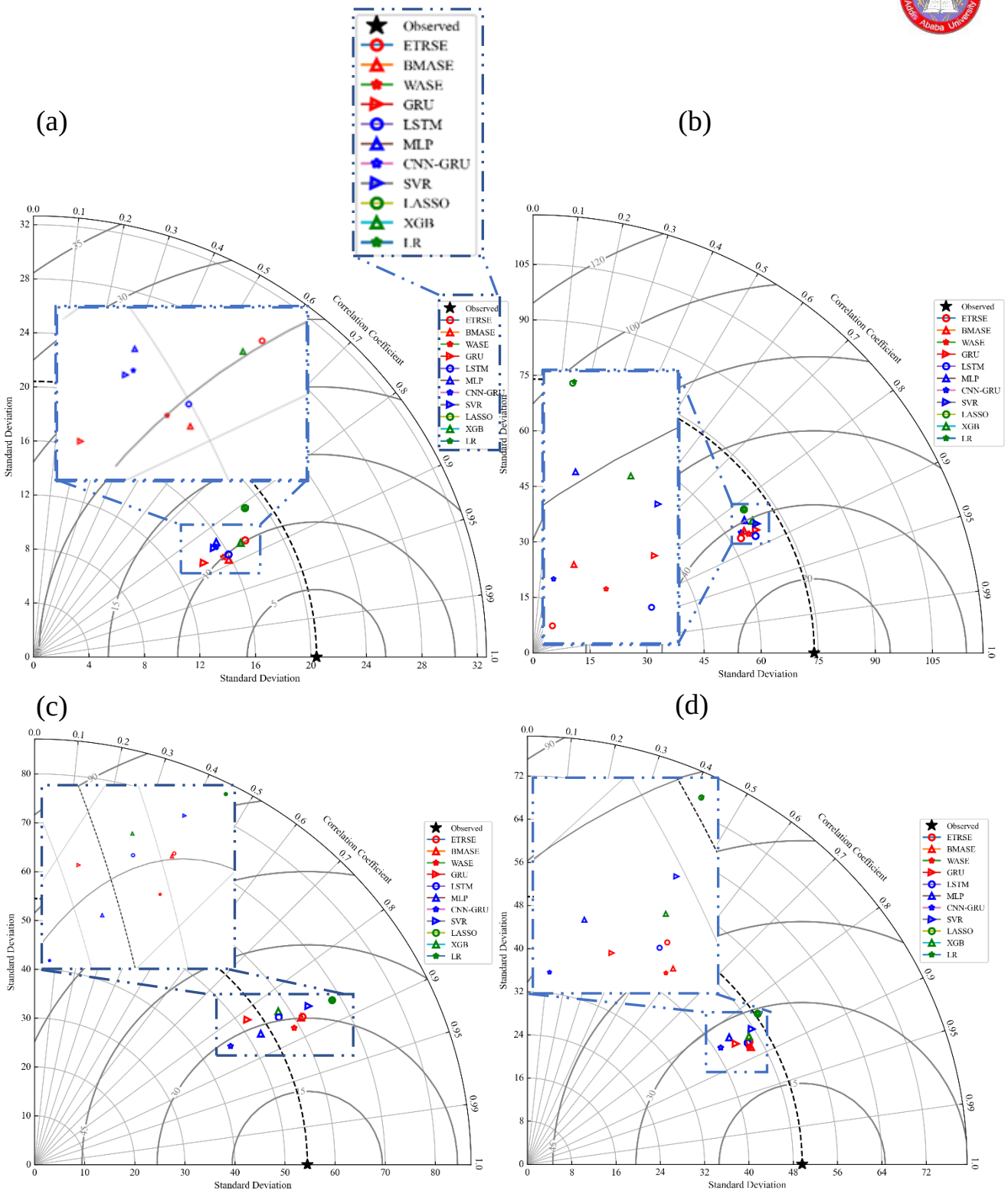


Figure 6. 10 The Taylor diagram displays the standard deviations, root mean square error, and correlation coefficient between observed and predicted streamflow for the proposed eleven models and three catchments. Borkena (a), Gummera (b), Sore (c), and the average for all catchments (d).

CHAPTER 7:

COMPARING CONCEPTUAL AND SUPERENSEMBLE DEEP LEARNING MODELS FOR STREAMFLOW SIMULATION IN DATA-SCARCE CATCHMENTS

Abstract[4]

Model ensembles, or integrating the benefits of many models without altering the core character of the data, are becoming more common in hydrology. This study expands on a prior study from chapter five, in which three superensemble machine-learning models were compared against eight single-base models. We evaluated the performance of these superensemble learners to that of a semidistributed HBV-light physical hydrological model in this work.

Ten years of daily time series data are extracted from two river catchments (Sore and Mashaa) in Ethiopia's Baro Akobo basin. The HBV model uses input variables such as the mean temperature, potential evapotranspiration, and ground precipitation. The superensemble model was evaluated with three remote sensing precipitation and vegetation indices data products to simulate single-step daily streamflow. The recursive feature elimination (RFE) technique is utilized in this research to extract the critical input parameters from the fused inputs.

The performance of the twelve models was evaluated using the root mean squared error (RMSE), mean absolute percentage error (MAPE), median absolute error (MEDAE), mean absolute error (MAE), and coefficient of determination (R^2). Finally, the average modelling performance in the two catchments demonstrated that the three superensemble learners consistently outperformed the eight base models and the HBV-light model. The top-ranked Extra Tree Regression Superensemble (ETRSE) model outperforms the HBV-light model by 24% in terms of the performance measure indice R^2. Choosing key input parameters also demonstrates that it improves the performance of machine learning models.

[4] *This chapter is based on the publication-*
https://www.sciencedirect.com/science/article/pii/S2214581824000429#ab0015

7.1. Introduction

Over the last century, hydrological models have developed with complexity and procedures, beginning with Mulvany's (1850) rational approach for peak discharge estimation (Singh, 2018). The various hydrological models presently accessible reflect the various aims for which they were built, such as operational flood and streamflow forecasting, water resource planning and management, and quantifying climate change-driven flood variances (Bourdin et al., 2012).

Hydrological models may be classified differently regarding hydrological cycle process descriptions, time scales, parameter estimate techniques, spatial resolutions of input data, and simulation methods (Sood & Smakhtin, 2014). The primary streamflow model classification technique relied on the system process or physical-based models generated from watershed characteristics and data-driven models based on previously obtained data. Process-based models often use the experimental formula, which provides insight into physical attributes but requires a large amount of data; SHE/MIKE SHE and SWAT are examples of this type of model. On the other hand, data-driven models are applicable and easily functional without addressing the fundamental physical processes of the watershed system. Machine learning algorithms are included in this group (Wegayehu & Muluneh, 2021). Hydrological models may also be classified as lumped or distributed depending on how the model parameters vary over time and space (Devia et al., 2015). Lumped models consider a watershed as a single unit, attempting to relate input parameters to discharge outputs while ignoring the attribute's spatial dynamics. Distributed models, on the other hand, attempt to account for spatial patterns of hydrological variables within a catchment area (Keith J Beven, 2005). The distributed models may be entirely or partially distributed; fully distributed models are regarded as the gold standard for hydrological modelling, despite their complex model structures and high processing costs (Yang et al., 2020).

The past decades of hydrological modelling owe a great deal to advances achieved before 1960. After computers were invented in the 1960s, the digital revolution started, and by the 1970s, computers were accessible to institutions, government agencies, and businesses (Singh, 2018). Machine learning-based data-driven models have been a prominent and rising research area in recent years, thanks to advancements in computer processing power and the availability of vast volumes of satellite data. Physical and statistical models have a variety of flaws, including inadequate uncertainty analysis, high processing costs, and the need for an extensive physical dataset (Faizollahzadeh et al., 2019).

Due to their effective artificial intelligence, machine learning models are particularly effective in addressing these difficulties. However, it is unclear whether data-driven machine learning will gradually overtake the physical hydrological model or vice versa and how traditional hydrologic models will advance further, given the disparity in philosophy used by the two model groups. Thus, it would be crucial to assess new ideas and identify their use and advancement patterns. It is also critical to thoroughly evaluate these two kinds of models and determine which models are more reliable and robust than others under what situations, rather than just building new models or applying them independently in other research fields (T. Kim et al., 2021).

For modelling streamflow in four watersheds, Kim et al. (2021) compared three physical models (SACramento Soil Moisture Accounting (SAC-SMA), Xin'anjiang (XAJ), and Coupled Routing and Excess STorage (CREST)) with two machine learning models (ANN and LSTM). The same research discovered that ML is competitive in many situations compared to physical models and advocated constructing hybrid hydrologic models to use the benefits of both models while addressing the disadvantages of individual approaches and associated data demands. L. Liu et al. (2022) investigated the conceptual rainfall-runoff model (SIMHYD) and the machine learning model (LSTM) for flood modelling capacities in 232 basins with varying agroclimatic conditions. In the calibration phase, the LSTM model outperformed SIMHYD but not in the validation phase. The research also shows that variations in basin characteristics have almost no effect on the performance of the LSTM and SIMHYD models. The same paper recommends machine learning approaches for flood simulation if suitable training data are available; otherwise, hydrological models are recommended. The literature mentioned above has limits in terms of the extent of ML model selection. Furthermore, recent research has shown that combining high-performing ML models with several meta-learners is better than employing a single model (Nourani et al., 2021; Tyralis et al., 2019).

Although model innovation has increased the accuracy of hydrological simulation, hydrological models are still incapable of explaining the hydrological processes of a watershed on a large scale without accounting for uncertainty from several sources. Each hydrological model has a unique approach to understanding and generalizing hydrophysical phenomena. Different models may provide different simulations when given the same inputs. As a result, various methods have been developed that combine the advantages of many models to aggregate the numerous simulation findings and improve forecast accuracy (Liang et al., 2013).

Ensemble learning may solve several shortcomings of traditional single ML models, such as underfitting in sparse data catchments, failure to calculate optimum performance effectively, and methods lacking proper fitness functions (Zounemat-Kermani, Batelaan, et al., 2021). Shamseldin (1997) was the first to create a hybrid-model simulation based on five rainfall-runoff models with three combinations (i.e., neural network, simple averaging, and a regression-based approach). According to the same research, a group of models is more accurate than a single model (C.C. Young et al., 2015). Since then, numerous studies have been undertaken, including different physical and/or data-driven hydrological models (Y. Li et al., 2019; Noori & Kalin, 2016; Troin et al., 2021; Wagena et al., 2020; Zounemat-Kermani, Batelaan, et al., 2021).

We used multisource remote sensing and ground-gauged rainfall data in the previous chapter to assess streamflow modelling using superensemble learning. The chapter also presented an innovative approach for using three satellite-based vegetation indices as input for streamflow modelling. In addition, three distinct meta-learners (weighted average (WA), extra tree regression (ETR), and Bayesian model averaging (BMA)) and eight different base models were used (LSTM, GRU, CNN-GRU, SVR, MLP, XGB, Lasso, and LR). According to the results of this research, ensemble models outperformed single machine learning models in five input scenarios and three Ethiopian river catchments. The present chapter builds on this conclusion by attempting to compare the top three ensemble learners (ETR, BMA, and WA) with a semidistributed conceptual physical hydrological model (Hydrologiska Byrans Vattenavdelning (HBV)). The HBV model uses precipitation, mean temperature, and potential evapotranspiration time series as inputs. Despite its simplicity, many applications under varied land cover and climatological circumstances have shown that its design is relatively stable and surprisingly general, which benefits its use in data-scarce catchments (Bergström, 1992; Devia et al., 2015; X. Zhang & Lindström, 1997).

To fully understand this research methodology, we recommend that the reader review the previous chapter. In this chapter, we will focus on the HBV model data analysis, integrating and comparing the three superensemble models with the HBV model result. In addition to the data-scarce Sore watershed used in the previous chapter, one additional study area will be investigated in this study.

7.2. Data and Methods

This research focuses on two data-scarce case study areas: the Sore and Mashaa subcatchments of Ethiopia's Baro Akobo River Basin.

7.2.1 Baro Akobo River Basin

The Baro Akobo River basin is Ethiopia's eighth-largest basin, with a drainage area of approximately 76,000 km^2 (6.9% of the country). The lower sections of the Baro and Akobo Rivers border Sudan; they join to create the Sobat River, a major tributary of the White Nile. The basin's total annual runoff output was assessed to be the second highest (23 Bm^3), accounting for approximately 8% of the total Nile River flow near the Egyptian border. Ethiopia's two bordering river basins are the Abay in the north and northeast and the Omo-Gibe in the east. The main rivers in the basin are the Baro, Akobo, Gilo, and Alwero.

7.2.2 Sore Subcatchment

The Sore subcatchment has an area of 1711 km^2 and is situated in Ethiopia's Baro-Akobo River basin's highlands. The Sore watershed's elevation ranges from 2658 to 1578 masl and is located between $8^0 21'$ and $7^0 49'$ latitudes and $35^0 31'$ and $35^0 57'$ longitudes, with the maximum elevation ranges found east and south. To the north and northwest, the elevation declines (Figure 7.1). Annual precipitation varies from 1804 to 2020 mm. Aridity rises from east to west throughout the Baro Akobo basin and the sore watershed, with monthly maximum temperatures ranging from 24°C to 28°C and monthly minimum temperatures ranging from 12°C to 14°C.

7.2.3 Mashaa Subcatchment

The Mashaa subcatchment has a surface area of 1668 km^2 bordering the Sore catchment. The Mashaa watershed is situated between latitudes $7^0 30'$ and $8^0 02'$ and longitudes $35^0 26'$ and $35^0 54'$, with elevations ranging from 2783 to 1669 m.a.s.l. The yearly precipitation ranges from 1800 to 2387 mm. The monthly high temperature ranges from 20 to 26 °C, while the monthly low temperature ranges from 10 to 13 °C. The hydrological and meteorological stations at Mashaa are used in this study. The details of the two case study sites are shown in Figure 7.1.

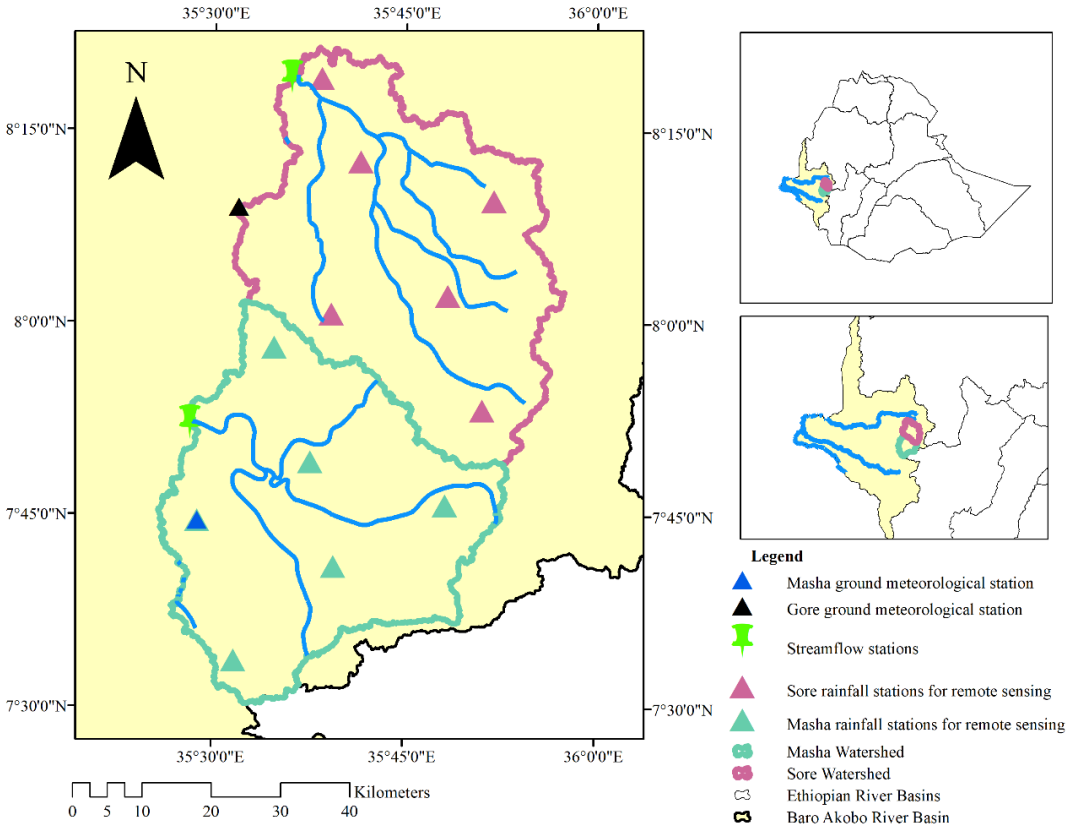


Figure 7. 1 The location of the case study areas (Sore and Mashaa subcatchments).

7.2.4 Data

- **Ground Data**

Three daily ground datasets are used in this study: precipitation, temperature, and discharge. The first six years of the time series are used to calibrate the model, while the latter two years are utilized to validate it. The entire data range is from 2003 to 2010. The Ethiopian Ministry of Water and Energy (MoWE) provided the hydrological data, and the Ethiopian National Meteorological Agency (NMA) provided the meteorological data.

Although we evaluated three ground precipitation data stations (Mettu, Gore, and Hurmu) and two temperature data stations (Mettu and Gore), only the Gore meteorological station had a relatively acceptable and continuous dataset in the time range (2003 – 2010 GC). Consequently, we only use Gore meteorological and Mettu hydrological time series data to calibrate and verify the HBV-Light model. Table 7.1 displays the time series statistical features of each watershed's chosen meteorological and hydrological stations.

Table 7. 1 For each case study area, the selected hydrological and meteorological stations and their daily time series statistical properties.

Catchment		Mashaa	Sore
Climatological Stations Name		Mashaa	Gore
Climatological Stations Position	Longitude	35.48	35.53
	Latitude	7.74	8.15
Streamflow Stations Name		Mashaa	Nr. Metu
Streamflow Stations Position	Longitude	35.47	35.60
	Latitude	7.87	8.32
Maximum	Rainfall (mm)	74.20	70.60
	Temperature (^{o}c)	22.85	25.10
	Q (m^3/sec)	307.25	213.38
Minimum	Rainfall (mm)	0.00	0.00
	Temperature (^{o}c)	7.30	11.30
	Q (m^3/sec)	1.10	0.97
Mean	Rainfall (mm)	5.93	4.57
	Temperature (^{o}c)	17.02	19.34
	Q (m^3/sec)	52.61	48.57
Standard deviation	Rainfall (mm)	9.01	7.84
	Temperature (^{o}c)	1.27	1.62
	Q (m^3/sec)	54.80	51.14

- **Remote Sensing Data**
  - **Meteorological Data**

The research consists of three remote sensing-based precipitation datasets (IMERG-final, CHIRPS, and MSWEP-V2) and three vegetation indices (NDVI, EVI, and NDWI), all of which have been well described in the previous chapter. The respective remote sensing-based precipitation data for the two case study locations were constructed using five typical meteorological stations selected randomly, together with one already existing station. Their time series statistical properties are shown in Table 7.2.

Table 7. 2 The five randomly selected sample meteorological stations with one existing meteorological station for each of the two case study locations and three precipitation products with their time series statistical features.

CATCHMENt	Remote Sensing Station (Rainfall)	LONGTIUDE	LATITUDE	Maximum (mm)			Mean (mm)			Standard Deviation (mm)		
				IMERG	CHIRPS	MSWEP	IMERG	CHIRPS	MSWEP	IMERG	CHIRPS	MSWEP
Mashaa	Mashaa	35.48	7.74	88.53	79.27	130.25	4.29	5.48	4.84	8.31	8.97	7.24
	Mashaa 1	35.58	7.97	71.73	60.23	89.63	4.23	5.19	4.54	8.36	8.22	5.78
	Mashaa 2	35.63	7.82	99.26	56.29	121.75	4.42	4.95	5.37	8.55	8.02	6.49
	Mashaa 3	35.80	7.76	105.87	57.97	169.00	4.84	5.13	5.07	9.20	8.69	6.45
	Mashaa 4	35.66	7.68	80.47	75.96	139.63	4.63	5.07	5.29	8.79	8.63	6.35
	Mashaa 5	35.53	7.56	93.25	87.10	99.63	4.52	5.83	5.16	8.57	9.76	6.06
Sore	Gore	35.64	8.32	88.31	73.70	93.06	4.03	4.69	4.47	8.37	8.23	5.69
	Sore 1	35.69	8.21	86.77	57.91	75.50	4.06	4.92	4.58	8.13	8.25	5.83
	Sore 2	35.87	8.16	89.56	62.96	113.31	4.42	4.89	4.75	8.47	8.63	6.29
	Sore 3	35.65	8.01	176.76	68.68	89.94	4.25	5.12	4.97	8.83	8.36	6.33
	Sore 4	35.81	8.04	74.91	58.44	138.31	4.51	5.04	4.92	8.42	8.37	6.48
	Sore 5	35.85	7.89	111.66	73.99	183.63	4.85	5.12	5.12	9.24	8.66	6.64

N.B. The minimum precipitation at all stations and data products is zero

- **Digital Elevation Model (DEM)**

The region's topography is well documented by the DEM, which specifies the elevation of each point at a specific place as a digital file with accurate spatial resolution. It is an essential geographic input for the HBV-light model to analyse and categorize the watershed into separate elevation zones. The raw DEM was obtained by the Shuttle Radar Topography Mission (SRTM) and had a resolution of 30 m X 30 m, which allowed the catchments to be separated into five elevation zones (see Figure 7.2 below).

- **Land Use /Land Cover Data**

Changes in land use and cover are one key element influencing the watershed's hydrological system. As a result, accurate identification of land use and land cover information has a considerable influence on the performance of the runoff prediction model. The data acquired from remote sensing sources are analysed using GIS software. The two catchments' wide land use and land cover may be classified into forest, cropland, and grassland. As a result, we divide each elevation zone into three vegetation zones to supply an input to the HBV-light model.

- **Potential Evapotranspiration Estimation (PET)**

PET combines transpiration from a vegetated surface with an unconstrained water supply and evaporation from an open water body. Due to the study area's high humidity and temperatures, the flora significantly affects the quantity of evapotranspiration. As PET is also impacted by sunlight and wind speed, determining the PET of the study region is critical. Since the project area has minimal access to weather data, we employed Enku's simple temperature technique, a novel basic empirical temperature methodology (Enku & M. Melesse, 2014). This equation was tested in many Ethiopian catchments and showed excellent results, which makes it ideal for use in data-limited catchments.

$$ET_o = (T_{max})^n / k \quad \text{(Eq.1)}$$

where ET_o is the reference evapotranspiration (mm/day), n = 2.5, and k is the coefficient; both can be calibrated for specific local circumstances. K can range from 600 to 1300 for lower and higher mean annual maximum temperature locations, respectively. $k = 73*T_{mm} - 1015$ for dry seasons, $k = 38*T_{mm} - 63$ for wet seasons, and $k = 48*T_{mm} - 330$ for mixed wet and dry seasons. where T_{max} (°C) is the long-term daily mean maximum temperature.

Table 7. 3 Calculated mean monthly areal potential evapotranspiration for the two catchments (mm/month)

Month	Sore	Mashaa
January	120.86	107.27
February	115.18	102.23
March	128.01	112.57
April	118.47	104.67
May	113.34	102.48
June	101.22	91.90
July	98.50	89.77
August	99.89	90.11
September	101.38	91.45
October	109.91	99.91
November	109.68	98.49
December	116.77	103.70

Table 7. 4 Areal coverage in decimal percentage for three vegetation zones in each of the five elevation zones of the two subcatchments.

Catchment	Sore	Mashaa	Catchment	Sore	Mashaa	Catchment	Sore	Mashaa
Catchment Area (Km^2)	1726.22	1667.86	Catchment Area (Km^2)	1726.22	1667.86	Catchment Area (Km^2)	1726.22	1667.86
Mean Elevation Zone 1 (meter)	1726.94	1753.7	Mean Elevation Zone 3 (meter)	1981.18	2106.75	Mean Elevation Zone 5 (meter)	2309.08	2465.87
Veg 1 (Zone 1)	0.25	0.25	veg 1 (Zone 3)	0.18	0.23	veg 1 (Zone 5)	0.1	0.07
veg 2 (Zone 1)	0	0.03	veg 2 (Zone 3)	0.05	0.02	veg 2 (Zone 5)	0.03	0.02
veg 3 (Zone 1)	0	0	veg 3 (Zone 3)	0	0	veg 3 (Zone 5)	0	0
Mean Elevation Zone 2 (meter)	1833.75	1931.05	Mean Elevation Zone 4 (meter)	2146.7	2288.07			
veg 1 (Zone 2)	0.2	0.18	veg 1 (Zone 4)	0.13	0.16			
veg 2 (Zone 2)	0.01	0.01	veg 2 (Zone 4)	0.04	0.02			
veg 3 (Zone 2)	0	0	veg 3 (Zone 4)	0	0			

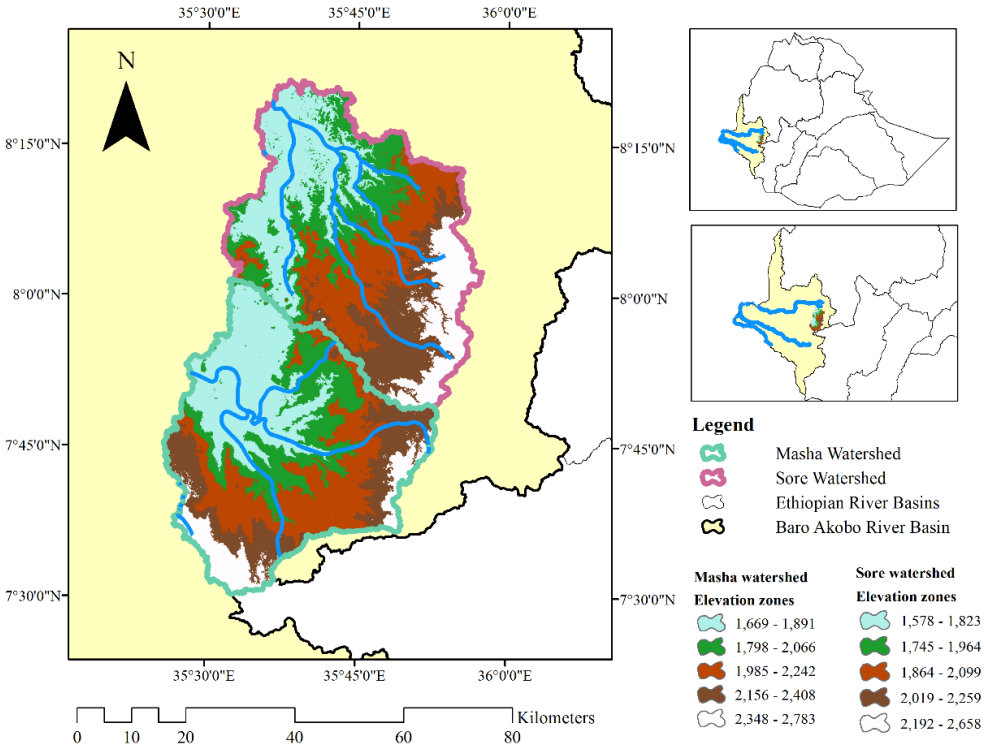


Figure 7. 2 DEM map showing each two catchments classified as five elevation zones

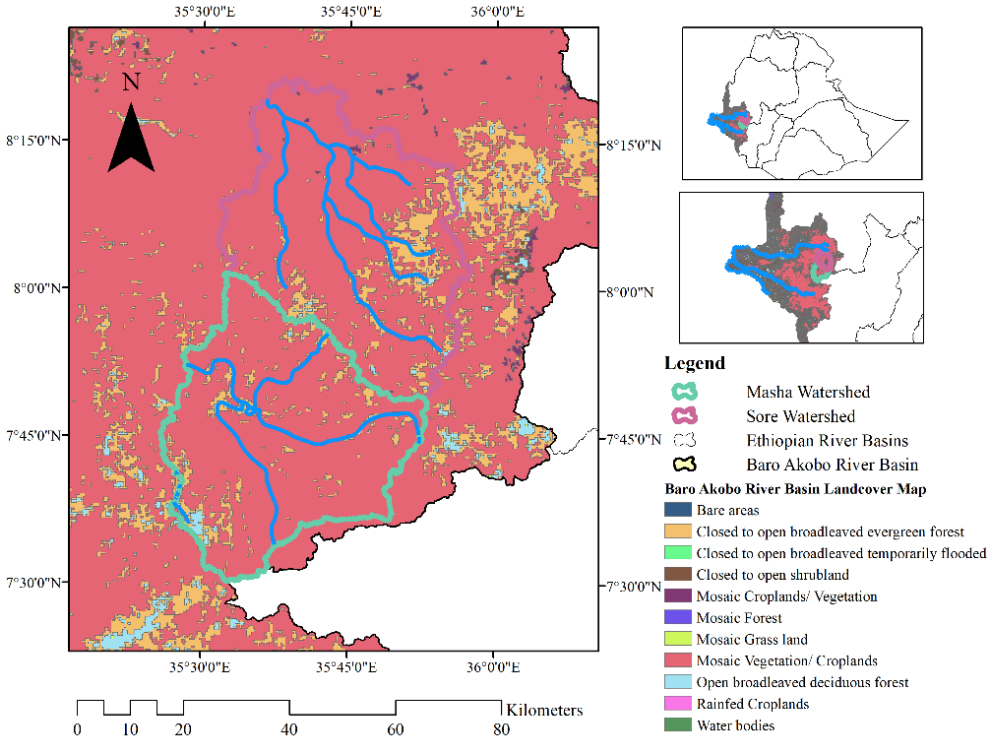


Figure 7. 3 The land cover map of the two case study area catchments.

7.2.5 Methods

Two types of models were employed in this research to simulate one-step daily discharge. The first model is a superensemble model that includes three meta-learners (extra tree regression (ETR), Bayesian model averaging (BMA), and weighted average (WA)) as well as eight base models (GRU, LSTM, MLP, CNN-GRU, SVR, lasso, XGB, and LR). This model group was evaluated and presented effectively in the previous chapter and demonstrated the best performance. In our present work, we will compare the results of this model, adding a new study area to a semidistributed physical hydrological model (HBV-light). As a result, the following sections will focus on the HBV model analysis.

- **Hydrologiska Byrans Vattenavdelning (HBV) Model**

The HBV hydrological model was created by the Swedish Meteorological and Hydrological Institute (SMHI) (X. Zhang & Lindström, 1997). The HBV model is a semidistributed conceptual hydrological model successfully used in 30 countries worldwide (Bergström, 1976, 1992). Initially, designed for Scandinavian catchments, it performed well in tropical and subtropical climates (X. Zhang & Lindström, 1997). The model requires daily rainfall, temperature, and monthly evapotranspiration for streamflow estimation. The watershed is segmented into subcatchments, which are further classified into elevation, lake area, and vegetation zones.

The "HBV light" version of the model used in this work (Seibert, 1997) is similar to the HBV-6 version described by (Bergström, 1992) with just two minor variations. Instead of beginning the simulation with some user-defined initial state values, the new version employs a warming-up phase during which state variables grow from standard initial values to their correct values in agreement with meteorological factors and parameter values. Furthermore, the restriction that only integer values may be used for the MAXBAS routing parameter has been relaxed. The HBV model was applied to five elevation zones and three land cover zones in this research.

The four HBV model routines are soil moisture, snow, response, and routing. However, since there is no considerable snowfall in the research area, we reject the snow routine, which was not considered in the parameter analysis. The three routines comprised nine parameters that might alter the magnitude of the simulated discharges.

The Monte Carlo simulation approach was used, and ranges of possible values for each parameter were determined based on the range of calibrated values from model default values (Table 7.5). During the Monte Carlo simulation, 1000 model runs were chosen to run. The model was run for each parameter set, and the objective function values were determined. The coefficient of efficiency (R_{eff}) is often used to assess HBV model simulations, and a perfect match occurs when R_{eff} equals one. The top 100 model runs with the best objective function (R_{eff}) are kept for further analysis.

$$R_{eff} = 1 - \frac{\sum(Q_{sim}(t) - Q_{obs}(t))^2}{\sum(Q_{obs}(t) - Q_{obs\,(mean)})^2} \quad \text{(Eq.2)}$$

Finally, to compare with the validation datasets, we simulated the discharge for two years (2009 - 2010) using calibrated parameters with the highest R_{eff} value.

Table 7. 5 Model parameters and their ranges used in the Monte Carlo calibration procedure

Parameter	Explanation	Minimum	Maximum	Calibrated Parameters		Unit
Soil routine				Sore	Mashaa	
FC Vegetation Zone 1	Maximum of SM (storage in soil box)	100	550	221.87	366.58	mm
FC Vegetation Zone 2				545.47	423.97	
FC Vegetation Zone 3				392.23	201.09	
LP Vegetation Zone 1	The threshold for reduction of evaporation (SM/FC)	0.3	1	0.66	0.97	-
LP Vegetation Zone 2				0.90	0.94	
LP Vegetation Zone 3				0.56	0.58	
BETA Vegitation Zone 1	Shape coefficient	1	6	4.53	3.07	-
BETA Vegitation Zone 2				2.83	3.89	
BETA Vegitation Zone 3				3.51	3.78	
Response Routine						
K_0	Recession coefficient (upper box)	0.1	0.5	0.10	0.25	day^{-1}
K_1	Recession coefficient (upper box)	0.01	0.4	0.04	0.05	day^{-1}
K_2	Recession coefficient (lower box)	0.001	0.15	0.05	0.04	day^{-1}
PERC	Maximum flow from upper to the lower box	0	4	3.98	3.46	$mm\ d^{-1}$
UZL	Threshold for K_o outflow	0	70	62.32	66.65	mm
Routing Routine						
MAXBAS	Routing, length of the weighting function	1	7	1.93	2.49	day

7.3. Results and Discussion

7.3.1 HBV Model Results

The prepared precipitation, temperature, discharge, and evapotranspiration input data from each catchment are imported individually into the HBV-light model. The top 100 runs with the highest objective function were maintained after calibration with R_{eff}. As stated in Table 7.5 above, these are the high-scoring key calibrated parameter metrics for both catchments. These optimal parameter values with a time series spanning from January 01, 2009, to December 31, 2010, were used for the validation. The performance of the HBV model in simulating discharge during the validation time series is displayed in Table 7.6 below, with five performance metrics for each catchment.

Even though the overall result is not satisfactory, the model in the Sore watershed generates better results than the model in the Mashaa watershed. Since each watershed only has one meteorological station, the HBV model fails to capture peak flows. In Figure 7.4 below, the observed and modelled discharge time series are clearly shown.

Table 7. 6 The individual and average performance of the HBV model in two subcatchments

	Mashaa					Sore					Average				
	RMSE	MAE	MDAE	MAPE	R^2	RMSE	MAE	MDAE	MAPE	R^2	RMSE	MAE	MDAE	MAPE	R^2
HBV Physical Model	42.31	26.89	8.79	0.40	0.420	24.59	16.05	8.61	0.49	0.763	33.45	21.47	8.70	0.44	0.591

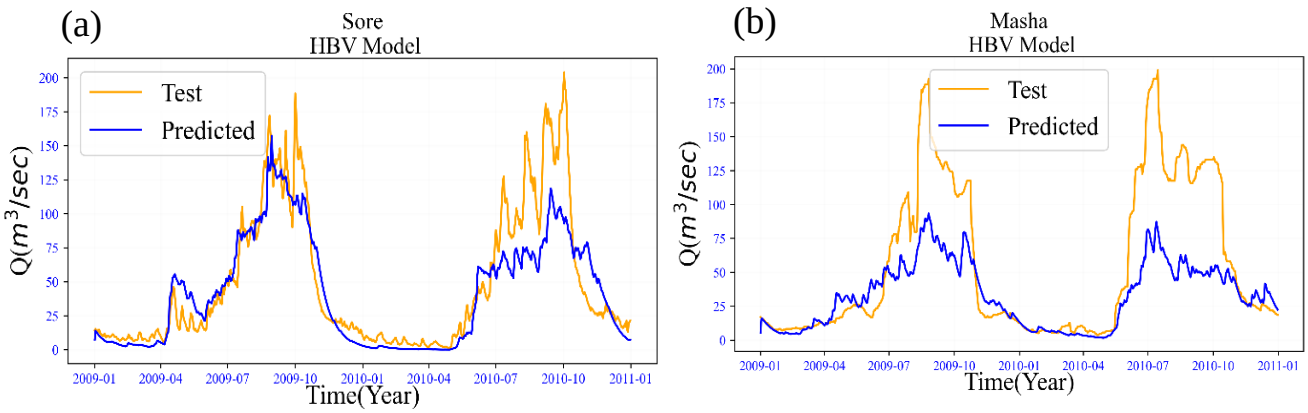


Figure 7. 4 Time series graph of observed and predicted values of the HBV model (a) Sore and (b) Mashaa catchments.

7.3.2 Superensemble Modelling with Fused Input Data

The eight base models and three meta-learners are evaluated using all fused datasets. Twenty-one variables are merged for the data-driven models: three remote sensing precipitation products for each of the six stations in each catchment and three areal vegetation indices. In contrast to the eight single machine learning base models, the three superensemble models, ETRSE, WASE, and BMASE, scored the highest R^2 values. These superensemble based data-driven models outperformed the HBV physical model for data-scarce catchments with insufficient meteorological data. Figure 7.5 depicts the observed and simulated discharge test time series for the top-ranked ETRSE model.

Table 7. 7 Individual and average performance of the eleven algorithms using all fused inputs and data from two subcatchments displayed with a heatmap.

	Mashaa					Sore					Average					
	RMSE	MAE	MEDAE	MAPE	R^2	RMSE	MAE	MEDAE	MAPE	R^2	RMSE	MAE	MEDAE	MAPE	R^2	RANk
ETRSE	24.64	16.26	8.39	0.39	0.803	19.68	13.02	7.18	0.49	0.848	22.16	14.64	7.79	0.44	0.826	1
WASE	23.05	16.61	10.64	0.55	0.828	22.79	15.89	9.8	0.82	0.796	22.92	16.25	10.22	0.69	0.812	2
BMASE	26.96	17.65	7.4	0.35	0.764	20.94	14.64	9.32	0.65	0.828	23.95	16.15	8.36	0.50	0.796	3
GRU	25.96	17.45	8.09	0.41	0.782	22.08	16.11	11.64	0.92	0.809	24.02	16.78	9.87	0.67	0.796	4
LSTM	27.27	18.77	9.65	0.45	0.76	21.25	14.45	7.33	0.54	0.823	24.26	16.61	8.49	0.50	0.792	5
XGB	28.99	19.09	8.37	0.37	0.728	21.65	15.05	8.98	0.49	0.816	25.32	17.07	8.68	0.43	0.772	6
CNN-GRU	25.48	15.89	6.93	0.32	0.789	25.25	17.15	8.61	0.49	0.75	25.37	16.52	7.77	0.41	0.770	7
MLP	20.67	16.03	14.14	0.75	0.862	29.41	17.41	7.68	0.48	0.661	25.04	16.72	10.91	0.62	0.762	8
SVR	30.24	21.19	12.98	0.6	0.703	26	17.97	11.11	0.69	0.735	28.12	19.58	12.05	0.65	0.719	9
LASSO	27.37	22.13	20.18	1.2	0.757	39.76	30.21	23.75	1.72	0.38	33.57	26.17	21.97	1.46	0.569	10
LR	27.53	22.29	20.08	1.21	0.754	40.18	30.54	24.46	1.75	0.367	33.86	26.42	22.27	1.48	0.561	11

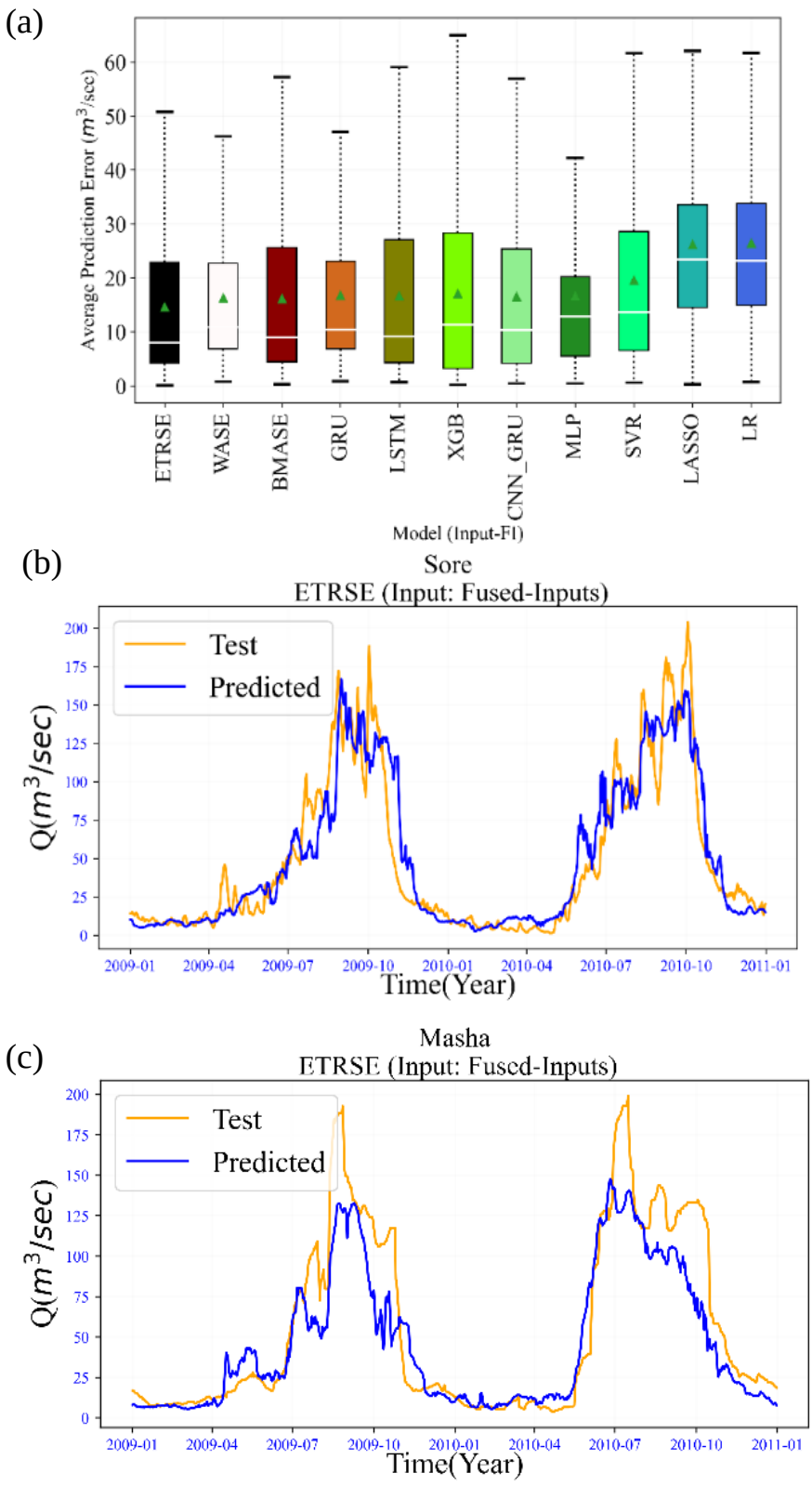


Figure 7. 5 Mean spread of prediction error (m^3/s) or box plot for the 11 models during the test period and fused inputs (a) and time series graph of observed and predicted values of the high score ETRSE model (b) Sore, and (c) Mashaa catchments.

7.3.3 Superensemble Modelling with Selected Input Data

Feature selection separates the essential inputs from the subset of the original datasets by eliminating unnecessary and redundant data. Hence strategies for speeding up training and increasing accuracy are classified as unsupervised, supervised, or semisupervised (Cai et al., 2018). The well-known supervised wrapper approach or recursive feature elimination (RFE) was purposefully selected for this experiment. When implementing RFE, the two configuration options are the number of features to choose and the best algorithm for feature selection. This feature selection strategy is simple to build and use (Lian et al., 2020). After some preliminary trials, we decided on ten variables and the decision tree regressor (DTR) as the optimization strategy.

As shown in Table 7.8, BMASE is ranked highest, followed by ETRSE, whose performance accuracy was improved by feature selection and is almost similar. The HBV-Light model exhibited the lowest average performance compared to other ensembled and single machine learning models. Figures 7.6 (b) and (c) demonstrate the contrast between the top-ranked model's real and simulated time series. Figure 7.6 (a) also depicts the average spread of prediction error for all algorithms using a box plot.

Table 7. 8 Eleven algorithms' individual and average performances using selected inputs and data from two subcatchments displayed with a heatmap.

	Mashaa					Sore					Average					
	RMSE	MAE	MDAE	MAPE	R^2	RMSE	MAE	MDAE	MAPE	R^2	RMSE	MAE	MDAE	MAPE	R^2	Rank
BMASE	21.61	14.18	6.91	0.36	0.849	21.29	13.58	6.96	0.39	0.822	21.45	13.88	6.94	0.38	0.836	1
ETRSE	19.55	12.18	6.10	0.31	0.876	22.98	14.59	7.40	0.46	0.793	21.27	13.39	6.75	0.39	0.835	2
WASE	22.41	14.47	5.94	0.34	0.837	21.45	14.21	8.09	0.49	0.819	21.93	14.34	7.02	0.42	0.828	3
GRU	22.56	15.25	8.33	0.40	0.835	24.24	14.61	5.26	0.35	0.769	23.40	14.93	6.80	0.38	0.802	4
CNN-GRU	26.67	17.45	9.72	0.35	0.769	23.13	14.89	6.83	0.45	0.790	24.90	16.17	8.28	0.40	0.780	5
LSTM	26.63	18.19	8.76	0.39	0.770	24.88	15.88	7.75	0.44	0.757	25.76	17.04	8.26	0.42	0.764	6
XGB	31.36	20.83	9.77	0.43	0.681	20.86	13.83	6.92	0.39	0.829	26.11	17.33	8.35	0.41	0.755	7
MLP	31.04	20.37	8.62	0.41	0.687	22.58	15.42	8.42	0.56	0.800	26.81	17.90	8.52	0.49	0.744	8
SVR	29.59	21.56	13.94	0.74	0.716	24.57	17.42	11.34	0.70	0.763	27.08	19.49	12.64	0.72	0.740	9
LASSO	25.24	17.89	11.47	0.71	0.794	28.67	22.20	19.18	1.19	0.678	26.96	20.05	15.33	0.95	0.736	10
LR	25.33	18.00	11.54	0.71	0.792	28.55	22.17	19.22	1.19	0.680	26.94	20.09	15.38	0.95	0.736	11
HBV Physical Model	42.31	26.89	8.79	0.40	0.420	24.59	16.05	8.61	0.49	0.763	33.45	21.47	8.70	0.44	0.591	12

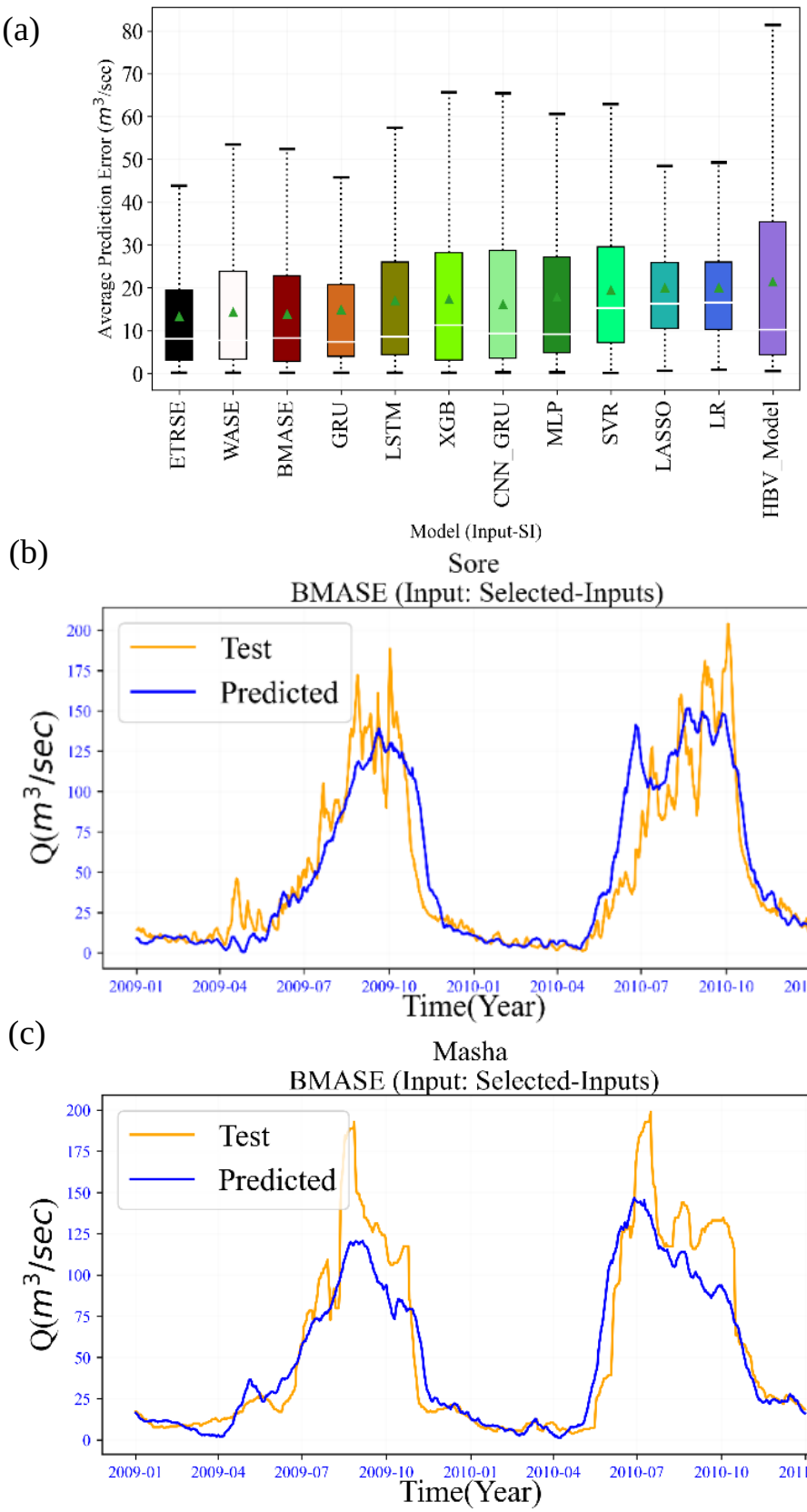


Figure 7. 6 Mean spread of prediction error (m^3/s) or box plot for the 12 models during the test period and selected inputs.

7.4. Conclusions

Our previous innovative study presented in chapter six aimed to estimate daily streamflow using a fusion of a remote sensing-based vegetation indice, a remote sensing-based precipitation product, and ground-gauged rainfall data. Superensemble learning was employed in the same investigation using three meta-learners (BMASE, ETRSE, and WASE) and eight base models (GRU, LSTM, MLP, CNN-GRU, SVR, Lasso, XGB, LR). This study examined the excellent performance of superensemble data-driven models to a semidistributed physical hydrological model (HBV-light). We selected two river catchments (Sore and Mashaa) in Ethiopia's Baro Akobo River basin due to our curiosity about modelling in data-limited regions, specifically meteorological data. The average performances of twelve algorithms in the two subcatchments are displayed with a heatmap in Table 7.9.

The following key results will summarize this study: -

I. According to the average performance of twelve models in two subcatchments, superensemble models offer high promise, with ETRSE, WASE, and BMASE ranking one to three using R^2 as a performance indicator. The top-ranked ETRSE model outperformed the HBV-light model by 24%, according to R^2.

II. The selected input scenario produces the highest average model performance score (BMASE, 0.836% R^2). This is consistent with our previous work, demonstrating that choosing influential input parameters positively impacts model performance.

III. These results suggest that modelling using conceptual hydrological models is difficult in data-scarce catchments since it requires large physical and meteorological parameter datasets. However, data-driven models, especially the superensemble models shown in this and preceding research, offer significant promise with different input variability possibilities and consistent performance.

Finally, the critical findings of this work should be broadened to incorporate agroclimatic variability, physical hydrological model variability, machine learning model variability, and optimal input data assimilation approaches. Furthermore, we invite the scientific community to advance these findings using a novel approach and the aforementioned variable situations.

Table 7. 9 The average performances of twelve algorithms in two subcatchments are displayed with a heatmap.

Model	RMSE	MAE	MEDAE	MAPE	R^2	Rank
ETRSE	21.81	14.26	7.36	0.41	0.831	1
WASE	22.09	14.82	8.49	0.54	0.823	2
BMASE	22.94	15.24	7.69	0.46	0.812	3
GRU	23.71	15.86	8.33	0.52	0.799	4
LSTM	24.58	16.39	8.38	0.45	0.786	5
XGB	25.54	17.05	8.47	0.42	0.768	6
CNN-GRU	25.74	16.93	8.06	0.41	0.762	7
MLP	25.93	17.31	9.72	0.55	0.753	8
SVR	27.60	19.54	12.34	0.68	0.729	9
LASSO	30.26	23.11	18.65	1.21	0.652	10
LR	30.40	23.25	18.83	1.22	0.648	11
HBV Physical Model	33.45	21.47	8.70	0.44	0.591	12

CHAPTER 8:

CONCLUSION AND RECOMMENDATION

8.1 Conclusions

This study aims to create novel deep-learning algorithms for investigating and simulating continuous streamflow time series using ground and remote sensing data sources. For this purpose, we arranged and presented the dissertation into four distinct objectives.

First, in chapter four, we started with a simple case study that evaluated the potential of single machine learning models for univariate streamflow forecasting. This chapter compares several deep learning algorithms for one-step daily streamflow forecasting at two subcatchment streamflow outlets. The four algorithms utilized in this work are MLP, S-LSTM, Bi-LSTM, and GRU. The research convincingly demonstrated the effects of climate (time series characteristics) and delayed temporal variability on the performance of several proposed deep-learning models.

Second, in Chapter five, we broaden the study's data sources from univariate to multivariate. The machine learning models we used advanced to hybrid deep learning models for one-step daily streamflow modelling at the two river basin subcatchments streamflow outlets. The CNN-LSTM and CNN-GRU hybrid deep learning models proposed in this study have one or two 1D CNN layers, as do the standard MLP, LSTM, and GRU models. This study conducted a series of tests to investigate the performance variation of the suggested models for streamflow simulation by varying input combinations, rolling time frames, and climatic variables.

Third, in chapter six, we proposed a novel superensemble deep learning model to assimilate remote sensing-based vegetation indice, precipitation product, and ground gauge rainfall data. Five input scenarios are designed to effectively comprehend the performance enhancements caused by these inputs: only vegetation indice, remote sensing-based precipitation, ground gauged precipitation, all fused inputs, and selected input variables. To train the input scenarios, we used superensemble learning with three separate meta-learners (BMASE, ETRSE, WASE) and eight base models (GRU, LSTM, MLP, CNN-GRU, SVR, LASSO, XGB, LR). We identified three Ethiopian river basin subcatchments with reasonably excellent datasets (Gummera and Borkena) and the Sore watershed with limited meteorological data as case studies. We employed 12 years of time series

(2003-2014) for all investigations, rolled the predictor variables monthly, and forecasted one-step streamflow value using data from the preceding thirty days.

Finally, in chapter seven, we carefully examined the superior performance of the superensemble data-driven models of the previous chapter over a well-known semidistributed physical hydrological model (HBV-light). We demonstrated the potential of ensemble deep learning models by simulating streamflow time series from two river catchments (Sore and Mashaa).

The following key points will conclude the findings of this study.

I. Catchment characteristics impacted univariate streamflow forecasting performance more than deep learning model structures and lagged time variations.

II. We should consider the variability of our parameters when choosing deep learning models for streamflow simulation. Single machine learning models are advised for univariate and small-scale streamflow data modelling and sophisticated ensemble deep learning models for catchments with various data sources, parameter options, and long time series.

III. According to this study, vegetation indices have the potential to be employed for data-driven streamflow modelling, especially in data-scarce catchments with no meteorological time series. However, more research on this outcome will be expected utilizing novel data assimilation approaches.

IV. This study also shows that one of LSTM's major flaws is that its performance degrades when feature selection criteria are not employed. XGB, on the other hand, performed better after controlling redundant inputs internally. Furthermore, even if certain single models outperform others in a specific catchment, this performance is not transferable to other catchments. On the contrary, superensemble models demonstrated consistent performance across all catchments and input circumstances.

V. While additional advanced investigations are needed to prove ensemble deep learning models' superiority to conceptual hydrological models, this study's findings are a start in the right direction for these scholarly arguments. Therefore, for catchments with little to no meteorological data, we advise using the promises in the superensemble deep learning model rather than the conceptual hydrological models, which need a large amount of physical data.

8.2 **Recommendations**

Deep learning models have advanced our understanding, analysis, and simulation of streamflow time series utilizing ground and remote sensing data sources. At this point, we would want to suggest policymakers, stakeholders in the water sector, research institutes or universities, and future young researchers in the field. However, this work is still open to further research that will enhance the findings.

I. As a result of combining the advantages of each model, proving consistency across all catchments, and reducing uncertainty, we can comprehend the promise of superensemble deep-learning models in this work. Thus, we suggest this approach as a front-line data-driven streamflow modelling tool.

II. The debate remains about the promise of data-driven models over physical hydrological models. Future studies should examine and design novel methodologies that integrate these two strategies' benefits and reduce uncertainty.

III. In addition to installing new ground-based streamflow gauging stations, we recommend that the ministry of water resources and energy re-establish a hydrology sub-institute and strengthen its capacity to organize, research, and transfer knowledge in the field to the industry by leveraging the vast amount of data stored in remote sensing to supplement the design and implementation of water resource monitoring systems.

IV. Under various uncertain circumstances, technologically advanced monitoring of our water resources for the welfare of humanity urges focused attention in terms of financial resources, knowledge incubation, development, and transfer to the industry from various stakeholders, including governments, private or social groups, research institutions, and universities. Hence, we advise that all parties collaborate and build a system until we monitor every drop of water that touches the land.

Satellite monitoring of the Earth's surface is a demanding research field. Hence, we encourage young scholars to examine new data assimilation strategies for optimum hydrological modelling. Moreover, in the age of artificial intelligence, deploying machine learning models for this purpose is a futuristic endeavor that we will never ignore.

REFERENCES

Abdelminaam, D. S., Ismail, F. H., Taha, M., Taha, A., Houssein, E. H., & Nabil, A. (2021). CoAID-DEEP: An Optimized Intelligent Framework for Automated Detecting COVID-19 Misleading Information on Twitter. *IEEE Access*, *9*, 27840–27867. https://doi.org/10.1109/ACCESS.2021.3058066

Abera, F. F., Arega, S., & Gedamu, B. H. (2020). Climate Change Induced Precipitation and Temperature Effects on Water Resources: The Case of Borkena Watershed in the Highlands of Wollo, Central Ethiopia. *Water Conservation Science and Engineering*, *5*(1), 53–66. https://doi.org/10.1007/s41101-020-00084-8

Adeyemo, J., Oyebode, O., & Stretch, D. (2018). River Flow Forecasting using an Improved Artificial Neural Network. In A.-A. Tantar, E. Tantar, M. Emmerich, P. Legrand, L. Alboaie, & H. Luchian (Eds.), *Evolve a bridge between probability, set oriented numerics, and evolutionary computation vi* (Vol. 674, pp. 179–193). Springer International Publishing. https://doi.org/10.1007/978-3-319-69710-9_13

Ahmed, A. A. M., Deo, R. C., Raj, N., Ghahramani, A., Feng, Q., Yin, Z., & Yang, L. (2021). Deep Learning Forecasts of Soil Moisture: Convolutional Neural Network and Gated Recurrent Unit Models Coupled with Satellite-Derived MODIS, Observations and Synoptic-Scale Climate Indice Data. *Remote Sensing*, *13*(4), Article 4. https://doi.org/10.3390/rs13040554

Alemayehu, T., McCartney, M., & Kebede, S. (2010). The Water Resource Implications of Planned Development in the Lake Tana Catchment, Ethiopia. *Ecohydrology & Hydrobiology*, *10*(2), 211–221. https://doi.org/10.2478/v10104-011-0023-6

Aljahdali, S., Sheta, A., & Turabieh, H. (2020). River Flow Forecasting: A Comparison Between Feedforward and Layered Recurrent Neural Network. *Innovation in Information Systems and Technologies to Support Learning Research*, 523–532. https://doi.org/10.1007/978-3-030-36778-7_58

Alquraish, M. M., & Khadr, M. (2021). Remote-Sensing-Based Streamflow Forecasting Using Artificial Neural Network and Support Vector Machine Models. *Remote Sensing*, *13*(20), Article 20. https://doi.org/10.3390/rs13204147

Althelaya, K. A., El-Alfy, E.-S. M., & Mohammed, S. (2018). Stock Market Forecast Using Multivariate Analysis with Bidirectional and Stacked (LSTM, GRU). *2018 21st Saudi Computer Society National Computer Conference (NCC)*, 1–7. https://doi.org/10.1109/NCG.2018.8593076

Andronache, C. (2018). Remote Sensing of Clouds and Precipitation. Springer.

Annis, A., & Nardi, F. (2019). Integrating VGI and 2d Hydraulic Models into a Data Assimilation Framework for Real Time Flood Forecasting and Mapping. *Geo-Spatial Information Science*, *22*(4), 223–236. https://doi.org/10.1080/10095020.2019.1626135

Apaydin, H., Feizi, H., Sattari, M. T., Colak, M. S., Shamshirband, S., & Chau, K.-W. (2020). Comparative Analysis of Recurrent Neural Network Architectures for Reservoir Inflow Forecasting. *Water*, *12*(5), Article 5. https://doi.org/10.3390/w12051500

Awad, M., & Khanna, R. (2015). Support Vector Regression. In M. Awad & R. Khanna (Eds.), *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers* (pp. 67–80). Apress. https://doi.org/10.1007/978-1-4302-5990-9_4

Awulachew, S. B., McCartney, M., Steenhuis, T. S., & Ahmed, A. A. (2009). *A review of hydrology, sediment and water resource use in the Blue Nile Basin*. IWMI.

Awulachew, S. B., Yilma, A. D., Loulseged, M., Loiskandl, W., Ayana, M., & Alamirew, T. (2007). Water Resources and Irrigation Development in Ethiopia. IWMI.

Baćak, V., & Kennedy, E. H. (2019). Principled Machine Learning Using the Superlearner: An Application to Predicting Prison Violence. *Sociological Methods & Research*, *48*(3), 698–721. https://doi.org/10.1177/0049124117747301

Bai, P., Liu, X., & Xie, J. (2021). Simulating Runoff under Changing Climatic Conditions: A Comparison of the Long Short-Term Memory Network with Two Conceptual Hydrologic Models. *Journal of Hydrology*, *592*, 125779. https://doi.org/10.1016/j.jhydrol.2020.125779

Bai, Y., Bezak, N., Sapač, K., Klun, M., & Zhang, J. (2019). Short-Term Streamflow Forecasting Using the Feature-Enhanced Regression Model. *Water Resources Management*, *33*(14), 4783–4797. https://doi.org/10.1007/s11269-019-02399-1

Bai, Y., Bezak, N., Zeng, B., Li, C., Sapač, K., & Zhang, J. (2021). Daily Runoff Forecasting Using a Cascade Long Short-Term Memory Model that Considers Different Variables. *Water Resources Management*, *35*(4), 1167–1181. https://doi.org/10.1007/s11269-020-02759-2

Barzegar, R., Aalami, M. T., & Adamowski, J. (2020). Short-Term Water Quality Variable Prediction Using a Hybrid CNN–LSTM Deep Learning Model. *Stochastic Environmental Research and Risk Assessment*, *34*(2), 415–433. https://doi.org/10.1007/s00477-020-01776-2

Bates, J. M., & Granger, C. W. J. (1969). The Combination of Forecasts*: Journal of the Operational Research Society*. *20*. https://doi.org/10.1057/jors.1969.103

Beck, H. E., Vergopolan, N., Pan, M., Levizzani, V., van Dijk, A. I. J. M., Weedon, G. P., Brocca, L., Pappenberger, F., Huffman, G. J., & Wood, E. F. (2017). Global-Scale Evaluation of 22 Precipitation Datasets Using Gauge Observations and Hydrological Modeling. *Hydrology and Earth System Sciences*, *21*(12), 6201–6217. https://doi.org/10.5194/hess-21-6201-2017

Beck, H. E., Wood, E. F., Pan, M., Fisher, C. K., Miralles, D. G., Dijk, A. I. J. M. van, McVicar, T. R., & Adler, R. F. (2019). MSWEP V2 Global 3-Hourly 0.1° Precipitation: Methodology and Quantitative Assessment. *Bulletin of the American Meteorological Society*, *100*(3), 473–500. https://doi.org/10.1175/BAMS-D-17-0138.1

Bergström, S. (1976). Development and Application of a Conceptual Runoff Model for Scandinavian Catchments. http://urn.kb.se/resolve?urn=urn:nbn:se:smhi:diva-5738

Bergström, S. (1992). The HBV Model – Its Structure and Applications. SMHI. http://urn.kb.se/resolve?urn=urn:nbn:se:smhi:diva-2672

Birkinshaw, S. J., Moore, P., Kilsby, C. G., O'Donnell, G. M., Hardy, A. J., & Berry, P. A. M. (2014). Daily Discharge Estimation at Ungauged River Sites Using Remote Sensing. *Hydrological Processes*, *28*(3), 1043–1054. https://doi.org/10.1002/hyp.9647

Biswas. (1970). History of Hydrology. http://www.history-of-hydrology.net/mediawiki/indice.php?title=Biswas_(1970)_History_of_Hydrology

Bjerklie, D. M., Birkett, C. M., Jones, J. W., Carabajal, C., Rover, J. A., Fulton, J. W., & Garambois, P.-A. (2018). Satellite Remote Sensing Estimation of River Discharge: Application to the Yukon River Alaska. *Journal of Hydrology*, *561*, 1000–1018. https://doi.org/10.1016/j.jhydrol.2018.04.005

Blumenfeld, J. (2015). From TRMM to GPM: The Evolution of NASA Precipitation Data. https://earthdata.nasa.gov/learn/articles/trmm-to-gpm

Bourdin, D. R., Fleming, S. W., & Stull, R. B. (2012). Streamflow Modelling: A Primer on Applications, Approaches and Challenges. *Atmosphere-Ocean*, *50*(4), 507–536. https://doi.org/10.1080/07055900.2012.734276

Bowden, G. J., Dandy, G. C., & Maier, H. R. (2005). Input Determination for Neural Network Models in Water Resources Applications. Part 1—Background and Methodology. *Journal of Hydrology*, *301*(1), 75–92. https://doi.org/10.1016/j.jhydrol.2004.06.021

Boyraz, C., & Engin, Ş. N. (2018). Streamflow Prediction with Deep Learning. *6th International Conference on Control Engineering Information Technology (CEIT)*, 1–5. https://doi.org/10.1109/CEIT.2018.8751915

Bui, H. T., Ishidaira, H., & Shaowei, N. (2019). Evaluation of the Use of Global Satellite–Gauge and Satellite-Only Precipitation Products in Streamflow Simulations. *Applied Water Science*, *9*(3), 53. https://doi.org/10.1007/s13201-019-0931-y

Burges, S. J. (2004). History of the Stanford Watershed Model. *ResearchGate*. https://www.researchgate.net/publication/242220953_HISTORY_OF_THE_STANFORD_WATERSHED_MODEL

Cai, J., Luo, J., Wang, S., & Yang, S. (2018). Feature Selection in Machine Learning: A New Perspective. *Neurocomputing*, *300*, 70–79. https://doi.org/10.1016/j.neucom.2017.11.077

Campos, L. C. D., Goliatt da Fonseca, L., Fonseca, T. L., de Abreu, G. D., Pires, L. F., & Gorodetskaya, Y. (2019). Short-Term Streamflow Forecasting for Paraíba do Sul River Using Deep Learning. *Progress in Artificial Intelligence*, 507–518. https://doi.org/10.1007/978-3-030-30241-2_43

Capolongo, D., Refice, A., Bocchiola, D., D'Addabbo, A., Vouvalidis, K., Soncini, A., Zingaro, M., Bovenga, F., & Stamatopoulos, L. (2019). Coupling Multitemporal Remote Sensing with Geomorphology and Hydrological Modeling for Post Flood Recovery in the Strymonas Dammed River Basin (Greece). *Science of The Total Environment*, *651*, 1958–1968. https://doi.org/10.1016/j.scitotenv.2018.10.114

Carmona, P., Climent, F., & Momparler, A. (2019). Predicting Failure in the U.S. Banking Sector: An Extreme Gradient Boosting Approach. *International Review of Economics & Finance*, *61*, 304–323. https://doi.org/10.1016/j.iref.2018.03.008

Cazenave, A., Champollion, N., Benveniste, J., & Chen, J. (Eds.). (2016). Remote Sensing and Water Resources (Vol. 55). Springer.

Chen, C., He, W., Li, J., & Tang, Z. (2020). A Novel Hybrid CNN-LSTM Scheme for Nitrogen Oxide Emission Prediction in FCC Unit. *Mathematical Problems in Engineering*, pp.1-12. https://doi.org/10.1155/2020/8071810

Chen, L., & Wang, L. (2018). Recent Advance in Earth Observation Big Data for Hydrology. *Big Earth Data*, *2*(1), 86–107. https://doi.org/10.1080/20964471.2018.1435072

Cho, K., van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, 103–111. https://doi.org/10.3115/v1/W14-4012

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *ArXiv:1406.1078 [Cs, Stat]*. http://arxiv.org/abs/1406.1078

Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. *Machine Learning*, *20*(3), 273–297. https://doi.org/10.1007/BF00994018

Couta, D., Zhang, Y., & Li, Y. (2019). River Flow Forecasting Using Long Short-term Memory. *DEStech Transactions on Computer Science and Engineering*, Article icaic. https://doi.org/10.12783/dtcse/icaic2019/29416

Cui, F., Salih, S. Q., Choubin, B., Bhagat, S. K., Samui, P., & Yaseen, Z. M. (2020). Newly Explored Machine Learning Model for River Flow Time Series Forecasting at Mary River, Australia.

Environmental Monitoring and Assessment, *192*(12), 761. https://doi.org/10.1007/s10661-020-08724-1

Dabhade, P., Agarwal, R., Alameen, K. P., Fathima, A. T., Sridharan, R., & Gopakumar, G. (2021). Educational Data Mining for Predicting Students' Academic Performance Using Machine Learning Algorithms. *Materials Today: Proceedings*, *47*, 5260–5267. https://doi.org/10.1016/j.matpr.2021.05.646

Danandeh Mehr, A., & Safari, M. J. S. (2021). Genetic Programming for Streamflow Forecasting: A Concise Review of Univariate Models with a Case Study. In P. Sharma & D. Machiwal (Eds.), *Advances in Streamflow Forecasting* (pp. 193–214). Elsevier. https://doi.org/10.1016/B978-0-12-820673-7.00007-X

Dehghani, R., Torabi Poudeh, H., & Izadi, Z. (2021). Dissolved Oxygen Concentration Predictions for Running Waters With using Hybrid Machine Learning Techniques. *Modeling Earth Systems and Environment*. https://doi.org/10.1007/s40808-021-01253-x

Devia, G. K., Ganasri, B. P., & Dwarakish, G. S. (2015). A Review on Hydrological Models. *Aquatic Procedia*, *4*, 1001–1007. https://doi.org/10.1016/j.aqpro.2015.02.126

Din, A. Z. U., Ayaz, Y., Hasan, M., Khan, J., & Salman, M. (2019). Bivariate Short-term Electric Power Forecasting using LSTM Network. *2019 International Conference on Robotics and Automation in Industry (ICRAI)*, 1–8. https://doi.org/10.1109/icrai47710.2019.8967378

Enku, T., & M. Melesse, A. (2014). A Simple Temperature Method for the Estimation of Evapotranspiration. *Hydrological Processes*. *28*(6), 2945–2960. https://doi.org/10.1002/hyp.9844

Essien, A., & Giannetti, C. (2019). A Deep Learning Framework for Univariate Time Series Prediction Using Convolutional LSTM Stacked Autoencoders. *IEEE International Symposium on Inovations in Intelligent Systems and Applications (INISTA)*, 1–6. https://doi.org/10.1109/INISTA.2019.8778417

Faizollahzadeh, S. ardabili, Mosavi, A., Dehghani, M., & R, A. V.-K. (2019). Deep Learning and Machine Learning in Hydrological Processes, Climate Change and Earth Systems: A Systematic Review. In *Engineering for Sustainable Future: Selected papers of the 18th International Conference on Global Research and Education Inter-Academia* (pp. 52-62). Springer International Publishing. https://doi.org/10.20944/preprints201908.0166.v1

Fereidoon, M., Koch, M., & Brocca, L. (2019). Predicting Rainfall, and Runoff Through Satellite Soil Moisture Data and Swat Modelling for a Poorly Gauged Basin in Iran. *Water*, *11*(3), 594. https://doi.org/10.3390/w11030594

Fiseha, B. M., Setegn, S. G., Melesse, A. M., Volpi, E., & Fiori, A. (2014). Impact of Climate Change on the Hydrology of Upper Tiber River Basin Using Bias Corrected Regional Climate Model. *Water Resources Management*, *28*(5), 1327–1343. https://doi.org/10.1007/s11269-014-0546-x

Fleig, A. K., Tallaksen, L. M., Hisdal, H., & Hannah, D. M. (2011). Regional Hydrological Drought in North-Western Europe: Linking a New Regional Drought Area Indice with Weather Types. *Hydrological Processes*, *25*(7), 1163–1179. https://doi.org/10.1002/hyp.7644

Gamage, N., Agrawal, R., Smakhtin, V. U., & Perera, B. J. C. (2011). An Artificial Neural Network Model for Simulating Streamflow using Remote Sensing Data. In *Proceedings of the 34th World Congress of the International Association for Hydro-Environment Research and Engineering: 33rd Hydrology and Water Resources Symposium and 10th Conference on Hydraulics in Water Engineering: 33rd Hydrology and Water Resources Symposium and 10th Conference on Hydraulics in Water Engineering* (pp. 1371-1378). Barton, ACT: Engineers Australia. https://cgspace.cgiar.org/handle/10568/38461

Gao, S., Huang, Y., Zhang, S., Han, J., Wang, G., Zhang, M., & Lin, Q. (2020). Short-Term Runoff Prediction with GRU and LSTM Networks without Requiring Time Step Optimization During Sample Generation. *Journal of Hydrology*, *589*, 125188. https://doi.org/10.1016/j.jhydrol.2020.125188

Gass, S. I. (1983). Decision Aiding Models: Validation, Assessment, and Related Issues for Policy Analysis. *Operations Research*, *31*(4), 603–631. https://doi.org/10.1287/opre.31.4.603

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely Randomized Trees. *Machine Learning*, *63*(1), 3–42. https://doi.org/10.1007/s10994-006-6226-1

Ghaith, M., Siam, A., Li, Z., & El-Dakhakhni, W. (2020). Hybrid Hydrological Data-Driven Approach for Daily Streamflow Forecasting. *Journal of Hydrologic Engineering*, *25*(2), 04019063. https://doi.org/10.1061/(ASCE)HE.1943-5584.0001866

Gleason, C. J., Smith, L. C., & Lee, J. (2014). Retrieval of River Discharge Solely from Satellite Imagery and at-Many-Stations Hydraulic Geometry: Sensitivity to River form and Optimization Parameters. *Water Resources Research*, *50*(12), 9604–9619. https://doi.org/10.1002/2014WR016109

Gnauck, A. (2004). Interpolation and Approximation of Water Quality Time Series and Process Identification. *Analytical and Bioanalytical Chemistry*, *380*(3), 484–492. https://doi.org/10.1007/s00216-004-2799-3

Gunathilake, M. B., Karunanayake, C., Gunathilake, A. S., Marasingha, N., Samarasinghe, J. T., Bandara, I. M., & Rathnayake, U. (2021). Hydrological Models and Artificial Neural Networks (ANNs) to Simulate Streamflow in a Tropical Catchment of Sri Lanka. *Applied Computational Intelligence and Soft Computing*, *2021*, e6683389. https://doi.org/10.1155/2021/6683389

Gunnar, R. (2004). A Brief Introduction into Machine Learning. *Friedrich Miescher Laboratory of the Max Planck Society*, 1-6.

Hamm, L., Brorsen, B. W., & Hagan, M. T. (2007). Comparison of Stochastic Global Optimization Methods to Estimate Neural Network Weights. *Neural Processing Letters*, *26*(3), 145–158. https://doi.org/10.1007/s11063-007-9048-7

Hamzah, F. B., Hamzah, F. M., Razali, S. F. M., & Samad, H. (2021). A Comparison of Multiple Imputation Methods for Recovering Missing Data in Hydrological Studies. *Civil Engineering Journal*, *7*(9), Article 9. https://doi.org/10.28991/cej-2021-03091747

Hinne, M., Gronau, Q. F., Bergh, D. van den, & Wagenmakers, E.-J. (2020). A Conceptual Introduction to Bayesian Model Averaging: *Advances in Methods and Practices in Psychological Science*. https://doi.org/10.1177/2515245919898657

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hu, Q., Li, Z., Wang, L., Huang, Y., Wang, Y., & Li, L. (2019). Rainfall Spatial Estimations: A Review from Spatial Interpolation to Multi-Source Data Merging. *Water*, *11*(3), 579. https://doi.org/10.3390/w11030579

Huang, C., Chen, Y., Zhang, S., & Wu, J. (2018). Detecting, Extracting, and Monitoring Surface Water from Space Using Optical Sensors: A Review. *Reviews of Geophysics*, *56*(2), 333–360. https://doi.org/10.1029/2018RG000598

Hussain, M. M., Bari, S. H., Mahmud, I., & Siddiquee, M. I. H. (2021). Chapter 5 - Application of Different Artificial Neural Network for Streamflow Forecasting. In P. Sharma & D. Machiwal (Eds.), *Advances in Streamflow Forecasting* (pp. 149–170). Elsevier. https://doi.org/10.1016/B978-0-12-820673-7.00006-8

Ibrahim, K. S. M. H., Huang, Y. F., Ahmed, A. N., Koo, C. H., & El-Shafie, A. (2022). A Review of the Hybrid Artificial Intelligence and Optimization Modelling of Hydrological Streamflow Forecasting. *Alexandria Engineering Journal*, *61*(1), 279–303. https://doi.org/10.1016/j.aej.2021.04.100

Jain, M., Manandhar, S., Lee, Y. H., Winkler, S., & Dev, S. (2020). Forecasting Precipitable Water Vapor Using LSTMs. *ArXiv*. https://app.dimensions.ai/details/publication/pub.1128846873

Jaiswal, R. K., Ali, S., & Bharti, B. (2020). Comparative Evaluation of Conceptual and Physical Rainfall–Runoff Models. *Applied Water Science*, *10*(1), 48. https://doi.org/10.1007/s13201-019-1122-6

Jiang, D., & Wang, K. (2019). The Role of Satellite-Based Remote Sensing in Improving Simulated Streamflow: A Review. *Water*, *11*(8), 1615. https://doi.org/10.3390/w11081615

Kazakov, O. D., & Mikheenko, O. V. (2020). Transfer Learning and Domain Adaptation Based on Modeling of Socio-Economic Systems. *Business Informatics*, *14*(2 (eng)), Article 2 (eng).

Keith J Beven. (2005). Rainfall-Runoff Modeling: Introduction. *Encyclopedia of Hydrological Sciences*.

https://www.researchgate.net/publication/229703907_Rainfall-Runoff_Modeling_Introduction

Khan, A. (2000). History of Remote Sensing and GIS. https://www.academia.edu/10815020/History_of_Remote_Sensing_and_GIS

Khorram, S., Wiele, C. F. van der, Koch, F. H., Nelson, S. A. C., & Potts, M. D. (2016). Principles of Applied Remote Sensing (pp. 21-31). New York: Springer.

Kim, D. Y., & Song, C. M. (2020). Developing a Discharge Estimation Model for Ungauged Watershed Using CNN and Hydrological Image. *Water*, *12*(12), Article 12. https://doi.org/10.3390/w12123534

Kim, T., Yang, T., Gao, S., Zhang, L., Ding, Z., Wen, X., Gourley, J. J., & Hong, Y. (2021). Can Artificial Intelligence and Data-Driven Machine Learning Models Match or Even Replace Process-Driven Hydrologic Models for Streamflow Simulation?: A Case Study of Four Watersheds with Different Hydro-Climatic Regions Across the CONUS. *Journal of Hydrology*, *598*, 126423. https://doi.org/10.1016/j.jhydrol.2021.126423

Konar, J., Khandelwal, P., & Tripathi, R. (2020). Comparison of Various Learning Rate Scheduling Techniques on Convolutional Neural Network. *2020 IEEE International Students' Conference on Electrical,Electronics and Computer Science (SCEECS)*, 1–5. https://doi.org/10.1109/SCEECS48394.2020.94

Kukreja, S. L., Löfberg, J., & Brenner, M. J. (2006). A Least Absolute Shrinkage and Selection Operator (Lasso) for Nonlinear System Identification. *IFAC Proceedings Volumes*, *39*(1), 814–819. https://doi.org/10.3182/20060329-3-AU-2901.00128

Kumar, S., Hussain, L., Banarjee, S., & Reza, M. (2018). Energy Load Forecasting using Deep Learning Approach-LSTM and GRU in Spark Cluster. *2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT)*, 1–4.https://doi.org/10.1109/EAIT.2018.8470406

Kumar, S., Roshni, T., & Himayoun, D. (2019). A Comparison of Emotional Neural Network (ENN) and Artificial Neural Network (ANN) Approach for Rainfall-Runoff Modelling. *Civil Engineering Journal*, *5*(10), 2120-2130–2130. https://doi.org/10.28991/cej-2019-03091398

Kun, R., Wei, F., Jihong, Q., Xia, Z., & Xiaoyu, S. (2020). Comparison of Eight Filter-Based Feature Selection Methods for Monthly Streamflow Forecasting – Three Case Studies on CAMELS Datasets. *Journal of Hydrology*, *586*, 124897. https://doi.org/10.1016/j.jhydrol.2020.124897

Kwan, C., Qi, H., & Tran, T. (2016). Recent Advances in Remote Spectral Sensing. *Journal of Sensors*, *2016*, 1–2. https://doi.org/10.1155/2016/6125729

Laan, M. J. van der, Polley, E. C., & Hubbard, A. E. (2007). Superlearner. *Statistical Applications in Genetics and Molecular Biology*, *6*(1). https://doi.org/10.2202/1544-6115.1309

Lara-Benítez, P., Carranza-García, M., & Riquelme, J. C. (2021). An Experimental Review on Deep Learning Architectures for Time Series Forecasting. *International Journal of Neural Systems*, *31*(03), 2130001. https://doi.org/10.1142/S0129065721300011

Li, C., Tang, G., & Hong, Y. (2018). Cross-Evaluation of Ground-Based, Multi-Satellite and Reanalysis Precipitation Products: Applicability of the Triple Collocation Method Across Mainland China. *Journal of Hydrology*, *562*, 71–83. https://doi.org/10.1016/j.jhydrol.2018.04.039

Li, T., Hua, M., & Wu, X. (2020). A Hybrid CNN-LSTM Model for Forecasting Particulate Matter (PM2.5). *IEEE Access*, *8*, 26933–26940. https://doi.org/10.1109/access.2020.2971348

Li, Y., Liang, Z., Hu, Y., Li, B., Xu, B., & Wang, D. (2019). A Multi-Model Integration Method for Monthly Streamflow Prediction: Modified Stacking Ensemble Strategy. *Journal of Hydroinformatics*, *22*(2), 310–326. https://doi.org/10.2166/hydro.2019.066

Lian, W., Nie, G., Jia, B., Shi, D., Fan, Q., & Liang, Y. (2020). An Intrusion Detection Method Based on Decision Tree-Recursive Feature Elimination in Ensemble Learning. *Mathematical Problems in Engineering*, *2020*, e2835023. https://doi.org/10.1155/2020/2835023

Liang, Z., Wang, D., Guo, Y., Zhang, Y., & Dai, R. (2013). Application of Bayesian Model Averaging Approach to Multimodel Ensemble Hydrologic Forecasting. *Journal of Hydrologic Engineering*, *18*(11), 1426–1436. https://doi.org/10.1061/(ASCE)HE.1943-5584.0000493

Lillesand, T. M., Kiefer, R. W., & Chipman, J. W. (2015). Remote Sensing and Image Interpretation. *John Wiley & Sons.*

Liu, L., Liu, X., Bai, P., Liang, K., & Liu, C. (2022). Comparison of Flood Simulation Capabilities of a Hydrologic Model and a Machine Learning Model. *International Journal of Climatology*, *n/a*(n/a). https://doi.org/10.1002/joc.7738

Liu, Y., Zhang, T., Kang, A., Li, J., & Lei, X. (2021). Research on Runoff Simulations Using Deep-Learning Methods. *Sustainability*, *13*(3), Article 3. https://doi.org/10.3390/su13031336

Livieris, I. E., Pintelas, E., & Pintelas, P. (2020). A CNN–LSTM Model for Gold Price Time-Series Forecasting. *Neural Computing and Applications*. https://doi.org/10.1007/s00521-020-04867-x

M L Tan, Z. D. (2014). Streamflow Modelling by Remote Sensing: A Contribution to Digital Earth. In *IOP Conference Series: Earth and Environmental Science* (Vol. 18, No. 1, p. 012060). IOP Publishing.

Macharia, J. M., Ngetich, F. K., & Shisanya, C. A. (2020). Comparison of Satellite Remote Sensing Derived Precipitation Estimates and Observed Data in Kenya. *Agricultural and Forest Meteorology*, *284*, 107875. https://doi.org/10.1016/j.agrformet.2019.107875

Maggioni, V., & Massari, C. (2018). On the Performance of Satellite Precipitation Products in Riverine Flood Modeling: A Review. *Journal of Hydrology*, *558*, 214–224. https://doi.org/10.1016/j.jhydrol.2018.01.039

Maier, H. R., Jain, A., Dandy, G. C., & Sudheer, K. P. (2010). Methods Used for the Development of Neural Networks for the Prediction of Water Resource Variables in River Systems: Current Status and Future Directions. *Environmental Modelling & Software*, *25*(8), 891–909. https://doi.org/10.1016/j.envsoft.2010.02.003

Manning, R. (1816). History of Hydrology. http://www.history-of-hydrology.net/mediawiki/indice.php?title=Manning,_Robert

Mehr, A. D., & Gandomi, A. H. (2021). MSGP-LASSO: An Improved Multi-Stage Genetic Programming Model for Streamflow Prediction. *Information Sciences*, *561*, 181–195. https://doi.org/10.1016/j.ins.2021.02.011

Meresa, H. (2019). Modelling of River Flow in Ungauged Catchment using Remote Sensing Data: Application of the Empirical (SCS-CN), Artificial Neural Network (ANN) and Hydrological Model (HEC-HMS). *Modeling Earth Systems and Environment*, *5*(1), 257–273. https://doi.org/10.1007/s40808-018-0532-z

Mesta, B., Akgun, O. B., & Kentel, E. (2021). Alternative Solutions for Long Missing Streamflow Data for Sustainable Water Resources Management. *International Journal of Water Resources Development*, *37*(5), 882–905. https://doi.org/10.1080/07900627.2020.1799763

Moradkhani, H., & Sorooshian, S. (2008). General Review of Rainfall-Runoff Modeling: Model Calibration, Data Assimilation, and Uncertainty Analysis. In S. Sorooshian, K.-L. Hsu, E. Coppola, B. Tomassetti, M. Verdecchia, & G. Visconti (Eds.), *Hydrological Modelling and the Water Cycle: Coupling the Atmospheric and Hydrological Models* (pp. 1–24). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-77843-1_1

Mosavi, A., Ozturk, P., & Chau, K. (2018). Flood Prediction Using Machine Learning Models: Literature Review. *Water*, *10*(11), 1536. https://doi.org/10.3390/w10111536

Muhammad, A. U., Li, X., & Feng, J. (2019). Using LSTM GRU and Hybrid Models for Streamflow Forecasting. In X. B. Zhai, B. Chen, & K. Zhu (Eds.), *Machine Learning and Intelligent Communications* (pp. 510–524). Springer International Publishing. https://doi.org/10.1007/978-3-030-32388-2_44

Mushore, T., Dube, T., Shoko, C., Mazvimavi, D., & Masocha, M. (2019). Progress in Rainfall-Runoff Modelling Contribution of Remote Sensing. *Transactions of the Royal Society of South Africa*, *74*(2), 173–179. https://doi.org/10.1080/0035919X.2019.1589600

Mwale, F. D., Adeloye, A. J., & Rustum, R. (2012). Infilling of Missing Rainfall and Streamflow Data in the Shire River Basin, Malawi – A Self Organizing Map Approach. *Physics and Chemistry of the Earth, Parts A/B/C*, *50–52*, 34–43. https://doi.org/10.1016/j.pce.2012.09.006

Naimi, A. I., & Balzer, L. B. (2018). Stacked Generalization: An Introduction to Super Learning. *European Journal of Epidemiology*, *33*(5), 459–464. https://doi.org/10.1007/s10654-018-0390-z

Ni, L., Wang, D., Singh, V. P., Wu, J., Wang, Y., Tao, Y., & Zhang, J. (2020). Streamflow and Rainfall Forecasting by Two Long Short-Term Memory-Based Models. *Journal of Hydrology*, *583*, 124296. https://doi.org/10.1016/j.jhydrol.2019.124296

Niu, W., & Feng, Z. (2021). Evaluating the Performances of Several Artificial Intelligence Methods in Forecasting Daily Streamflow Time Series for Sustainable Water Resources Management. *Sustainable Cities and Society*, *64*, 102562. https://doi.org/10.1016/j.scs.2020.102562

Noori, N., & Kalin, L. (2016). Coupling SWAT and ANN Models for Enhanced Daily Streamflow Prediction. *Journal of Hydrology*, *533*, 141-151.

Nourani, V., Gökçekuş, H., & Gichamo, T. (2021). Ensemble Data-Driven Rainfall-Runoff Modeling Using Multi-Source Satellite and Gauge Rainfall Data Input Fusion. *Earth Science Informatics*. https://doi.org/10.1007/s12145-021-00615-4

Nourani, V., Molajou, A., Najafi, H., & Danandeh Mehr, A. (2019). Emotional ANN (EANN): A New Generation of Neural Networks for Hydrological Modeling in Iot. *Artificial intelligence in IoT*, 45–61. https://doi.org/10.1007/978-3-030-04110-6_3

Nourani, V., Molajou, A., Uzelaltinbulat, S., & Sadikoglu, F. (2019). Emotional Artificial Neural Networks (Eanns) for Multi-Step Ahead Prediction of Monthly Precipitation; Case Study: Northern Cyprus. *Theoretical and Applied Climatology*, *138*(3–4), 1419–1434. https://doi.org/10.1007/s00704-019-02904-x

Oyebode, O. (2019). Evolutionary Modelling of Municipal Water Demand with Multiple Feature Selection Techniques. *Journal of Water Supply: Research and Technology-Aqua*, *68*(4), 264–281. https://doi.org/10.2166/aqua.2019.145

Oyebode, O., & Stretch, D. (2019). Neural Network Modeling of Hydrological Systems: A Review of Implementation Techniques. *Natural Resource Modeling*, *32*(1), e12189. https://doi.org/10.1111/nrm.12189

Papacharalampous, G., Tyralis, H., & Koutsoyiannis, D. (2018a). One-Step Ahead Forecasting of Geophysical Processes within a Purely Statistical Framework. *Geoscience Letters*, *5*(1), 12. https://doi.org/10.1186/s40562-018-0111-1

Papacharalampous, G., Tyralis, H., & Koutsoyiannis, D. (2018b). Univariate Time Series Forecasting of Temperature and Precipitation with a Focus on Machine Learning Algorithms: A Multiple-Case

Study from Greece. *Water Resources Management*, *32*(15), 5207–5239. https://doi.org/10.1007/s11269-018-2155-6

Parisouj, P., Mohebzadeh, H., & Lee, T. (2020). Employing Machine Learning Algorithms for Streamflow Prediction: A Case Study of Four River Basins with Different Climatic Zones in the United States. *Water Resources Management*, *34*(13), 4113–4131. https://doi.org/10.1007/s11269-020-02659-5

Peters-Lidard, C. D., Hossain, F., Leung, L. R., McDowell, N., Rodell, M., Tapiador, F. J., Turk, F. J., & Wood, A. (2019). 100 Years of Progress in Hydrology. *Meteorological Monographs*. https://doi.org/10.1175/AMSMONOGRAPHS-D-18-0019.1

Polley, E. C., Rose, S., & van der Laan, M. J. (2011). Super Learning. In M. J. van der Laan & S. Rose (Eds.), *Targeted Learning: Causal Inference for Observational and Experimental Data* (pp. 43–66). Springer. https://doi.org/10.1007/978-1-4419-9782-1_3

Polley, E., & Laan, M. van der. (2010). Superlearner in Prediction. *U.C. Berkeley Division of Biostatistics Working Paper Series*. https://biostats.bepress.com/ucbbiostat/paper266

Pradhan, R. K., Markonis, Y., Vargas Godoy, M. R., Villalba-Pradas, A., Andreadis, K. M., Nikolopoulos, E. I., Papalexiou, S. M., Rahim, A., Tapiador, F. J., & Hanel, M. (2022). Review of GPM IMERG Performance: A Global Perspective. *Remote Sensing of Environment*, *268*, 112754. https://doi.org/10.1016/j.rse.2021.112754

Prodhan, F. A., Zhang, J., Hasan, S. S., Pangali Sharma, T. P., & Mohana, H. P. (2022). A Review of Machine Learning Methods for Drought Hazard Monitoring and Forecasting: Current Research Trends, Challenges, and Future Research Directions. *Environmental Modelling & Software*, *149*, 105327. https://doi.org/10.1016/j.envsoft.2022.105327

Qihao Weng. (2016). Introduction to Remote Sensing Systems, Data, and Applications. *Remote Sensing of Natural Resources, 3-20.*

Rahimzad, M., Moghaddam Nia, A., Zolfonoon, H., Soltani, J., Danandeh Mehr, A., & Kwon, H.-H. (2021a). Performance Comparison of an LSTM-based Deep Learning Model versus Conventional Machine Learning Algorithms for Streamflow Forecasting. *Water Resources Management*. https://doi.org/10.1007/s11269-021-02937-w

Rahimzad, M., Moghaddam Nia, A., Zolfonoon, H., Soltani, J., Danandeh Mehr, A., & Kwon, H.-H. (2021b). Performance Comparison of an LSTM-based Deep Learning Model versus Conventional Machine Learning Algorithms for Streamflow Forecasting. *Water Resources Management*. https://doi.org/10.1007/s11269-021-02937-w

Ramapriyan, H. K., & Murphy, K. (2017). Collaborations and Partnerships in NASA's Earth Science Data Systems. *Data Science Journal*, *16*(0), 51. https://doi.org/10.5334/dsj-2017-051

Rania, S., Waad, B., & Nadia, E. (2000). Hybrid Feature Selection Method based on the Genetic Algorithm and Pearson Correlation Coefficient. *Machine learning paradigms: theory and application*, 3-24. https://doi.org/10.1007/978-3-030-02357-7_1

Ratner, B. (2009). The Correlation Coefficient: Its Values Range Between +1/−1, or Do They? *Journal of Targeting, Measurement and Analysis for Marketing*, *17*(2), 139–142. https://doi.org/10.1057/jt.2009.5

Refsgaard, J. C., & Knudsen, J. (1996). Operational Validation and Intercomparison of Different Types of Hydrological Models. *Water Resources Research*, *32*(7), 2189–2202. https://doi.org/10.1029/96WR00896

Sagi, O., & Rokach, L. (2018). Ensemble Learning: A Survey. *WIREs Data Mining and Knowledge Discovery*, *8*(4), e1249. https://doi.org/10.1002/widm.1249

Sahoo, B. B., Jha, R., Singh, A., & Kumar, D. (2019). Long Short-Term Memory (LSTM) Recurrent Neural Network for Low-Flow Hydrological Time Series Forecasting. *Acta Geophysica*, *67*(5), 1471–1481. https://doi.org/10.1007/s11600-019-00330-1

Seibert, J. (1997). Estimation of Parameter Uncertainty in the HBV Model: Paper Presented at the Nordic Hydrological Conference (Akureyri, Iceland - August 1996). *Hydrology Research*, *28*(4–5), 247–262. https://doi.org/10.2166/nh.1998.15

Serrat-Capdevila, A., Merino, M., Valdes, J. B., & Durcik, M. (2016). Evaluation of the Performance of Three Satellite Precipitation Products over Africa. *Remote Sensing*, *8*(10), 836. https://doi.org/10.3390/rs8100836

Sezer, O. B., Gudelek, M. U., & Ozbayoglu, A. M. (2020). Financial Time Series Forecasting with Deep Learning: A Systematic Literature Review: 2005–2019. *Applied Soft Computing*, *90*, 106181. https://doi.org/10.1016/j.asoc.2020.106181

Shahid, F., Zameer, A., & Muneeb, M. (2020). Predictions for COVID-19 with Deep Learning Models of LSTM, GRU and Bi-LSTM. *Chaos Solitons & Fractals*, *140*, 110212. https://doi.org/10.1016/j.chaos.2020.110212

Shamseldin, A. Y. (1997). Application of a Neural Network Technique to Rainfall-Runoff Modelling. *Journal of Hydrology*, *199*(3), 272–294. https://doi.org/10.1016/S0022-1694(96)03330-6

Sharma, P., & Machiwal, D. (2021). Chapter 1 - Streamflow Forecasting: Overview of Advances in Data-Driven Techniques. In P. Sharma & D. Machiwal (Eds.), *Advances in Streamflow Forecasting* (pp. 1–50). Elsevier. https://doi.org/10.1016/B978-0-12-820673-7.00013-5

Shin, H.-C., Roth, H. R., Gao, M., Lu, L., Xu, Z., Nogues, I., Yao, J., Mollura, D., & Summers, R. M. (2016). Deep Convolutional Neural Networks for Computer-Aided Detection: CNN

Architectures, Dataset Characteristics and Transfer Learning. *IEEE Transactions on Medical Imaging*, *35*(5), 1285–1298. https://doi.org/10.1109/TMI.2016.2528162

Sichangi, A. W., Wang, L., Yang, K., Chen, D., Wang, Z., Li, X., Zhou, J., Liu, W., & Kuria, D. (2016). Estimating Continental River Basin Discharges using Multiple Remote Sensing Datasets. *Remote Sensing of Environment*, *179*, 36–53. https://doi.org/10.1016/j.rse.2016.03.019

Sichangi, A., Wang, L., & Hu, Z. (2018). Estimation of River Discharge Solely from Remote-Sensing Derived Data: An Initial Study over the Yangtze River. *Remote Sensing*, *10*(9), 1385. https://doi.org/10.3390/rs10091385

Singh, V. P. (2018). Hydrologic Modeling: Progress and Future Directions. *Geoscience Letters*, *5*(1), 15. https://doi.org/10.1186/s40562-018-0113-z

Singh Vijay P. & Woolhiser David A. (2002). Mathematical Modeling of Watershed Hydrology. *Journal of Hydrologic Engineering*, *7*(4), 270–292. https://doi.org/10.1061/(ASCE)1084-0699(2002)7:4(270)

Sinisi, S. E., Polley, E. C., Petersen, M. L., Rhee, S.-Y., & Laan, M. J. van der. (2007). Super Learning: an Application to the Prediction of HIV-1 Drug Resistance. *Statistical Applications in Genetics and Molecular Biology*, *6*(1). https://doi.org/10.2202/1544-6115.1240

Skofronick-Jackson, G., Berg, W., Kidd, C., Kirschbaum, D. B., Petersen, W. A., Huffman, G. J., & Takayabu, Y. N. (2018). Global Precipitation Measurement (GPM): Unified Precipitation Estimation from Space. In C. Andronache (Ed.), *Remote Sensing of Clouds and Precipitation* (pp. 175–193). Springer International Publishing. https://doi.org/10.1007/978-3-319-72583-3_7

Sood, A., & Smakhtin, V. (2014). Global Hydrological Models: A Review. https://doi.org/10.1080/02626667.2014.950580

Sorooshian, S., AghaKouchak, A., Arkin, P., Eylander, J., Foufoula-Georgiou, E., Harmon, R., Hendrickx, J. M. H., Imam, B., Kuligowski, R., Skahill, B., & Skofronick-Jackson, G. (2011). Advanced Concepts on Remote Sensing of Precipitation at Multiple Scales. *Bulletin of the American Meteorological Society*, *92*(10), 1353–1357.

Su, X., Yan, X., & Tsai, C.-L. (2012). Linear Regression. *WIREs Computational Statistics*, *4*(3), 275–294. https://doi.org/10.1002/wics.1198

Sulugodu, B., & Deka, P. C. (2019). Evaluating the Performance of CHIRPS Satellite Rainfall Data for Streamflow Forecasting. *Water Resources Management*, *33*(11), 3913–3927. https://doi.org/10.1007/s11269-019-02340-6

Sun, Q., Miao, C., Duan, Q., Ashouri, H., Sorooshian, S., & Hsu, K.-L. (2018). A Review of Global Precipitation Datasets: Data Sources, Estimation, and Intercomparisons. *Reviews of Geophysics*, *56*(1), 79–107. https://doi.org/10.1002/2017RG000574

Sun, W. C., Ishidaira, H., & Bastola, S. (2010). Towards Improving River Discharge Estimation in Ungauged Basins: Calibration of Rainfall-Runoff Models based on Satellite Observations of River Flow Width at Basin Outlet. *Hydrology and Earth System Sciences*, *14*(10), 2011–2022. https://doi.org/10.5194/hess-14-2011-2010

Sun, W., Ishidaira, H., Bastola, S., & Yu, J. (2015). Estimating Daily Time Series of Streamflow Using Hydrological Model Calibrated based on Satellite Observations of River Water Surface Width: Toward Real World Applications. *Environmental Research*, *139*, 36–45. https://doi.org/10.1016/j.envres.2015.01.002

Suradhaniwar, S., Kar, S., Durbha, S. S., & Jagarlapudi, A. (2021). Time Series Forecasting of Univariate Agrometeorological Data: A Comparative Performance Evaluation via One-Step and Multi-Step Ahead Forecasting Strategies. *Sensors*, *21*(7), Article 7. https://doi.org/10.3390/s21072430

T., C., Ren, L., Yuan, F., & Tang, T. (2019). Merging Ground and Satellite-Based Precipitation Datasets for Improved Hydrological Simulations in the Xijiang River Basin of China. *Stochastic Environmental Research and Risk Assessment*, *33*, 1893-1905. https://doi.org/10.1007/s00477-019-01731-w

Taddese, G., Sonder, K., & Peden, D. (2009). The Water of the Awash River Basin: A Future Challenge To Ethiopia. *ILRI, Addis Ababa*.

Tahsin, S., Medeiros, S. C., & Singh, A. (2018). Assessing the Resilience of Coastal Wetlands to Extreme Hydrologic Events using Vegetation Indices: A Review. *Remote Sensing*, *10*(9), 1390. https://doi.org/10.3390/rs10091390

Tanbeer, S. K., Ahmed, C. F., Jeong, B.-S., & Lee, Y.-K. (2009). Sliding Window-based Frequent Pattern Mining over Data Streams. *Information Sciences*, *179*(22), 3843–3865. https://doi.org/10.1016/j.ins.2009.07.012

Tarpanelli, A., Santi, E., Tourian, M. J., Filippucci, P., Amarnath, G., & Brocca, L. (2019). Daily River Discharge Estimates by Merging Satellite Optical Sensors and Radar Altimetry through Artificial Neural Network. *IEEE Transactions on Geoscience and Remote Sensing*, *57*(1), 329–341. https://doi.org/10.1109/TGRS.2018.2854625

Tegegne, G., Park, D. K., & Kim, Y.-O. (2017). Comparison of Hydrological Models for the Assessment of Water Resources in a Data-Scarce Region, the Upper Blue Nile River Basin. *Journal of Hydrology: Regional Studies*, *14*, 49–66. https://doi.org/10.1016/j.ejrh.2017.10.002

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794). https://doi.org/10.1145/2939672.2939785

Tibshirani, R. (1996). Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, *58*(1), 267–288. https://doi.org/10.1111/j.2517-6161.1996.tb02080.x

Torres, J. F., Gutiérrez-Avilés, D., Troncoso, A., & Martínez-Álvarez, F. (2019). Random Hyper-Parameter Search-Based Deep Neural Network for Power Consumption Forecasting. In I. Rojas, G. Joya, & A. Catala (Eds.), *Advances in Computational Intelligence* (pp. 259–269). Springer International Publishing. https://doi.org/10.1007/978-3-030-20521-8_22

Torres, J. F., Hadjout, D., Sebaa, A., Martínez-Álvarez, F., & Troncoso, A. (2021). Deep Learning for Time Series Forecasting: A Survey. *Big Data*, *9*(1), 3–21. https://doi.org/10.1089/big.2020.0159

Tourian, M. J., Sneeuw, N., & Bárdossy, A. (2013). A Quantile Function Approach to Discharge Estimation from Satellite Altimetry (ENVISAT). *Water Resources Research*, *49*(7), 4174–4186. https://doi.org/10.1002/wrcr.20348

Troin, M., Arsenault, R., Wood, A. W., Brissette, F., & Martel, J.-L. (2021). Generating Ensemble Streamflow Forecasts: A Review of Methods and Approaches over the Past 40 Years. *Water Resources Research*, *57*(7), e2020WR028392. https://doi.org/10.1029/2020WR028392

Tyralis, H., & Papacharalampous, G. (2017). Variable Selection in Time Series Forecasting Using Random Forests. *Algorithms*, *10*(4), Article 4. https://doi.org/10.3390/a10040114

Tyralis, H., Papacharalampous, G., & Langousis, A. (2019). Superensemble Learning for Daily Streamflow Forecasting: Large-Scale Demonstration and Comparison with Multiple Machine Learning Algorithms. *Neural Computing and Applications*, *33*(8), 3053-3068. https://doi.org/10.1007/s00521-020-05172-3

Van, S. P., Le, H. M., Thanh, D. V., Dang, T. D., Loc, H. H., & Anh, D. T. (2020). Deep Learning Convolutional Neural Network in Rainfall–Runoff Modelling. *Journal of Hydroinformatics*, *22*(3), 541–561. https://doi.org/10.2166/hydro.2020.095

WWDSE in association with MCE and WAPCOS (2005).Wabi Shebele River Basin Integrated Development Master Plan Study Project – Phase III – Master Plan Main Report (p. 578).

Wagena, M. B., Goering, D., Collick, A. S., Bock, E., Fuka, D. R., Buda, A., & Easton, Z. M. (2020). Comparison of Short-Term Streamflow Forecasting Using Stochastic Time Series, Neural Networks, Process-Based, and Bayesian Models. *Environmental Modelling & Software*, *126*, 104669. https://doi.org/10.1016/j.envsoft.2020.104669

Wallis, K. F. (2011). Combining Forecasts – Forty Years Later. *Applied Financial Economics*, *21*(1–2), 33–41. https://doi.org/10.1080/09603107.2011.523179

Wang, B., & Gong, N. Z. (2018). Stealing Hyperparameters in Machine Learning. *IEEE Symposium on Security and Privacy (SP)*, 36–52. https://doi.org/10.1109/SP.2018.00038

Wang, N., Liu, W., Sun, F., Yao, Z., Wang, H., & Liu, W. (2020). Evaluating Satellite-Based and Reanalysis Precipitation Datasets with Gauge-Observed Data and Hydrological Modeling in the Xihe River Basin, China. *Atmospheric Research*, *234*, 104746. https://doi.org/10.1016/j.atmosres.2019.104746

Wang, Y., Liao, W., & Chang, Y. (2018). Gated Recurrent Unit Network-Based Short-Term Photovoltaic Forecasting. *Energies*, *11*(8), Article 8. https://doi.org/10.3390/en11082163

Wegayehu, E. B., & Muluneh, F. B. (2021). Multivariate Streamflow Simulation Using Hybrid Deep Learning Models. *Computational Intelligence and Neuroscience*, *2021*, e5172658. https://doi.org/10.1155/2021/5172658

Worqlul, A. W., Ayana, E. K., Yen, H., Jeong, J., MacAlister, C., Taylor, R., Gerik, T. J., & Steenhuis, T. S. (2018). Evaluating Hydrologic Responses to Soil Characteristics Using SWAT Model in a Paired-Watersheds in the Upper Blue Nile Basin. *CATENA*, *163*, 332–341. https://doi.org/10.1016/j.catena.2017.12.040

Wu, W., Dandy, G. C., & Maier, H. R. (2014). Protocol for Developing ANN Models and Its Application to the Assessment of the Quality of the ANN Model Development Process in Drinking Water Quality Modelling. *Environmental Modelling & Software*, *54*, 108–127. https://doi.org/10.1016/j.envsoft.2013.12.016

Wu, W., May, R., Dandy, G. C., & Maier, H. R. (2012). A Method for Comparing Data Splitting Approaches for Developing Hydrological ANN Models. *International Congress on Environmental Modelling and Software*. https://scholarsarchive.byu.edu/iemssconference/2012/Stream-B/394

Xiong, L., & Zeng, L. (2019). Impacts of Introducing Remote Sensing Soil Moisture in Calibrating a Distributed Hydrological Model for Streamflow Simulation. *Water*, *11*(4), 666. https://doi.org/10.3390/w11040666

Xu, T., & Liang, F. (2021). Machine Learning for Hydrologic Sciences: An Introductory Overview. *WIREs Water*, *8*(5), e1533. https://doi.org/10.1002/wat2.1533

Yamak, P. T., Yujian, L., & Gadosey, P. K. (2019). A Comparison between ARIMA, LSTM, and GRU for Time Series Forecasting. *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*, 49–55. https://doi.org/10.1145/3377713.3377722

Yan, J., Jia, S., Lv, A., & Zhu, W. (2019). Water Resources Assessment of China's Transboundary River Basins Using a Machine Learning Approach. *Water Resources Research*, *55*(1), 632–655. https://doi.org/10.1029/2018WR023044

Yan, L., Feng, J., & Hang, T. (2019). Small Watershed Stream-Flow Forecasting Based on LSTM. *Advances in Intelligent Systems and Computing*, *935*, 1006–1014. https://doi.org/10.1007/978-3-030-19063-7_79

Yang, S., Yang, D., Chen, J., Santisirisomboon, J., Lu, W., & Zhao, B. (2020). A Physical Process and Machine Learning Combined Hydrological Model for Daily Streamflow Simulations of Large Watersheds with Limited Observation Data. *Journal of Hydrology*, *590*, 125206. https://doi.org/10.1016/j.jhydrol.2020.125206

Yaseen, Z. M., El-shafie, A., Jaafar, O., Afan, H. A., & Sayl, K. N. (2015). Artificial Intelligence Based Models for Stream-Flow Forecasting: 2000–2015. *Journal of Hydrology*, *530*, 829–844. https://doi.org/10.1016/j.jhydrol.2015.10.038

Yaseen, Z. M., Mohtar, W. H. M. W., Ameen, A. M. S., Ebtehaj, I., Razali, S. F. M., Bonakdari, H., Salih, S. Q., Al-Ansari, N., & Shahid, S. (2019). Implementation of Univariate Paradigm for Streamflow Simulation Using Hybrid Data-Driven Model: Case Study in Tropical Region. *IEEE Access*, *7*, 74471–74481. https://doi.org/10.1109/access.2019.2920916

Yaseen, Z. M., Sulaiman, S. O., Deo, R. C., & Chau, K.-W. (2019). An Enhanced Extreme Learning Machine Model for River Flow Forecasting: State-of-the-Art, Practical Applications in Water Resource Engineering Area and Future Research Direction. *Journal of Hydrology*, *569*, 387–408. https://doi.org/10.1016/j.jhydrol.2018.11.069

Yawson, D. K., Kongo, V. M., & Kachroo, R. K. (2005). Application of Linear and Nonlinear Techniques in River Flow Forecasting in the Kilombero River Basin, Tanzania. *Hydrological Sciences Journal*, *50*(5), null-796. https://doi.org/10.1623/hysj.2005.50.5.783

Yin, J., Deng, Z., Ines, A. V. M., Wu, J., & Eeswaran, R. (2020). Forecast of Short-Term Daily Reference Evapotranspiration under Limited Meteorological Variables Using a Hybrid Bi-Directional Long Short-Term Memory Model (Bi-LSTM). *Agricultural Water Management*, *242*, 106386. https://doi.org/10.1016/j.agwat.2020.106386

Ymeti, I. (2007). Rainfall Estimation by Remote Sensing for Conceptual Rainfall-Runoff Modeling in the Upper Blue Nile Basin. Enschede, The Netherlands: ITC.

Young, C.-C., Liu, W.-C., & Chung, C.-E. (2015). Genetic Algorithm and Fuzzy Neural Networks Combined with the Hydrological Modeling System for Forecasting Watershed Runoff Discharge. *Neural Computing and Applications*, *26*(7), 1631–1643. https://doi.org/10.1007/s00521-015-1832-0

Young, S., Abdou, T., & Bener, A. (2018). Deep Superlearner: A Deep Ensemble for Classification Problems. *ArXiv:1803.02323 [Cs, Stat]*. http://arxiv.org/abs/1803.02323

Yu, C., Hu, D., Liu, M., Wang, S., & Di, Y. (2020). Spatio-Temporal Accuracy Evaluation of Three High-Resolution Satellite Precipitation Products in China Area. *Atmospheric Research*, *241*, 104952. https://doi.org/10.1016/j.atmosres.2020.104952

Yuan, X., Chen, C., Lei, X., Yuan, Y., & Muhammad Adnan, R. (2018). Monthly Runoff Forecasting Based on LSTM–ALO Model. *Stochastic Environmental Research and Risk Assessment*, *32*(8), 2199–2212. https://doi.org/10.1007/s00477-018-1560-y

Yuvaraj, N., Chang, V., Gobinathan, B., Pinagapani, A., Kannan, S., Dhiman, G., & Rajan, A. R. (2021). Automatic Detection of Cyberbullying Using Multi-Feature Based Artificial Intelligence with Deep Decision Tree Classification. *Computers & Electrical Engineering*, *92*, 107186. https://doi.org/10.1016/j.compeleceng.2021.107186

Zaji, A. H., Bonakdari, H., & Gharabaghi, B. (2019). Developing an AI-Based Method for River Discharge Forecasting Using Satellite Signals. *Theoretical and Applied Climatology*. https://doi.org/10.1007/s00704-019-02833-9

Zhang, F., & O'Donnell, L. J. (2020). Chapter 7 - Support Vector Regression. In A. Mechelli & S. Vieira (Eds.), *Machine Learning* (pp. 123–140). Academic Press. https://doi.org/10.1016/B978-0-12-815739-8.00007-9

Zhang, X., & Lindström, G. (1997). Development of an Automatic Calibration Scheme for the HBV Hydrological Model. *Hydrological Processes*, *11*(12), 1671–1682. https://doi.org/10.1002/(SICI)1099-1085(19971015)11:12<1671::AID-HYP497>3.0.CO;2-G

Zhang, Z., Zhang, Q., & Singh, V. P. (2018). Univariate Streamflow Forecasting Using Commonly Used Data-Driven Models: Literature Review and Case Study. *Hydrological Sciences Journal*, *63*(7), 1091–1111. https://doi.org/10.1080/02626667.2018.1469756

Zhao, X., Lv, H., Lv, S., Sang, Y., Wei, Y., & Zhu, X. (2021). Enhancing Robustness of Monthly Streamflow Forecasting Model Using Gated Recurrent Unit Based on Improved Grey Wolf Optimizer. *Journal of Hydrology*, *601*, 126607. https://doi.org/10.1016/j.jhydrol.2021.126607

Zhou, Z.-H. (2009). Ensemble Learning. In S. Z. Li & A. Jain (Eds.), *Encyclopedia of Biometrics* (pp. 270–273). Springer US. https://doi.org/10.1007/978-0-387-73003-5_293

Zhu, S., Luo, X., Yuan, X., & Xu, Z. (2020). An Improved Long Short-Term Memory Network for Streamflow Forecasting in the Upper Yangtze River. *Stochastic Environmental Research and Risk Assessment*. https://doi.org/10.1007/s00477-020-01766-4

Zhu, S., Shen, Y., & Ma, Z. (2021). A New Perspective for Charactering the Spatio-temporal Patterns of the Error in GPM IMERG over Mainland China. *Earth and Space Science*, *8*(1). https://doi.org/10.1029/2020EA001232

Zivot, E., & Wang, J. (2003). Rolling Analysis of Time Series. In E. Zivot & J. Wang (Eds.), *Modeling Financial Time Series with S-Plus*, pp. 299–346. Springer. https://doi.org/10.1007/978-0-387-21763-5_9

Zou, Q., Xiong, Q., Li, Q., Yi, H., Yu, Y., & Wu, C. (2020). A Water Quality Prediction Method Based on the Multi-Time Scale Bidirectional Long Short-Term Memory Network. *Environmental Science and Pollution Research*, *27*(14), 16853–16864. https://doi.org/10.1007/s11356-020-08087-7

Zounemat-Kermani, M., Batelaan, O., Fadaee, M., & Hinkelmann, R. (2021). Ensemble Machine Learning Paradigms in Hydrology: A Review. *Journal of Hydrology*, *598*, 126266. https://doi.org/10.1016/j.jhydrol.2021.126266

Zounemat-Kermani, M., Mahdavi-Meymand, A., & Hinkelmann, R. (2021). A Comprehensive Survey on Conventional and Modern Neural Networks: Application to River Flow Forecasting. *Earth Science Informatics*, *14*(2), 893–911. https://doi.org/10.1007/s12145-021-00599-1

Zounemat-Kermani, M., Matta, E., Cominola, A., Xia, X., Zhang, Q., Liang, Q., & Hinkelmann, R. (2020). Neurocomputing in Surface Water Hydrology and Hydraulics: A Review of Two Decades Retrospective, Current Status and Future Prospects. *Journal of Hydrology*, *588*, 125085. https://doi.org/10.1016/j.jhydrol.2020.125085

APPENDICES

Appendix A: Codes for Chapter Four

Import the required packages

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from statsmodels.tools.eval_measures import rmse
from keras.layers import Dense
from keras.preprocessing.sequence import TimeseriesGenerator
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import GRU
from keras.layers import Dropout
from keras.layers import Bidirectional
from tensorflow.keras import regularizers
from keras.layers import TimeDistributed
from tensorflow import keras
from tensorflow.keras import layers
from kerastuner.tuners import RandomSearch
from keras.layers import RepeatVector
from keras.optimizers import Adam
from timeit import default_timer as timer
from keras.callbacks import EarlyStopping
from sklearn.metrics import mean_squared_error,mean_absolute_percentage_error,r2_score
from sklearn.metrics import mean_absolute_error
```

```
from keras.metrics import Mean
from scipy.stats import pearsonr
import hydroeval as he
from math import sqrt
from pandas import DataFrame
from matplotlib.ticker import PercentFormatter
from keras.utils.vis_utils import plot_model
import os
import csv
import warnings
warnings.filterwarnings("ignore")
```

Read input data

```
data = pd.read_excel("filename.xlsx")
timeseries = pd.read_excel("filename.xlsx",indice_col='date')
timeseries.streamflow = pd.to_numeric(timeseries.streamflow,errors='coerce')
timeseries.tail(5)
```

Fill missing data

```
ts = timeseries.interpolate(method ="linear",inplace = False)
```

Splitting data for training and testing (80/20)

```
A=length of test data set
train, test = ts[:-A], ts[-A:]
```

Timeseries plot for split data

```
plt.rcParams["font.family"]="Times New Roman"
plt.figure(figsize=(27,4),dpi = 300, edgecolor="y")
plt.plot(train['streamflow'],color = "blue",alpha = 1,linestyle ="-",label = "Training")
plt.plot(test["streamflow"],color = "orange",alpha =1,linestyle ="--",label = "Testing")
plt.title("Catchment_Name_Streamflow_Timeseries", fontsize = 32)
plt.legend(fontsize = 25)
plt.xlabel("Time(days)",size=27)
plt.ylabel("Q(m3/sec)",size=27)
```

```
plt.xticks(size=27,color = "black")
plt.yticks(size=27,color = "black")
plt.grid(True,alpha = 0.1)
plt.savefig("Splited_Catchment_Name")
```

Box plot for split data

```
plt.rcParams["font.family"]="Times New Roman"
fig = plt.figure(figsize=(7,4),dpi = 300, edgecolor="y")
Total = ts["streamflow"]
Train = train['streamflow']
Test=test["streamflow"]
data = [Total,Train,Test]
```

Creating axes instance

```
ax = fig.add_axes([1,1,1,1])
```

Creating plot

```
bp = ax.boxplot(data,0,'',showmeans=True, patch_artist = True)
ax.set_xticklabels(['Total', 'Train', 'Test'])
colors = ['black', 'blue', 'orange']
for patch, color in zip(bp['boxes'], colors):
    patch.set_facecolor(color)
```

Changing color and linewidth of whiskers

```
for whisker in bp['whiskers']:
    whisker.set(color ='black', linewidth = 1.5, linestyle =":")
```

Changing color and linewidth of caps

```
for cap in bp['caps']:
    cap.set(color ='black', linewidth = 2)
```

Changing color and linewidth of medians

```
for median in bp['medians']:
    median.set(color ='white', linewidth = 3)
```

Changing style of fliers

```
for flier in bp['fliers']:
```

```
    flier.set(marker ='D', color ='black', alpha = 1)
ax.get_xaxis().tick_bottom()
ax.get_yaxis().tick_left()
plt.xlabel("Splitted data",size=15)
plt.ylabel("Q(m3/sec)",size=15)
plt.xticks(size=17,color="black")
plt.yticks(size=17,color="black")
plt.grid(True,alpha = 0.1)
plt.savefig("discriptive_boxplot_catchment_name",bbox_inches="tight")
```

Data normalization for training data

```
lag= number of lags
scaler = StandardScaler()
train_data = scaler.fit_transform(train)
x_train=[]
y_train=[]
for i in range (lag,train.shape[0]):
    x_train.append(train_data[i-lag:i])
    y_train.append(train_data[i,0])
```

Data normalization for test Data¶

```
past_days = train.tail(lag)
test_data = past_days.append(test,ignore_indice=True)
inputs_test = scaler.fit_transform(test_data)
x_test = []
y_test = []
for i in range(lag,test_data.shape[0]):
    x_test.append(inputs_test[i-lag:i])
    y_test.append(inputs_test[i,0])
x_train,y_train = np.array(x_train),np.array(y_train)
x_test,y_test = np.array(x_test),np.array(y_test)
n_input= x_train.shape[1]
```

```
x_train = x_train.reshape((x_train.shape[0], n_input))
x_test = x_test.reshape((x_test.shape[0], n_input))
n_features = 1
```

Parameter optimization using Keras tuner

```
def build_model(hp):
    model = keras.Sequential([
        keras.layers.Dense(units=hp.Int('MLP_l1_units',min_value=5,max_value=40,step=5),
                           activation='relu',input_dim = n_input),
        keras.layers.Dropout(hp.Float('dropout1',min_value=0.0,max_value=0.3,default=0.2,step=0.1)),
        keras.layers.Dense(units=hp.Int('MLP_l2_units',min_value=5,max_value=40,step=5),activation='relu'),
        keras.layers.Dense(1,activation='linear') ])
    model.compile(optimizer=keras.optimizers.Adam(
        hp.Choice('learning_rate',values=[1e-2, 1e-3, 1e-4])),loss="mse",metrics=['mse','mae','mape'])
    model.fit(x_train,y_train,epochs=hp.Int("num_epoches",min_value=10,max_value=100,
                                step=10),batch_size=hp.Int("num_batch_size",min_value=10,max_value=100,
                                step=10),validation_data=(x_test,y_test),verbose=1,shuffle = False)
    return model
tuner = RandomSearch(build_model,objective="val_loss",max_trials=20,executions_per_trial=3,
    directory='Catchment_name_MLP',project_name=' Catchment_name _daily_time_step_MLP')
tuner.search_space_summary()
tuner.search(x_train,y_train,validation_data=(x_test,y_test),verbose=1,shuffle = False)
tuner.results_summary()
```

Model training using optimal parameters

```
class TimingCallback(keras.callbacks.Callback):
    def __init__(self, logs={}):
        self.logs=[]
    def on_epoch_begin(self, epoch, logs={}):
        self.starttime = timer()
    def on_epoch_end(self, epoch, logs={}):
        self.logs.append(timer()-self.starttime)
```

```
cb = TimingCallback()
train_loss = pd.DataFrame()
val_loss = pd.DataFrame()
train_mae = pd.DataFrame()
val_mae = pd.DataFrame()
for i in range(number of times model trains):
    modelT = Sequential()
```

For MLP

```
    modelT.add(Dense(25,activation ="relu", input_dim = n_input))
    modelT.add(Dropout(0.0))
    modelT.add(Dense(40, activation ="relu"))
    modelT.add(Dropout(0.1))
    modelT.add(Dense(1))
```

For Bi-LSTM

```
    modelT.add(Bidirectional(LSTM(10,activation ="relu", return_sequences = True)))
    modelT.add(Dropout(0.0))
    modelT.add(Bidirectional(LSTM(5, activation ="relu")))
    modelT.add(Dropout(0.1))
    modelT.add(Dense(1))
```

For GRU

```
    modelT.add(GRU(25,activation ="relu", return_sequences = True))
    modelT.add(Dropout(0.1))
    modelT.add(GRU(20, activation ="relu"))
    modelT.add(Dropout(0.1))
    modelT.add(Dense(1))
```

For LSTM

```
    modelT.add(LSTM(25,activation ="relu", return_sequences = True))
    modelT.add(Dropout(0.1))
    modelT.add(LSTM(20, activation ="relu"))
    modelT.add(Dropout(0.1))
    modelT.add(Dense(1))
```

```
opt = Adam(learning_rate=0.0001)

modelT.compile(optimizer = opt, loss = "mse", metrics=['mse','mae','mape'])

history
  =modelT.fit(x_train,y_train,epochs=90,batch_size=60,callbacks=[cb],validation_data=(x_test,y_test),verb
  ose=1,shuffle = False)

 train_loss[str(i)] = history.history["loss"]

 val_loss[str(i)] = history.history["val_loss"]

 train_mae[str(i)] = history.history["mae"]

 val_mae[str(i)] = history.history["val_mae"]
```

Model internal network structure graph

```
plot_model(modelT, to_file='Catchment_name_MLP_time_step_plot.png', show_shapes=True,
           show_layer_names=True,   dpi=300)

plt.savefig(' Catchment_name _GRU_time_step_plot.png')
```

Model loss function graph

```
plt.rcParams["font.family"]="Times New Roman"

plt.figure(figsize=(7,4),dpi = 300, edgecolor="y")

plt.plot(train_loss.iloc[0:0,0:1],color = "gray",alpha =1,linestyle ="-",label = "Train", linewidth = 0.6)

plt.plot(train_loss,color = "gray",alpha =1,linestyle ="-", linewidth = 0.6)

plt.legend(  loc= "upper right",fontsize = 7)

plt.plot(val_loss.iloc[0:0,0:1],color = "cyan",alpha =1,linestyle ="--",label = "Test", linewidth = 0.6)

plt.plot(val_loss,color = "cyan",alpha =1,linestyle ="--",linewidth = 0.6)

plt.legend(  loc= "upper right",fontsize = 19)

plt.title( "Catchment_name \n MLP time step",fontsize=17,pad=2)

plt.ylabel( "loss")

plt.xlabel( "epoch")

plt.xlabel("Epochs",size=19,labelpad=-2)

plt.ylabel("Loss",size=19)

plt.xticks(size=15,color = "black")

plt.yticks(size=15,color = "black")

plt.savefig("Catchment_name_MLP_time_step_loss")
```

Prediction test with the trained model

```
y_pred = modelT.predict(x_test)

y_pred = scaler.inverse_transform(y_pred)

y_test = scaler.inverse_transform(y_test)

timeseriesdate = list(data)[0:1]

test = data[-1315:]

date = test[timeseriesdate]

y_pred_data = pd.DataFrame(y_pred)

y_test_data = pd.DataFrame(y_test)

date.set_indice("date")

xt=date["date"]

xt= pd.DataFrame(xt)

xt=xt.reset_indice(inplace=False)

xt=xt.drop("indice",axis=1)

yt= pd.DataFrame(y_test)

yt=yt.reset_indice(inplace=False)

yt=yt.drop("indice",axis=1)

yp=y_pred_data[0]

yp=pd.DataFrame(yp)

y_pred.shape
```

Predict and test timeseries comparison graph

```
plt.rcParams["font.family"]="Times New Roman"

plt.figure(figsize=(7,4),dpi =300, edgecolor="y")

plt.plot(y_test,color = "cyan",alpha = 1,linestyle ="--",label = "Test")

plt.plot(y_pred,color = "grey",alpha =1,linestyle ="-",label = "Forecasted")

plt.title(" Catchment_name \n Model, Time_step",fontsize = 17, pad=1)

plt.legend(fontsize = 19)

plt.xlabel("Time(days)",size=19,labelpad=-2)

plt.ylabel("Q(m3/sec)",size=19)

plt.xticks(size=15,color = "black")
```

```
plt.yticks(size=15,color = "black")
plt.grid(True,alpha = 0.03)
plt.savefig("Catchment_name_Model_time_step_test+forecasted")
```

Evaluate the prediction with scatter Plot

```
plt.rcParams["font.family"]="Times New Roman"
plt.figure(figsize=(6,4),dpi =300, edgecolor="y")
plt.scatter(yp,yt,color = "grey",alpha = 0.3,linestyle ="-",label = "Data Points")
plt.plot(y_test,y_test,color = "cyan",alpha = 1,linestyle ="-",label = "Linear Fit")
plt.xlabel("Predicted Q(m3/sec)",size=19,labelpad=-1)
plt.ylabel("Test Q(m3/sec)",size=19)
plt.xticks(size=10,color = "black")
plt.yticks(size=10,color = "black")
plt.grid(True,alpha = 0.03)
plt.savefig("Catchment_Model_Time_Step_scatter")
```

Model evaluations

```
mse = mean_squared_error(y_test,y_pred)
mae = mean_absolute_error(y_test,y_pred)
mape= mean_absolute_percentage_error(y_test,y_pred)
r2 = r2_score(y_test,y_pred)
std = np.std(y_pred)
corr=pearsonr(y_pred_data[0],y_test_data[0])[0]
rmse = sqrt(mse)
epochs = number of epoches
print('RMSE: %f' % rmse)
print('MAE: %f' %mae)
print('MAPE: %f' %mape)
print('corr: %f' %corr)
print('std: %f' %std)
print('R2: %f' %r2)
print(sum(cb.logs)/epochs)
```

Save model residuals to draw box plots of spread of prediction error (PE, m3 s−1)

```
residual = [abs(y_test[i]-y_pred[i]) for i in range(len(y_pred))]
residuals = DataFrame(residual)
p = plt.hist(residuals)
y=[]
x=p[1][0:5]
for i in range(len(p[0][0:5])):
   y.append((p[0][i]/sum(p[0]))*100)
y=np.round(y,1)
def Rho (residuals):
      residuals.to_csv('Residuals_model_timestep.csv')
      path = r'C:\Users\EYOB\Desktop\manuscript_experiment'
      extension = 'csv'
      os.chdir(path)
Rho (residuals)
```

Appendix B: Codes for Chapter Five

Import the required packages

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from statsmodels.tools.eval_measures import rmse
from keras.layers import Dense
from keras.preprocessing.sequence import TimeseriesGenerator
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import GRU
from keras.layers import Dropout
from keras.layers import Bidirectional
from keras.layers import Flatten
from tensorflow.keras import regularizers
from keras.layers import TimeDistributed
from tensorflow import keras
from tensorflow.keras import layers
from kerastuner.tuners import RandomSearch
from keras.layers import RepeatVector
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D,AveragePooling1D
from keras.callbacks import EarlyStopping
from keras.optimizers import Adam
from timeit import default_timer as timer
from keras.utils.vis_utils import plot_model
import warnings
```

```
warnings.filterwarnings("ignore")
```

Read input data

```
timeseries = pd.read_excel("Catchment _data.xlsx")

timeseries.date = pd.to_datetime(timeseries.date,errors='coerce')

ts = timeseries

ts=ts.rolling(30).mean()

ts.drop([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28],axis=0,inplace=True)
```

Split the data

```
A = length of test data

n_past = 30

train,test = ts[:-A],ts[-A-n_past:]
```

Data preparation

```
colsx = list(ts)[1:20]

colsy = list(ts)[0]

df_for_trainingx = train[colsx].astype(float)

df_for_testx = test[colsx].astype(float)

df_for_trainingy = train[colsy].astype(float)

df_for_testy = test[colsy].astype(float)

df_for_trainingy = np.array(df_for_trainingy)

df_for_testy = np.array(df_for_testy)

df_for_trainingy=df_for_trainingy.reshape(df_for_trainingy.shape[0], 1)

df_for_testy=df_for_testy.reshape(df_for_testy.shape[0], 1)
```

Normalize and convert training data to supervised learning

```
scalerx = StandardScaler()

train_data_scaledx = scalerx.fit_transform(df_for_trainingx)

scalery = StandardScaler()

train_data_scaledy = scalery.fit_transform(df_for_trainingy)

trainx = []

trainy = []

n_past = 30
```

```
n_step = 0
for i in range(n_past, len(train_data_scaledy)):
    trainx.append(train_data_scaledx[i-n_past:i-n_step, 0:df_for_trainingx.shape[1]])
    trainy.append(train_data_scaledy[i-n_step-1:i-n_step,0])
trainx, trainy = np.array(trainx), np.array(trainy)
```

Normalize and convert test data to supervised learning

```
scalerx = StandardScaler()
test_data_scaledx = scalerx.fit_transform(df_for_testx)
scalery = StandardScaler()
test_data_scaledy = scalery.fit_transform(df_for_testy)
testx = []
testy = []
n_past = 30
n_step = 0
for i in range(n_past, len(test_data_scaledy)):
    testx.append(test_data_scaledx[i-n_past:i-n_step, 0:df_for_trainingx.shape[1]])
    testy.append(test_data_scaledy[i-n_step-1:i-n_step,0])
testx, testy = np.array(testx), np.array(testy)
```

Shape the input into the required dimensions for MLP

```
n_input = trainx.shape[1] * trainx.shape[2]
trainx = trainx.reshape((trainx.shape[0], n_input))
testx = testx.reshape((testx.shape[0], n_input))
trainx.shape
```

Shape the input into the required dimensions for hybrid models

```
n_input= trainx.shape[1]
n_features = 5
n_seq = 1
trainx = trainx.reshape((trainx.shape[0],n_seq, n_input,n_features))
testx = testx.reshape((testx.shape[0],n_seq, n_input,n_features))
trainx.shape
```

Parameter optimization using Keras tuner for hybrid models

```
def build_model(hp):
    model = keras.Sequential([
     keras.layers.TimeDistributed(keras.layers.Conv1D(filters=hp.Int("conv_1_filter",min_value=8,
    max_value=32, step=8),
                        kernel_size=hp.Choice("conv_1_kernal",values= [2,3]),
                        activation ="relu",
                        input_shape = ( None, n_input, n_features))),
     keras.layers.TimeDistributed(keras.layers.AveragePooling1D(pool_size=hp.Choice
("conv_1_pool_size",values=[2,3]))),
     keras.layers.TimeDistributed(keras.layers.Conv1D(filters=hp.Int
("conv_2_filter",min_value=8, max_value=32, step=8),
                        kernel_size=hp.Choice("conv_2_kernal",values= [2,3]),
                        activation ="relu",
                        input_shape = ( None, n_input, n_features))),
     keras.layers.TimeDistributed(keras.layers.AveragePooling1D(pool_size=hp.Choice
("conv_2_pool_size",values=[2,3]))),
     keras.layers.TimeDistributed(keras.layers.Flatten()),
     keras.layers.LSTM(units=hp.Int('lstm_l1_units',min_value=5,max_value=30,step=5),
                              activation='relu',return_sequences = True),
     keras.layers.Dropout(hp.Float(
               'dropout1',  min_value=0.0,max_value=0.3,default=0.2,  step=0.1)),
     keras.layers.LSTM(units=hp.Int('lstm_l2_units',min_value=5,max_value=30,step=5),activation='relu'),
     keras.layers.Dropout(hp.Float( 'dropout2',min_value=0.0,max_value=0.3,default=0.2,step=0.1)),
     keras.layers.Dense(1,activation='linear')])
    model.compile(optimizer=keras.optimizers.Adam(
          hp.Choice('learning_rate',values=[1e-2, 1e-3, 1e-4])),loss="mse",metrics=['mse','mae','mape'])
    model.fit(trainx,trainy,epochs=hp.Int("num_epoches",min_value=10,max_value=100,
                              step=10),batch_size=hp.Int("num_batch_size",min_value=10,max_value=100,
                              step=10),validation_data=(testx, testy),verbose=1,shuffle = False)
    return model
```

```
tuner = RandomSearch(build_model,objective="val_loss",max_trials=20,executions_per_trial=3,
    directory='catchment_monthly_p_cnn-lstm',project_name='catchment_cnn-lstm')
tuner.search_space_summary()
tuner.search(trainx,trainy,validation_data=(testx, testy),verbose=1,shuffle = False)
tuner.results_summary()
```

Model training using optimal parameters for hybrid models

```
class TimingCallback(keras.callbacks.Callback):
    def __init__(self, logs={}):
        self.logs=[]
    def on_epoch_begin(self, epoch, logs={}):
        self.starttime = timer()
    def on_epoch_end(self, epoch, logs={}):
        self.logs.append(timer()-self.starttime)
cb = TimingCallback()
train_loss = pd.DataFrame()
val_loss = pd.DataFrame()
train_mae = pd.DataFrame()
val_mae = pd.DataFrame()
for i in range(1):
    modelT = Sequential()
    modelT.add(TimeDistributed(Conv1D(filters=16, kernel_size=2, activation ="relu"),input_shape = ( None, n_input, n_features)))
    modelT.add(TimeDistributed(AveragePooling1D(pool_size=2)))
    modelT.add(TimeDistributed(Conv1D(filters=24, kernel_size=2, activation ="relu"),input_shape = ( None, n_input, n_features)))
   modelT.add(TimeDistributed(AveragePooling1D(pool_size=3)))#modelT.add(TimeDistributed(Conv1D(filters=8, kernel_size=2, activation ="relu")))
    modelT.add(TimeDistributed(Flatten()))
    modelT.add(LSTM(25,activation ="relu", return_sequences = True))
    modelT.add(Dropout(0.1))
    modelT.add(LSTM(20, activation ="relu"))
    modelT.add(Dropout(0.2))
```

```
modelT.add(Dense(1))

opt = Adam(learning_rate=0.0001)

modelT.compile(optimizer = opt, loss = "mse", metrics=['mse','mae','mape'])

history = modelT.fit(trainx,trainy,epochs=80,batch_size=40,callbacks=[cb],validation_data=(testx,
testy),verbose=1,shuffle = False)

train_loss[str(i)] = history.history["loss"]

val_loss[str(i)] = history.history["val_loss"]

train_mae[str(i)] = history.history["mae"]

val_mae[str(i)] = history.history["val_mae"]
```

Plot the loss function to inspect under and over fitting

```
plot_model(modelT, to_file='model_plot.png', show_shapes=True, show_layer_names=True)
```

Appendix C: Codes for Chapter Six and Seven

Import the required packages

```
import numpy as np
import pandas as pd
from math import sqrt
from numpy import hstack
from numpy import vstack
from numpy import asarray
from sklearn.metrics import mean_squared_error,mean_absolute_percentage_error,r2_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import median_absolute_error
from keras.metrics import Mean
from scipy.stats import pearsonr
import hydroeval as he
from math import sqrt
from math import log
from keras.metrics import Mean
from hydroeval import nse,evaluator
from math import sqrt
from pandas import DataFrame
from timeit import default_timer as timer
from matplotlib.ticker import PercentFormatter
import os
import csv
from sklearn.linear_model import Lasso
import xgboost
from xgboost import XGBRegressor
from sklearn.datasets import make_regression
from sklearn.model_selection import KFold
from sklearn.model_selection import TimeSeriesSplit
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from statsmodels.tools import add_constant
from itertools import combinations
from statsmodels.regression.linear_model import OLS, GLS, WLS
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import ElasticNet
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from keras.models import Sequential
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import BaggingRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import ExtraTreesRegressor
from keras.optimizers import Adam
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from statsmodels.tools.eval_measures import rmse
from keras.layers import Dense
from keras.preprocessing.sequence import TimeseriesGenerator
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import GRU
from keras.layers import Dropout
from keras.layers import Bidirectional
from keras.layers import Flatten
from tensorflow.keras import regularizers
from keras.layers import TimeDistributed
```

```
from tensorflow import keras
from tensorflow.keras import layers
from keras.optimizers import Adam
from timeit import default_timer as timer
from kerastuner.tuners import RandomSearch
from keras.layers import RepeatVector
from keras.models import load_model
import pickle
from timeit import default_timer as timer
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D,AveragePooling1D
from keras.callbacks import EarlyStopping
import warnings
warnings.filterwarnings("ignore")
```

Read input data

```
timeseries = pd.read_excel("Catchment_data.xlsx")
timeseries.date = pd.to_datetime(timeseries.date,errors='coerce')
```

Splitting data for training and testing (80/20)

```
n_past = 59 # 30 past days for training and 29 for rolling(mean)
A = Length of test data
train,test = ts[:-A],ts[-A-n_past:]
```

Prepare input and output columns here

```
colsx = list(ts)[2:54]
colsy = list(ts)[1]
```

Draw all variables into training and testing

```
df_for_trainingx = train[colsx].astype(float)
df_for_testx = test[colsx].astype(float)
df_for_trainingy = train[colsy].astype(float)
df_for_testy = test[colsy].astype(float)
df_for_trainingy = np.array(df_for_trainingy)
```

```
df_for_testy = np.array(df_for_testy)

df_for_trainingy=df_for_trainingy.reshape(df_for_trainingy.shape[0], 1)

df_for_testy=df_for_testy.reshape(df_for_testy.shape[0], 1)
```

Roll training data monthly and drop null time window

```
df_for_trainingx=df_for_trainingx.rolling(30).mean()

df_for_testx=df_for_testx.rolling(30).mean()

d=df_for_trainingy.shape[0]-n_past # number of training datasets-rolling time window

df_for_trainingx.drop([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28],axis=0,
inplace=True)

df_for_testx.drop([0+d,1+d,2+d,3+d,4+d,5+d,6+d,7+d,8+d,9+d,10+d,11+d,12+d,13+d,14+d,15+d,16+d,17+
d,18+d,19+d,20+d,21+d,22+d,23+d,24+d,25+d,26+d,27+d,28+d],axis=0,inplace=True)

df_for_testy=pd.DataFrame(df_for_testy)

df_for_trainingy=pd.DataFrame(df_for_trainingy)

df_for_trainingy.drop([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28],axis=0,
inplace=True)

df_for_testy.drop([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28],axis=0,inpl
ace=True)

df_for_trainingy.shape,df_for_testy.shape,df_for_trainingx.shape,df_for_testx.shape
```

Standardize the data using MinMaxScaler

```
scalerx = MinMaxScaler()

train_data_scaledx = scalerx.fit_transform(df_for_trainingx)

scalery = MinMaxScaler()

train_data_scaledy = scalery.fit_transform(df_for_trainingy)

test_data_scaledx = scalerx.fit_transform(df_for_testx)

test_data_scaledy = scalery.fit_transform(df_for_testy)
```

Arrange the training data for supervised learning

```
trainx = []

trainy = []

n_past = 30

n_step = 0

for i in range(n_past, len(train_data_scaledy)):

    trainx.append(train_data_scaledx[i-n_past:i-n_step, 0:df_for_trainingx.shape[1]])
```

```
        trainy.append(train_data_scaledy[i-n_step-1:i-n_step,0])
    trainx, trainy = np.array(trainx), np.array(trainy)
```

Arrange the test data for supervised learning

```
    testx = []
    testy = []
    n_past = 30
    n_step = 0
    for i in range(n_past, len(test_data_scaledy)):
        testx.append(test_data_scaledx[i-n_past:i-n_step, 0:df_for_trainingx.shape[1]])
        testy.append(test_data_scaledy[i-n_step-1:i-n_step,0])
    testx, testy = np.array(testx), np.array(testy)
```

Design the eight base models for superensemble model

GRU base model

```
    def get_out_of_fold_predictions(X, y):
        meta_X3, meta_y3 = list(), list()
        n_input = X.shape[1] * X.shape[2]
        X = X.reshape((X.shape[0], n_input))
```

Define split of data

```
        kfold = KFold(n_splits=5, shuffle=False)
```

Enumerate splits

```
        for train_ix, test_ix in kfold.split(X):
            fold_yhats = list()
```

Get data

```
            train_X, test_X = X[train_ix], X[test_ix]
            train_y, test_y = y[train_ix], y[test_ix]
            meta_y3.extend(test_y)
```

Fit and make predictions with each sub-model

```
            class TimingCallback(keras.callbacks.Callback):
                def __init__(self, logs={}):
                    self.logs=[]
```

```
            def on_epoch_begin(self, epoch, logs={}):
                self.starttime = timer()
            def on_epoch_end(self, epoch, logs={}):
                self.logs.append(timer()-self.starttime)
        cb = TimingCallback()
        train_loss = pd.DataFrame()
        train_mae = pd.DataFrame()
        model = Sequential()
        model.add(Dense(35, activation ="relu",input_dim = n_input))#
        model.add(Dropout(0))
        model.add(Dense(10, activation ="relu"))
        model.add(Dense(1))
        opt = Adam(learning_rate=0.001)
        model.compile(optimizer = opt, loss = "mse", metrics=['mse','mae','mape'])
        history = model.fit(train_X,train_y,epochs=30,batch_size=10,callbacks=[cb],verbose=1,shuffle = False)
        train_loss[str(i)] = history.history["loss"]
        train_mae[str(i)] = history.history["mae"]
```

Store columns

```
        filename = 'Borkena_BMASE_Models_FI/model_' + "3"+ '.h5'
        model.save(filename)
        print('>Saved %s' % filename)
        yhat = model.predict(test_X)
        fold_yhats.append(yhat.reshape(len(yhat),1))
```

Store fold yhats as columns

```
        meta_X3.append(hstack(fold_yhats))
    return vstack(meta_X3), asarray(meta_y3)
```

LSTM base model

```
def get_out_of_fold_predictions2(X, y):
    meta_X2, meta_y2 = list(), list()
    n_input= X.shape[1]
```

```
n_features = 51
n_seq = 1
```

Define split of data

```
kfold = KFold(n_splits=5, shuffle=False)
```

Enumerate splits

```
for train_ix, test_ix in kfold.split(X):
  fold_yhats = list()
```

Get data

```
train_X, test_X = X[train_ix], X[test_ix]
train_y, test_y = y[train_ix], y[test_ix]
meta_y2.extend(test_y)
class TimingCallback(keras.callbacks.Callback):
  def __init__(self, logs={}):
    self.logs=[]
  def on_epoch_begin(self, epoch, logs={}):
    self.starttime = timer()
  def on_epoch_end(self, epoch, logs={}):
    self.logs.append(timer()-self.starttime)
cb = TimingCallback()
train_loss = pd.DataFrame()
train_mae = pd.DataFrame()
model = Sequential()
model.add(LSTM(25,activation ="relu", return_sequences = True, input_shape = (n_input, n_features)))
model.add(Dropout(0.1))
model.add(LSTM(20, activation ="relu"))
model.add(Dense(1))
opt = Adam(learning_rate=0.01)
model.compile(optimizer = opt, loss = "mse", metrics=['mse','mae','mape'])
history = model.fit(train_X,train_y,epochs=20,batch_size=60,callbacks=[cb],verbose=1,shuffle = False)
train_loss[str(i)] = history.history["loss"]
train_mae[str(i)] = history.history["mae"]
```

```
            filename = 'Catchment_SE_Models_inputs/model_' + "2"+ '.h5'
            model.save(filename)
            print('>Saved %s' % filename)
            yhat = model.predict(test_X)
```

Store columns

```
            fold_yhats.append(yhat.reshape(len(yhat),1))
```

Store fold yhats as columns

```
            meta_X2.append(hstack(fold_yhats))
        return vstack(meta_X2), asarray(meta_y2)
```

MLP base model

```
    def get_out_of_fold_predictions(X, y):
        meta_X3, meta_y3 = list(), list()
        n_input = X.shape[1] * X.shape[2]
        X = X.reshape((X.shape[0], n_input))
```

Define split of data

```
        kfold = KFold(n_splits=5, shuffle=False)
```

Enumerate splits

```
        for train_ix, test_ix in kfold.split(X):
            fold_yhats = list()
```

Get data

```
            train_X, test_X = X[train_ix], X[test_ix]
            train_y, test_y = y[train_ix], y[test_ix]
            meta_y3.extend(test_y)
```

Fit and make predictions with each sub-model

```
            class TimingCallback(keras.callbacks.Callback):
                def __init__(self, logs={}):
                    self.logs=[]
                def on_epoch_begin(self, epoch, logs={}):
                    self.starttime = timer()
                def on_epoch_end(self, epoch, logs={}):
```

```
            self.logs.append(timer()-self.starttime)
        cb = TimingCallback()
        train_loss = pd.DataFrame()
        train_mae = pd.DataFrame()
        model = Sequential()
        model.add(Dense(35, activation ="relu",input_dim = n_input))#
        model.add(Dropout(0))
        model.add(Dense(10, activation ="relu"))
        model.add(Dense(1))
        opt = Adam(learning_rate=0.001)
        model.compile(optimizer = opt, loss = "mse", metrics=['mse','mae','mape'])
        history = model.fit(train_X,train_y,epochs=30,batch_size=10,callbacks=[cb],verbose=1,shuffle = False)
        train_loss[str(i)] = history.history["loss"]
        train_mae[str(i)] = history.history["mae"]
```

Store columns

```
        filename = 'Borkena_SE_Models_inputs/model_' + "3"+ '.h5'
        model.save(filename)
        print('>Saved %s' % filename)
        yhat = model.predict(test_X)
        fold_yhats.append(yhat.reshape(len(yhat),1))
```

Store fold yhats as columns

```
        meta_X3.append(hstack(fold_yhats))
    return vstack(meta_X3), asarray(meta_y3)
```

CNN-GRU base model

```
def get_out_of_fold_predictions4(X, y):
    meta_X4, meta_y4 = list(), list()
    n_input= X.shape[1]
    n_features = 51
    n_seq = 1
    X = X.reshape((X.shape[0],n_seq, n_input,n_features))
```

Define split of data

```
kfold = KFold(n_splits=5, shuffle=False)
```

Enumerate splits

```
for train_ix, test_ix in kfold.split(X):
  fold_yhats = list()
```

Get data

```
train_X, test_X = X[train_ix], X[test_ix]
train_y, test_y = y[train_ix], y[test_ix]
meta_y4.extend(test_y)
class TimingCallback(keras.callbacks.Callback):
  def __init__(self, logs={}):
    self.logs=[]
  def on_epoch_begin(self, epoch, logs={}):
    self.starttime = timer()
  def on_epoch_end(self, epoch, logs={}):
    self.logs.append(timer()-self.starttime)
cb = TimingCallback()
train_loss = pd.DataFrame()
train_mae = pd.DataFrame()
model = Sequential()
model.add(TimeDistributed(Conv1D(filters=24, kernel_size=2, activation ="relu"),input_shape = (
                                    None, n_input, n_features)))
model.add(TimeDistributed(AveragePooling1D(pool_size=2)))
model.add(TimeDistributed(Flatten()))
model.add(GRU(10,activation ="relu", return_sequences = True))
model.add(Dropout(0))
model.add(GRU(25, activation ="relu"))
model.add(Dense(1))
opt = Adam(learning_rate=0.0001)
model.compile(optimizer = opt, loss = "mse", metrics=['mse','mae','mape'])
```

```
            history = model.fit(train_X,train_y,epochs=100,batch_size=80,callbacks=[cb],verbose=1,shuffle =
                                False)
            train_loss[str(i)] = history.history["loss"]
            train_mae[str(i)] = history.history["mae"]
            filename = 'Catchment_SE_Models_inputs/model_' + "4"+ '.h5'
            model.save(filename)
            print('>Saved %s' % filename)
            yhat = model.predict(test_X)
```

Store columns

```
            fold_yhats.append(yhat.reshape(len(yhat),1))
```

Store fold yhats as columns

```
            meta_X4.append(hstack(fold_yhats))
        return vstack(meta_X4), asarray(meta_y4)
```

SVR base model

```
    def get_out_of_fold_predictions5(X, y):
        meta_X5, meta_y5 = list(), list()
        n_input = X.shape[1] * X.shape[2]
        X = X.reshape((X.shape[0], n_input))
```

Define split of data

```
        kfold = KFold(n_splits=5, shuffle=False)
```

Enumerate splits

```
        for train_ix, test_ix in kfold.split(X):
            fold_yhats = list()
```

Get data

```
            train_X, test_X = X[train_ix], X[test_ix]
            train_y, test_y = y[train_ix], y[test_ix]
            meta_y5.extend(test_y)
            class TimingCallback(keras.callbacks.Callback):
                def __init__(self, logs={}):
                    self.logs=[]
                def on_epoch_begin(self, epoch, logs={}):
```

```
            self.starttime = timer()
        def on_epoch_end(self, epoch, logs={}):
            self.logs.append(timer()-self.starttime)
    cb = TimingCallback()
    train_loss = pd.DataFrame()
    train_mae = pd.DataFrame()
    model= SVR(C=0.1, gamma=0.001, kernel = 'rbf')
    model.fit(train_X, train_y)
    filename = 'Catchment_SE_Models_input/model_' + "5"+ '.h5'
    pickle.dump(model, open(filename, 'wb'))
    print('>Saved %s' % filename)
    yhat = model.predict(test_X)
```

Store columns

```
    fold_yhats.append(yhat.reshape(len(yhat),1))
```

Store fold yhats as columns

```
    meta_X5.append(hstack(fold_yhats))
return vstack(meta_X5), asarray(meta_y5)
```

LASSO base model

```
def get_out_of_fold_predictions6(X, y):
    meta_X6, meta_y6 = list(), list()
    n_input = X.shape[1] * X.shape[2]
    X = X.reshape((X.shape[0], n_input))
```

Define split of data

```
    kfold = KFold(n_splits=5, shuffle=False)
```

Enumerate splits

```
    for train_ix, test_ix in kfold.split(X):
        fold_yhats = list()
```

Get data

```
        train_X, test_X = X[train_ix], X[test_ix]
        train_y, test_y = y[train_ix], y[test_ix]
```

```
        meta_y6.extend(test_y)
        class TimingCallback(keras.callbacks.Callback):
          def __init__(self, logs={}):
            self.logs=[]
          def on_epoch_begin(self, epoch, logs={}):
            self.starttime = timer()
          def on_epoch_end(self, epoch, logs={}):
            self.logs.append(timer()-self.starttime)
        cb = TimingCallback()
```

Fit model

```
        model = Lasso(alpha=0)
        model.fit(train_X, train_y)
        train_loss = pd.DataFrame()
        train_mae = pd.DataFrame()
        filename = 'Catchment_SE_Models_input/model_' + "6"+ '.h5'
        pickle.dump(model, open(filename, 'wb'))
        print('>Saved %s' % filename)
        yhat = model.predict(test_X)
```

Store columns

```
        fold_yhats.append(yhat.reshape(len(yhat),1))
```

Store fold yhats as columns

```
        meta_X6.append(hstack(fold_yhats))
      return vstack(meta_X6), asarray(meta_y6)
```

XGB base model

```
    def get_out_of_fold_predictions7(X, y):
     meta_X7, meta_y7 = list(), list()
     n_input = X.shape[1] * X.shape[2]
     X = X.reshape((X.shape[0], n_input))
```

Define split of data

```
     kfold = KFold(n_splits=5, shuffle=False)
```

Enumerate splits

```
for train_ix, test_ix in kfold.split(X):
    fold_yhats = list()
```

Get data

```
    train_X, test_X = X[train_ix], X[test_ix]
    train_y, test_y = y[train_ix], y[test_ix]
    meta_y7.extend(test_y)
    class TimingCallback(keras.callbacks.Callback):
      def __init__(self, logs={}):
        self.logs=[]
      def on_epoch_begin(self, epoch, logs={}):
        self.starttime = timer()
      def on_epoch_end(self, epoch, logs={}):
        self.logs.append(timer()-self.starttime)
    cb = TimingCallback()
```

Fit model

```
    model = XGBRegressor(base_score=1, booster='gbtree', colsample_bylevel=1,
      colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
      gamma=0, gpu_id=-1, importance_type=None,
      interaction_constraints='', learning_rate=0.05, max_delta_step=0,
      max_depth=15, min_child_weight=3,
      monotone_constraints='()', n_estimators=1500, n_jobs=4,
      num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
      reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
      validate_parameters=1, verbosity=None)
    model.fit(train_X, train_y)
    train_loss = pd.DataFrame()
    train_mae = pd.DataFrame()
    filename = 'Catchment_SE_Models_input/model_' + "7"+ '.h5'
    pickle.dump(model, open(filename, 'wb'))
    print('>Saved %s' % filename)
```

```
        yhat = model.predict(test_X)
```

Store columns

```
        fold_yhats.append(yhat.reshape(len(yhat),1))
```

Store fold yhats as columns

```
        meta_X7.append(hstack(fold_yhats))
    return vstack(meta_X7), asarray(meta_y7)
```

Linear base model

```
def get_out_of_fold_predictions8(X, y):
    meta_X8, meta_y8 = list(), list()
    n_input = X.shape[1] * X.shape[2]
    X = X.reshape((X.shape[0], n_input))
```

Define split of data

```
    kfold = KFold(n_splits=5, shuffle=False)
```

Enumerate splits

```
    for train_ix, test_ix in kfold.split(X):
        fold_yhats = list()
```

Get data

```
        train_X, test_X = X[train_ix], X[test_ix]
        train_y, test_y = y[train_ix], y[test_ix]
        meta_y8.extend(test_y)
        class TimingCallback(keras.callbacks.Callback):
            def __init__(self, logs={}):
                self.logs=[]
            def on_epoch_begin(self, epoch, logs={}):
                self.starttime = timer()
            def on_epoch_end(self, epoch, logs={}):
                self.logs.append(timer()-self.starttime)
        cb = TimingCallback()
```

Fit model

```
        model = LinearRegression()
```

```
            model.fit(train_X, train_y)
            train_loss = pd.DataFrame()
            train_mae = pd.DataFrame()
            filename = 'Catchment_SE_Models_input/model_' + "8"+ '.h5'
            pickle.dump(model, open(filename, 'wb'))
            print('>Saved %s' % filename)
            yhat = model.predict(test_X)
```

Store columns

```
            fold_yhats.append(yhat.reshape(len(yhat),1))
```

Store fold yhats as columns

```
            meta_X8.append(hstack(fold_yhats))
        return vstack(meta_X8), asarray(meta_y8)
```

Generate base model predictions (meta data)

```
meta_X1, meta_y= get_out_of_fold_predictions1(trainx,trainy)
print('Meta ', meta_X1.shape, meta_y.shape)
meta_X2, meta_y= get_out_of_fold_predictions2(trainx,trainy)
print('Meta ', meta_X2.shape, meta_y.shape)
meta_X3, meta_y = get_out_of_fold_predictions(trainx,trainy)
print('Meta ', meta_X3.shape, meta_y.shape)
meta_X4, meta_y= get_out_of_fold_predictions4(trainx,trainy)
print('Meta ', meta_X4.shape, meta_y.shape)
meta_X5, meta_y= get_out_of_fold_predictions5(trainx,trainy)
print('Meta ', meta_X5.shape, meta_y.shape)
meta_X5, meta_y= get_out_of_fold_predictions5(trainx,trainy)
print('Meta ', meta_X5.shape, meta_y.shape)
meta_X6, meta_y= get_out_of_fold_predictions6(trainx,trainy)
print('Meta ', meta_X6.shape, meta_y.shape)
meta_X7, meta_y= get_out_of_fold_predictions7(trainx,trainy)
print('Meta ', meta_X7.shape, meta_y.shape)
meta_X8, meta_y= get_out_of_fold_predictions8(trainx,trainy)
```

```
print('Meta ', meta_X8.shape, meta_y.shape)
```

The function to call trained ensemble base models

```
def get_models(n_models):
    all_models = list()
    for i in range(n_models):
```

Define filename for this ensemble

```
        filename = 'Catchment_SE_Models_input/model_' + str(i + 1) + '.h5'
```

Load model from file

```
        try:
            modell = load_model(filename)
        except:
            modelp = pickle.load(open(filename, 'rb'))
```

Add to list of members

```
        try:
            all_models.append(modelp)
        except:
            all_models.append(modell)
        print('>loaded %s' % filename)
    return all_models
```

The function to call trained non-ensemble base models

```
def get_basemodels(n_models):
    all_models = list()
    for i in range(n_models):
```

Define filename for this ensemble

```
        filename = 'Catchment_Base_Models_input/model_' + str(i + 1) + '.h5'
```

Load model from file

```
        try:
            modell = load_model(filename)
        except:
            modelp = pickle.load(open(filename, 'rb'))
```

Add to list of members

```
        try:
            all_models.append(modelp)
        except:
            all_models.append(modell)
        print('>loaded %s' % filename)
    return all_models
```

Call trained ensemble base models

```
n_members = 8
models =get_models(n_members)
print('Loaded %d models' % len(models))
```

Call trained non-ensemble base models

```
n_members = 8
basemodels =get_basemodels(n_members)
print('Loaded %d models' % len(basemodels))
```

Stack base model predictions of all models (stacked meta data)

```
meta_x = np.hstack((meta_X1,meta_X2,meta_X3,meta_X4,meta_X5,meta_X6,meta_X7,meta_X8))
```

The function for Bayesian model averaging with linear regression (BMA)

```
class BMA:
    def __init__(self, y, X, **kwargs):
        self.y = y
        self.X = X
        self.names = list(X.columns)
        self.nRows, self.nCols = np.shape(X)
        self.likelihoods = np.zeros(self.nCols)
        self.coefficients = np.zeros(self.nCols)
        self.probabilities = np.zeros(self.nCols)
        self.names = list(X.columns)
        if 'MaxVars' in kwargs.keys():
            self.MaxVars = kwargs['MaxVars']
```

```
        else:
            self.MaxVars = self.nCols
        if 'Priors' in kwargs.keys():
            if np.size(kwargs['Priors']) == self.nCols:
                self.Priors = kwargs['Priors']
            else:
                print("WARNING: Provided priors error.  Using equal priors instead.")
                print("The priors should be a numpy array of length equal tot he number of regressor variables.")
                self.Priors = np.ones(self.nCols)
        else:
            self.Priors = np.ones(self.nCols)
    def fit(self):
        likelighood_sum = 0
        for num_elements in range(1,self.MaxVars+1):
            model_indice_sets = list(combinations(list(range(self.nCols)), num_elements))
            for model_indice_set in model_indice_sets:
                model_X = self.X.iloc[:,list(model_indice_set)]
                model_regr = LinearRegression()
                model_regr.fit(model_X,self.y)
                num_params = len(model_regr.coef_) + 1
                yhat = model_regr.predict(model_X)
                mse = mean_squared_error(self.y, yhat)
                model_likelihood = len(y)*log(mse) + num_params * log(len(y))
                print("Model Variables:",model_indice_set,"likelihood=",model_likelihood)
                likelighood_sum = likelighood_sum + model_likelihood
                for idx, i in zip(model_indice_set, range(num_elements)):
                    self.likelihoods[idx] = self.likelihoods[idx] + model_likelihood
                    self.coefficients[idx] = self.coefficients[idx] + model_regr.coef_[i]*model_likelihood
        self.probabilities = self.likelihoods/likelighood_sum
        self.coefficients = self.coefficients/likelighood_sum
        return self
```

```
    def summary(self):
        df = pd.DataFrame([self.names, list(self.probabilities), list(self.coefficients)],
            ["Variable Name", "Probability", "Avg. Coefficient"]).T
        return df
```

Run BMA function and save average coefficients for each variable

```
 meta_x = pd.DataFrame(meta_x,columns
=["GRU","LSTM","MLP","CNN_GRU","SVR","LASSO","XGBOOST","LR"])

meta_y = pd.DataFrame(meta_y,columns =["Qobs"])

X = meta_x[["GRU","LSTM","MLP","CNN_GRU","SVR","LASSO","XGBOOST","LR"]]

y = meta_y["Qobs"]

result = BMA(y,add_constant(X)).fit()

coeff = result.summary()

score_m1 = coeff.loc[1,"Avg. Coefficient"]

score_m2 = coeff.loc[2,"Avg. Coefficient"]

score_m3 = coeff.loc[3,"Avg. Coefficient"]

score_m4 = coeff.loc[4,"Avg. Coefficient"]

score_m5 = coeff.loc[5,"Avg. Coefficient"]

score_m6 = coeff.loc[6,"Avg. Coefficient"]

score_m7 = coeff.loc[7,"Avg. Coefficient"]

score_m8 = coeff.loc[8,"Avg. Coefficient"]

score = np.hstack((score_m1,score_m2,score_m3,score_m4,score_m5,score_m6,score_m7,score_m8))

score = score.reshape((1,score.shape[0]))

df = pd.DataFrame(meta_x, columns=['GRU', 'LSTM', 'MLP', 'CNN-GRU', 'SVR', 'Lasso', 'XGB', 'LR'])

s = pd.DataFrame(score, columns=['GRU', 'LSTM', 'MLP', 'CNN-GRU', 'SVR', 'Lasso', 'XGB', 'LR'])

multiply =df.multiply(np.array(s), axis='columns')

multiply.loc[:,'Total'] = multiply.sum(axis=1)

prediction = multiply['Total']

prediction.shape
```

Predict with fitted meta model (BMA) and test data set

```
def BMA_Ensemble_predictions(X, models):
    meta_X = list()
    yhat= list()
    for model in models:
        try:
            yhat = model.predict(X)
            meta_X.append(yhat.reshape(len(yhat),1))
        except:
            try:
                yhat = model.predict(X.reshape((X.shape[0], (X.shape[1] * X.shape[2]))))
                meta_X.append(yhat.reshape(len(yhat),1))
            except:
                try:
                    n_input= X.shape[1]
                    n_features = 51
                    n_seq = 1
                    yhat = model.predict(X.reshape((X.shape[0],n_seq, n_input,n_features)))
                    meta_X.append(yhat.reshape(len(yhat),1))
                except:
                    print("data error")
    meta_X = hstack(meta_X)
    score = np.hstack((score_m1,score_m2,score_m3,score_m4,score_m5,score_m6,score_m7,score_m8))
    score = score.reshape((1,score.shape[0]))
    df = pd.DataFrame(meta_X, columns=['GRU', 'LSTM', 'MLP', 'CNN-GRU', 'SVR', 'Lasso', 'XGB', 'LR'])
    s = pd.DataFrame(score, columns=['GRU', 'LSTM', 'MLP', 'CNN-GRU', 'SVR', 'Lasso', 'XGB', 'LR'])
    multiply =df.multiply(np.array(s), axis='columns')
    multiply.loc[:,'Total'] = multiply.sum(axis=1)
    prediction = multiply['Total']
    yhat = pd.DataFrame(prediction)
```

```
    yhat.append(yhat)
    return yhat
```

Stack base model predictions and predict using the performance measures (WA)

```
score_m1 = sum(score_m1)/len(score_m1)
score_m2 = sum(score_m2)/len(score_m2)
score_m3 = sum(score_m3)/len(score_m3)
score_m4 = sum(score_m4)/len(score_m4)
score_m5 = sum(score_m5)/len(score_m5)
score_m6 = sum(score_m6)/len(score_m6)
score_m7 = sum(score_m7)/len(score_m7)
score_m8 = sum(score_m8)/len(score_m8)
meta_x = np.hstack((meta_X1,meta_X2,meta_X3,meta_X4,meta_X5,meta_X6,meta_X7,meta_X8))
score = np.hstack((score_m1,score_m2,score_m3,score_m4,score_m5,score_m6,score_m7,score_m8))
score = score.reshape((1,score.shape[0]))
df = pd.DataFrame(meta_x, columns=['GRU', 'LSTM', 'MLP', 'CNN-GRU', 'SVR', 'Lasso', 'XGB', 'LR'])
s = pd.DataFrame(score, columns=['GRU', 'LSTM', 'MLP', 'CNN-GRU', 'SVR', 'Lasso', 'XGB', 'LR'])
multiply =df.multiply(np.array(s), axis='columns')
multiply.loc[:,'Total'] = multiply.sum(axis=1)
summation = s.sum(axis=1)
summation = pd.DataFrame(summation)
prediction = multiply['Total']/summation.iloc[0,0]
prediction.shape
```

Predict with fitted meta model and test data set (WA)

```
def WA_Ensemble_predictions(X, models):
    meta_X = list()
    yhat= list()
    for model in models:
        try:
            yhat = model.predict(X)
            meta_X.append(yhat.reshape(len(yhat),1))
```

```
            except:
                try:
                    yhat = model.predict(X.reshape((X.shape[0], (X.shape[1] * X.shape[2]))))
                    meta_X.append(yhat.reshape(len(yhat),1))
                except:
                    try:
                        n_input= X.shape[1]
                        n_features = 51
                        n_seq = 1
                        yhat = model.predict(X.reshape((X.shape[0],n_seq, n_input,n_features)))
                        meta_X.append(yhat.reshape(len(yhat),1))
                    except:
                        print("data error")
        meta_X = hstack(meta_X)
        score = np.hstack((score_m1,score_m2,score_m3,score_m4,score_m5,score_m6,score_m7,score_m8))
        score = score.reshape((1,score.shape[0]))
        df = pd.DataFrame(meta_X, columns=['GRU', 'LSTM', 'MLP', 'CNN-GRU', 'SVR', 'Lasso', 'XGB', 'LR'])
        s = pd.DataFrame(score, columns=['GRU', 'LSTM', 'MLP', 'CNN-GRU', 'SVR', 'Lasso', 'XGB', 'LR'])
        multiply =df.multiply(np.array(s), axis='columns')
        multiply.loc[:,'Total'] = multiply.sum(axis=1)
        summation = s.sum(axis=1)
        summation = pd.DataFrame(summation)
        prediction = multiply['Total']/summation.iloc[0,0]
        yhat = pd.DataFrame(prediction)
        yhat.append(yhat)
        return yhat
```

The function for meta model (ETR)

```
    def fit_meta_model(X, y):
        model = ExtraTreesRegressor(n_estimators=500)
        model.fit(X, y)
```

```
    return model
```

Fit meta model (ETR)

```
meta_model = fit_meta_model(meta_x, meta_y)
```

Predict with fitted meta model and test data set for ETRSE

```
def super_learner_predictions_ETRSE(X, models, meta_model):
  meta_X = list()
  for model in models:
    try:
      yhat = model.predict(X)
      meta_X.append(yhat.reshape(len(yhat),1))
    except:
      try:
        yhat = model.predict(X.reshape((X.shape[0], (X.shape[1] * X.shape[2]))))
        meta_X.append(yhat.reshape(len(yhat),1))
      except:
        try:
          n_input= X.shape[1]
          n_features = 51
          n_seq = 1
          yhat = model.predict(X.reshape((X.shape[0],n_seq, n_input,n_features)))
          meta_X.append(yhat.reshape(len(yhat),1))
        except:
          print("data error")
  meta_X = hstack(meta_X)
  return meta_model.predict(meta_X)
```

Predict and evaluate the performance of superensemble models (BMA, WA, and ETR)

```
y_test = scalery.inverse_transform(testy)
```

Predict with superensemble model (BMA)

```
y_pred = BMA_Ensemble_predictions(testx, models)
y_pred = scalery.inverse_transform(y_pred)
```

Predict with superensemble model (WA)

```
y_pred = WA_Ensemble_predictions(testx, models)
y_pred = scalery.inverse_transform(y_pred)
```

Predict with superensemble model (ETR)

```
y_pred = super_learner_predictions_ETRSE(testx, models, meta_model)
y_pred = y_pred.reshape((y_pred.shape[0],1))
y_pred = scalery.inverse_transform(y_pred)
timeseriesdate = list(timeseries)[0:1]
test = timeseries[-730:]
date = test[timeseriesdate]
y_pred_data = pd.DataFrame(y_pred)
y_test_data = pd.DataFrame(y_test)
date.set_indice("date")
xt=date["date"]
xt= pd.DataFrame(xt)
xt=xt.reset_indice(inplace=False)
xt=xt.drop("indice",axis=1)
yt= pd.DataFrame(y_test)
yt=yt.reset_indice(inplace=False)
yt=yt.drop("indice",axis=1)
yp=y_pred_data[0]
yp=pd.DataFrame(yp)
xt.shape
yp.shape
```

Evaluating the performance

```
mse = mean_squared_error(y_test,y_pred)
mae = mean_absolute_error(y_test,y_pred)
medae = median_absolute_error(y_test, y_pred)
mape= mean_absolute_percentage_error(y_test,y_pred)
r2 = r2_score(y_test,y_pred)
std = np.std(y_pred)
```

```
corr=pearsonr(y_pred_data[0],y_test_data[0])[0]

rmse = sqrt(mse)

print('Superlearner RMSE: %f' % rmse)

print('Superlearner MAE: %f' % mae)

print('Superlearner MEDAE: %f' %medae)

print('Superlearner corr: %f' %corr)

print('Superlearner std: %f' %std)

print('Superlearner R2: %f' %r2)
```