



Evaluation of deep-learning and tree-boosting machine learning models in automatic error correction of forecasts from a physics-based model: A case study on Storå river, Denmark.



Master of Science Thesis
by

Sivarama Krishna Reddy Chidepudi

Supervisors

Dr. Nicola Balbarini (DHI)
Dr. Laura Frølich (DHI)
Prof. Niels Schütze (TU Dresden)

Examination Committee

Prof. Niels Schütze (TU Dresden), Chairman
Prof. Dimitri Solomatine (IHE Delft)
Dr. Nicola Balbarini (DHI)
Dr. Laura Frølich (DHI)

This research is submitted in partial fulfillment of requirements for the MSc degree in Hydro Science and Engineering at the Technische Universität Dresden, Germany

Dresden
20th August 2021

Declaration of Independence

I hereby declare that I submitted my MSc thesis to the examination board of the department of hydro-science and engineering entitled:

Evaluation of deep-learning and tree-boosting machine learning models in automatic error correction of forecasts from a physics-based model: A case study on Storå river, Denmark.

I completed this work independently without using any other sources or aids other than those specified, and I marked the citations appropriately.

Dresden: 20/08/2021



Sivarama Krishna Reddy Chidepudi

Abstract

Accurate real-time flood predictions play a vital role in flood early warning systems, which further helps in mitigating the damage and saving lives. Error correction using machine learning (ML) in physics-based models (alternatively known as physically-based models) has been widely considered and recommended in the literature to improve forecast accuracy. This study mainly focuses on evaluating the ability of novel tree-based ML methods and Bidirectional LSTM (BLSTM) at different lead times and high flow conditions. Also, the performance of these methods is compared with the traditionally used autoregression (AR), Multilayer perceptron (MLP), and naïve models. So overall, we evaluated six data-driven models and one naïve model on Storå river to correct the errors in the physics-based model: Two tree-boosting ML models (XGBoost, Gradient boosting), two deep learning-based models (MLP, BLSTM), and then simple models like autoregression (AR) & persistence (or naïve). Then, a stacked model combining XGBoost, and AR is developed and tested. Hyperparameter tuning is performed using Bayesian optimization. Results on the independent test set show that all the methods can improve the discharge simulations from a physics-based model. However, the Bidirectional LSTM and stacked model are consistently performed slightly better than other models in all lead times. At shorter lead times, tree-boosting approaches marginally underperformed. While gradient boosting performed better at longer lead times and produced results comparable to BLSTM and stacked models, XGBoost continues to underperform but gave better results than AR and PERS & MLP. The BLSTM and stacked models performed well under high flow conditions as well. Even though the difference is minor, they consistently outperformed all the other models. Furthermore, while tree-based methods (XGBoost & gradient boosting) fared somewhat worse than BLSTM & stacked model, they outperformed basic methods (AR/Pers) and MLP at high flow conditions. One additional key finding in this study is that even when the stacked model was built using less computationally intensive methods (XGBoost & AR), it produced equivalent results to BLSTM.

Keywords

Error correction, AR, ANN, XGBoost, Bidirectional LSTM, Gradient Boosting, Stacking, Physics-based model.

Acknowledgment

I want to thank my supervisors Dr. Laura Frølich & Dr. Nicola Balbarini from DHI, for their continuous support and direction throughout my thesis. I am highly indebted for their help since the beginning of the thesis through regular meetings, which helped me focus on achieving the targeted objectives with enormous freedom to explore all the possibilities in approaching the challenges faced at different phases in this study. Further, I am also thankful to my supervisors in DHI for arranging all the necessary things for my relocation and stay in Denmark during the tough times of the ongoing COVID-19 pandemic.

My special thanks to Prof. Dimitri Solomatine and Prof. Biswa Bhattacharya for their continuous support and valuable feedback throughout this MSc program. I am also highly thankful to Prof. Niels Schütze for his timely response and arranging all the academic requirements for my thesis examination at TU Dresden.

I sincerely thank all the Flood risk management MSc program professors who taught us in all the four universities (TU Dresden, IHE Delft, UPC Barcelona & University of Ljubljana) of the consortium. Including the staff members for their help in our mobility to all the four locations as planned. Especially Ms. Judith Pöschmann & Ms. Ineke Melis have played a crucial role in the mobility of this program. Thanks to European Commission for providing the funding through the Erasmus Mundus flood risk management program.

Finally, I am indebted to all my friends who are part of this MSc program and my family members who supported me in all possible ways.

Conceptual Formulation

Applying machine learning & deep learning models to mitigate errors in river predictions.

Introduction

High-accuracy flood prediction is important for real-time decision-making, such as early warning systems and emergency response planning. Sometimes, errors in predictions from physical models increase over time as assumptions underlying the physical model change. Automatic monitoring of predictions and subsequent observations may enable automatic correction of predictions.

Objectives

This study investigates the use of deep learning & machine learning models to automatically correct predictions from physics-based models based on recent predictions and observations. So, as to evaluate whether machine learning & deep learning methods can improve predictions from physical models, corrected predictions from machine learning methods are compared to the original predictions, both with reference to historical observations.

Methodology

Machine learning methods like artificial neural networks (ANNs) can be used to correct predictions of complex systems (e.g., Watson,(2019)). If the machine learning or deep learning models that work well are found, these models can be inspected to improve the modeled discharge normally and at high flow conditions. Also, the possibilities of combining multiple models are to be assessed.

Expected outcome

One or more machine learning & deep learning models are trained and tested at a site with an available flood model to assess their ability to improve predictions. The performance of predictions corrected with machine learning solutions is compared with existing flood model predictions.

Required skills

Experience in Python programming, knowledge of machine learning models.

Table of Contents

Declaration of Independence	ii
Abstract.....	iii
Acknowledgment	iv
Conceptual Formulation.....	v
List of Figures	ix
List of Tables.....	xi
List of Symbols and abbreviations	xii
Chapter 1 Introduction.....	2
1.1 Purpose	2
1.2 Background.....	2
1.3 Scope	2
1.4 Research Questions	3
1.5 Innovation	4
1.6 Practical Value.....	4
1.7 Objective.....	4
1.8 Structure of the thesis	4
Chapter 2 Literature Review.....	5
2.1 Auto-Regressive (AR) Models.....	5
2.1.1 AR models with exogenous input variables.....	5
2.2 Machine Learning models	5
2.3 Combination of approaches & hybrid models	6
2.4 Hydrologic post-processing.....	7
2.5 Knowledge-based modular models.....	8
Chapter 3 Methodology	9
3.1 Data splitting.....	9
3.2 Selection of input variables	10
3.3 Data scaling.....	11

3.4	Hyperparameter optimization.	12
3.5	Performance evaluation measures	13
3.6	Models evaluated	15
3.6.1	Persistence Model	16
3.6.2	Auto Regressive Model.....	16
3.6.3	Deep Learning methods	16
3.6.4	Tree-boosting methods.....	22
3.6.5	Stacking Regressor	23
3.7	Software Used.....	23
Chapter 4	Case Study: Storå River, Denmark	24
4.1	Study area	24
4.2	Climate and Topography	25
4.3	Physics-based model.....	26
4.4	Data Availability	26
4.4.1	Discharge (Observed and Modelled)	27
4.4.2	Precipitation (Observed and Forecasted).....	27
4.5	Previous research in the study area	27
Chapter 5	Experimental design.....	28
5.1	Data Preprocessing	31
5.1.1	Data Splitting.....	31
5.1.2	Input variables selection	32
5.1.3	Data scaling	33
5.2	Hyperparameter tuning.....	33
5.2.1	Neural network-based methods	33
5.2.2	Boosting methods	35
5.3	Final Testing	36
Chapter 6	Results.....	38
6.1	Results on the validation set	39

6.2	Results on the final testing set	41
6.3	Discussion on Results	48
Chapter 7	Conclusions and future scope	49
7.1	Conclusions	49
7.2	Limitations & future scope	50
References	52
Appendices	60
Appendix A:	Comparison of Pearson correlation and Average mutual information at different lead times.....	60
Appendix B:	Distribution plots of hyperparameters in Bayesian optimization for tree- based and deep learning methods.	62
Appendix C:	Learning curves of ANN at different lead times during validation and test	69
Appendix D:	Learning curves of BLSTM at different lead times for validation and test sets	71
Appendix E:	Final best hyperparameters for the models evaluated	73
Appendix F:	Comparison of corrected discharges at different lead times.....	75

List of Figures

Figure 1: Categories of data splitting	9
Figure 2: Representation of time series cross-validation.....	10
Figure 3: Classification of models.....	15
Figure 4: Schematic representation of multilayer perceptron with one hidden layer (adapted from (Solomatine, 2017)	18
Figure 5: Schematic representation of simple LSTM information flow at three consecutive timesteps $t-1$, t , $t+1$ (Modified and adapted from (Alizadeh et al., 2021)).	19
Figure 6: Schematic representation of Bidirectional LSTM (adapted and modified from (Saeed et al., 2020))	21
Figure 7: Base map of Storå River, Denmark	24
Figure 8: Average precipitation(top) and temperature(bottom) in Denmark between 1873-2008. (Source: (EEA, 2020)	25
Figure 9: Modelled and observed discharge time-series at SKÆRUM BRO station for 2011-2019	27
Figure 10: General evaluation methodology	29
Figure 11: Schematic representation of workflow	30
Figure 12: Candidate variables time-series at SKÆRUM BRO station (Validation)	32
Figure 13: Candidate variables time-series at SKÆRUM BRO station (Test)	32
Figure 14: Average mutual information with input parameters Vs. E_{t+1}	38
Figure 15: Pearson correlation in input parameters vs. E_{t+1}	38
Figure 16: Performance statistics on validation data at different lead times	39
Figure 17: Comparison plots of corrected discharge time series for the validation data at 48 hours lead time.....	40
Figure 18: Performance statistics on final training data at different lead times.	42
Figure 19: Performance statistics on the test data at different lead times.....	43
Figure 20: Comparison plots of corrected discharge time series at 48 hours lead time on the test set.....	47
Figure 21: Comparison plots of corrected discharge time series at 12 hours lead time on the validation set.	75
Figure 22: Comparison plots of corrected discharge time series at 12 hours lead time on the test set.....	76
Figure 23: Comparison plots of corrected discharge time series at 24 hours lead time on the validation set.	77

Figure 24: Comparison plots of corrected discharge time series at 24 hours lead time on the test set.....	78
Figure 25: Comparison plots of corrected discharge time series at 36 hours lead time on the validation set.	79
Figure 26: Comparison plots of corrected discharge time series at 36 hours lead time on the test set.....	80

List of Tables

Table 1: Loss functions	18
Table 2: Timeline of the data sets after splitting.....	31
Table 3: Statistical Properties of error data in different splits (Mean, Min, Max, Std dev, No of samples)	31
Table 4: Range of hyperparameter values used for multilayer perceptron.....	34
Table 5: Range of hyperparameter values used for Bidirectional LSTM	35
Table 6: Range of hyperparameter values used for Gradient boosting	35
Table 7: Range of hyperparameter values used for newton boosting (XGBoost)	36
Table 8: Best hyperparameters for a lead time of 12 hours.....	37
Table 9: RMSE obtained on training dataset at different lead times	41
Table 10: RMSE obtained on the test data set at different lead times.	42
Table 11: Change (%) in RMSE after correction on the test set.	44
Table 12: Change (%) in MAE after correction on the test set.	44
Table 13: Change (%) in R^2 after correction on the test set.....	45
Table 14: Change (%) in RMSE after correction in high flow events within the test set.	45
Table 15: Change (%) in MAE after correction in high flow events within the test set.	46
Table 16: % Change in R^2 after correction in high flow events within the test set. ...	46
Table 17: Final best hyperparameters for tree-based and neural network-based methods.....	73
Table 18: Final best hyperparameters for neural network-based methods.....	74

List of Symbols and abbreviations

ANN: Artificial Neural Network	MCP: Model Conditional Processor
AI: Artificial Intelligence	MCD: Monte-Carlo Dropout
ARXM: linear Auto-Regressive Exogenous-input model	MDN: Mixture Density Networks
AR: Autoregressive	MAE: Mean Absolute Error
ARMASA: automatic time series program (MATLAB Toolbox)	MS-EnsPost: multiscale postprocessor for ensemble streamflow prediction,
ARMA: Autoregressive Moving Average	NARXM: Non-linear Auto-Regressive eXogenous-Input Model
BR: Bagged Regression	NNU: Neural Network Updating
BP: Back Propagation	PERS: Persistence
BLSTM: bidirectional LSTM	PGRNN: Physics-Guided Recurrent Neural Networks model
CNN: Convolutional Neural Network	PEs: Processing Elements
DNN: Deep Neural Network	ReLU: Rectified Linear Unit
DMI: Danish Meteorological Institute	RMSE: Root Mean Square Error
ECMWF: European Centre for Medium-Range Weather Forecasts	RNN: Recurrent Neural Networks
FEC: Flood Error Correction	SMAR: Soil Moisture Accounting and Routing
FFNN: Feed-Forward Neural Network	SB: Stacked Boosting
GBRT: Gradient Boosting Regression Tree	SAC-SMA: Sacramento Soil Moisture Accounting model
HBV: Hydrologiska byråns vattenbalansavdelning (Hydrological Model)	SANN: Sequential Artificial Neural Network
IoT: Internet of Things	TIGGE: THORPEX Interactive Grand Global Ensemble
LSTM: Long Short-Term Memory	TANH: Hyperbolic Tangent
LTF: Linear Transfer Function	XGBoost: Extreme Gradient Boosting
MLP: Multilayer Perceptron	

Chapter 1 Introduction

This chapter introduces the current study with its scope, practical value, and research objectives.

1.1 Purpose

Accurate real-time flood predictions play a vital role in decision making, as in Early warning systems & Emergency response planning. Improved predictions also contribute to the enhanced safety of residents and reduced material damage. However, errors in predictions from physics-based models can increase over time as assumptions underlying the model change. Some of these errors are inevitable due to uncertainties in the process. However, recent studies show forecast accuracy can be improved in physics-based models using real-time error correction methods (L. Chen et al., 2015; Madsen & Skotner, 2005).

1.2 Background

Error Correction (updating the output variables) methods widely improve the accuracy of flood forecasts (Sun et al., 2018)). Since the beginning of hydrological forecasting, many studies have evaluated the potential of error correction methods in the past few decades. These studies focused on traditionally used AR Models (Goswami et al., 2005; Lundberg, 1982; Refsgaard, 1997; Wu et al., 2012), Neural Network models (Abrahart & See, 2000; Prakash et al., 2014), and KNN ((Akbari & Afshar, 2014; Wani et al., 2017).

1.3 Scope

Previous works mainly used error correction methods like autoregressive-moving-average (ARMA) Moore, (2007), Wavelet transforms (Bogner & Kalas, (2008)) & Artificial Neural Networks (Babovic et al., (2000); Watson, (2019)) in real-time flood forecasting. However, there are still some persistent deficiencies in these methods like deficient theoretical basis, truncated forecast period & requirement of additional parameters as highlighted in Li et al. (2020).

With recent advancements in machine learning (ML) & deep-learning (DL), many new error correction methods for physics-based models gained traction across various fields. Watson (2019) tested error correction (using ANN) in predicting the chaotic Lorenz'96 system and highlighted that it's easier to go for error correction than replace

the physically-based models. Jia et al. (2018) proposed a Physics-Guided Recurrent Neural Networks model (PGRNN) using LSTM and RNN in simulating lake water temperature. Ellenson et al. (2020) showed that Bagged regression (BR) successfully detected error patterns in wave model outputs. Nearing et al. (2020) implemented a post-processing strategy using LSTM on a conceptual model SAC-SMA and found improvements in catchments with more snow. However, Siami-Namini et al. (2019) showed that BLSTMs outperformed regular LSTMs and ARIMA in time series forecasting because of the additional training layer in BLSTM, which improves learning long-term dependencies.

In addition to this, novel ML methods like Gradient Boosting regression Tree (GBRT) performed well in River Stage Forecasting Fu et al. (2019) and in predicting Mean Wave Overtopping Discharge (den Bieman et al., 2020). Moreover, Ibrahim Ahmed Osman et al. (2021) showed that the Extreme Gradient Boosting (XGBoost) model outperformed the ANN and Support Vector Regression models in predicting groundwater levels. Sigrist (2021) showed that Newton boosting performed better than gradient boosting in predictive accuracy. So far, BLSTM, GBRT & NBRT were not tested on error correction in physics-based models simulating discharge. It would be interesting to try these ML & DL methods as earlier ML methods have shown impressive performance in error correction. Thus, the main objective in the current study is to develop the framework for BLSTM, Newton (XGBoost) & Gradient Boosting regression trees for automatic error correction and then evaluate the performance against the traditional autoregressive (AR) & ANN models. Further, the possibilities of combining multiple models are to be assessed.

1.4 Research Questions

Specific research questions related to error correction within the framework of this study are as follows:

- Can the Machine learning model (NBRT, GBRT, BLSTM) improve the performance accuracy (R^2 , RMSE&MAE) over shorter (12hrs) and longer lead times (36-42hrs)?
- Will this model perform better than the traditionally used AR & ANN models?
- How will the proposed model, after combining multiple models using stacking regressor perform?
- How will these models perform at high flow conditions?

1.5 Innovation

We evaluated the novel methods in machine learning and deep learning in error correction of physics-based models in discharge predictions.

1.6 Practical Value

Automatic monitoring of predictions and subsequent observations may enable automatic correction of forecasts. Effective error correction leads to more accurate real-time flood predictions and operational flow control systems, which helps in effective decision making. The current study can also be helpful in flood damage mitigation, saving lives, and efficient operation of flow structures (e.g., reservoir). Moreover, the machine learning or deep learning models that showed better accuracy in higher lead times can replace the traditional methods, delivering better results only in shorter lead times. Further, models are tested using the operational physics-based model. So, the proposed models that gave better accuracy can directly implement for operational flow forecasting.

1.7 Objective

This study mainly aims to evaluate the potential of new data-driven models (GBRT, NBRT, BLSTM & Stacked model) in error correction of discharge simulations from physics-based models.

1.8 Structure of the thesis

1st chapter starts with the introduction to the study and briefly describes the purpose and practical value along with the primary objective and research questions. Then the 2nd chapter includes a literature review on the existing methods of error correction and improving the forecast accuracy in rainfall-runoff models and uncertainty estimation. 3rd chapter consists of theoretical aspects of the methods used. Then comes the 4th chapter, which describes the case study area details and analysis of the available data. 5th chapter begins with the experimental design adopted and a brief description of the hyperparameter tuning performed in this study. Also, it discusses the approaches adopted in developing and testing the models in detail. Followed by the 6th chapter highlighting the main results obtained from the models considered and analyses the results in the research questions. Finally, the last chapter discusses the conclusions and limitations and the future scope of the current study.

Chapter 2 Literature Review

This chapter mainly focuses on the literature review of existing methods for error correction. First simple methods like AR are explored, then studies using machine-learning models are mentioned, followed by the combined approaches.

2.1 Auto-Regressive (AR) Models

Lundberg (1982) used a hydrological model (HBV) in conjunction with an AR error model and showed that the AR model considerably improved the short-term forecasts while not having much improvement in the long-term forecasts (10 days or more). Broersen & Weerts (2005) studied the automatic time series program ARMASA for error correction in an HBV-96 hourly model and performed better than AR.

Refsgaard (1997) compared two updating procedures (ARMA & Extended Kalman filtering) and concluded that they significantly improve the performance of short-range hydrological forecasting. Abrahart & See (2000) compared Neural Network and ARMA for continuous river flow forecasting and found similar results in both methods. Xiong & O'Connor (2002) evaluated methods for error correction, including AR, ANN, and a fuzzy autoregressive threshold, and showed that AR performed better ANN.

2.1.1 AR models with exogenous input variables

Shamseldin & O'Connor (2001) tested Non-linear Auto-Regressive exogenous-Input Model (NARXM) on simulating discharge forecasts (SMAR) and compared it with Linear Auto-Regressive Exogenous-input (ARXM) method and then suggested NARXM as a decent alternative to AR models. Goswami et al. (2005) compared eight error updating models in discharge forecasts (SMAR) and found them to give relatively good 1-day ahead predictions. Still, for higher lead times (6-days ahead), only three models (Non-linear Auto-Regressive exogenous-Input Model (NARXM), Linear Transfer Function (LTF), and Neural Network Updating (NNU)) performed well. They used rainfall observations as an ideal representation of rainfall forecasts.

2.2 Machine Learning models

Babovic *et al.* (2000) used ANNs to forecast errors in operational forecasting of current speed (MIKE 21) in the Danish Øresund Strait and found these to have good forecast skills. Khu *et al.* (2001) employed genetic programming for error correction in a rainfall-

runoff model. They found accurate results in predicting runoff for lead times within the time of concentration of the catchment. Torres-Rua *et al.* (2012) used two machine learning algorithms, relevance vector machine (RVM) and multilayer perceptron (MLP), in a simulation model with its application in canal flow control scheme and found them to be efficient in minimizing the error. (Prakash, Sudheer, and Srinivasan (2014)) proposed Sequential ANN (SANN) with error updating in river flow forecasting, and the results showed that SANN outperforms conventional ANN by providing accurate forecasts at higher lead times (up to 8 days ahead).

(Wunsch *et al.*, 2021) compared LSTM, CNN & NARX in groundwater level forecasting and found that NARX outperformed the other two when the training period is shorter but mentioned that LSTM and CNN could perform better if larger training periods are available.

2.3 Combination of approaches & hybrid models

Yu & Chen (2005) proposed an error correction model using fuzzy rules in a real-time flood forecasting system which improved discharge forecasts for one to four hours ahead. Similar results were obtained in real-time river stage correction using the Least Squares Method (Hsu, Fu, and Liu, (2003)) & a combination of forecast errors (AR & MA) (Wu *et al.*, (2012)). Shen *et al.* (2015) further incorporated a Kalman filter in the method proposed by Wu *et al.* (2012) and concluded that the accuracy improved on average by 50%.

Similarly, Madsen and Skotner (2005) proposed a data assimilation approach using hybrid filtering and error correction (Harmonic Error & Autoregressive(AR)) in operational flood forecasting (MIKE11). They found significant improvement in forecast accuracy for lead times up to 24 h.

Bogner and Kalas (2008) combined wavelet transformations & state-space models for error correction in discharge forecasts and found that the timing accuracy of the estimates improved. Further L. Chen *et al.* (2015) compared different combinations of a) real-time flood error correction (FEC) using AR with b) multi-model combination (MC) technique. They concluded that a combination of these two methods could increase reliability and accuracy.

Babel et al. (2020) Implemented a combined filtering and error correction forecast method with data assimilation in a physically-based (Urban Drainage) model. They found it to have increased forecast skill and lead time. Farchi et al. (2020) proposed a hybrid surrogate model by combining data assimilation and machine learning (DNN & CNN) for error correction in numerical weather Prediction.

Apart from this, in real-time discharge forecasts, the system response curve method is widely used for error correction. (Li et al., (2020); Liang et al., (2021)) and Pagano et al. (2011) proposed a dual-pass error correction technique for long and short-term memory corrections.

2.4 Hydrologic post-processing

Several studies developed hydrologic post-processors; Krzysztofowicz & Kelly (2000) proposed a hydrologic uncertainty processor (HUP) to obtain probabilistic river stage forecasts by aggregating all the uncertainties. Raftery et al. (2005) proposed a Bayesian model averaging (BMA) based statistical post-processing method to calibrate the ensemble forecasts.

Later, Todini (2008) proposed a Model Conditional Processor (MCP), an alternative to BMA & HUP for predictive uncertainty assessment. After that, (Bogner & Pappenberger, 2011) combined HUP with error correction methods to estimate the predictive uncertainty in the corrected flow of the flow forecasting system.

Ehlers et al. (2019) tested the k-Nearest Neighbors (kNN)- resampling method to generate residual uncertainty estimates and found it to give robust results for hydrological variables (e.g., soil moisture, hydraulic head, etc.).

This approach was before tested by Wani et al. (2017) for residual uncertainty in streamflow forecasting and found the accuracy to be comparable to other techniques like uncertainty estimation based on local errors and clustering (UNEEC; (Shrestha & Solomatine, 2006, 2008) and Quantile regression (QR; Dogulu et al., 2015; Weerts et al., 2011).

Tyralis et al. (2019) proposed a new post-processing approach where quantile regression is stacked with quantile regression forests to improve probabilistic predictions. Acharya et al. (2020) compared different quantile regression methods for

hydrological post-processing and found the results similar to earlier studies on QR configurations (López López et al., 2014).

Other methodologies adopted in the deep learning framework are: (Klotz et al., 2020) tested four strategies (3 Mixture Density Networks and 1 Monte Carlo Dropout) for uncertainty estimation in deep learning for rainfall-runoff modeling. Some researchers combined the LUBE framework with LSTM to generate prediction intervals in wind power forecasting (Banik et al., 2020; Saeed et al., 2020).

2.5 Knowledge-based modular models

A modular model (Corzo, 2009) is a model with structural representation using particular domain knowledge, also known as a committee machine. (Corzo & Solomatine, 2007a) employed three baseflow separation techniques into ANN models and found that modular models integrating hydrological knowledge performed better than traditional ANN-based models in streamflow forecasting. (Corzo & Solomatine, 2007b) studied different data partitioning techniques incorporating domain knowledge and found that developing local specialized models is effective in predictive modeling.

Further, Kayastha et al. (2013) formed a fuzzy committee model employing specialized hydrological models for different flow regimes to minimize the error at high and low flow conditions separately and showed that fuzzy committee outperformed the individual models.

(Pianosi et al., 2014) proposed an error correction model with a prior classification system based on flow condition and forecasted rainfall to identify the source of error and then use a data-driven model specific to the classified error source. Further, the results from this study show that even the combination of simple classification(if-then) and linear correction improved the forecast capabilities of the hydrological model.

Chapter 3 Methodology

This chapter mainly focuses on theoretical aspects of the approaches used during developing and evaluating the data-driven models.

3.1 Data splitting

The primary purpose of data splitting into three sets is to avoid overfitting and evaluate the model's performance on unseen data. Specifically, a validation set avoids the overfitting of the model and a testing set to verify the model performance on new data before putting it into operation. The total available data is split into three sets, as shown in Figure 1. Generally, it is required to split these subsets to have similar statistical distribution in all (Solomatine, 2017).

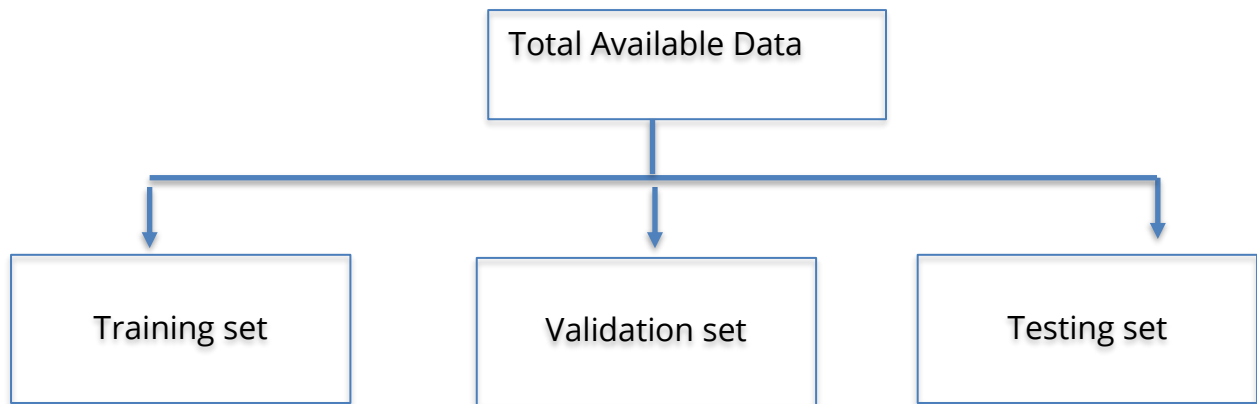


Figure 1: Categories of data splitting

In this study, splitting is performed as shown in Figure 1, i.e., three different sets in the proportion of 60:20:20. In addition to this, especially when there is limited data available, there are some other ways of cross-validation. The commonly used approach is K-fold cross-validation which splits the whole data into K parts. Each iteration uses one subset as a validation set to compute the performance metric by training the remaining subsets. Finally, this approach computes the average of all metrics to get the overall representation of the model performance.

Like K-fold, another approach called time-series split cross-validation is used in time series problems where chronological order must be maintained, and future values cannot be used to predict past values. Figure 2 shows the representation of splits in this approach.

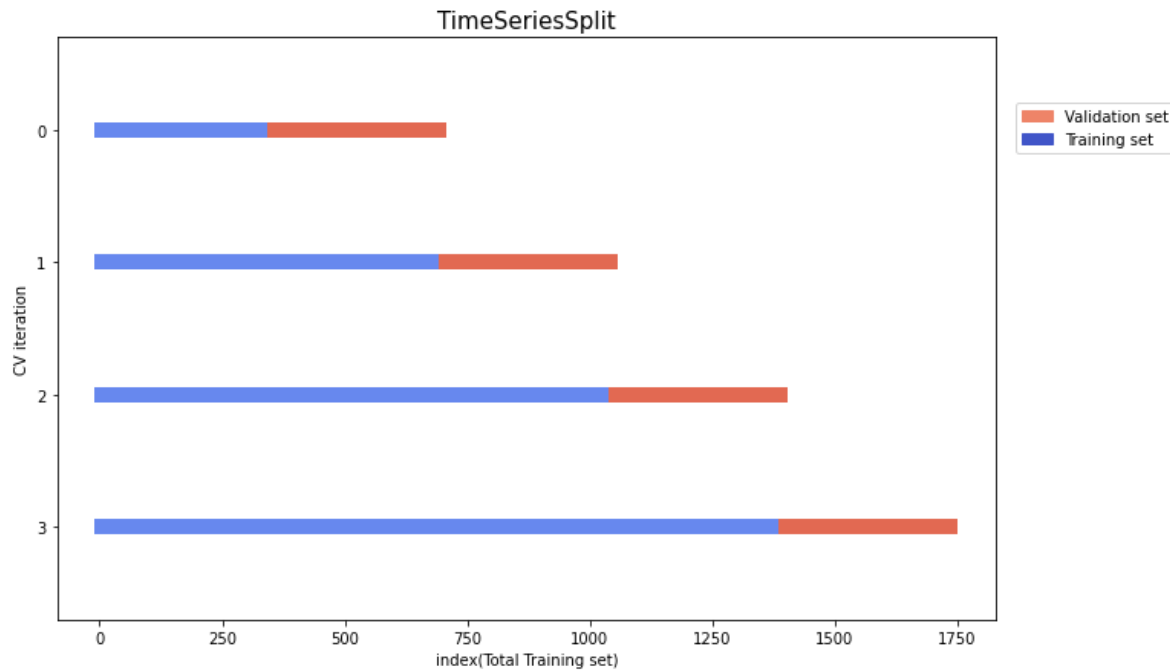


Figure 2: Representation of time series cross-validation

3.2 Selection of input variables

Input variables selection is a crucial step in building data-driven models for hydrological applications. It is not always possible to select all the possible variables even with enough data, as it makes the model complex, challenging to interpret, and more prone to overfitting. There are several methods to select "important" variables, and the most used ones are described below:

Correlation: The Pearson correlation coefficient R , between two variables X and Y , can be obtained using the equation (1).

$$R = \frac{\sum_{i=1}^k (x_i - \bar{x}) (y_i - \bar{y})}{\sqrt{\sum_{i=1}^k (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^k (y_i - \bar{y})^2}} \quad (1)$$

It quantifies the linear relationship strength between the observations and forecasts. Candidate variables with higher correlation with the target variables should be selected for model development.

Average mutual information (AMI): The AMI(Fraser & Swinney, 1986; Shrestha et al., 2009) between two random variables, X and Y, can be computed in equation (2)

$$AMI = \sum_{i,j} P_{XY}(x_i, y_i) \log_2 \left[\frac{P_{XY}(x_i, y_i)}{P_X(x_i)P_Y(y_i)} \right] \quad (2)$$

It quantifies the non-linear relationship between two variables. Like correlation, variables with high AMI values should be selected.

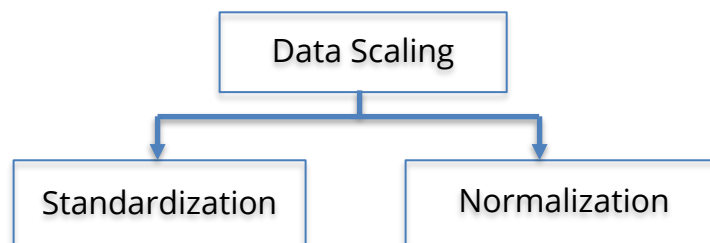
The selection of important variables has other advantages like reduced training times and improved computational efficiency in data-driven models. When there are many variables, dimensionality reduction techniques like principal component analysis (PCA) are used to develop a few composite variables, also known as features.

In addition to the variable selection methods mentioned above, the tree-based machine learning models also have impurity-based feature importance and permutation feature importance for input variable selection.

But in our current study, we focused on using simple methods like correlation and AMI for selecting the input variables.

3.3 Data scaling

Scaling of input and output variables can be performed by standardizing or normalizing. Scaling of input variables helps in making the learning stable and leads to faster convergence. While target variables scaling helps in avoiding the exploding gradients problem. Specifically, it helps when the input variables have different units and high variance in the values.



1. Standard scaler standardizes the data using the mean (x_{mean}) and standard deviation (σ) as shown in equation (3).

$$x_{scale} = \frac{(x - x_{mean})}{\sigma} \quad (3)$$

2. Min-max Scaler normalizes the data into the range of (0,1) using the equation shown in equation (4).

$$x_{scale} = (x - x_{min}) / (x_{max} - x_{min}) \quad (4)$$

Where x and x_{scale} Denotes the original and scaled data.

We performed scaling after splitting the data to avoid data leakages in the validation and test set. So, we used training data (Known data) to fit the scalar and then transform it on validation and test set.

For our current study, we used MinMaxScaler for data scaling (i.e., to normalize features).

3.4 Hyperparameter optimization.

Data-driven models have hyperparameters that need to be set during model initialization. They play a crucial role by controlling the overall training behavior of the model and significantly impact the model's overall performance. Hence, finding hyperparameters that suit our problem requirements is necessary. This process of finding the hyperparameters is known as hyperparameter optimization. There are several ways for optimizing the hyperparameters as described below:

1. Grid Search

This method first divides the domain of the hyperparameters into a discrete grid. Then, It evaluates every combination of values in this grid using performance metrics (RMSE, MSE, etc.) in a cross-validation set. While this method gives the best group of hyperparameters, it is prolonged.

2. Random Search

It is like grid search, but instead of checking all combinations of values as in grid search, this approach tests a randomly selected subset of the points in the grid. It may not give the best set of values as in grid search but still can provide a good set of values which can result in a good model.

3. Bayesian Optimization

Unlike random and grid search, Bayesian approaches use Bayes' theorem to obtain the minimum or maximum of the objective function. This approach develops a probabilistic model based on past evaluations. Then uses it to select the hyperparameter set for the subsequent assessment.

After testing all the methods mentioned above, we chose Bayesian optimization through the python implementation optuna to perform final hyper-parameter tuning. Optuna (Akiba et al., (2019)) uses Tree-structured Parzen estimators, also a Bayesian optimization form.

3.5 Performance evaluation measures

The performance of the models can be quantified using statistical indices/measures. The most used performance evaluation indices to estimate the quality of predictions from different models are root mean squared error, mean absolute error, & coefficient of determination, as described below.

1. Mean Absolute Error (MAE):

MAE gives the mean absolute difference between the corrected discharge and observed discharge, irrespective of sign, and it does not penalize the high errors, unlike RMSE. The MAE of corrected model discharges ($Q_{t+1_{Corr}}$) at 1 step ahead relative to observed discharge ($Q_{t+1_{obs}}$) takes the form of the equation (5). Values of MAE closer to zero are desirable.

$$MAE = \frac{1}{N} \sum_{i=1}^n |Q_{t+1_{Corr}} - Q_{t+1_{obs}}| \quad (5)$$

$$0 \leq MAE < +\infty$$

2. Root Mean Squared Error (RMSE):

RMSE gives the square root of the mean square error of the corrected discharges, and unlike MAE, it penalizes the high errors. More importantly, RMSE has the same units as the discharge (m^3/s) in our current study. The RMSE of corrected model discharges ($Q_{t+1_{Corr}}$) at one step ahead, relative to observed discharge ($Q_{t+1_{obs}}$) takes the form of the equation (6). Values of RMSE closer to zero are desirable.

$$RMSE = \sqrt{MSE} = \sqrt{\frac{\sum_{i=1}^n (Q_{t+1_{Corr}} - Q_{t+1_{obs}})^2}{n}} \quad (6)$$

$$0 \leq RMSE < +\infty$$

Further to assess the improvement, we used % of reduction in RMSE, which can be calculated as the equation (7).

$$\% \text{ Change} = \frac{(New \text{ Value} - Original \text{ Value})}{Original \text{ Value}} * 100 \quad (7)$$

In our case study, the new value refers to RMSE obtained from corrected discharges from different models. The original value refers to RMSE obtained from simulation discharge (i.e., before correction).

3. Coefficient of Determination:

The Coefficient of Determination (R^2) (also referred to as Nash Sutcliffe efficiency (Nash & Sutcliffe, 1970)) indicates the closeness of the variable data to the fitted regression line. The R^2 of corrected model discharges ($Q_{t+1_{Corr}}$) at 1 step ahead relative to observed discharge ($Q_{t+1_{obs}}$) takes the form of the equation (8). Values of R^2 closer to one are desirable.

$$R^2 = 1 - \frac{\sum (Q_{t+1_{Corr}} - Q_{t+1_{obs}})^2}{\sum (Q_{t+1_{Corr}} - Q_{t+1_{mean}})^2} \quad (8)$$

$$-\infty < R^2 \leq 1$$

4. High flow threshold

Further to testing the overall dataset, we are also evaluating the performance of the models at high flow events. The flow thresholds defined are as follows: High flow (Observed discharge > 80 percentile). We are focusing on only high flow events separately as it's the priority for the study area.

All the equations mentioned above are adapted and modified from (Solomatine 2017).

3.6 Models evaluated

The main objective here is to evaluate the BLSTM, tree-boosting machine learning methods, and stacked models to correct the error in the physics-based model. But the traditional models like persistence, AR, and MLP are also evaluated to understand the novel methods' performance relative to these traditional methods. The complete list of models evaluated and the broad classification is shown in Figure 3, followed by the description of each method and the hyperparameters involved.

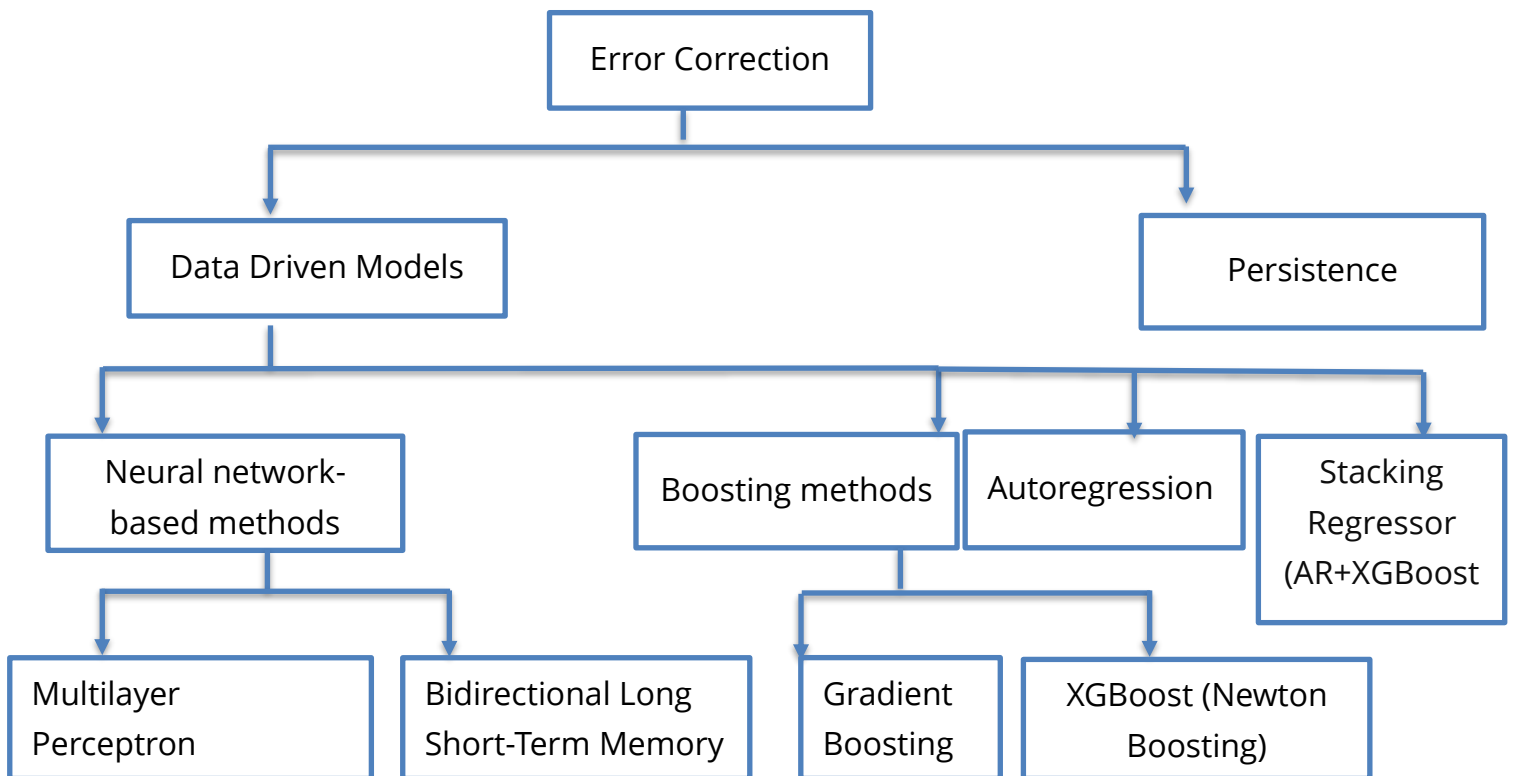


Figure 3: Classification of models

3.6.1 Persistence Model

The persistence model uses the values from the most recent time step available to determine the value for the upcoming time step. In other words, the error at time $t+i$ (where i is the number of time steps ahead) is equivalent to the error at time t .

$$E_{t+i} = E_t$$

3.6.2 Auto Regressive Model

The autoregressive model uses values from the previous time steps as an input to predict the values for the upcoming time steps with linear regression. A linear univariate autoregressive error correction model using four lags:

$$E_{t+i} = \text{AR}(5) = f(E_t, E_{t-1}, E_{t-2}, E_{t-3}, E_{t-4})$$

where i = number of time steps ahead.

3.6.3 Deep Learning methods

Two deep learning methods are being studied, one being the simple multilayer perceptron and the second one is the Bidirectional LSTM. The detailed description of each method, along with the simple LSTM for better understanding, is as follows:

1. Multilayer Perceptron

Multilayer Perceptron (MLP) is a commonly used variant of an artificial neural network (ANN) interconnected with several nodes (also known as neurons, units, or processing elements (PEs)). A simple MLP, as shown in Figure 4, consists of an input layer, an output layer, and a hidden layer.

The lines signify connection weights among nodes. The number of nodes present in the input layer is equivalent to the size of input features. This layer sends the input features (or variables) (x_i) to the units in hidden layers without performing any operation. Then the nodes in the hidden layer multiply the input using a set of weights. The output value will usually obtain using a bounded non-linear transfer function in the hidden layer (e.g., ReLU or tanh), which transforms the result.

The number of hidden nodes indicates the network complexity and determines its ability to approximate. Finally, the weights are updated using the backpropagation

algorithm, which mainly deals with computed errors and propagating them back through the network (Solomatine, 2017).

Hyperparameters involved in MLP are hidden layer, hidden units, optimizers, learning rate, dropout, epochs, loss functions, batch size, and activation functions.

The hyperparameter "number of epochs" signifies the complete passes the model performs during training.

Activation functions in deep learning models allow the model to learn nonlinearity in the data. Here we considered two commonly used activation functions for optimization as described below:

1. ReLU: Rectified Linear Activation function $[0, \infty)$

This function returns the input as output when the input is positive and returns zero when the input is negative.

2. Tanh: Hyperbolic tangent activation function. Range $(-1,1)$

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

Optimizers are the algorithms used to modify the attributes (weights and learning rate) in MLP to reduce the losses. Here we checked three different optimizers, and differences among them are described as follows:

1. Adam

Adaptive Moment Estimation (also known as Adam) takes adaptive learning rate from estimates of first and second-order moments.

2. SGD

Stochastic Gradient Descent does not change the learning rate during training, i.e., it maintains a constant learning rate.

3. RMSprop

Root Mean Squared Propagation also uses an adaptive learning rate.

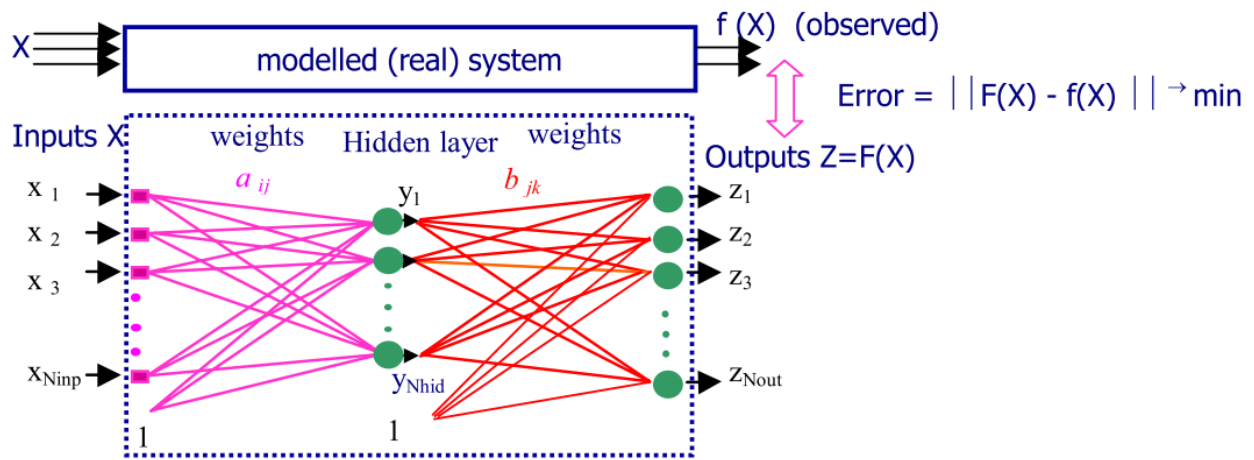


Figure 4: Schematic representation of multilayer perceptron with one hidden layer (adapted from (Solomatine, 2017))

Different loss functions penalize the outliers(errors) differently; hence five loss functions are optimized within the Bayesian optimization and other hyperparameters by choosing the metric (RMSE) as an objective function in this study.

Table 1 shows the outliers considered in this study and their effect on outliers.

Table 1: Loss functions

Loss function	Reaction to outliers
Mean square error	Penalizes the outliers heavily
Mean absolute error	Robust to outliers
Mean square logarithmic error	penalizes underestimates more than overestimates
Huber loss	Less sensitive to outliers
Logcosh	Less sensitive to outliers

2. Simple LSTM

LSTM (Alizadeh et al., 2021; Hochreiter, 1997) overcomes vanishing gradient problems and long-term dependency by presenting a novel hidden state, named cell state c_t , that retains the historical information. In addition, LSTM also has internal mechanisms to regulate the flow of information in the form of control gates. The gates are named forget, input & output gates.

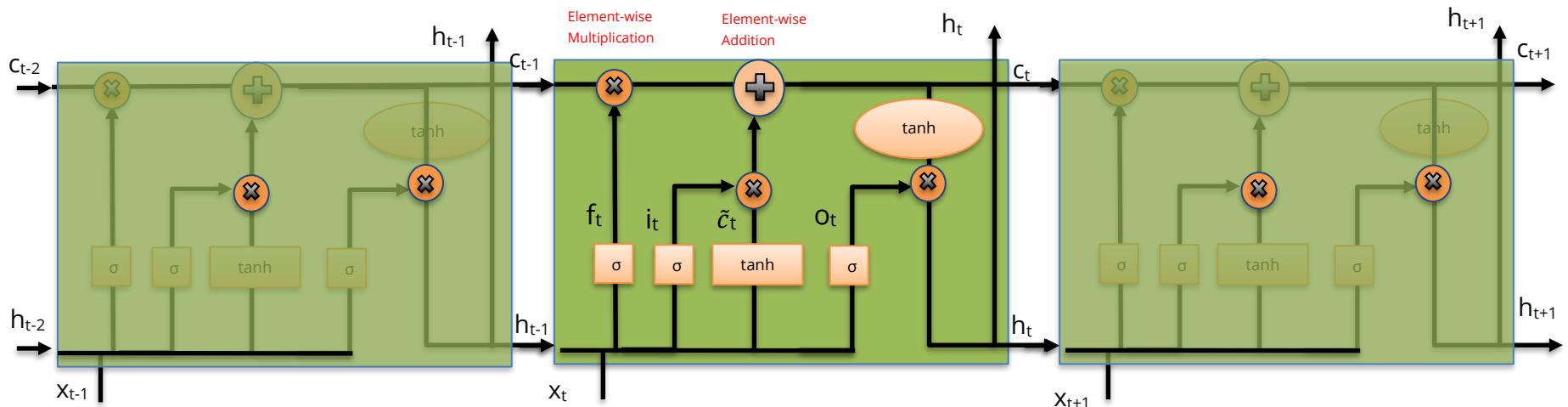


Figure 5: Schematic representation of simple LSTM information flow at three consecutive timesteps $t-1$, t , $t+1$ (Modified and adapted from (Alizadeh et al., 2021)).

The flow of information (Alizadeh et al., 2021) in simple LSTM is as depicted in Figure 5 and described below in three steps:

1. In step 1, the forget gate manages the information from the prior cell state c_{t-1} , which would then be added to the present state with the help of element-wise multiplication operator (\otimes) in the form as $f_t \otimes c_{t-1}$. This gate gives the binary output (0,1) with 0 indicating deletion of all previous information, and one means retaining all the information.

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

Then in the second step, the present cell state is computed in three phases

- a. The first phase involves converting the values of x_t and h_{t-1} into the range of (-1,1) to get a new cell state \tilde{c}_t using an activation function (tanh).

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c)$$

- b. In the second phase, values resulting from the input gate (i_t) are used to reorganize the present cell state c_t as $i_t \otimes \tilde{c}_t$. The input gate regulates both the sequence of input data at present (x_t) and hidden state information at t-1 (h_{t-1}), which incorporate into the cell state as:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

- c. In the final phase, the new cell state, c_t is obtained by adding a revised cell state in step 1 ($f_t \otimes c_{t-1}$) with the updated cell state in previous phases (2.b) ($i_t \otimes \tilde{c}_t$).

$$c_t = f_t \otimes c_{t-1} + i_t \otimes \tilde{c}_t$$

In the third and final step, the information in the new cell state that has to pass as an output of the present LSTM and the new hidden state to the upcoming cell is managed by the output gate (Alizadeh et al., 2021).

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

$$h_t = o_t \otimes \tanh(c_t)$$

Abbreviations used in the above equations:

W, U: matrix of network weights

f_t, i_t, o_t : outputs from forget, input, and output gates

c_t, c_{t-1} : Cell states at t and t-1

σ : Sigmoid function

h_t, h_{t-1} : Current and previous hidden states

\tilde{c}_t : cell candidate value

b: bias vector

3. Bidirectional LSTM

Bidirectional LSTM trains two LSTM models. The first model learns the input sequence, i.e., in the forward state, while the second model learns from the opposite direction of the input sequence, i.e., through backward states(Saeed et al., 2020), as depicted in Figure 6. Both models get merged using the concatenation mechanism by default. In other words, BLSTMs include an additional layer of training data than simple LSTMs.

Moreover, Siami-Namini et al. (2019) showed that BLSTMs outperformed regular LSTMs and ARIMA in time series forecasting because of the additional training layer in BLSTM, which improves learning long-term dependencies.

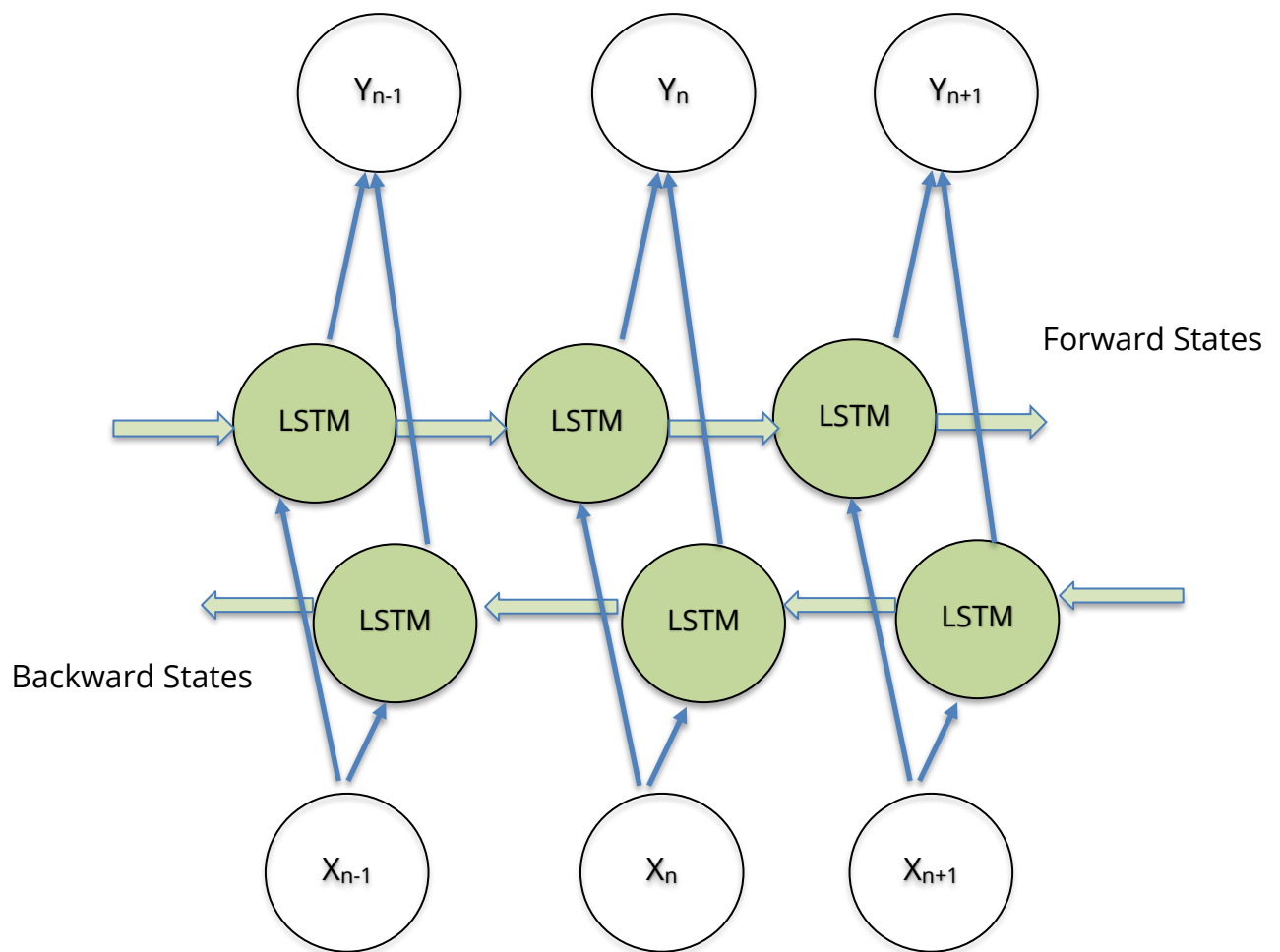


Figure 6: Schematic representation of Bidirectional LSTM (adapted and modified from (Saeed et al., 2020))

In addition to the hyperparameters mentioned in MLP, BLSTM has one other parameter, i.e., sequence length helps the model retain more training samples.

3.6.4 Tree-boosting methods

The current section describes the two tree-boosting machine learning methods (Gradient and Newton) of boosting used in this study.

Gradient Boosting(GB) ((Friedman, 2002) is an ensemble machine learning model commonly used for regression and classification tasks. GB uses the steepest gradient descent to minimize the loss function. The main idea behind these boosting or ensemble-based models is to adapt weak learners into strong learners by fitting weak learners to the previously evaluated negative gradient vector of the loss function. Here, weak learners are the regression trees that have low predictive ability. In simple terms, GB is an ensemble method that sequentially includes new models to the original ensemble. Specifically, “A new model is trained at each iteration to minimize the error of ensemble learned so far” (Papacharalampous et al., 2019).

Hyperparameters in gradient boosting are classified as boosting and tree-specific parameters based on their effect on boosting operations and an individual tree, respectively.

1. Tree-specific parameters are `min_samples_leaf`, `max_depth`.
 - `max_depth`; maximum depth in each tree, and
 - `min_samples_leaf`: minimum number of samples in the leaf node.

2. Boosting parameters are `n_estimators`, `subsample`, and learning rate.
 - `n_estimators` represent the number of trees,
 - `learning_rate` indicates the contribution of each tree on the final prediction,
 - the sub-sample shows the sample proportion to be used.

XGBoost (T. Chen & Guestrin, (2016)) is a scalable tree-boosting machine learning system. It solves the minimization of loss function using newton's method, i.e., through the second derivative. Hence some researchers called it an implementation of newton boosting (Didrick, 2016; Sigrist, 2021). Alternatively, it is an improved variant of Gradient boosting with regularization. It uses a more advanced regularized model form (L1 & L2) to control over-fitting, resulting in better performance through model generalization capabilities. Apart from hyperparameters in gradient boosting, XGBoost has L1 & L2 regularisation parameters (`alpha` and `lambda`) and `gamma`, which affect the model's performance.

(Sigris, 2021) differentiated the Gradient boosting and newton boosting techniques based on the updating steps (“Gradient descent and Second-order newton updates” (Sigris, 2021)) for finding the tree structures. The python package XGBoost implements the Newton boosting and Python library scikit-learn (Pedregosa et al., 2011) implements Gradient boosting regressor based on the approach of (Friedman 2001).

3.6.5 Stacking Regressor

Stacking regressor is a type of ensemble learning that combines the skills of different models to generate final estimates. SR trains all the models considered to make the predictions and uses them to generalize the final output. Using stacking regressor, we build a new model with XGBoost being the base estimator and autoregressive model as the final estimator.

3.7 Software Used

The Deep-Learning models are built using TensorFlow ((Abadi et al., 2016)) and Keras ((Chollet, 2015)). The machine learning framework used is Scikit learn (Pedregosa et al., 2011). we prepared all figures using Matplotlib ((Hunter, 2007)), Pandas((McKinney, 2010), & NumPy ((Van Der Walt et al., 2011)). The random seed function enables the models to generate reproducible results. Model development is entirely in the form of Jupyter notebooks.

All this work is carried out in python version 3.8.3 using a laptop with hardware configurations of Intel(R) Core (TM) i7-10510U CPU @ 1.80GHz 2.30 GHz and 8GB RAM

Chapter 4 Case Study: Storå River, Denmark

This chapter presents details of the case study, including the details of the study area, climate and topographical information, details of the physics-based model, and available data with some details of previous research in this area.

4.1 Study area

The present study is based on the Storå river and the SKÆRUM BRO discharge station (Figure 7). Vandkraftsøen is the large dam on the Storå river near the town Holstebro. It was built in 1941 for hydropower generation, but now it's partially used to manage the floods in the area. This dam and downstream of the river are famous for water sports like kayaking and canoeing. The river has flooding problems near the town of Holstebro and surrounding areas. The river originates between Silkeborg and Herning from a small town named Ikast, spanning over 104 kilometers. It's the second largest river in Denmark (DHI, 2021). The catchment area of the river is 1,100 km². Figure 7 shows the base map of the Storå river along with the discharge & precipitation stations locations.

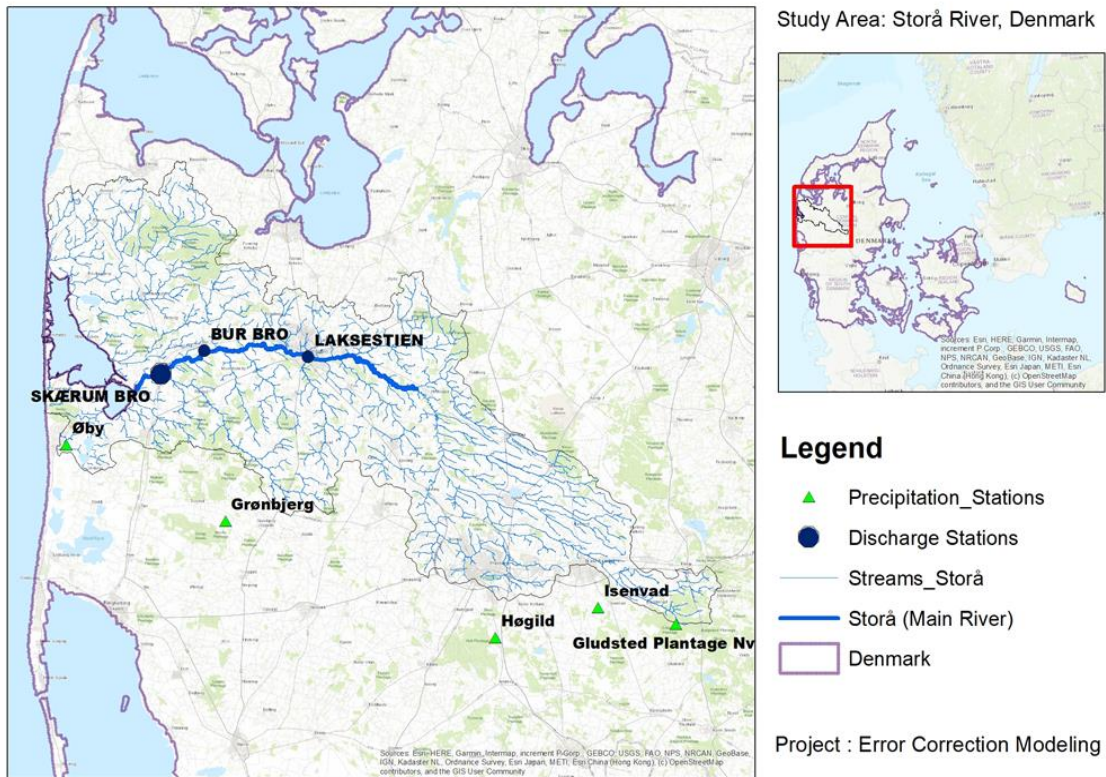


Figure 7: Base map of Storå River, Denmark

4.2 Climate and Topography

Denmark's climate is temperate, with an average annual temperature of 8.5°C, which has increased approximately by 1.5°C since 1873. And during the same period, the yearly average precipitation has increased by 100 mm, from 650mm per year before 1950 to 750 mm recently per year. The variation in rainfall and temperature is shown in Figure 8. Also, Denmark is a low-lying country with the highest altitude of 173m above the mean sea level (EEA, 2020):

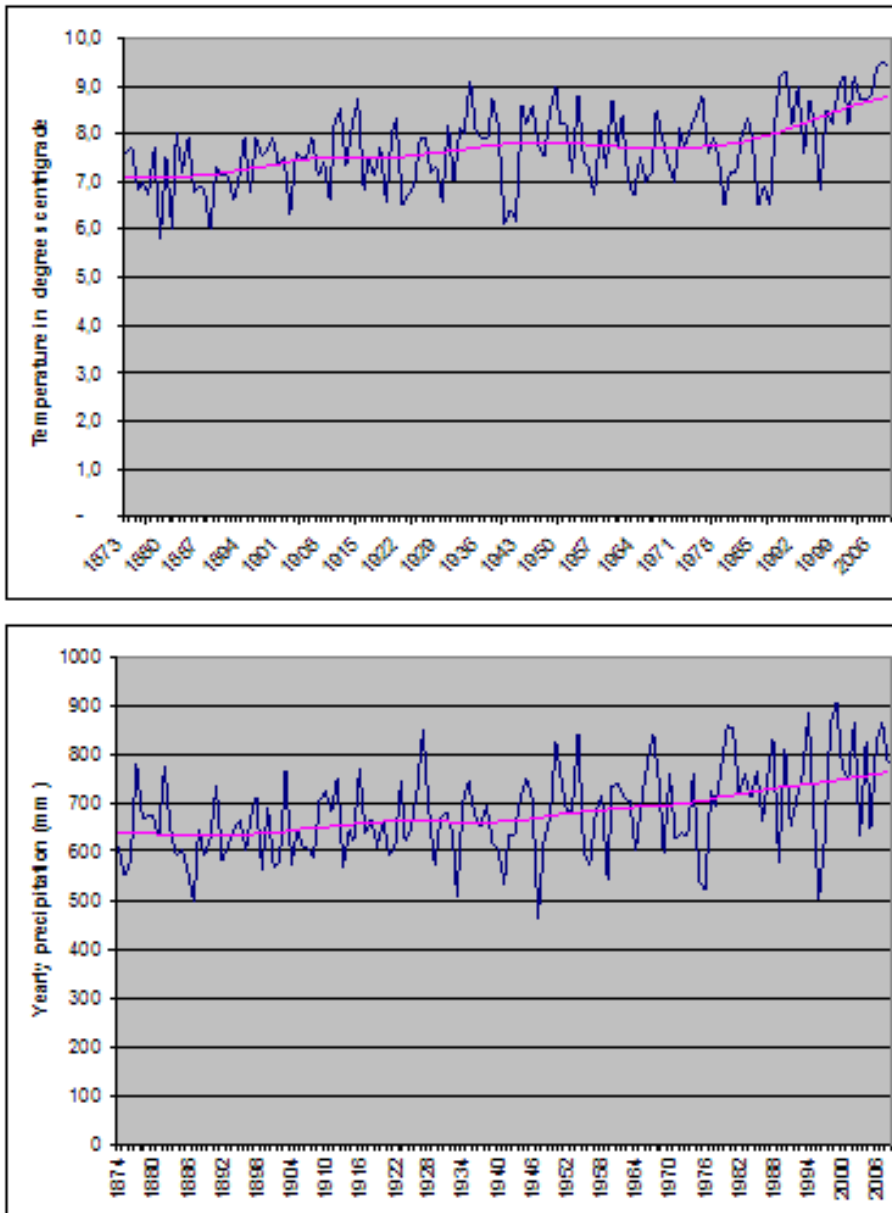


Figure 8: Average precipitation(top) and temperature(bottom) in Denmark between 1873-2008. (Source: (EEA, 2020)

4.3 Physics-based model

A downscaled hydrologic model (also commonly referred to as the NAM model) is built by DHI for the whole country, Denmark, using the framework of DHI's global hydrological model. The current NAM model is developed to provide forecasted discharge in areas outside of current detailed models. Inputs for the NAM model are precipitation, temperature, and potential evaporation. Historical weather data is taken from the Danish meteorological institute (DMI) and forecasted weather data from IBM, then used in country-wide distributed NAM models in each catchment of 1 Km². Runoff from each model is kinematically routed through the catchments to generate discharge simulations and other results from NAM models. This model operates every hour as it gets the updated real-time measurements and forecasts.

4.4 Data Availability

Observed and modeled discharge time series of the selected points (Figure 9) were plotted for 2011-2019 and the errors (residuals). Errors are calculated as the difference between the modeled and observed discharge. We use the observed & modeled data corresponding to 12:00 & 00:00 of each day in our study.

Data	Temporal resolution	Period
Observed Discharge	15 mins	2011-2019*(With some missing data)
Modelled Discharge	12 Hours	2011-2019
Forecasted Precipitation (ECMWF TIGGE)	12 hours	2011-2016,2018-2019
Observed Precipitation	Hourly	2010-2020

*41 days of observed discharge data values are missing, removing those dates in all other data sets.

4.4.1 Discharge (Observed and Modelled)

Observed discharge values are available at 15 mins temporal resolution, but to match with modeled discharge, Instantaneous observed values were taken corresponding to the modeled discharge values (i.e., at 00:00 & 12:00 of each day).

4.4.2 Precipitation (Observed and Forecasted)

Figure 7 shows the location of discharge and precipitation stations. Two precipitation stations (Øby & Grønbjerg) are available near the Skærum Bro discharge station. THORPEX Interactive Grand Global Ensemble (TIGGE) rainfall forecasts from ECMWF are available. Forecasted data (TIGGE) is retrieved from ECMWF's Meteorological Archival and Retrieval System (MARS) using python.

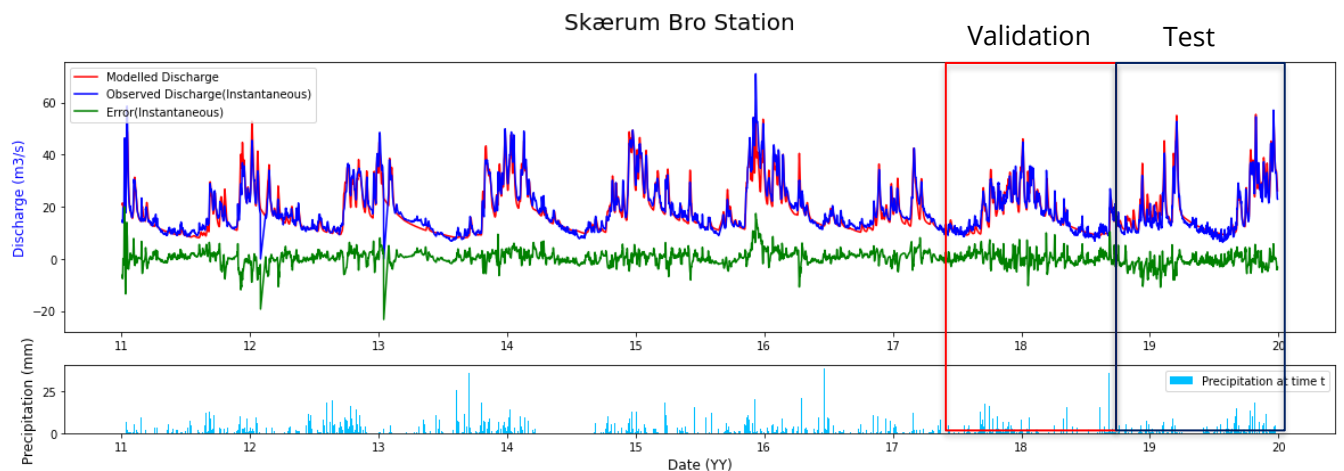


Figure 9: Modelled and observed discharge time-series at SKÆRUM BRO station for 2011-2019

4.5 Previous research in the study area

DHI performed the Initial research to implement AI and IoT-based prediction of water levels and flooding for the flood mitigation in Storå River. This project aims to identify the applicability of AI in flood early warning systems in Storå River (DHI, 2021). Under the coast-to-coast (C2C) climate challenge project, a pilot study on the Storå river established jointly by three municipalities (Brande, Herning, and Holstebro) is under progress. This project aims to examine the impacts of open area flooding in agricultural areas to minimize the flood risk in urban areas through stakeholder engagement. (C2CCC, 2018). River restoration was carried out on Storå river for the length of 0.5 kilometers under the EU-funded project REFORM (Reformrivers.EU, 2015).

Chapter 5 Experimental design

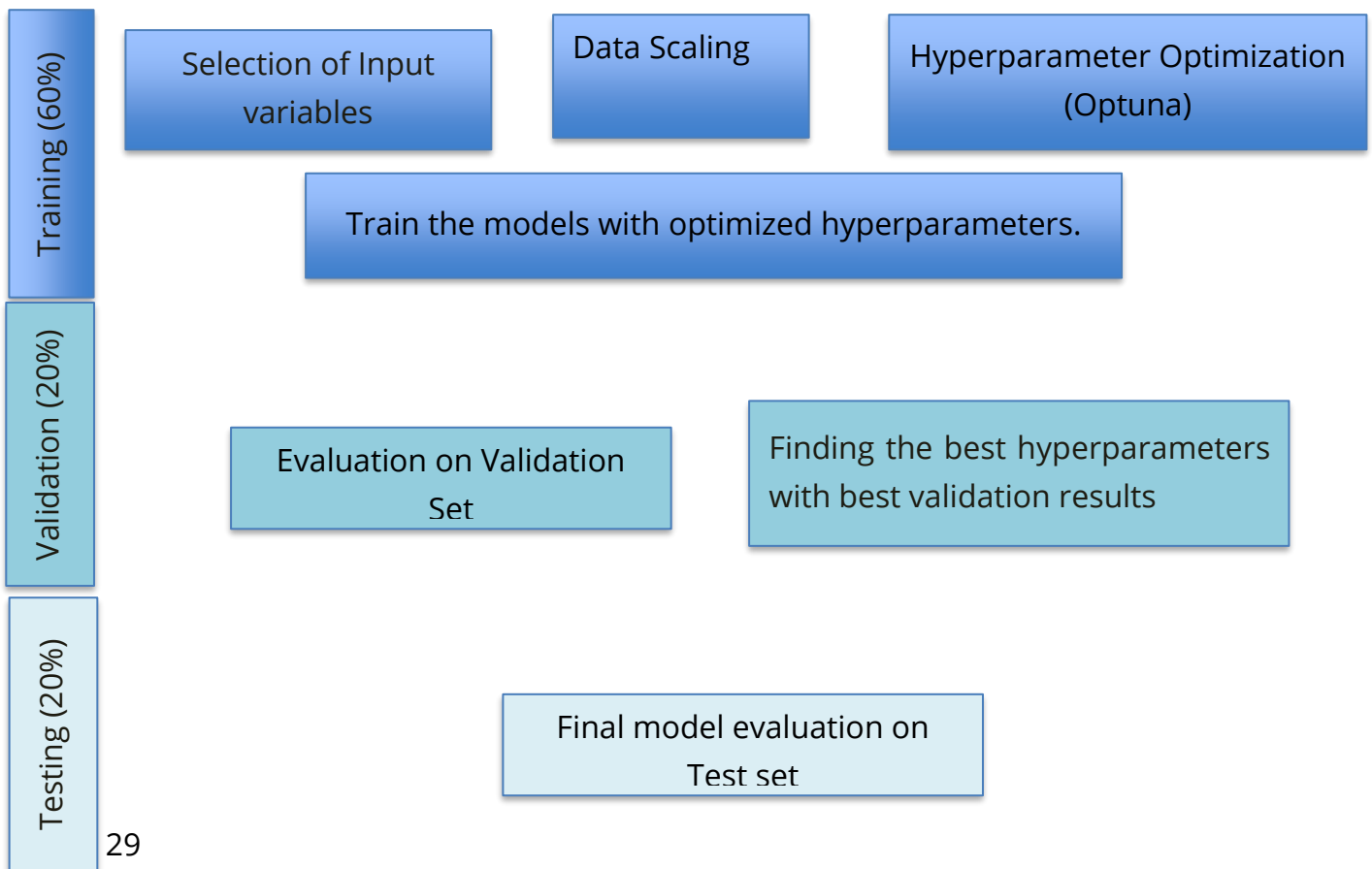
This chapter explains the procedure adopted for evaluating the different data-driven models for error correction. A brief description of the process and then details in each step involved follows.

The whole process of error correction using static multi-step ahead error forecasting for a lead time of 12hrs to 48hrs can be as follows in three main steps and further sub-phases/stages in each step:

1. The first step involves the data preprocessing, which involves three phases:
 - a. The first phase splits the total available data into three different sets as follows: Training (60%), Validation (20%), and Test (20%)
 - b. The second phase uses training data to select input variables based on cross-correlation and average mutual information between target and candidate variables.
 - c. The third phase normalizes the data using MinMaxscaler, where the trained data is used to fit the scaler and then transformed on train and validation data.
2. The second step deals with the Bayesian optimization, which further has three stages
 - a. The first stage optimizes the hyperparameters using optuna through several trials by fitting the model on 70% of training data and minimizing the RMSE on the last 30% of training data. This phase gives the best parameters as output
 - b. The second stage fits the model using the best parameters from the previous stage and using complete training data. Then this model is used to predict the errors on validation data.
 - c. The Third stage validates the predictions from the model using performance metrics (R^2 , RMSE, MAE) on the unseen validation set.
3. Finally, the third step involves testing on the final test set to get unbiased estimates of the models' performance, which begins after finalizing the models in the previous step with the best hyperparameters and best performance on validation data, which involves four phases:

- The first phase in this step involves normalizing the data by fitting the scaler to 80% of the total data available (i.e., Training + Validation) and transform it on this new training set and final test set (20%). Then fit the models with a new training set (80%) and predict the error on the test set (20%)
- Then the second phase involves correcting the model simulations by adding the denormalized forecasted error from different data-driven models.
- The third phase involves performance evaluation (R^2 , RMSE, MAE) of the corrected simulation concerning the observed discharge in the overall test set and the high flow events (i.e., > 80 Percentile). Further, also generate the plots comparing the corrected discharges with original modeled and observed discharges
- The final phase involves fitting the model on training data and predicting the error using training data to check its generalization ability by comparing it with test results obtained in the previous phase (3.c).

Figure 10 shows the dataset used in various phases explained above, and Figure 11 outlines the whole procedure in terms of final output at each step.



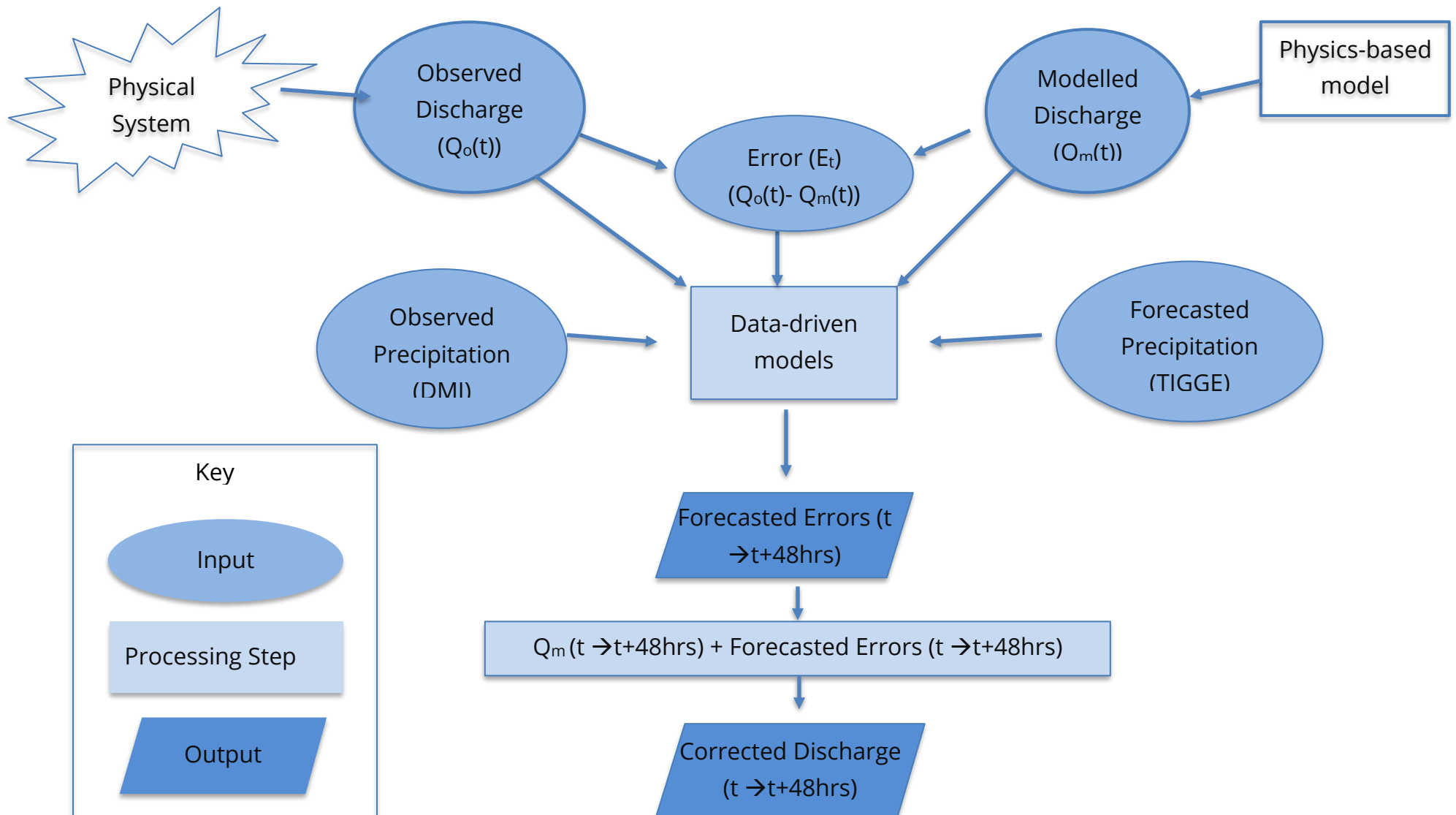


Figure 11: Schematic representation of workflow

Candidate input variables chosen for this case study includes Observed discharge, Modelled discharge, Forecast Precipitation, Observed Precipitation, and error at different lags. (See Figure 11)

5.1 Data Preprocessing

Then the first step begins with data preprocessing which includes data splitting, input variable selection & data scaling.

Abbreviations used in the analysis below

Odt- Observed Discharge at time t

Mdt- Modeled Discharge at time t

Pt- Observed Precipitation at time t

Pt+1- One step ahead forecasted precipitation

Et- Error between modeled and observed discharge at time t

Et= Mdt-Odt

5.1.1 Data Splitting

In our current study, we followed the procedure mentioned in section 3.1. To split our dataset into three parts while considering the requirement of similar statistical distribution. We divided the total data available into three splits (Training (60%), Validation (20%), and Test (20%)) in chronological order. Table 2 shows the timeline of the data in each set after splitting. Table 3 shows the statistical properties of the error time series in different splits used for the current study.

Table 2: Timeline of the data sets after splitting

Split	Timeline
Training Period (60%)	03/01/2011-08/06/2016
Validation Period (20%)	09/06/2016-20/03/2018
Test Period (20%)	21/03/2018-29/12/2019

Table 3: Statistical Properties of error data in different splits (Mean, Min, Max, Std dev, No of samples)

Et	Mean	Minimum	Maximum	Standard deviation	Number of samples
Training	0.58	-23.21	20.01	2.74	3892.00
Validation	0.82	-10.22	9.98	2.18	1296.00
Test	-0.91	-10.86	10.44	2.53	1296.00

Figure 12 & Figure 13 shows the time series of the candidate variables in the validation and test sets, respectively. First, hyperparameters are optimized using the training set. Then a validation set is used to select the best model with the best hyperparameters. After finalizing the model with the best hyperparameters with the best validation performance, the test set is used to evaluate the final model's unbiased evaluation.

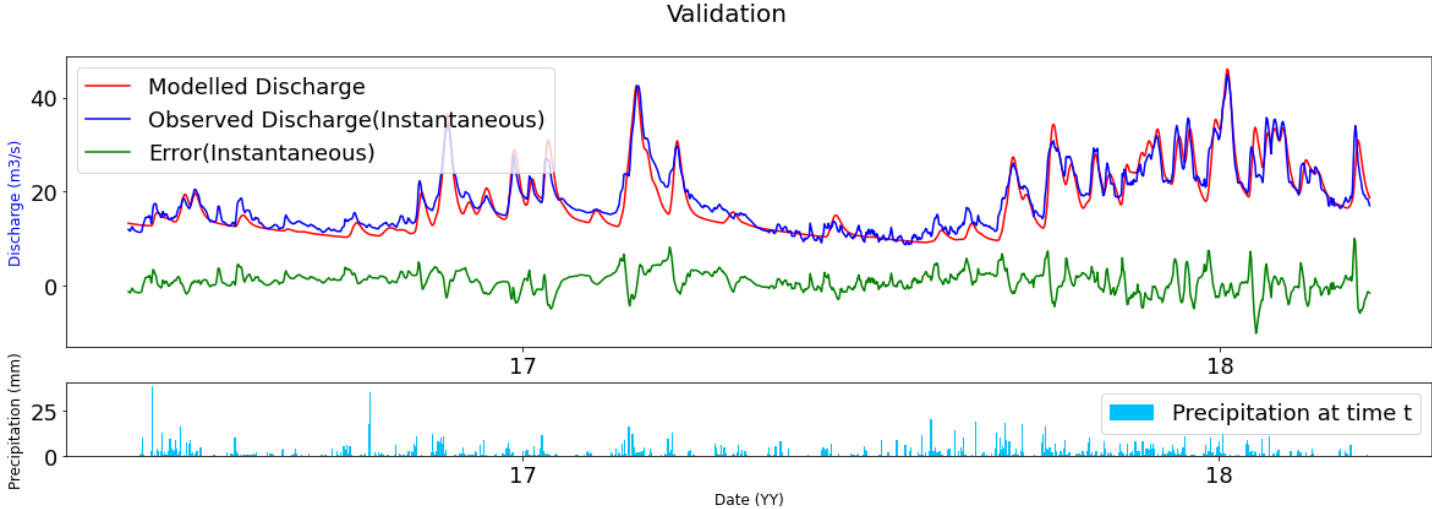


Figure 12: Candidate variables time-series at SKÆRUM BRO station (Validation)

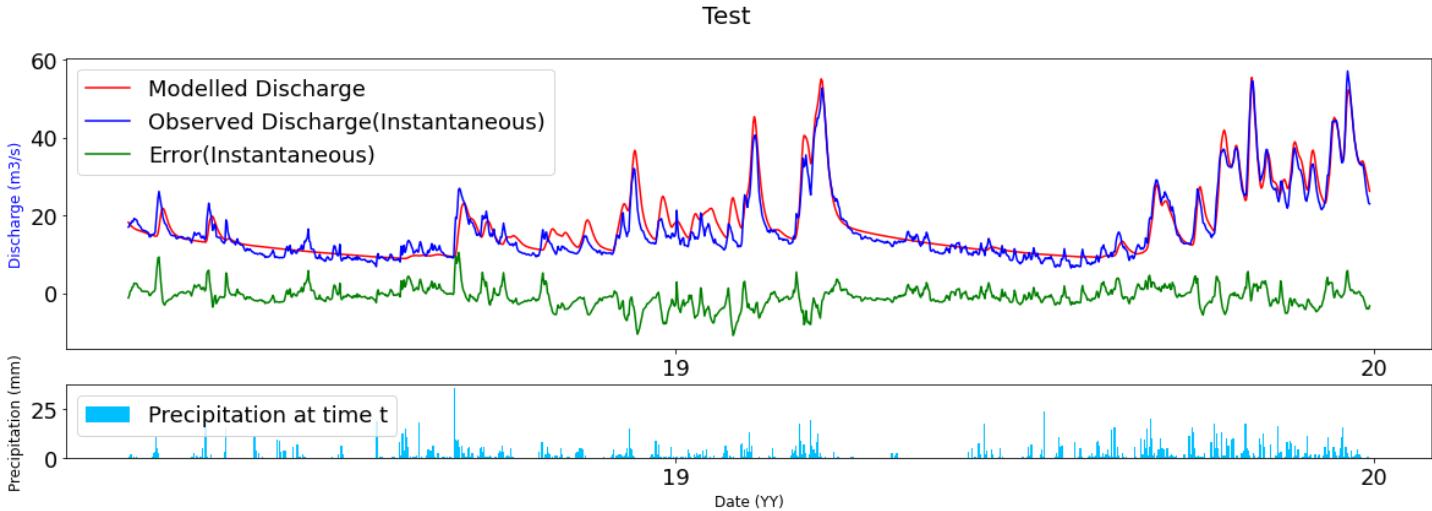


Figure 13: Candidate variables time-series at SKÆRUM BRO station (Test)

5.1.2 Input variables selection

Pearson Correlation and Average Mutual Information are used for selecting the input variables, as explained in section 3.2. Only training data is used in this phase. After few trials with different combinations of input variables and better validation results, the threshold of 0.2 is determined for both correlation and AMI.

(Moreido et al., 2021) concluded that in mixed catchments (i.e., Snowmelt and baseflow driven), It's preferable to allow automatic selection of parameters, but for rainfall-driven catchments, it's better to go with hydrologists' suggestion. In the current study, the performance of models is found to be either similar or slightly degraded after using all the variables available.

5.1.3 Data scaling

Data normalization is carried out using a minmax scalar, as explained in section 3.3.

5.2 Hyperparameter tuning.

Hyperparameters define the network structure and decide how the network trains. We tuned hyperparameters of ML models to minimize the RMSE on the last 30% of training data while fitting the model using 70% of training data. Pruning callback is used in the optimization, which monitors the validation loss in each trail and automatically stops the unpromising trials during the training. Best parameters are then saved to build the final model.

5.2.1 Neural network-based methods

This section describes the procedure adopted to develop two methods (MLP and BLSTM) and hyperparameter optimization.

1. Multilayer perceptron (MLP)

A simple MLP is developed using the sequential class from Keras ((Chollet, 2015)) & TensorFlow ((Abadi et al., 2016)) which allows us to build a linear stack of layers into the model. A maximum of 2 hidden layers is considered. Dropout is also used to regularize the model and reduce overfitting by improving the generalization power, considered a dropout of up to 20% for the search space.

Here a maximum of 200 epochs are used and Two callbacks (Early Stopping and Model Checkpoint) to avoid overfitting. Higher epochs improved the training accuracy but degraded the performance on a validation set. Early Stopping callback monitors the validation loss and interrupts the model training when there is no improvement depending on the threshold we set in patience. For MLP, it is determined as 20 by the trial-and-error method. When the validation loss does not improve continuously for 20 epochs, it halts training. Then Model checkpoint saves the model weights each time it sees an improvement in validation loss during the training.

Root mean square error (RMSE) is chosen as a metric while compiling the model. The optimization run returns RMSE after every trial in Bayesian optimization, and the process continues to minimize the RMSE.

Loss functions optimized are Mean square logarithmic error, Huber loss, mean absolute error, Mean square error, & logcosh. Optimizers considered are Adam, Stochastic Gradient Descent, RMSprop, as explained in section 3.6.3. Batch Size is optimized in the range of [32,256] with an interval of 32. Table 4 shows the complete hyperparameter search space used for Bayesian optimization of multilayer perceptron. Learning curves for MLP are plotted to check the model's generalization (See Appendix C) at each lead time.

Table 4: Range of hyperparameter values used for multilayer perceptron

Hyperparameters for neural network	Name	Values
1.Number of hidden layers required	n_layers	1,2
2. Number of neurons required in hidden layers	n_units_	[1,256]
3.Learning rate	Learning_rate	[0.001,0.01]
4.Activation function	activation_	[tanh, relu]
5.Optimizer	Optimizer	[Adam, SGD, RMSprop]
6.Dropout	dropout_	[0,0.2]
7.epochs	Epochs	[10,200]
8.Loss	Loss	[mse, mae, msle, huber, logcosh]
9.Batch Size	batch_size	[32, 256], interval=32

2. Bidirectional long-short term memory

A standard Bidirectional LSTM (BLSTM) neural network is trained and tested to predict the error time series. BLSTM also has similar hyperparameters as explained above for MLP, and In addition to them, sequence length is also optimized. Some researchers used up to 365 days for sequence length (Kratzert et al., 2018). Still, it will reduce the training data by one year and increase the computational burden while tuning the hyperparameters. So, because of time and computational limits, we limited the sequence length to 5 days. But when tested during validation, using longer sequence length did not improve the performance. Callbacks (Patience and Model Checkpoints) and metrics are defined in MLP, except that patience of 40 is used for BLSTM, obtained by trial and error. Table 5 shows the search space used for hyperparameter tuning. Learning curves plotted for BLSTM are shown in Appendix D.

Table 5: Range of hyperparameter values used for Bidirectional LSTM

Hyperparameters for BLSTM	Name	Values
Tuning the Number of Epochs	Epochs	[10,200]
Tuning the Batch Size	Batch_size	[32, 256], interval=32
Tuning the Number of Neurons	n_units_	[1,256]
Number of BLSTM layers	n_layers	1,2
Sequence Length	seq_length	[2,10]
Optimizers	Optimizer	[Adam, SGD, RMSProp]
Learning rate	learning_rate	[0.001,0.01]
Loss	Loss	[mse, mae, msle, huber, logcosh]
Dropout rate	dropout_l	[0.05,0.2]

5.2.2 Boosting methods

This section describes the procedure adopted for developing tree-boosting machine learning methods along with hyperparameter optimization. One thousand trials are performed for boosting techniques (Gradient and Newton) to find a global minimum in optuna.

1. Gradient Boosting

A gradient boosting regressor is developed and for more details on its description and hyperparameters, see section 3.6.4. Lower values of subsamples help in avoiding the overfitting. Table 6 shows the hyperparameters and their ranges considered for Bayesian optimization.

Table 6: Range of hyperparameter values used for Gradient boosting

Hyperparameters for Gradient Boosting	Name	Values
Minimum samples required in a leaf	min_samples_leaf	[1,10]
Learning rate	learning_rate	[0.01,0.1]
Fraction of samples to be chosen for each tree	Subsample	[0,1]
Maximum tree depth	max_depth	[2,20]
number of boosted trees	n_estimators	[100,300]

2. Newton boosting

The python implementation XGBoost uses the newton boosting approach. For most details on the approach and hyperparameters, see section 3.6.4. The search for hyperparameters in XGBoost is performed in two stages. In the first stage, all the parameters except `n_estimators` are optimized using Bayesian optimization. Early stopping criteria are used to identify `n_estimators` corresponding to optimized hyperparameters in the previous step, which stops when there is no improvement after ten rounds. Table 7 shows the search space for hyperparameter optimization in XGBoost.

Table 7: Range of hyperparameter values used for newton boosting (XGBoost)

Hyperparameters tuned for XGBoost	Name	Values
Subsample ratio of the training instances	<code>subsample</code>	[0.4,0.9]
Maximum tree depth	<code>max_depth</code>	[4,12]
L1 regularization parameter	<code>Alpha</code>	[0.01,10]
L2 regularization parameter	<code>lambda</code>	[1e-8,10]
Minimum loss reduction	<code>gamma</code>	[0,1]
The minimum data points per leaf	<code>min_child_weight</code>	[0,5]
Learning rate	<code>learning_rate</code>	[0.01,0.1]
Number of boosted trees	<code>n_estimators</code>	[0,600]

Further, a stacking regressor with XGBoost as a base estimator and an autoregressive model as the final estimator is built to exploit the benefits from both methods.

5.3 Final Testing

Table 8 shows the hyperparameters for 12 hours lead time. For hyperparameters at other lead times, see Table 17 &

Table 18 for tree-based & deep-learning-based methods, respectively. (in Appendix E). After finalizing the best models, unbiased estimates of the model's performance are obtained using the test set. For the hyperparameters variance at different trials in Bayesian optimization, Please see Appendix B.

Table 8: Best hyperparameters for a lead time of 12 hours

Updating Model	Optimal Hyperparameters
ANN(MLP)	{'learning_rate': 0.0015798028419481372, 'optimizer': 'Adam', 'loss': 'msle', 'epochs': 140, 'batch_size': 64, 'n_layers': 1, 'n_units_l0': 35, 'activation_l0': 'relu', 'dropout_l0': 0.0013014518954329451}
XGBoost	{'max_depth': 6, 'learning_rate': 0.09976512440998135, 'subsample': 0.6714706798105836, 'alpha': 0.013528920895993416, 'lambda': 3.5267815447831284e-06, 'n_estimators':60 'gamma': 0.0007976125851248869, 'min_child_weight': 0}
Gradient Boosting	{'max_depth': 13, 'learning_rate': 0.035092865547259953, 'subsample': 0.16502226493209313, 'min_samples_leaf': 2, 'n_estimators': 158}
BLSTM	{'seq_length': 3, 'learning_rate': 0.007141029339645854, 'optimizer': 'adam', 'loss': 'msle', 'epochs': 195, 'batch_size': 96, 'n_layers': 1, 'n_units_l0': 6, 'activation_l0': 'tanh', 'dropout_l0': 0.05791790765558189}

Chapter 6 Results

This chapter mainly focuses on the results obtained at various stages of model evaluation (validation & testing) and includes a discussion on the obtained results in this study.

The best variables in each method were filtered using Pearson Correlation (Figure 14) and Average mutual Information (Figure 15) for independent variables. By trial-and-error approach, It is found that the intersection of best variables with a threshold of 0.2 from both ways gave the best final input variables for the ML models. So, the final variables are Et, Et-1, Et-2, Et-3, Et-4, Odt, Odt-1, Odt-2, Odt-3, Odt-4. For complete plots on AMI and correlation, see Appendix A. Figure 15 shows significant autocorrelation in error time series. Figure 14 shows that the top contributors from AMI are also previous lags from the error time series.

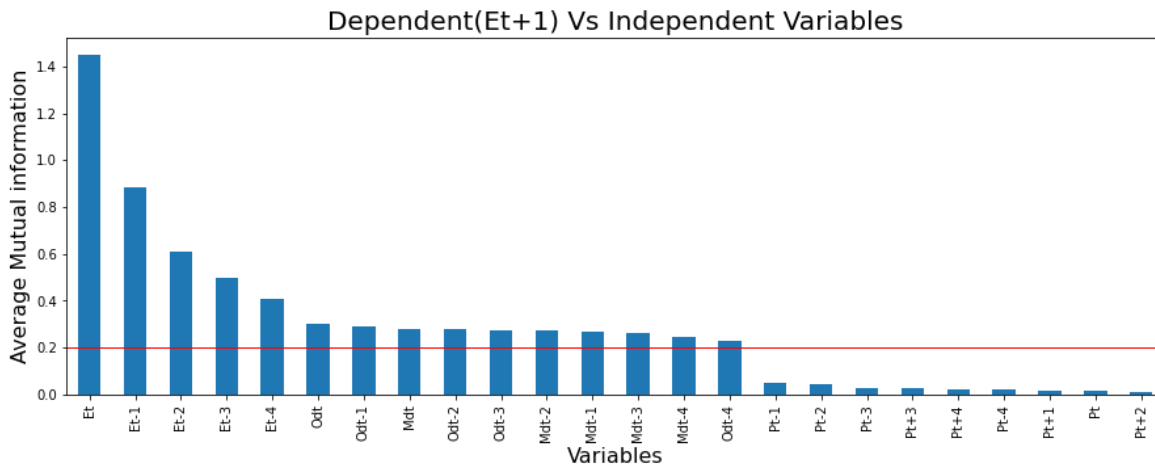


Figure 14: Average mutual information with input parameters Vs. Et+1

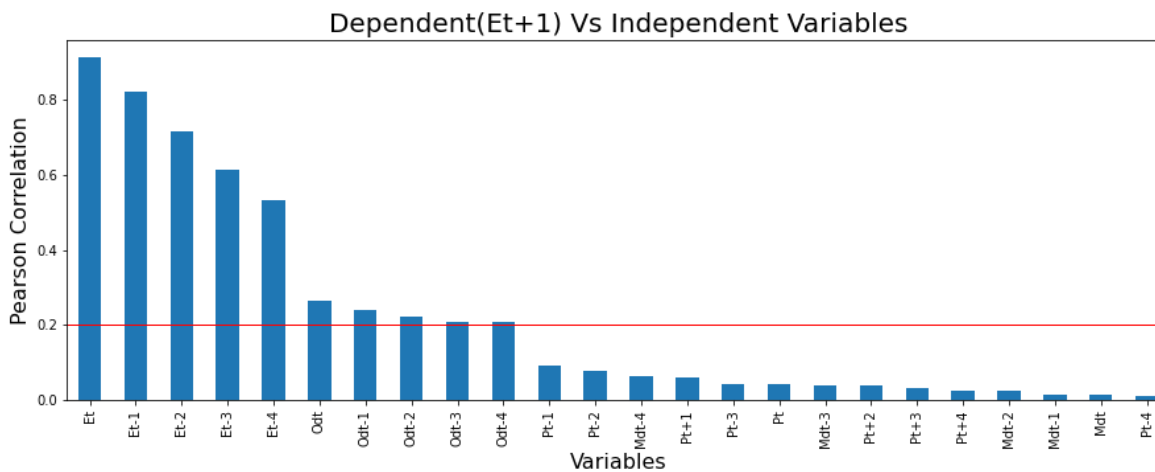


Figure 15: Pearson correlation in input parameters vs. Et+1

Even though we got better results with this approach, one possible drawback is that a feature has high non-linear relation to the target. It would get a low correlation and hence be excluded in this way, even though it might be a good predictor when used in non-linear methods.

6.1 Results on the validation set

The best hyperparameters for the final models are chosen based on the model's performance on the validation set. Figure 16 compares results obtained on the validation data at different lead times and high flow conditions.

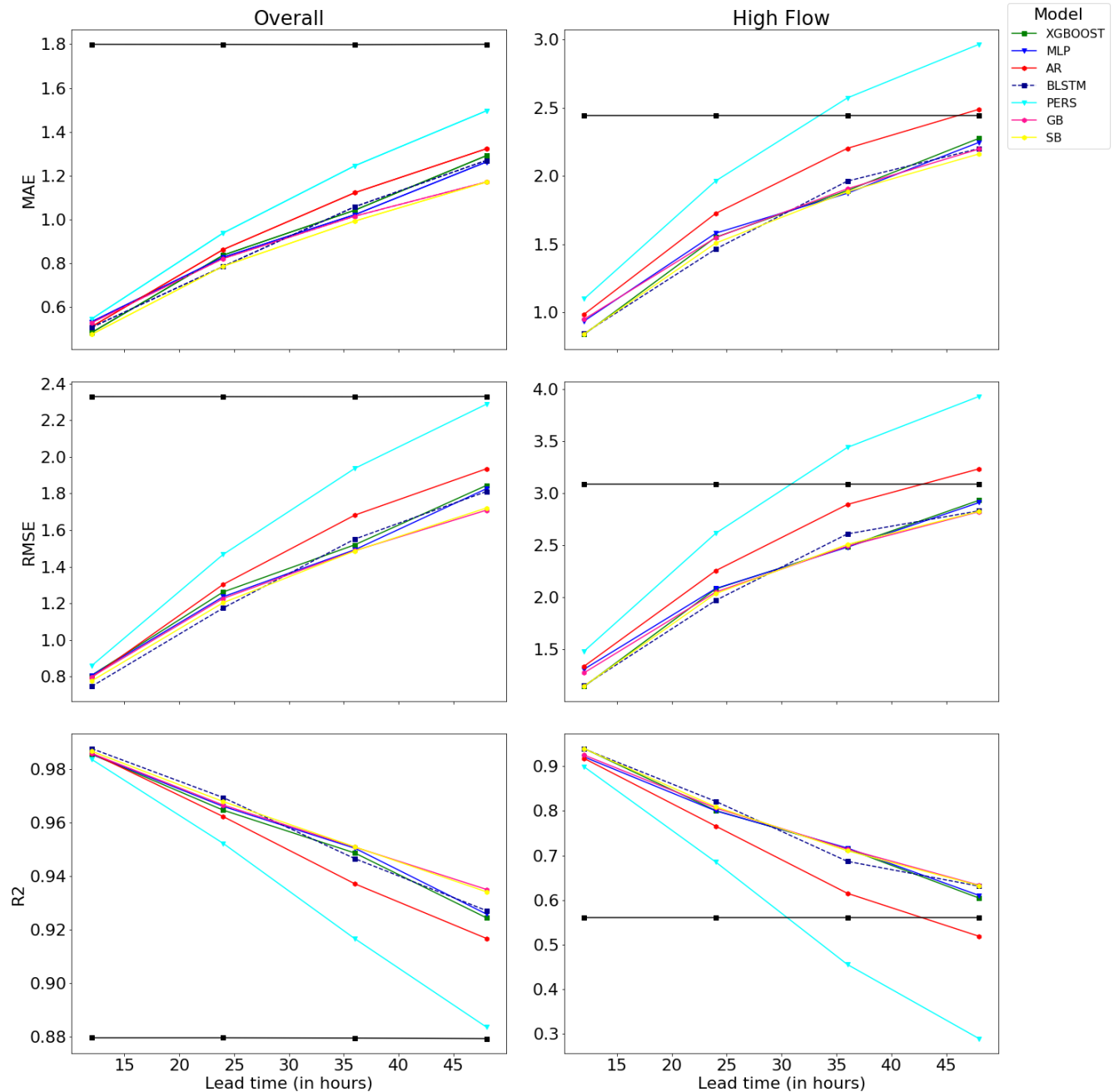


Figure 16: Performance statistics on validation data at different lead times

Figure 17 shows the comparison of time series plots for corrected discharges at 48 hours lead time. For comparison plots at other lead times (12,24,36 hours), please refer to Appendix F.

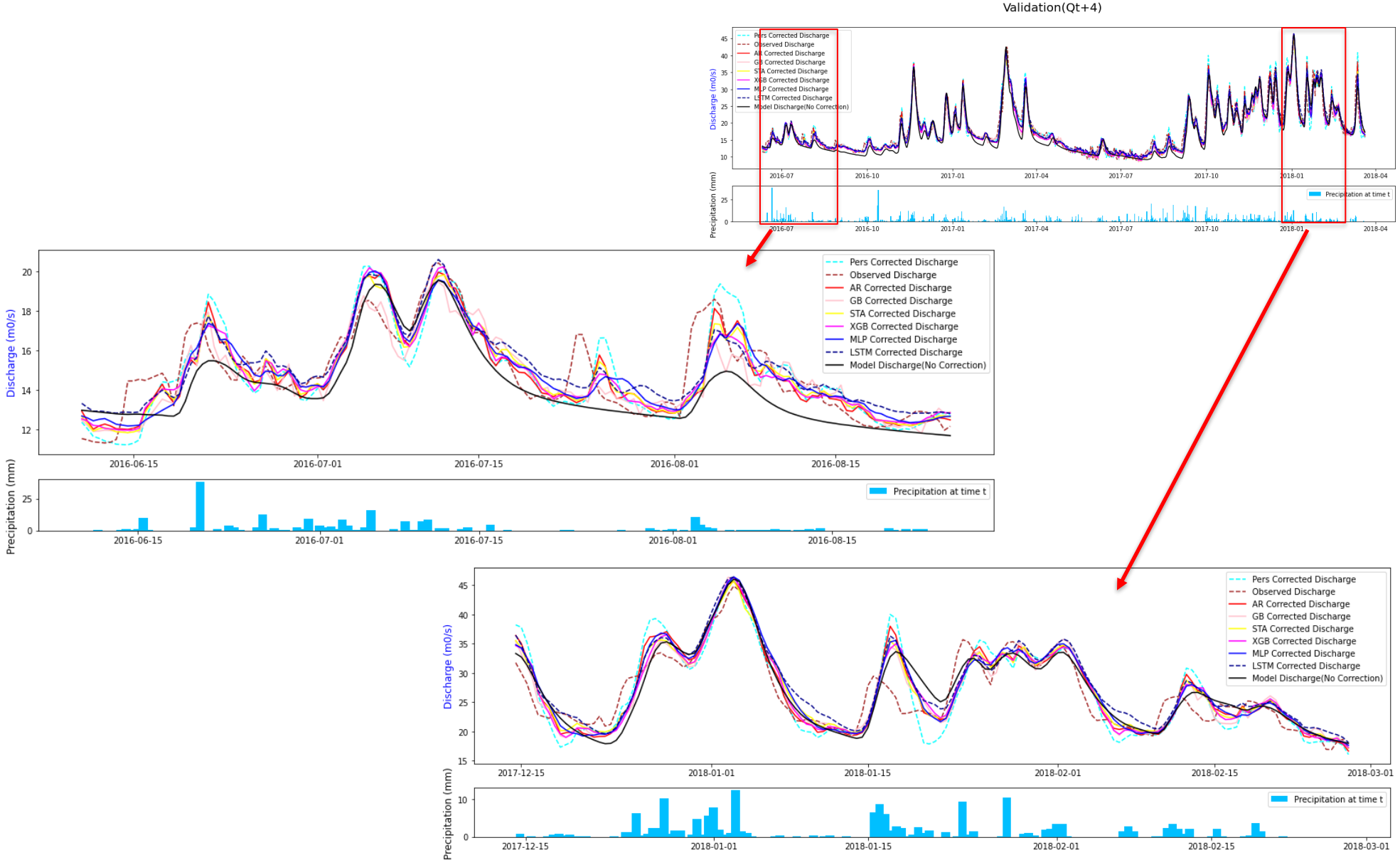


Figure 17: Comparison plots of corrected discharge time series for the validation data at 48 hours lead time

Results indicate that all deep learning and tree-based models performed better than the simple autoregression and persistence models on the overall validation set and high flow events.

In shorter lead times, performance difference among the models is very less and is only noticeable in terms of RMSE. And the RMSE of BLSTM is slightly better, indicating its ability to correct large errors. The comparable results in all the models at the shorter lead time, especially at 12 hours, could be because of high autocorrelation in error time series at 1st lag. At high flow conditions, BLSTM and stacked model gave similar results.

However, the model's performance difference is noticeable at longer lead times; the gradient boosting and stacked model performed similarly and are slightly better than other models. But the improvement is very little at high flow conditions at longer lead times, and all the tree-based and deep learning models performed similarly.

Even though all the models successfully improve the model simulations after correction on the overall validation set, simple models (i.e., Persistence and Autoregression) could not improve the simulations at longer lead times in high flow conditions.

6.2 Results on the final testing set

Results in this section are obtained on evaluating the independent test set (20%), which is not used anywhere in finalizing the model parameters or training. Table 9 & Table 10 show the RMSE values obtained from different models at various lead times on training and test data, respectively. Further, the three performance evaluation statistics (R^2 , MAE, & RMSE) on the training set and test set are also plotted in Figure 18 & Figure 19, respectively. Results indicate that gradient boosting and XGBoost are slightly overfitting in shorter lead times. But the results on the test set using the XGBoost and gradient boosting are comparable with other models with marginal underperformance at these times.

Table 9: RMSE obtained on training dataset at different lead times

Model	12 hours	24 hours	36 hours	48 hours
AR	1.03	1.47	1.82	2.07
PERS	1.08	1.59	2.03	2.37
MLP	0.92	1.21	1.57	1.80
Gradient Boosting	0.61	1.22	1.49	1.68
XGBoost	0.53	0.88	1.57	1.91
BLSTM	0.82	1.22	1.61	1.79
Stacking	0.69	1.14	1.63	1.88
No correction	2.69	2.69	2.69	2.69

Table 10: RMSE obtained on the test data set at different lead times.

Model	12 hours	24 hours	36 hours	48 hours
AR	1.00	1.54	1.91	2.17
PERS	1.03	1.63	2.05	2.39
MLP	1.01	1.48	1.81	2.02
Gradient Boosting	1.05	1.54	1.76	1.98
XGBoost	1.06	1.59	1.78	2.12
BLSTM	0.97	1.48	1.85	1.96
Stacking	0.99	1.48	1.77	2.01
No Correction	2.69	2.69	2.69	2.69

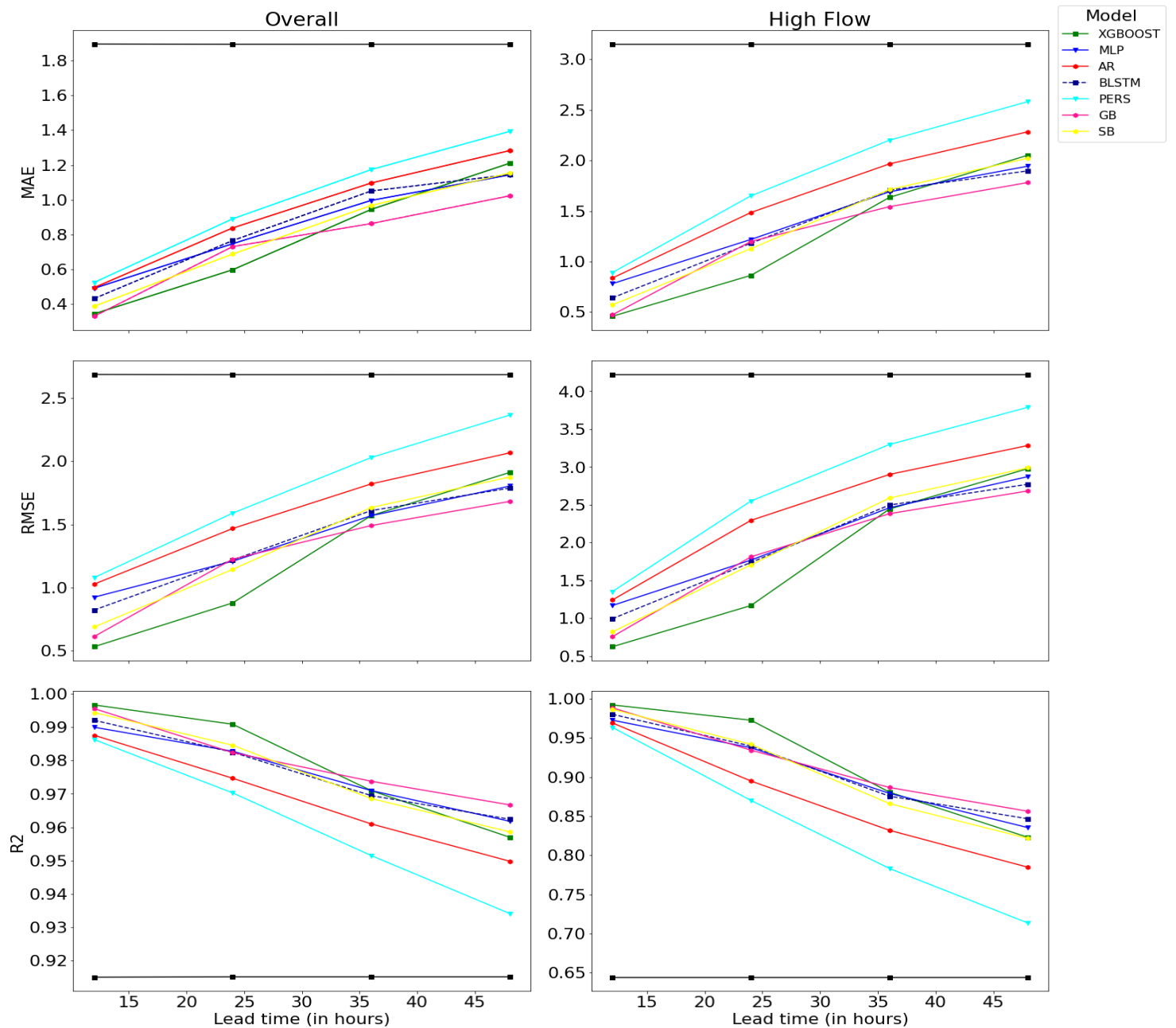


Figure 18: Performance statistics on final training data at different lead times.

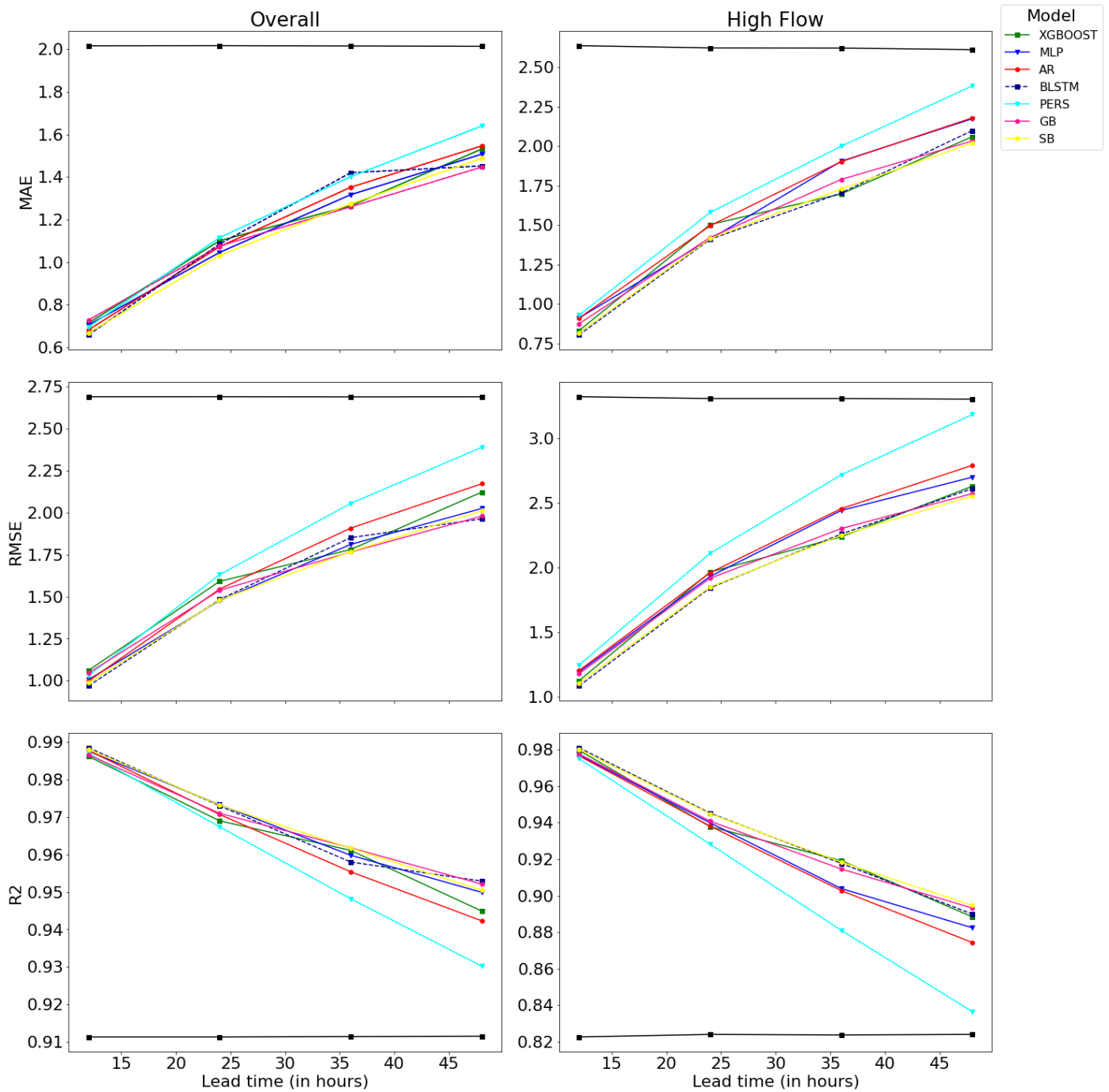


Figure 19: Performance statistics on the test data at different lead times.

Figure 18 and Figure 19 compare the performance statistics (R^2 , MAE, & RMSE) with the training and test set, respectively, at the different lead times and high flow conditions. Performance results on the independent test set at different lead times on test data (Figure 19) show that all the models successfully improve the simulated discharges from the physics-based model and are like results on the validation set (Figure 16). Since the difference among the models is less, it's hard to notice all the patterns from the figures alone. Hence percentage change in performance statistics after error correction relative to originally modeled discharge is computed further for better understanding.

The top two models at each lead time are highlighted in bold, and the negative values in MAE & RMSE indicate the % decrease in error and positive values in R². As already highlighted before, RMSE penalizes the large errors (outliers), and MAE is robust to these outliers.

Table 11, Table 12, & Table 13 shows the percentage change in RMSE, MAE & R² respectively for different models and lead times on the overall test set. The top two models at each lead time are highlighted in bold, and the negative values in MAE & RMSE indicate the % decrease in error and positive values in R². As already highlighted before, RMSE penalizes the large errors (outliers), and MAE is robust to these outliers.

Table 11: Change (%) in RMSE after correction on the test set.

Lead Time	AR	PERS	MLP	XGBoost	Gradient Boosting	BLSTM	Stacked model
12	-62.90%	-61.70%	-62.60%	-60.50%	-61.10%	-64.0%	-63.30%
24	-42.50%	-39.40%	-45.10%	-40.80%	-42.80%	-44.90%	-45.00%
36	-29.0%	-23.60%	-32.60%	-33.80%	-34.30%	-31.20%	-34.30%
48	-19.20%	-11.20%	-24.70%	-21.10%	-26.40%	-27.00%	-25.20%

Regarding RMSE (Table 11 & Figure 19) on the overall test set, the difference among the models' performance is less. The BLSTM & stacked model gave comparable results and performed better than AR & PERS in all the lead times. Further, they performed slightly better than tree-based models (XGBoost & gradient boosting) at shorter lead times (up to 24 hours). Gradient boosting gave comparable results with BLSTM & stacked model at longer lead times (>24 hours), but XGBoost continued to underperform. MLP gave better results than AR, PERS & XGBoost but marginally underperformed or gave almost comparable results with the BLSTM & Stacked model. It marginally gave better results than gradient boosting at shorter lead times but underperformed at longer lead times. Also, both XGBoost and gradient boosting are slightly underperforming the AR at shorter times. This underperformance in tree-based methods at shorter lead times is possibly due to the overfitting observed in training data (Table 9 & Figure 18).

Table 12: Change (%) in MAE after correction on the test set.

Lead Time	AR	PERS	MLP	XGBoost	Gradient Boosting	BLSTM	Stacked model
12	-66.20%	-65.50%	-65.00%	-64.50%	-63.80%	-67.30%	-66.90%
24	-46.80%	-44.70%	-48.10%	-45.40%	-46.60%	-46.20%	-48.80%
36	-32.90%	-30.40%	-34.60%	-37.20%	-37.50%	-29.50%	-36.80%
48	-23.20%	-18.60%	-25.10%	-23.90%	-28.10%	-27.90%	-26.10%

Now comparing MAE results (Table 12&Figure 19), the overall performance trend in models as observed in RMSE is still intact with slight underperformance of BLSTM at 36 hours lead time. However, BLSTM performed better than all other models at 48 hours lead time, as observed in RMSE.

Table 13: Change (%) in R² after correction on the test set.

Lead Time	AR	PERS	MLP	XGBoost	Gradient Boosting	BLSTM	Stacked model
12	8.39%	8.31%	8.37%	8.22%	8.26%	8.47%	8.43%
24	6.52%	6.16%	6.80%	6.33%	6.55%	6.77%	6.79%
36	4.83%	4.04%	5.31%	5.45%	5.53%	5.11%	5.52%
48	3.38%	2.05%	4.21%	3.66%	4.45%	4.54%	4.28%

Finally, In terms of R2 (Table 13 & Figure 19), the overall trend in performance is like that observed in RMSE. But the difference among the models is less.

So overall, test set results showed that the BLSTM and stacked model performed slightly better than the other models in all lead times. Xgboost and gradient boosting are marginally underperforming in shorter lead times than traditional methods (AR, PERS&MLP) due to the overfitting. And at longer lead times, gradient boosting, BLSTM, and stacked model performed significantly better (~8% in RMSE) than simple methods (AR/Pers) and slightly better (~2% in RMSE) than MLP but XGBoost slightly underperformed than MLP.

Table 14, Table 15, & Table 16 shows the percentage change in RMSE, MAE & R², respectively, for different models and lead times in high flow events.

Table 14: Change (%) in RMSE after correction in high flow events within the test set.

Lead Time	AR	PERS	MLP	XGBoost	Gradient Boosting	BLSTM	Stacked model
12	-63.7%	-62.4%	-64.0%	-66.1%	-64.4%	-67.3%	-66.6%
24	-40.7%	-36.2%	-41.5%	-40.5%	-42.0%	-44.2%	-44.0%
36	-25.7%	-17.8%	-26.1%	-32.3%	-30.4%	-31.7%	-31.9%
48	-15.5%	-3.6%	-18.3%	-20.4%	-22.1%	-20.9%	-22.6%

From RMSE results (Table 14 & Figure 19) at the high flow events, the outperformance of the BLSTM & stacked model is more clearly visible with comparable results among them. As both these models consistently performed better than other models. Interestingly, although gradient boosting and XGBoost marginally underperforming the BLSTM & stacked model, they performed better than all the traditional methods (AR, PERS & MLP).

Table 15: Change (%) in MAE after correction in high flow events within the test set.

Lead Time	AR	PERS	MLP	XGBoost	Gradient Boosting	BLSTM	Stacked model
12	-65.50%	-64.80%	-65.40%	-68.50%	-66.80%	-69.50%	-69.00%
24	-43.00%	-39.70%	-46.20%	-42.70%	-45.80%	-46.20%	-46.10%
36	-27.50%	-23.70%	-27.40%	-35.30%	-31.80%	-35.00%	-34.10%
48	-16.60%	-8.80%	-16.80%	-21.10%	-22.20%	-19.60%	-22.60%

Comparing the results from MAE (Table 15 & Figure 19) at the high flow events, While the similar trend as observed with RMSE is also applicable here, MAE of the BLSTM is slightly outperforming the other models opposite to its performance on the overall test set.

Table 16: % Change in R² after correction in high flow events within the test set.

Lead Time	AR	PERS	MLP	XGBoost	Gradient Boosting	BLSTM	Stacked model
12	18.72%	18.53%	18.77%	19.09%	18.84%	19.26%	19.16%
24	13.85%	12.65%	14.07%	13.81%	14.17%	14.71%	14.66%
36	9.60%	6.96%	9.74%	11.59%	11.04%	11.41%	11.50%
48	6.10%	1.52%	7.09%	7.82%	8.40%	8.00%	8.58%

Comparing the R² (Table 16 & Figure 19) at the high flow events, similar trends observed above with RMSE are observed here. But the difference among the models is more noticeable, unlike the less difference observed on the overall dataset.

Finally, the comparison of time series plots for corrected discharges of the test set at 48 hours lead time is plotted in Figure 20 to verify the results obtained from performance evaluation indices. Further, please see Appendix F for the comparison plots for validation and test sets at the other lead times.

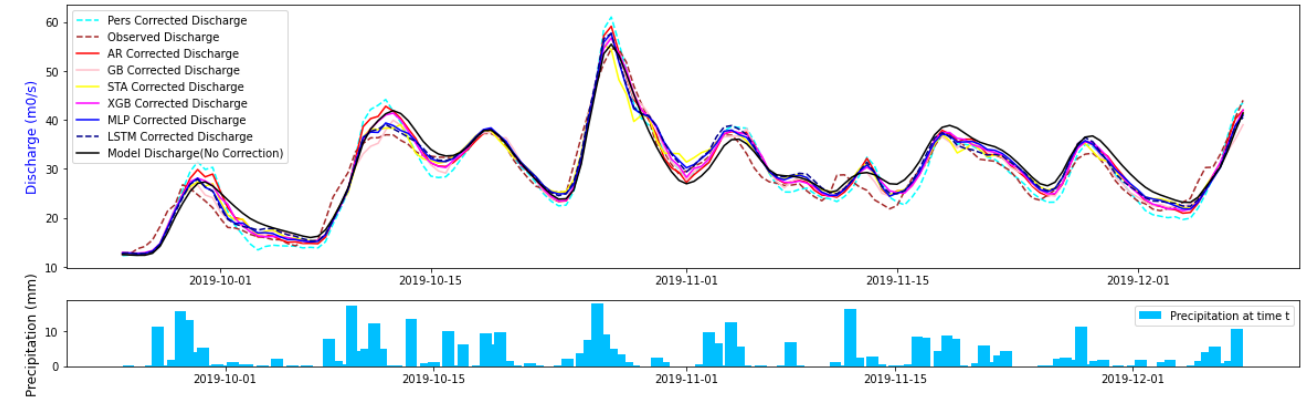
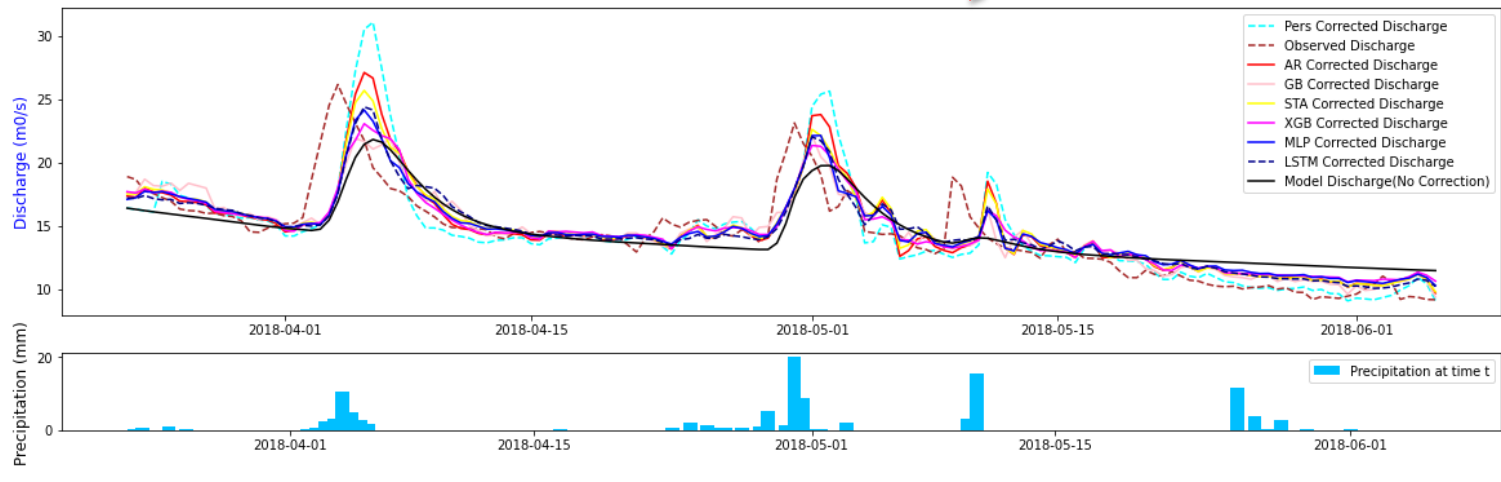
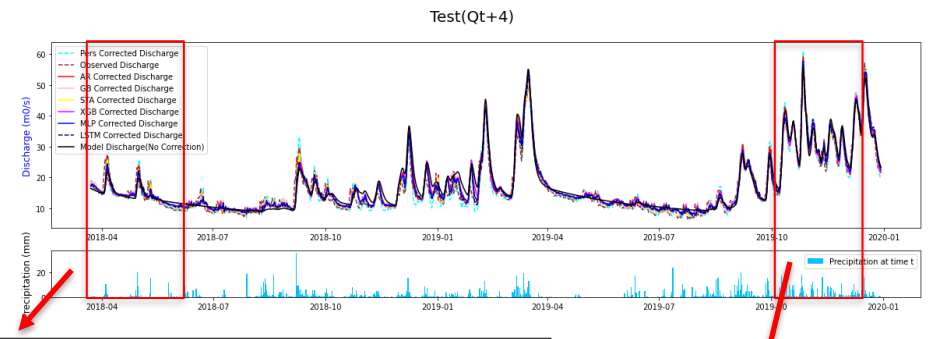


Figure 20: Comparison plots of corrected discharge time series at 48 hours lead time on the test set

6.3 Discussion on Results

The minimal difference among the model's performance at shorter lead times is possibly due to the significant autocorrelation of error time series at the initial lags, resulting in better performance even with the simple methods like AR & PERS. Eventually, this autocorrelation diminished with increasing lead time, and non-linear methods performed better at longer lead times with considerable differences. Further, The clear outperformance of the novel (ML&DL) methods at high flow events with all lead times compared to traditional methods could be partly because of the objective function (RMSE) used in the Bayesian optimization. Since the optimization is done by minimizing the RMSE, and the RMSE penalizes the large errors, it has a more prominent role on high flow events leading to better performance in all evaluation metrics.

(Gauch et al., 2021) studied tree- and lstm-based models for streamflow forecasting and found that LSTM and XGBoost gave similar accurate results when trained on smaller datasets; however, LSTM outperformed with larger training data. (Alizadeh et al., 2021) proposed attention-based LSTM cell post-processor and compared with Gradient boosting, LSTM, and GRU with deterministic forecasts. They found comparable results in 3 deep learning techniques, which also performed better than gradient boosting. The two studies mentioned above are in line with the findings in the current study, indicating slightly better performance in deep learning-based techniques. Also, (Lee & Ahn, 2021) proposed a stacking ensemble model for post-processing streamflow forecasts using quantile regression and showed that it could be used for short lead times. (Sikorska-Senoner & Quilty, 2021) proposed a conceptual data-driven approach for error correction ensemble hydrological model simulations, making it flexible for any DDM. They also tested eight data-driven models and found out that XGBoost and Random forest gave the best performance. But they did not consider LSTM in the above study. Our study got slightly better results in XGBoost & gradient boosting than random forest during validation. So, we dropped random forest in further analysis. So overall, all the three approaches stacking, tree-based, and deep learning methods, have shown the potential to improve the discharge forecasts, which is also observed in the current study.

Chapter 7 Conclusions and future scope

This chapter summarizes the conclusions, followed by the limitation of this study and the further research recommendations.

7.1 Conclusions

This study evaluated six data-driven models and a naïve model to improve the forecasts from the physics-based model. For finding the best models, hyperparameter tuning was performed using Bayesian optimization to minimize the RMSE. Results on the validation set indicated comparable results with a stacked model, tree-based, and deep-learning-based methods. Hence all the methods were chosen for testing on the final independent test set.

The results on the final test set show that all the six data-driven models tested in this study successfully improved the simulations. However, the Bidirectional LSTM and Stacked model continuously performed slightly better than other models. Tree-boosting methods slightly underperformed at the shorter lead times because of the overfitting. While gradient boosting improved at higher lead times and gave comparable results with BLSTM & stacked model, XGBoost continues to underperform slightly but gave better results than AR and PERS & MLP. Also, MLP gave comparable results with BLSTM & stacked model at shorter lead times but showed slight underperformance at higher lead times.

At high flow conditions, BLSTM and stacked model gave robust results. Even though the difference is marginal, they still performed consistently better than all the other models. Then the tree-based methods (XGBoost & gradient boosting) slightly underperformed the BLSTM & stacked model but performed better than simple methods (AR/Pers) and MLP.

A further highlight in this study is that even though the stacked model is developed from less computationally expensive methods (XGBoost & AR), it still gave comparable results with BLSTM.

7.2 Limitations & future scope

Finally, this section aims to summarize the limitations of the current study and its associated future recommendations. Even though the targeted objectives were achieved, there is still some room for further research to obtain more generalized results.

The current study is performed only at the outlet point as we have complete data available for all variables with a longer period at this station. Still, this study can also be extended to other stations based on data availability. Also, Gauch et al. (2021) mentioned that the model accuracy increased when they used data from additional basins to train the model. So, the possibility of training models across the basins to get improved results can be further explored.

(Todini, 2008) mentioned that the performance of proposed models at rising and falling limbs is important as the correlations at these key locations may be low. But these flow regimes are not considered in the current study, which could be further evaluated.

Extreme flood events may come up with unknown deficiencies that were not identified by the hydrological model before. Then error models may not accurately represent the observed and modeled values (Liu et al., 2012). The possible solution for this challenge is to get sufficiently enough data for the training. Then more extreme events will be seen by the model during the training. But for ungauged basins or basins with less data availability, the possibilities of using the data from hydrologically similar catchments for training can be explored.

Modular models incorporating hydrological knowledge can be explored to account for human-induced error or external influences like dam releases upstream.

Some studies computed integrated gradients to interpret input variables to understand their contribution to LSTM, and permutation-based feature selection approaches can also be explored. Tree-based models have an additional feature to estimate the importance of input variables which can also be used for input variable selection. But to maintain uniformity for comparing different DDMs, it is not used for final input variable selection.

Even though the error correction improved the simulation, the remaining (residual) modeling uncertainty can be quantified using methods like Quantile regression in conjunction with the machine learning methods. For example, tree-based ensemble methods can generate quantile regression-based prediction intervals by choosing quantile loss function and defining one additional hyperparameter(α) in the model. It is also possible with neural networks-based methods by defining a quantile loss function. But this is not analyzed explicitly in this study due to time constraints. Extension of XGBoost for probabilistic forecasting is currently available as XGBoostLSS; this can be evaluated for its capability to generate predictive intervals. In addition to this, Catboost and NGBost also give prediction intervals which can also be explored. Also, the uncertainty associated with the initializing of parameters can be assessed.

The current study uses a stacking regressor only for XGBoost and AR, but this can also be tested using multiple models as base estimators. As BLSTM already gave better results, further improvement can be expected if included in the stacked models.

References

- Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., ... Zheng, X. (2016). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*. <http://arxiv.org/abs/1603.04467>
- Abrahart, R. J., & See, L. (2000). Comparing neural network and autoregressive moving average techniques for the provision of continuous river flow forecasts in two contrasting catchments. *Hydrological Processes*, *14*(11–12), 2157–2172. [https://doi.org/10.1002/1099-1085\(20000815/30\)14:11/12<2157::AID-HYP57>3.0.CO;2-S](https://doi.org/10.1002/1099-1085(20000815/30)14:11/12<2157::AID-HYP57>3.0.CO;2-S)
- Acharya, S. C., Babel, M. S., Madsen, H., Sisomphon, P., & Shrestha, S. (2020). Comparison of different quantile regression methods to estimate predictive hydrological uncertainty in the Upper Chao Phraya River Basin, Thailand. *Journal of Flood Risk Management*, *13*(1). <https://doi.org/10.1111/jfr3.12585>
- Akbari, M., & Afshar, A. (2014). Similarity-based error prediction approach for real-time inflow forecasting. *Hydrology Research*, *45*(4–5), 589–602. <https://doi.org/10.2166/nh.2013.098>
- Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A Next-generation Hyperparameter Optimization Framework. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2623–2631. <https://doi.org/10.1145/3292500.3330701>
- Alizadeh, B., Ghaderi Bafti, A., Kamangir, H., Zhang, Y., Wright, D. B., & Franz, K. J. (2021). A Novel Attention-Based LSTM Cell Post-Processor Coupled with Bayesian Optimization for Streamflow Prediction. *Journal of Hydrology*, *601*(February), 126526. <https://doi.org/10.1016/j.jhydrol.2021.126526>
- Babel, M., Tansar, H., Mark, O., Weesakul, S., & Madsen, H. (2020). Data assimilation for flow forecasting in urban drainage systems by updating a hydrodynamic model of Damhusåen Catchment, Copenhagen. *Urban Water Journal*, *17*(10), 847–859. <https://doi.org/10.1080/1573062X.2020.1828938>
- Babovic, V., Cañizares, R., Jensen, H. R., & Klinting, A. (2001). Neural Networks as Routine for Error Updating of Numerical Models. *Journal of Hydraulic Engineering*, *127*(3), 181–193. [https://doi.org/10.1061/\(asce\)0733-9429\(2001\)127:3\(181\)](https://doi.org/10.1061/(asce)0733-9429(2001)127:3(181))
- Banik, A., Behera, C., Sarathkumar, T. V., & Goswami, A. K. (2020). Uncertain wind power forecasting using LSTM-based prediction interval. *IET Renewable Power Generation*, *14*(14), 2657–2667. <https://doi.org/10.1049/iet-rpg.2019.1238>
- Bogner, K., & Kalas, M. (2008). Error-correction methods and evaluation of an ensemble based hydrological forecasting system for the Upper Danube catchment. *Atmospheric Science Letters*, *9*(2), 95–102.

<https://doi.org/10.1002/asl.180>

- Bogner, K., & Pappenberger, F. (2011). Multiscale error analysis, correction, and predictive uncertainty estimation in a flood forecasting system. *Water Resources Research*, 47(7), 7524. <https://doi.org/10.1029/2010WR009137>
- Broersen, P. M. T., & Weerts, A. H. (2005). Automatic error correction of Rainfall-Runoff models in flood forecasting systems. *Conference Record - IEEE Instrumentation and Measurement Technology Conference*, 2(May), 963–968. <https://doi.org/10.1109/imtc.2005.1604281>
- C2CCC. (2018). C2C. C2ccc Website. <https://www.c2ccc.eu/english/subprojects/c13-storaen/>
- Chen, L., Zhang, Y., Zhou, J., Singh, V. P., Guo, S., & Zhang, J. (2015). Real-time error correction method combined with combination flood forecasting technique for improving the accuracy of flood forecasting. *Journal of Hydrology*, 521, 157–169. <https://doi.org/10.1016/j.jhydrol.2014.11.053>
- Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 13-17-Aug*, 785–794. <https://doi.org/10.1145/2939672.2939785>
- Chollet, F. (2015). *Keras*. GitHub. <https://github.com/fchollet/keras>
- Corzo, G. (2009). Hybrid models for hydrological forecasting. In *PhD Thesis*. http://repository.tudelft.nl/assets/uuid:78bf7973-25bb-491a-a931-421973081a23/PHD_THESIS_CORZO-PEREZ.pdf
- Corzo, G., & Solomatine, D. (2007a). Baseflow separation techniques for modular artificial neural network modelling in flow forecasting. *Hydrological Sciences Journal*, 52(3), 491–507. <https://doi.org/10.1623/hysj.52.3.491>
- Corzo, G., & Solomatine, D. (2007b). Knowledge-based modularization and global optimization of artificial neural network models in hydrological forecasting. *Neural Networks*, 20(4), 528–536. <https://doi.org/10.1016/j.neunet.2007.04.019>
- DHI. (2021). *DHI helps the Danish EPA use AI and IoT to predict flooding on Denmark's second largest river*. DHI Website. <https://www.dhigroup.com/global/news/2021/03/dhi-helps-the-danish-epa-use-ai-and-iot-to-predict-flooding-on-denmark's-second-largest-river>
- Didrick, N. (2016). Tree Boosting With XGBoost thesis Master. In *Undefined* (Issue December).
- Dogulu, N., López López, P., Solomatine, D. P., Weerts, A. H., & Shrestha, D. L. (2015). Estimation of predictive hydrologic uncertainty using the quantile regression and UNEEC methods and their comparison on contrasting catchments. *Hydrology and Earth System Sciences*, 19(7), 3181–3201. <https://doi.org/10.5194/hess-19-3181-2015>
- EEA. (2020). *Denmark climate*. <https://www.eea.europa.eu/soer/2010/countries/dk/country-introduction-denmark>

- Ehlers, L. B., Wani, O., Koch, J., Sonnenborg, T. O., & Refsgaard, J. C. (2019). Using a simple post-processor to predict residual uncertainty for multiple hydrological model outputs. *Advances in Water Resources*, *129*(April 2018), 16–30. <https://doi.org/10.1016/j.adwatres.2019.05.003>
- Ellenson, A., Pei, Y., Wilson, G., Özkan-Haller, H. T., & Fern, X. (2020). An application of a machine learning algorithm to determine and describe error patterns within wave model output. *Coastal Engineering*, *157*. <https://doi.org/10.1016/j.coastaleng.2019.103595>
- Farchi, A., Laloyaux, P., Bonavita, M., & Bocquet, M. (2020). *Using machine learning to correct model error in data assimilation and forecast applications*. October. <http://arxiv.org/abs/2010.12605>
- Fraser, A. M., & Swinney, H. L. (1986). Independent coordinates for strange attractors from mutual information. In *Physical Review A* (Vol. 33, Issue 2, pp. 1134–1140). <https://doi.org/10.1103/PhysRevA.33.1134>
- Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, *29*(5), 1189–1232. <https://doi.org/10.1214/aos/1013203451>
- Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics and Data Analysis*, *38*(4), 367–378. [https://doi.org/10.1016/S0167-9473\(01\)00065-2](https://doi.org/10.1016/S0167-9473(01)00065-2)
- Fu, J. C., Huang, H. Y., Jang, J. H., & Huang, P. H. (2019). River Stage Forecasting Using Multiple Additive Regression Trees. *Water Resources Management*, *33*(13), 4491–4507. <https://doi.org/10.1007/s11269-019-02357-x>
- Gauch, M., Mai, J., & Lin, J. (2021). The proper care and feeding of CAMELS: How limited training data affects streamflow prediction. *Environmental Modelling and Software*, *135*(November 2020), 104926. <https://doi.org/10.1016/j.envsoft.2020.104926>
- Goswami, M., O'Connor, K. M., Bhattarai, K. P., & Shamseldin, A. Y. (2005). Assessing the performance of eight real-time updating models and procedures for the Brosna River. *Hydrology and Earth System Sciences*, *9*(4), 394–411. <https://doi.org/10.5194/hess-9-394-2005>
- Hochreiter, S. (1997). Long Short-Term Memory. *Neural Computation*, *17*(9), 1735–1780.
- Hsu, M. H., Fu, J. C., & Liu, W. C. (2003). Flood routing with real-time stage correction method for flash flood forecasting in the Tanshui River, Taiwan. *Journal of Hydrology*, *283*(1–4), 267–280. [https://doi.org/10.1016/S0022-1694\(03\)00274-9](https://doi.org/10.1016/S0022-1694(03)00274-9)
- Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science and Engineering*, *9*(3), 90–95. <https://doi.org/10.1109/MCSE.2007.55>
- Ibrahim Ahmed Osman, A., Najah Ahmed, A., Chow, M. F., Feng Huang, Y., & El-Shafie, A. (2021). Extreme gradient boosting (Xgboost) model to predict the groundwater levels in Selangor Malaysia. *Ain Shams Engineering Journal*, *12*(2), 1545–1556. <https://doi.org/10.1016/j.asej.2020.11.011>
- Jia, X., Willard, J., Karpatne, A., Read, J., Zwart, J., Steinbach, M., & Kumar, V. (2019).

- Physics guided RNNs for modeling dynamical systems: A case study in simulating lake temperature profiles. *SIAM International Conference on Data Mining, SDM 2019*, 558–566. <https://doi.org/10.1137/1.9781611975673.63>
- Kayastha, N., Ye, J., Fenicia, F., Kuzmin, V., & Solomatine, D. P. (2013). Fuzzy committees of specialized rainfall-runoff models: Further enhancements and tests. *Hydrology and Earth System Sciences*, *17*(11), 4441–4451. <https://doi.org/10.5194/hess-17-4441-2013>
- Khu, S. T., Liong, S. Y., Babovic, V., Madsen, H., & Muttill, N. (2001). Genetic programming and its application in real-time runoff forecasting. *Journal of the American Water Resources Association*, *37*(2), 439–451. <https://doi.org/10.1111/j.1752-1688.2001.tb00980.x>
- Klotz, D., Kratzert, F., Gauch, M., Sampson, A. K., Klambauer, G., Hochreiter, S., & Nearing, G. (2020). Uncertainty Estimation with Deep Learning for Rainfall-Runoff Modelling. *Hydrology and Earth System Sciences Discussions*, *April*, 1–32. <http://arxiv.org/abs/2012.14295>
- Kratzert, F., Klotz, D., Brenner, C., Schulz, K., & Herrnegger, M. (2018). Rainfall-runoff modelling using Long Short-Term Memory (LSTM) networks. *Hydrology and Earth System Sciences*, *22*(11), 6005–6022. <https://doi.org/10.5194/hess-22-6005-2018>
- Krzysztofowicz, R., & Kelly, K. S. (2000). Hydrologic uncertainty processor for probabilistic river stage forecasting. *Water Resources Research*, *36*(11), 3265–3277. <https://doi.org/10.1029/2000WR900108>
- Lee, D., & Ahn, K. (2021). A stacking ensemble model for hydrological post-processing to improve streamflow forecasts at medium-range timescales over South Korea. *Journal of Hydrology*, *600*(February), 126681. <https://doi.org/10.1016/j.jhydrol.2021.126681>
- Li, Q., Li, C., Yu, H., Qian, J., Hu, L., & Ge, H. (2020). System response curve correction method of runoff error for real-time flood forecast. *Hydrology Research*, *51*(6), 1312–1331. <https://doi.org/10.2166/nh.2020.048>
- Liang, Z., Huang, Y., Singh, V. P., Hu, Y., Li, B., & Wang, J. (2021). Multi-source error correction for flood forecasting based on dynamic system response curve method. *Journal of Hydrology*, *594*(January), 125908. <https://doi.org/10.1016/j.jhydrol.2020.125908>
- Liu, Y., Weerts, A. H., Clark, M., Hendricks Franssen, H. J., Kumar, S., Moradkhani, H., Seo, D. J., Schwanenberg, D., Smith, P., Van Dijk, A. I. J. M., Van Velzen, N., He, M., Lee, H., Noh, S. J., Rakovec, O., & Restrepo, P. (2012). Advancing data assimilation in operational hydrologic forecasting: Progresses, challenges, and emerging opportunities. *Hydrology and Earth System Sciences*, *16*(10), 3863–3887. <https://doi.org/10.5194/hess-16-3863-2012>
- López López, P., Verkade, J. S., Weerts, A. H., & Solomatine, D. P. (2014). Alternative configurations of quantile regression for estimating predictive uncertainty in water level forecasts for the upper Severn River: A comparison. *Hydrology and*

- Earth System Sciences*, 18(9), 3411–3428. <https://doi.org/10.5194/hess-18-3411-2014>
- Lundberg, A. (1982). Combination of a conceptual model and an autoregressive error model for improving short time forecasting (Eman catchment). *Nordic Hydrology*, 13(4), 233–246. <https://doi.org/10.2166/nh.1982.0019>
- Madsen, H., & Skotner, C. (2005). Adaptive state updating in real-time river flow forecasting - A combined filtering and error forecasting procedure. *Journal of Hydrology*, 308(1–4), 302–312. <https://doi.org/10.1016/j.jhydrol.2004.10.030>
- McKinney, W. (2010). Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference*, 1(Scipy), 56–61. <https://doi.org/10.25080/majora-92bf1922-00a>
- Moore, R. J. (2007). The PDM rainfall-runoff model. *Hydrology and Earth System Sciences*, 11(1), 483–499. <https://doi.org/10.5194/hess-11-483-2007>
- Moreido, V., Gartsman, B., Solomatine, D. P., & Suchilina, Z. (2021). How Well Can Machine Learning Models Perform without Hydrologists? Application of Rational Feature Selection to Improve Hydrological Forecasting. *Water*, 13(12), 1696. <https://doi.org/10.3390/w13121696>
- Nash, J. E., & Sutcliffe, J. V. (1970). River flow forecasting through conceptual models part I - A discussion of principles. *Journal of Hydrology*, 10(3), 282–290. [https://doi.org/10.1016/0022-1694\(70\)90255-6](https://doi.org/10.1016/0022-1694(70)90255-6)
- Nearing, G., Sampson, A. K., Kratzert, F., & Frame, J. (2020). *Post-Processing a Conceptual Rainfall-Runoff Model with an LSTM*. <https://doi.org/10.31223/OSF.IO/53TE4>
- Papacharalampous, G., Tyralis, H., Langousis, A., Jayawardena, A. W., Sivakumar, B., Mamassis, N., Montanari, A., & Koutsoyiannis, D. (2019). Probabilistic Hydrological Post-Processing at Scale: *Water*, 2126(ii).
- Pedregosa, F., Michel, V., Grisel OLIVIERGRISEL, O., Blondel, M., Prettenhofer, P., Weiss, R., Vanderplas, J., Cournapeau, D., Pedregosa, F., Varoquaux, G., Gramfort, A., Thirion, B., Grisel, O., Dubourg, V., Passos, A., Brucher, M., Perrot and Édouardand, M., Duchesnay, A., & Duchesnay EDOUARD DUCHESNAY, Fré. (2011). Scikit-learn: Machine Learning in Python Gaël Varoquaux Bertrand Thirion Vincent Dubourg Alexandre Passos PEDREGOSA, VAROQUAUX, GRAMFORT ET AL. Matthieu Perrot. In *Journal of Machine Learning Research* (Vol. 12). <http://scikit-learn.sourceforge.net>.
- Pianosi, F., Castelletti, A., Mancusi, L., & Garofalo, E. (2014). Improving flow forecasting by error correction modelling in altered catchment conditions. *Hydrological Processes*, 28(4), 2524–2534. <https://doi.org/10.1002/hyp.9788>
- Prakash, O., Sudheer, K. P., & Srinivasan, K. (2014). Improved higher lead time river flow forecasts using sequential neural network with error updating. *Journal of Hydrology and Hydromechanics*, 62(1), 60–74. <https://doi.org/10.2478/johh-2014-0010>

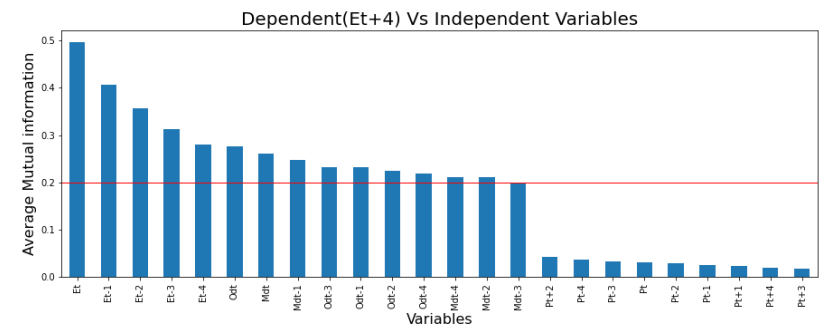
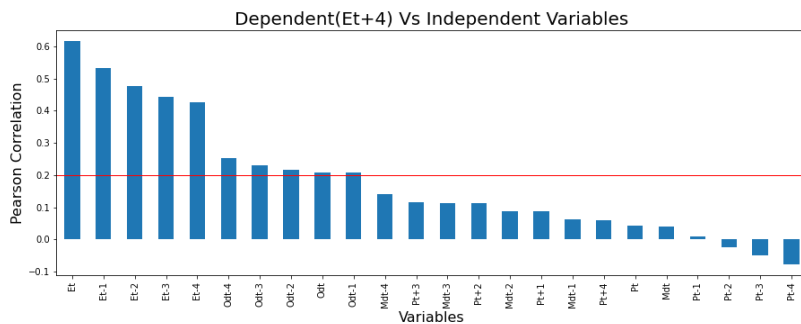
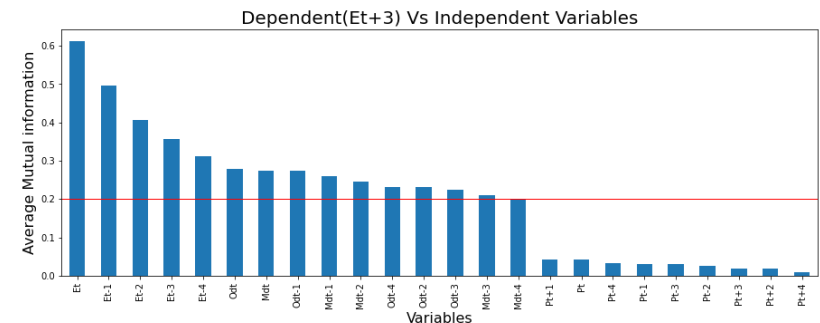
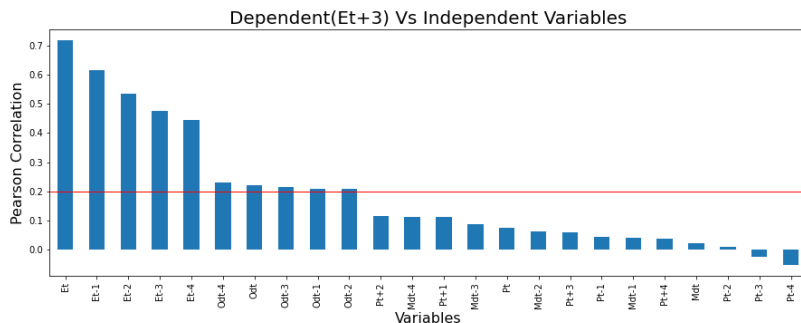
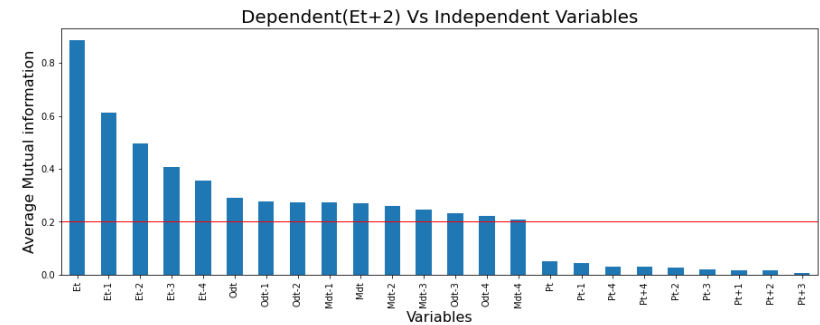
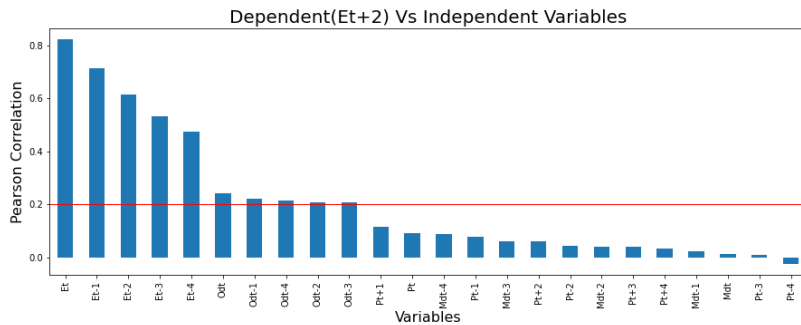
- Raftery, A. E., Gneiting, T., Balabdaoui, F., & Polakowski, M. (2005). Using Bayesian model averaging to calibrate forecast ensembles. *Monthly Weather Review*, *133*(5), 1155–1174. <https://doi.org/10.1175/MWR2906.1>
- Reformrivers.eu. (2015). *River restoration*. REFORM. <https://reformrivers.eu/river-stora---denmark-small-restoration-measure>
- Refsgaard, J. C. (1997). Validation and intercomparison of different updating procedures for real-time forecasting. *Nordic Hydrology*, *28*(2), 65–84. <https://doi.org/10.2166/nh.1997.0005>
- Saeed, A., Li, C., Danish, M., Rubaiee, S., Tang, G., Gan, Z., & Ahmed, A. (2020). Hybrid bidirectional lstm model for short-term wind speed interval prediction. *IEEE Access*, *8*, 182283–182294. <https://doi.org/10.1109/ACCESS.2020.3027977>
- Shamseldin, A. Y., & O'Connor, K. M. (2001). A non-linear neural network technique for updating of river flow forecasts. *Hydrology and Earth System Sciences*, *5*(4), 577–597. <https://doi.org/10.5194/hess-5-577-2001>
- Shen, J. C., Chang, C. H., Wu, S. J., Hsu, C. T., & Lien, H. C. (2015). Real-time correction of water stage forecast using combination of forecasted errors by time series models and Kalman filter method. *Stochastic Environmental Research and Risk Assessment*, *29*(7), 1903–1920. <https://doi.org/10.1007/s00477-015-1074-9>
- Shrestha, D. L., Kayastha, N., & Solomatine, D. P. (2009). A novel approach to parameter uncertainty analysis of hydrological models using neural networks. *Hydrology and Earth System Sciences*, *13*(7), 1235–1248. <https://doi.org/10.5194/hess-13-1235-2009>
- Shrestha, D. L., & Solomatine, D. P. (2006). Machine learning approaches for estimation of prediction interval for the model output. *Neural Networks*, *19*(2), 225–235. <https://doi.org/10.1016/j.neunet.2006.01.012>
- Shrestha, D. L., & Solomatine, D. P. (2008). 2008Shrestha.pdf. *International Journal of River Basin Management*, *6*(2), 109–122.
- Siarni-Namini, S., Tavakoli, N., & Namin, A. S. (2019). The Performance of LSTM and BiLSTM in Forecasting Time Series. *Proceedings - 2019 IEEE International Conference on Big Data, Big Data 2019*, 3285–3292. <https://doi.org/10.1109/BigData47090.2019.9005997>
- Sigrist, F. (2021). Gradient and Newton boosting for classification and regression. In *Expert Systems with Applications* (Vol. 167). <https://doi.org/10.1016/j.eswa.2020.114080>
- Sikorska-Senoner, A. E., & Quilty, J. M. (2021). A novel ensemble-based conceptual-data-driven approach for improved streamflow simulations. *Environmental Modelling & Software*, *143*(May), 105094. <https://doi.org/10.1016/j.envsoft.2021.105094>
- Solomatine, D. P. (2017). Data-driven modelling: machine learning, data mining and knowledge discovery. *Lecture Notes, IHE Delft*, 141.
- Sun, Y., Bao, W., Jiang, P., Ji, X., Gao, S., Xu, Y., Zhang, Q., & Si, W. (2018). Development

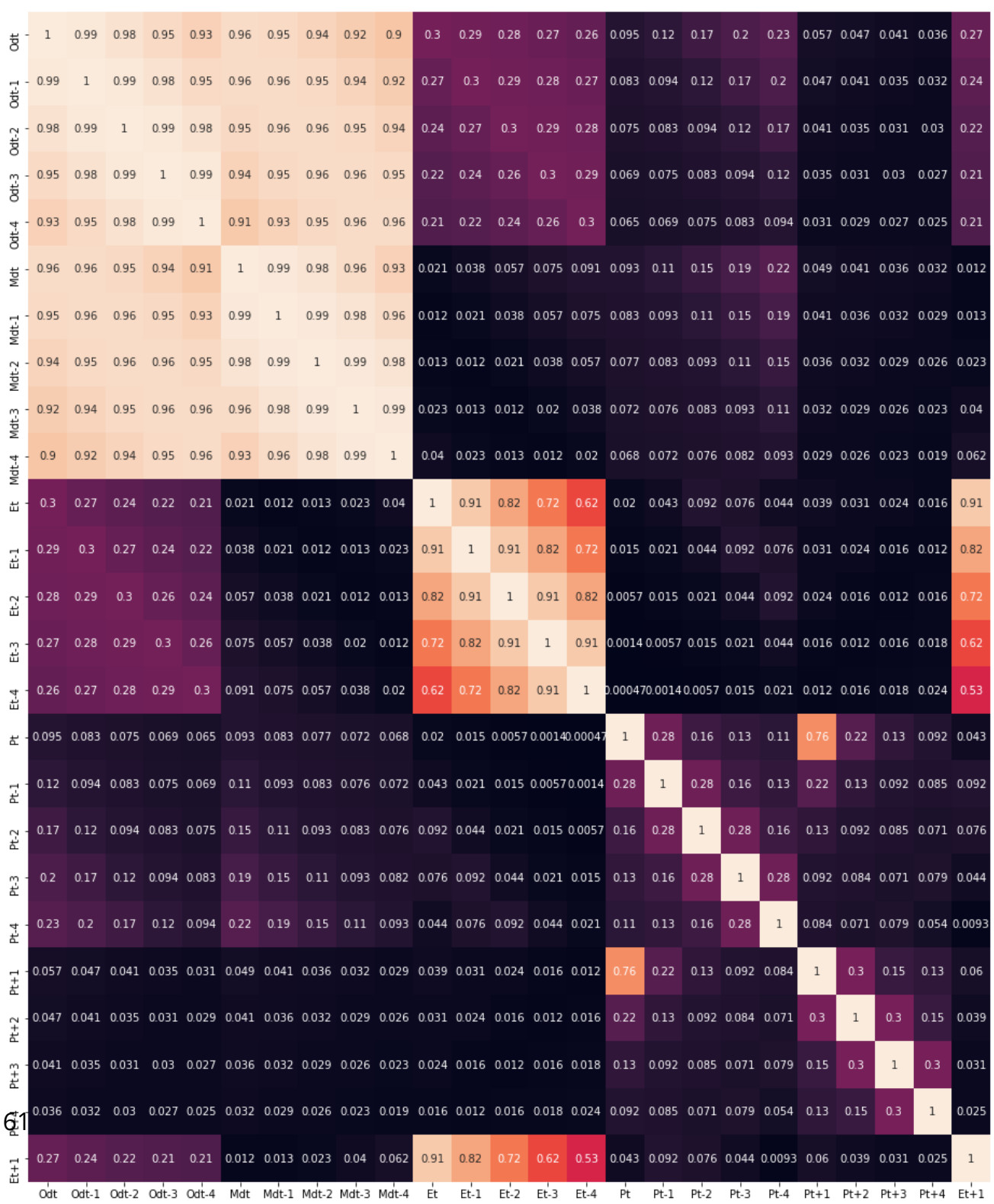
- of Multivariable Dynamic System Response Curve Method for Real-Time Flood Forecasting Correction. *Water Resources Research*, 54(7), 4730–4749. <https://doi.org/10.1029/2018WR022555>
- Todini, E. (2008). A model conditional processor to assess predictive uncertainty in flood forecasting. *International Journal of River Basin Management*, 6(2), 123–137. <https://doi.org/10.1080/15715124.2008.9635342>
- Torres-Rua, A. F., Tíclavilca, A. M., Walker, W. R., & McKee, M. (2012). Machine Learning Approaches for Error Correction of Hydraulic Simulation Models for Canal Flow Schemes. *Journal of Irrigation and Drainage Engineering*, 138(11), 999–1010. [https://doi.org/10.1061/\(asce\)ir.1943-4774.0000489](https://doi.org/10.1061/(asce)ir.1943-4774.0000489)
- Tyrallis, H., Papacharalampous, G., Burnetas, A., & Langousis, A. (2019). Hydrological post-processing using stacked generalization of quantile regression algorithms: Large-scale application over CONUS. *Journal of Hydrology*, 577(January), 123957. <https://doi.org/10.1016/j.jhydrol.2019.123957>
- Van Der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy array: A structure for efficient numerical computation. *Computing in Science and Engineering*, 13(2), 22–30. <https://doi.org/10.1109/MCSE.2011.37>
- Wani, O., Beckers, J. V. L., Weerts, A. H., & Solomatine, D. P. (2017). Residual uncertainty estimation using instance-based learning with applications to hydrologic forecasting. *Hydrology and Earth System Sciences*, 21(8), 4021–4036. <https://doi.org/10.5194/hess-21-4021-2017>
- Watson, P. A. G. (2019). Applying Machine Learning to Improve Simulations of a Chaotic Dynamical System Using Empirical Error Correction. *Journal of Advances in Modeling Earth Systems*, 11(5), 1402–1417. <https://doi.org/10.1029/2018MS001597>
- Weerts, A. H., Winsemius, H. C., & Verkade, J. S. (2011). Estimation of predictive hydrological uncertainty using quantile regression: Examples from the National Flood Forecasting System (England and Wales). *Hydrology and Earth System Sciences*, 15(1), 255–265. <https://doi.org/10.5194/hess-15-255-2011>
- Wu, S. J., Lien, H. C., Chang, C. H., & Shen, J. C. (2012). Real-time correction of water stage forecast during rainstorm events using combination of forecast errors. *Stochastic Environmental Research and Risk Assessment*, 26(4), 519–531. <https://doi.org/10.1007/s00477-011-0514-4>
- Wunsch, A., Liesch, T., & Broda, S. (2021). Groundwater level forecasting with artificial neural networks: A comparison of long short-term memory (LSTM), convolutional neural networks (CNNs), and non-linear autoregressive networks with exogenous input (NARX). *Hydrology and Earth System Sciences*, 25(3), 1671–1687. <https://doi.org/10.5194/hess-25-1671-2021>
- Xiong, L., & O'Connor, K. M. (2002). Comparison of four updating models for real-time river flow forecasting. *Hydrological Sciences Journal*, 47(4), 621–639. <https://doi.org/10.1080/02626660209492964>

Yu, P.-S., & Chen, S.-T. (2005). Updating Real-Time Flood Forecasting Using a Fuzzy Rule-Based Model/Mise à Jour de Prédiction de Crue en Temps Réel Grâce à un Modèle à Base de Règles Floues. *Hydrological Sciences Journal*, 50(2), 2-278. <https://doi.org/10.1623/hysj.50.2.265.61796>

Appendices

Appendix A: Comparison of Pearson correlation and Average mutual information at different lead times

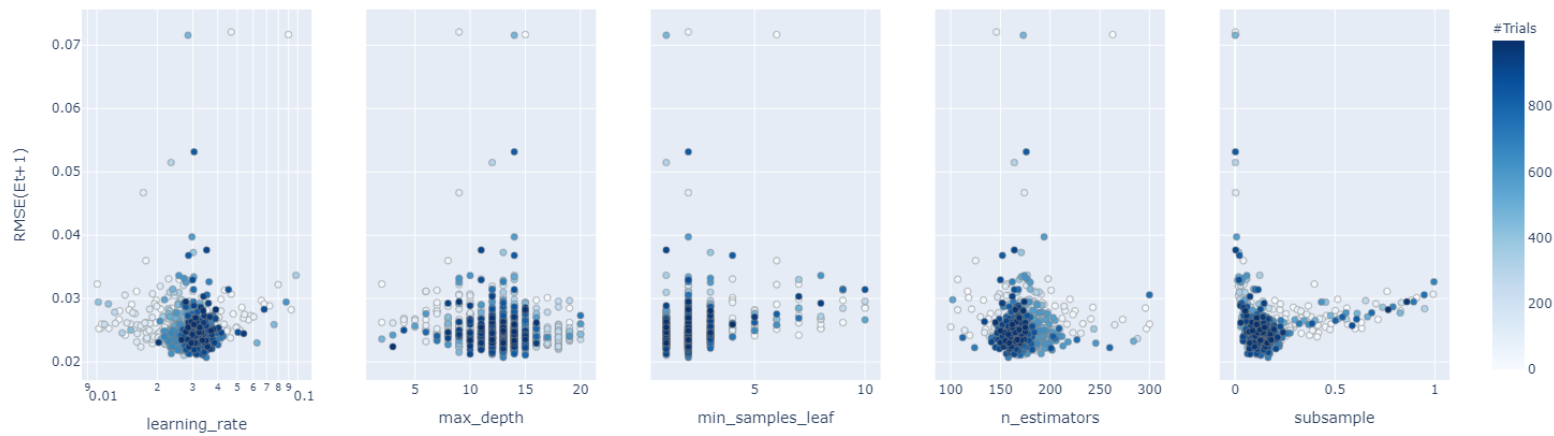




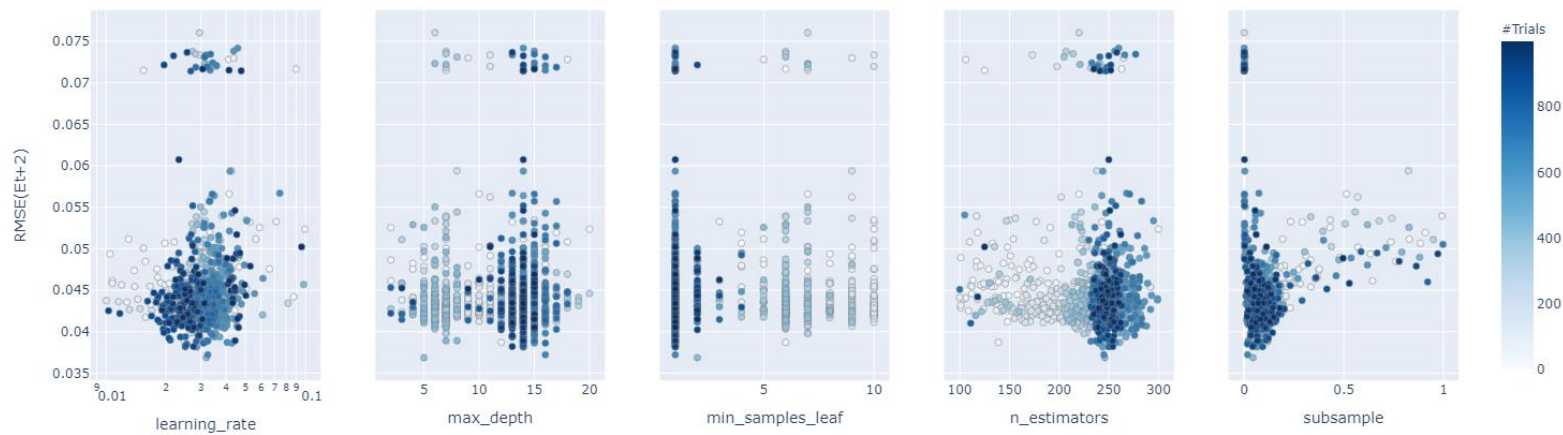
Appendix B: Distribution plots of hyperparameters in Bayesian optimization for tree-based and deep learning methods.

Gradient Boosting

Slice Plot 12 hours



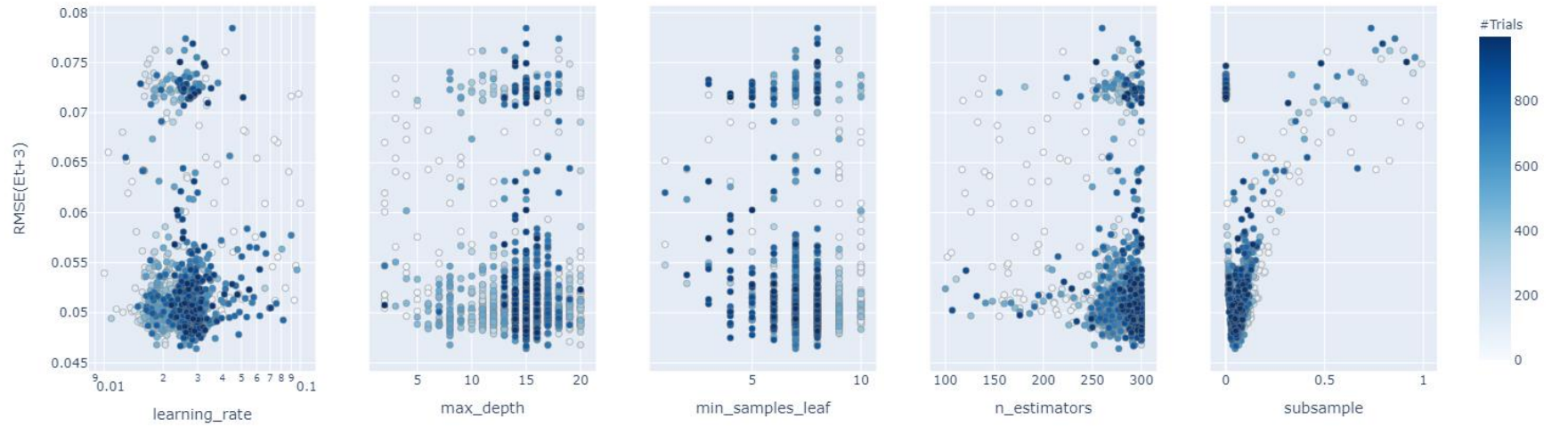
Slice Plot 24 hours



Gradient Boosting

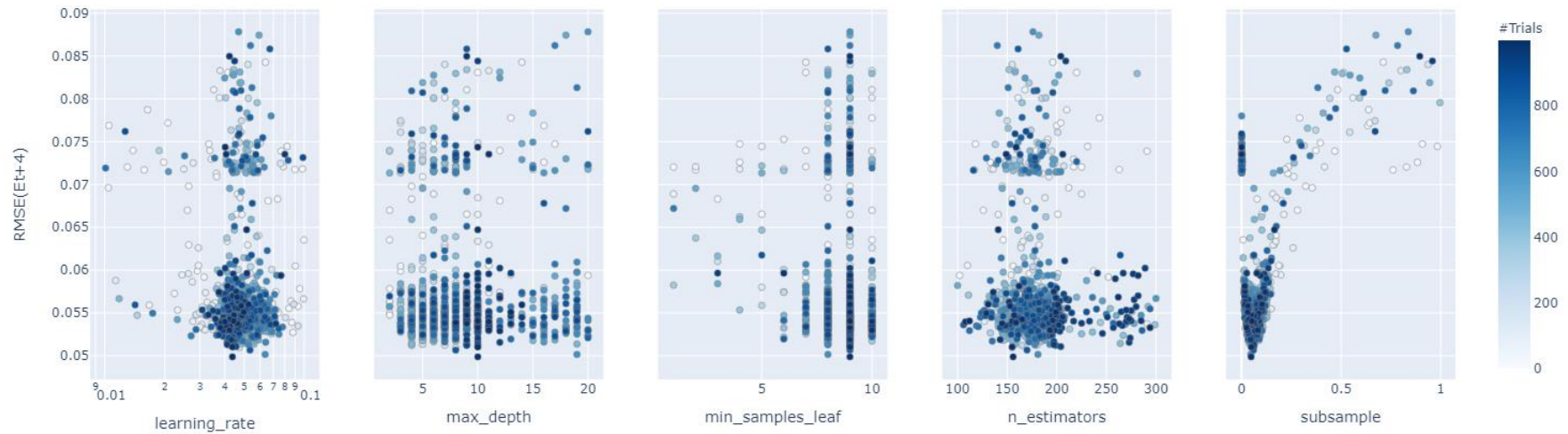
Slice Plot

36 hours



Slice Plot

48 hours

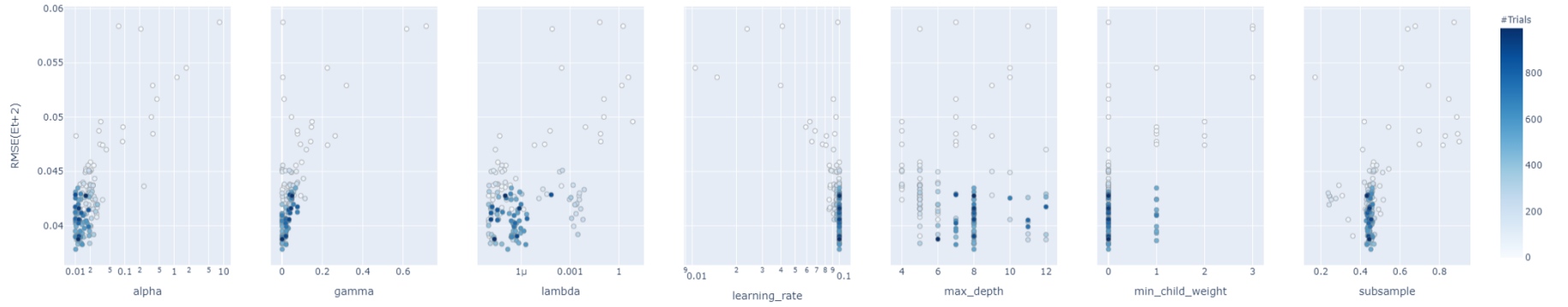


XGBoost

Slice Plot 12 hours

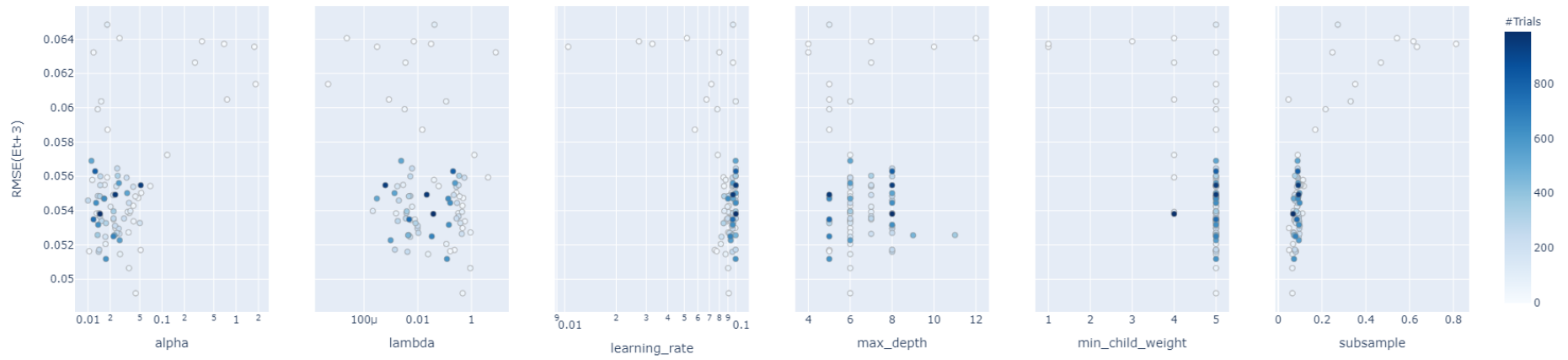


Slice Plot 24 hours

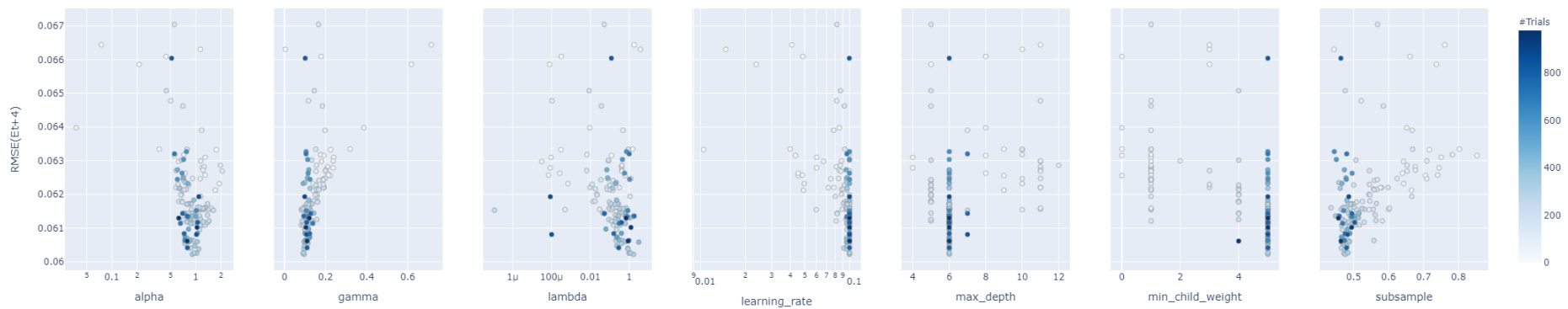


XGBoost

Slice Plot 36 hours

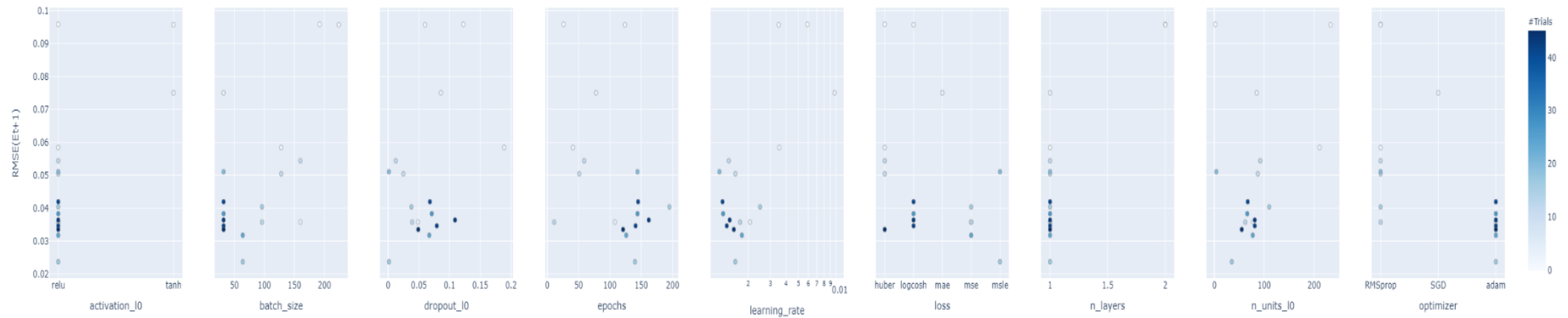


Slice Plot 48 hours

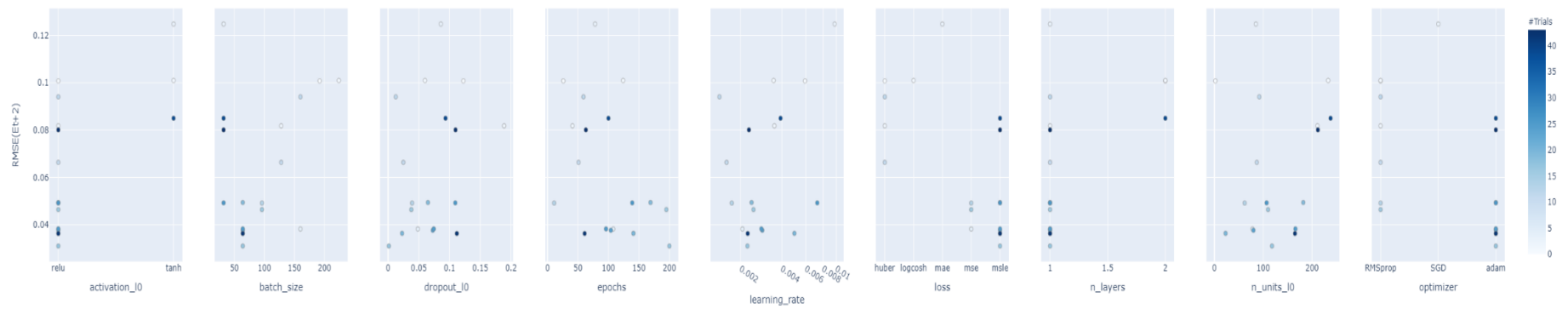


ANN

Slice Plot 12 hours

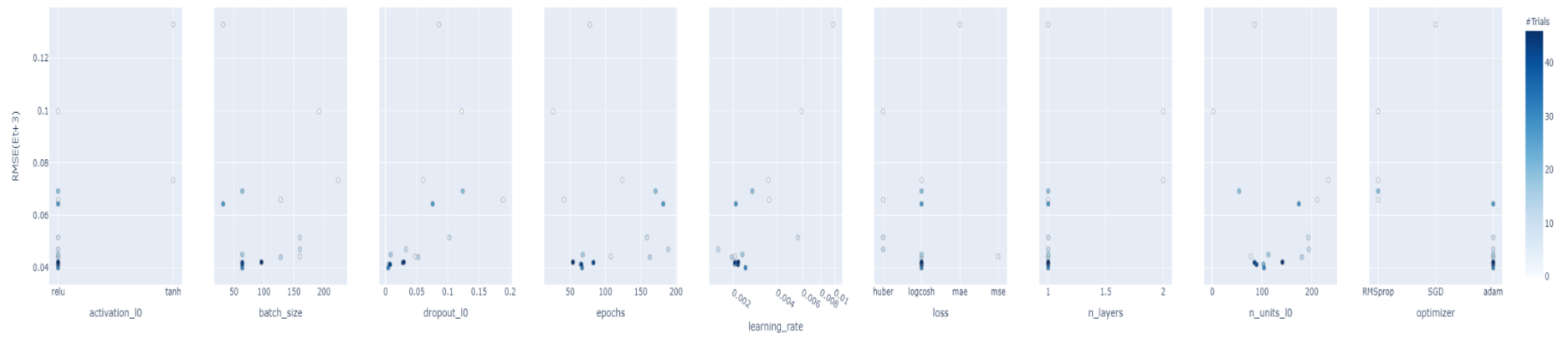


Slice Plot 24 hours

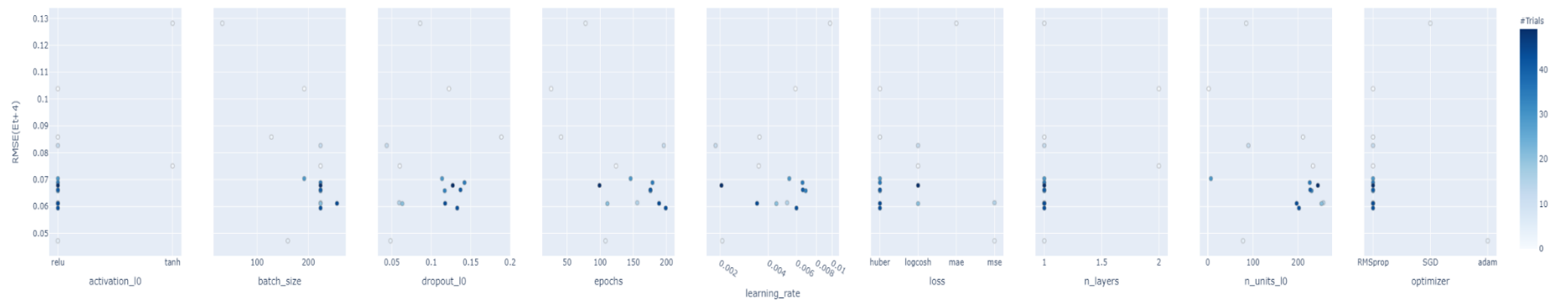


ANN

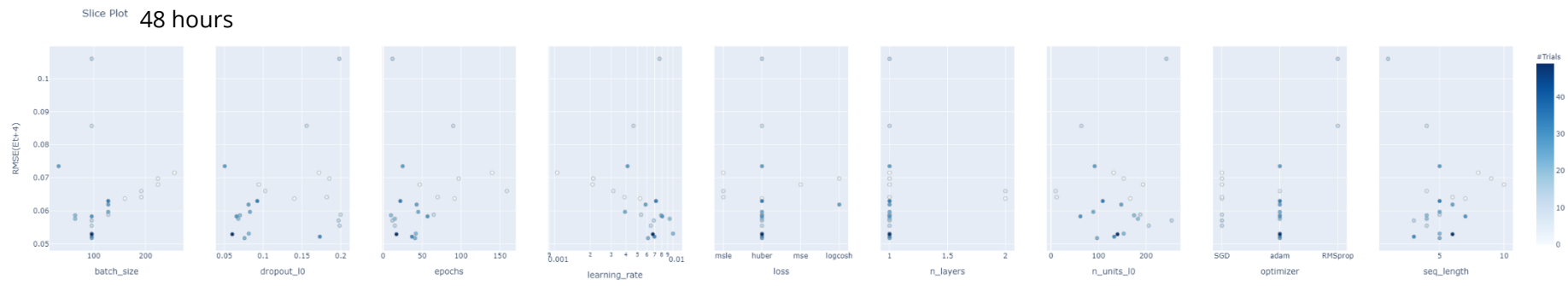
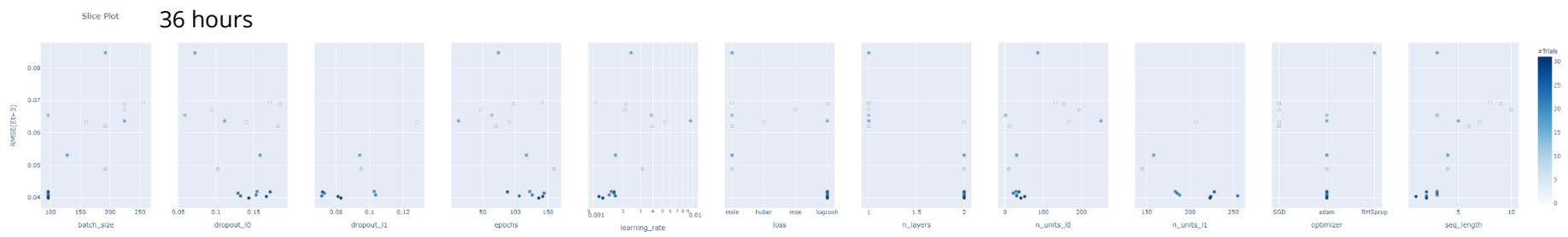
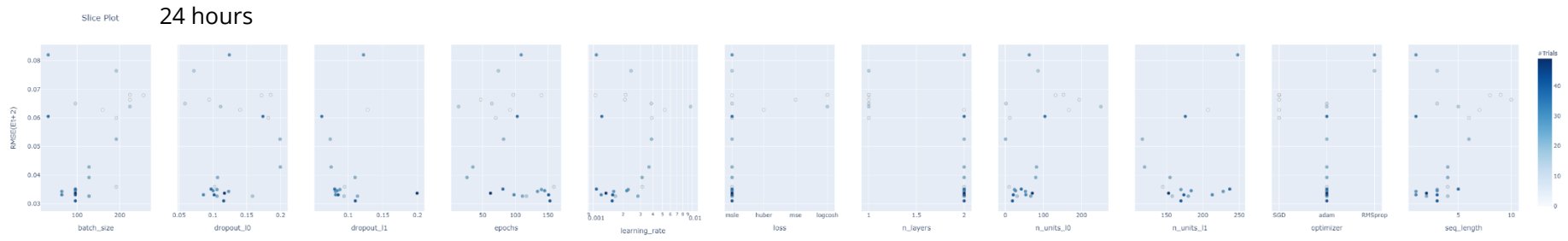
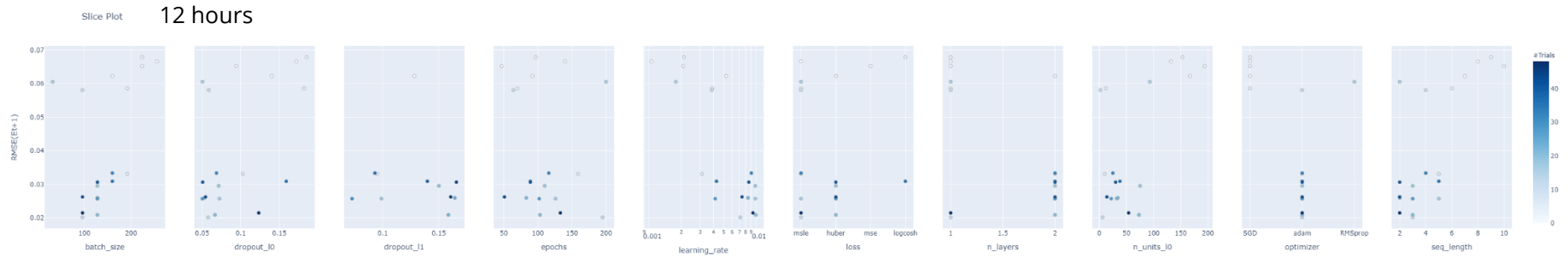
Slice Plot 36 hours



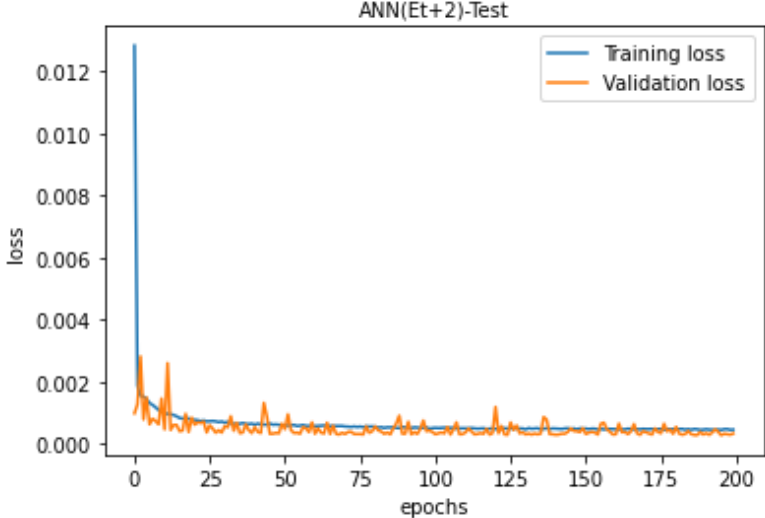
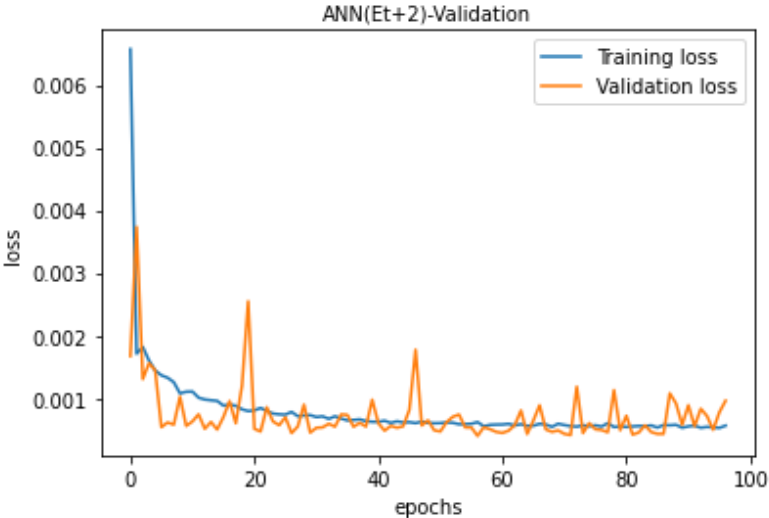
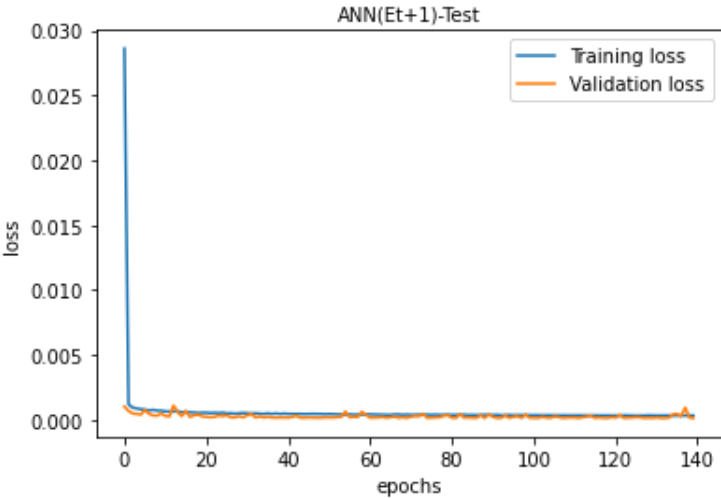
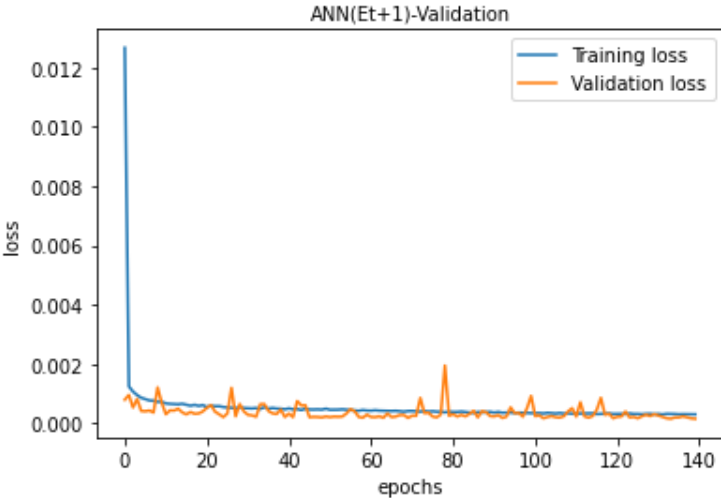
Slice Plot 48 hours

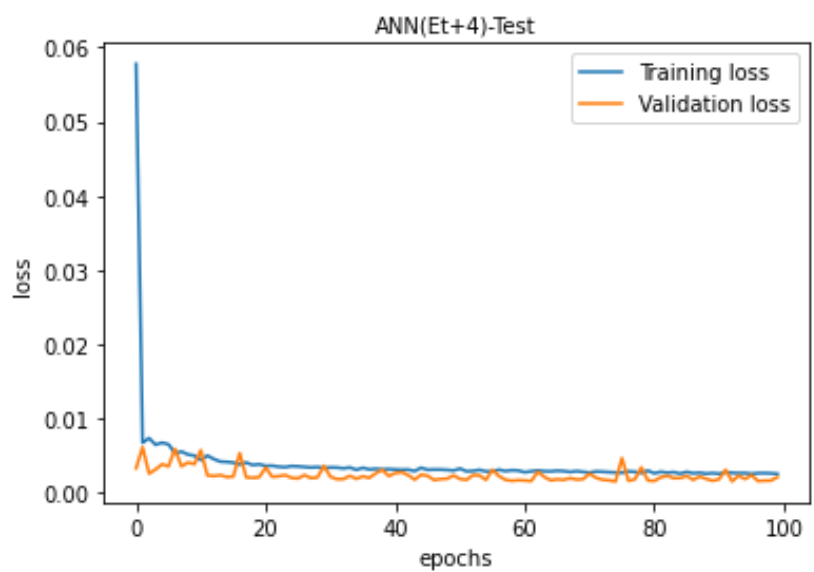
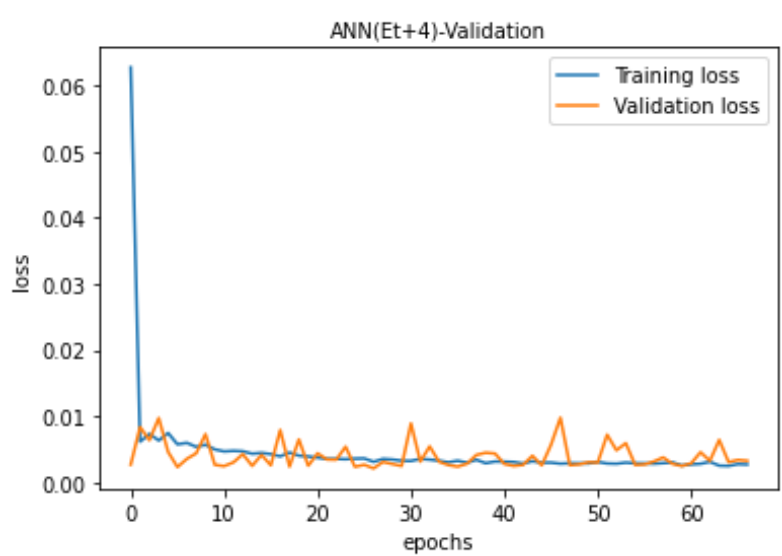
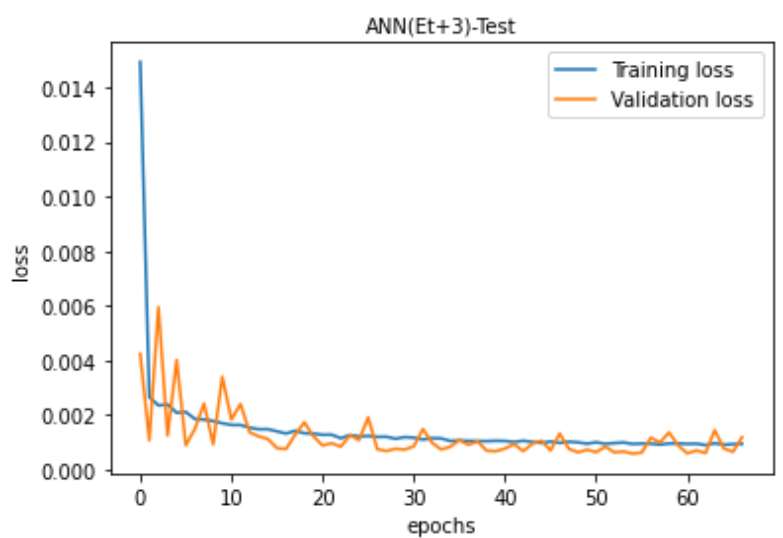
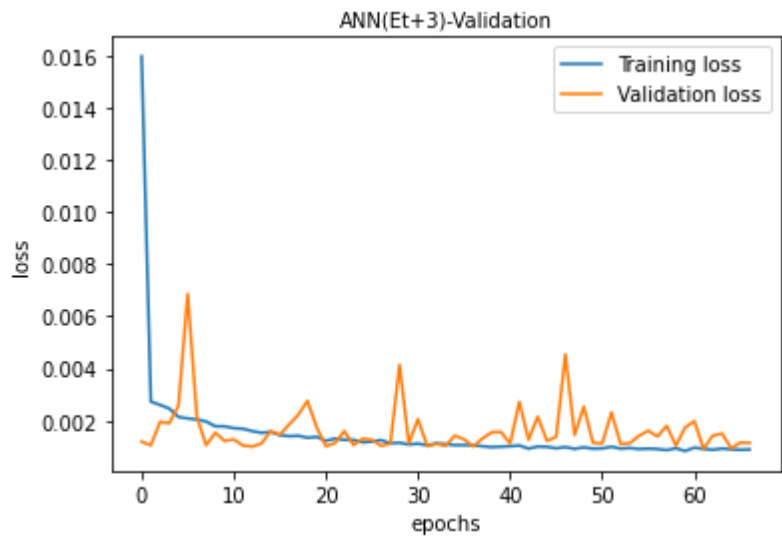


BLSTM

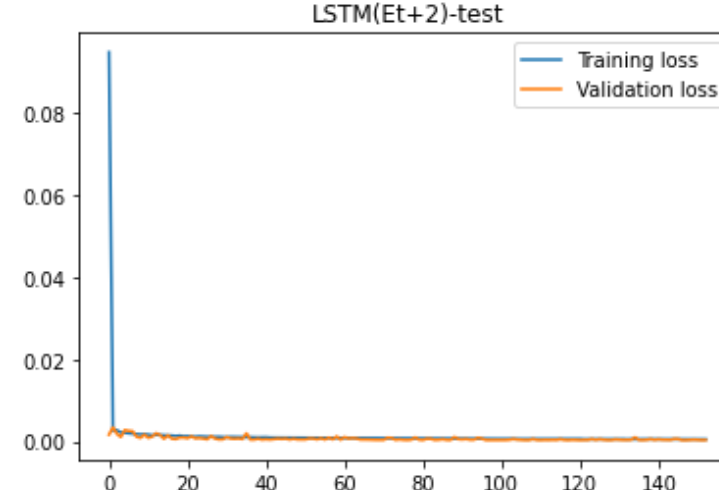
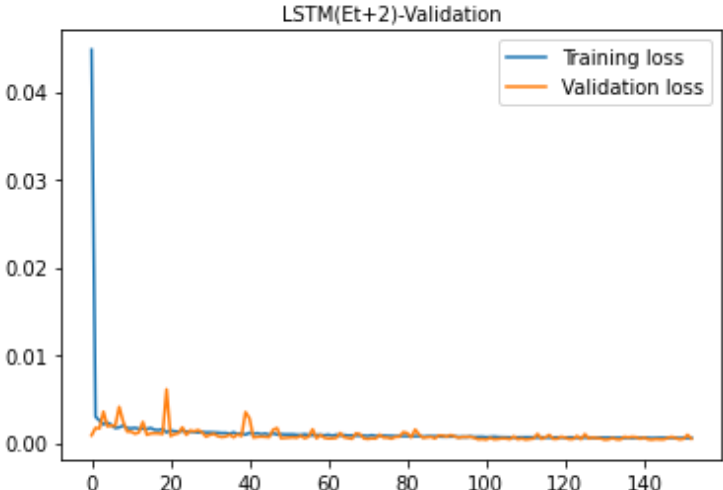
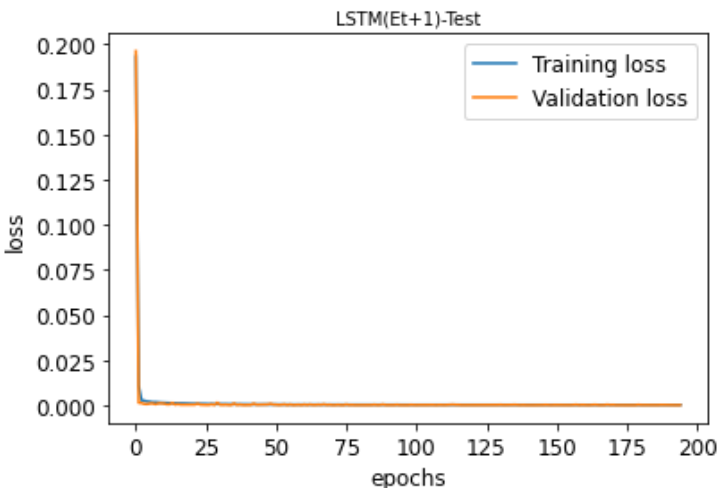
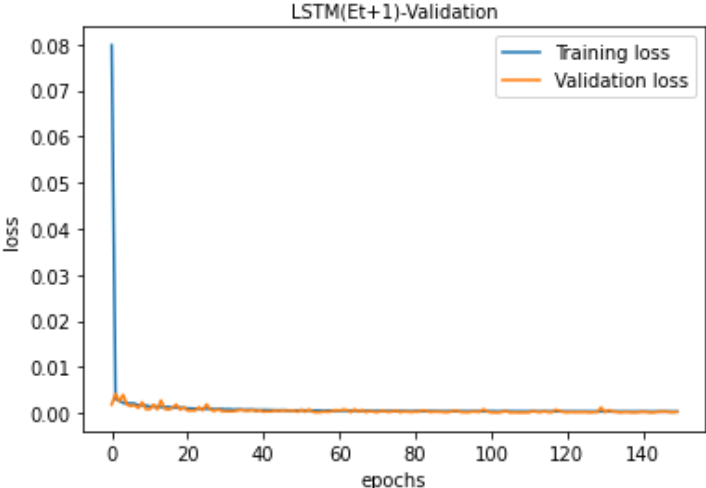


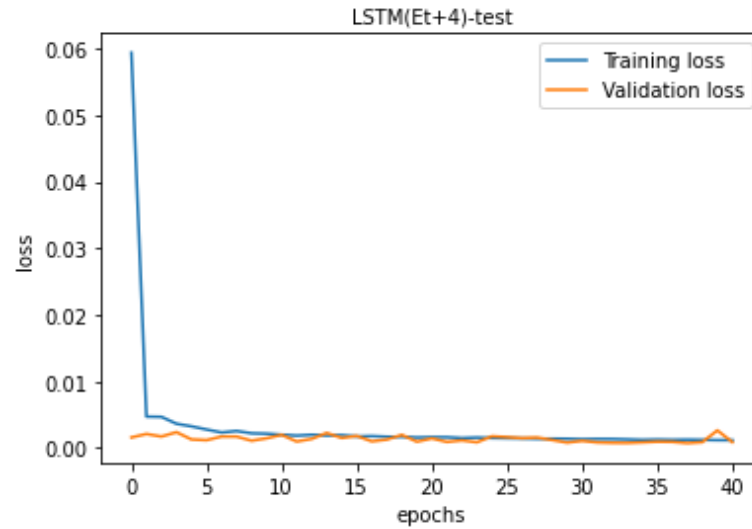
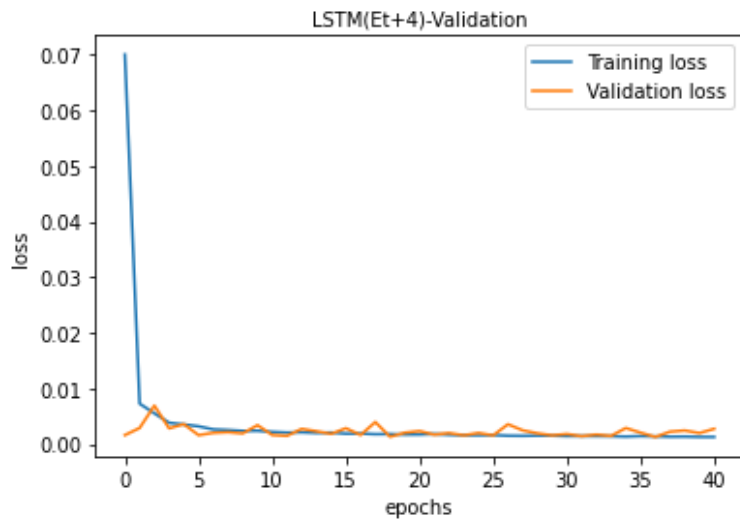
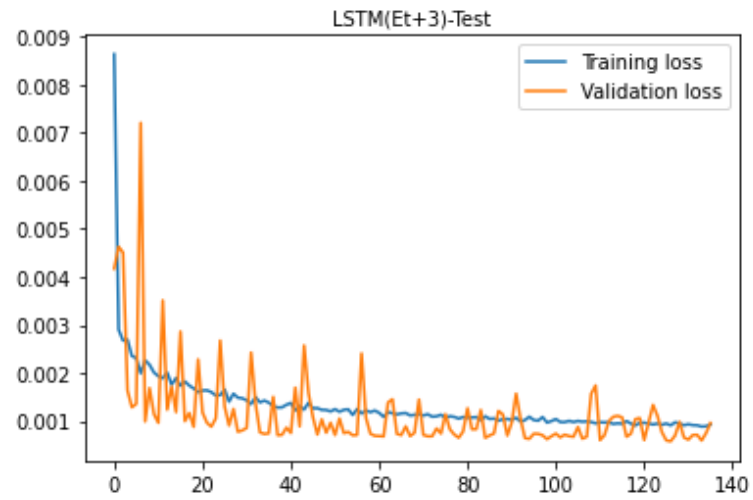
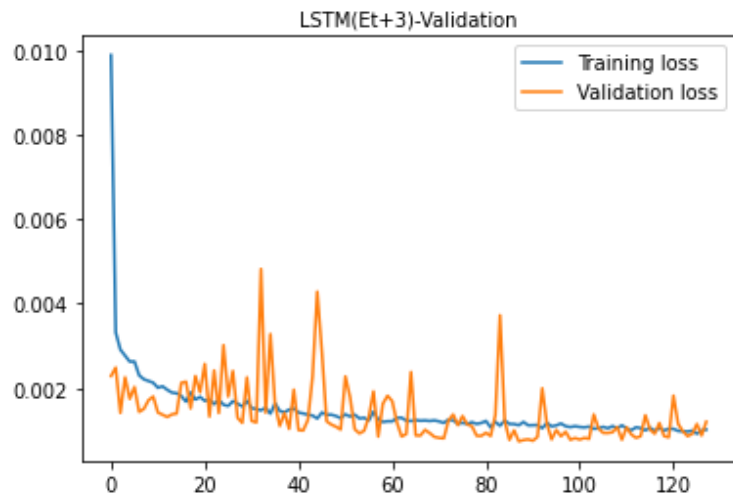
Appendix C: Learning curves of ANN at different lead times during validation and test





Appendix D: Learning curves of BLSTM at different lead times for validation and test sets





Appendix E: Final best hyperparameters for the models evaluated

Table 17: Final best hyperparameters for tree-based and neural network-based methods

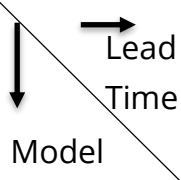
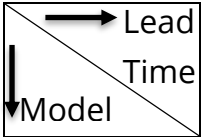
 Lead Time Model	12 hours	24 hours	36 hours	48 hours
XGBoost	{'max_depth': 6, 'learning_rate': 0.09976512440998135, 'subsample': 0.6714706798105836, 'alpha': 0.013528920895993416, 'lambda': 3.5267815447831284e-06, 'n_estimators': 60, 'gamma': 0.0007976125851248869, 'min_child_weight': 0}	{'max_depth': 8, 'learning_rate': 0.09982384902621824, 'subsample': 0.4529709764793954, 'alpha': 0.010140093256853989, 'lambda': 1.5078295673336042e-07, 'n_estimators': 33, 'gamma': 0.0008994193435131781, 'min_child_weight': 0}	{'max_depth': 6, 'learning_rate': 0.09089086924986431, 'subsample': 0.06463720484806382, 'alpha': 0.04408963914623115, 'lambda': 0.46938301088901824, 'n_estimators': 39, 'gamma': 0, 'min_child_weight': 5}	{'max_depth': 6, 'learning_rate': 0.09983079154027447, 'subsample': 0.47630344992475326, 'alpha': 0.9080283123276469, 'lambda': 1.6941246182655454, 'n_estimators': 95, 'gamma': 0.09538367105491903, 'min_child_weight': 5}
Gradient Boosting	{'max_depth': 13, 'learning_rate': 0.035092865547259953, 'subsample': 0.16502226493209313, 'min_samples_leaf': 2, 'n_estimators': 158}	{'max_depth': 5, 'learning_rate': 0.03184503113868548, 'subsample': 0.04477894948162988, 'min_samples_leaf': 2, 'n_estimators': 246}	{'max_depth': 15, 'learning_rate': 0.029490164484633367, 'subsample': 0.04724542865430821, 'min_samples_leaf': 7, 'n_estimators': 277}	{'max_depth': 10, 'learning_rate': 0.04366914991680411, 'subsample': 0.047875027401341676, 'min_samples_leaf': 9, 'n_estimators': 156}

Table 18: Final best hyperparameters for neural network-based methods

	12 hours	24 hours	36 hours	48 hours
ANN	{'learning_rate': 0.0015798028419481372, 'optimizer': 'adam', 'loss': 'msle', 'epochs': 140, 'batch_size': 64, 'n_layers': 1, 'n_units_l0': 35, 'activation_l0': 'relu', 'dropout_l0': 0.0013014518954329451}	{'learning_rate': 0.0022452983732626896, 'optimizer': 'adam', 'loss': 'msle', 'epochs': 200, 'batch_size': 64, 'n_layers': 1, 'n_units_l0': 118, 'activation_l0': 'relu', 'dropout_l0': 0.0013014518954329451}	{'learning_rate': 0.002433138709536434, 'optimizer': 'adam', 'loss': 'logcosh', 'epochs': 67, 'batch_size': 64, 'n_layers': 1, 'n_units_l0': 104, 'activation_l0': 'relu', 'dropout_l0': 0.004314768089342425}	{'learning_rate': 0.0020650324085196487, 'optimizer': 'adam', 'loss': 'mse', 'epochs': 108, 'batch_size': 160, 'n_layers': 1, 'n_units_l0': 78, 'activation_l0': 'relu', 'dropout_l0': 0.048415175080705475}
BLSTM	{'seq_length': 3, 'learning_rate': 0.007141029339645854, 'optimizer': 'adam', 'loss': 'msle', 'epochs': 195, 'batch_size': 96, 'n_layers': 1, 'n_units_l0': 6, 'activation_l0': 'tanh', 'dropout_l0': 0.05791790765558189}	{'seq_length': 3, 'learning_rate': 0.0015542464923596429, 'optimizer': 'adam', 'loss': 'msle', 'epochs': 153, 'batch_size': 96, 'n_layers': 2, 'n_units_l0': 19, 'activation_l0': 'tanh', 'dropout_l0': 0.11589575420708235, 'n_units_l1': 170, 'activation_l1': 'tanh', 'dropout_l1': 0.11033620463155927}	{'seq_length': 2, 'learning_rate': 0.0012494524325071636, 'optimizer': 'adam', 'loss': 'logcosh', 'epochs': 136, 'batch_size': 96, 'n_layers': 2, 'n_units_l0': 42, 'activation_l0': 'tanh', 'dropout_l0': 0.14362520285391292, 'n_units_l1': 223, 'activation_l1': 'tanh', 'dropout_l1': 0.0829144604491681}	{'seq_length': 5, 'learning_rate': 0.006142381627495895, 'optimizer': 'adam', 'loss': 'huber', 'epochs': 41, 'batch_size': 96, 'n_layers': 1, 'n_units_l0': 97, 'activation_l0': 'tanh', 'dropout_l0': 0.0760273092979798}

Appendix F: Comparison of corrected discharges at different lead times

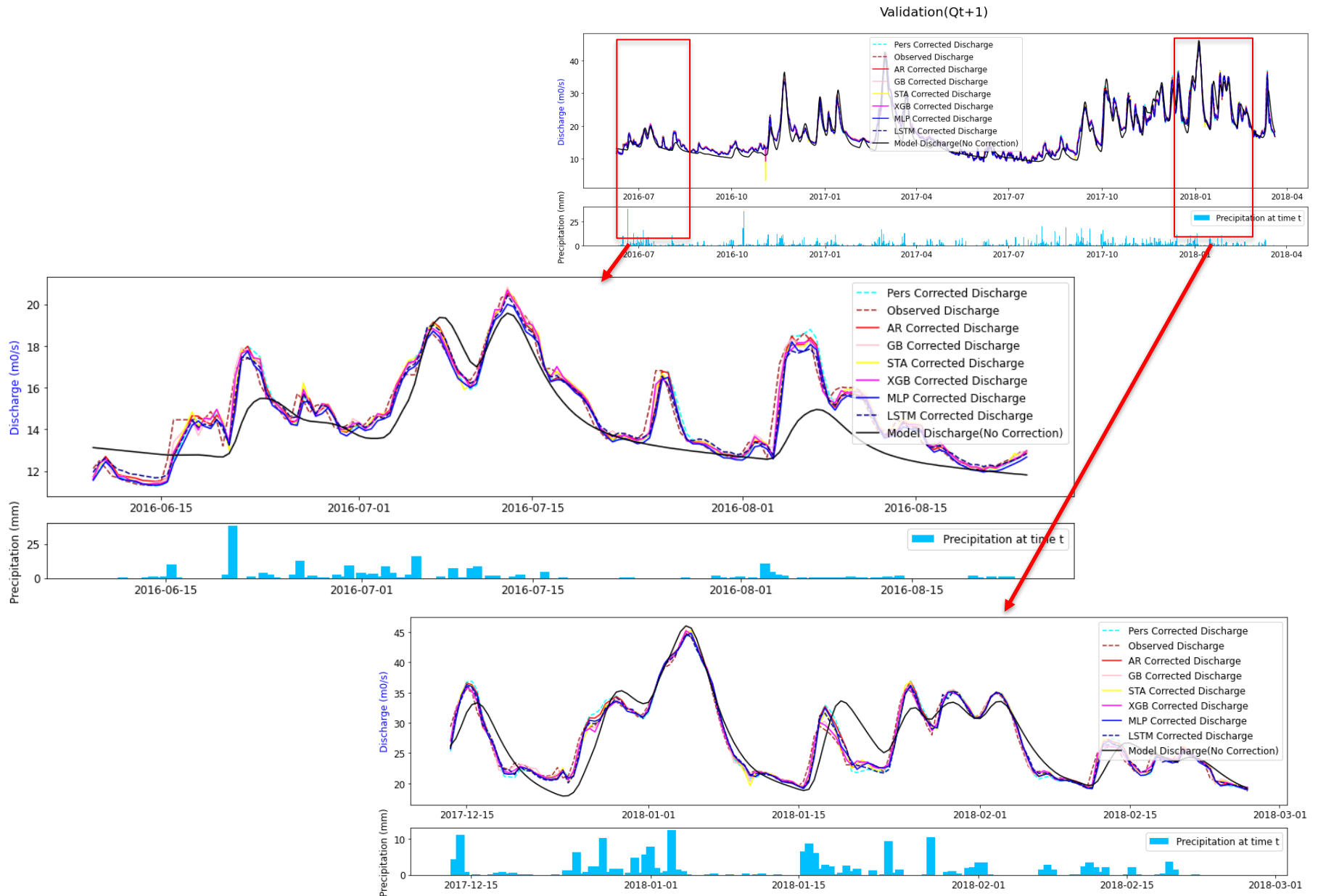


Figure 21: Comparison plots of corrected discharge time series at 12 hours lead time on the validation set.

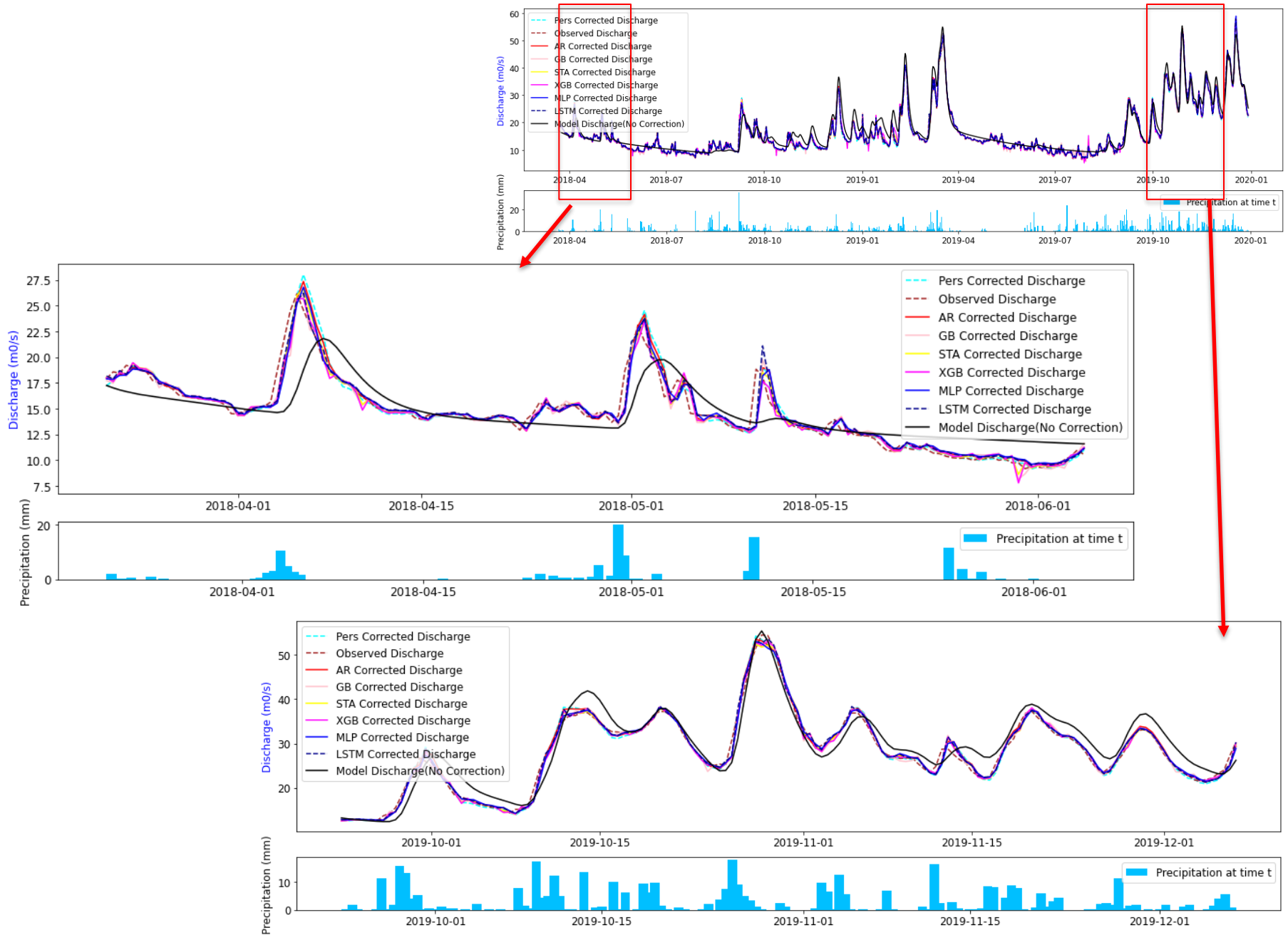


Figure 22: Comparison plots of corrected discharge time series at 12 hours lead time on the test set.

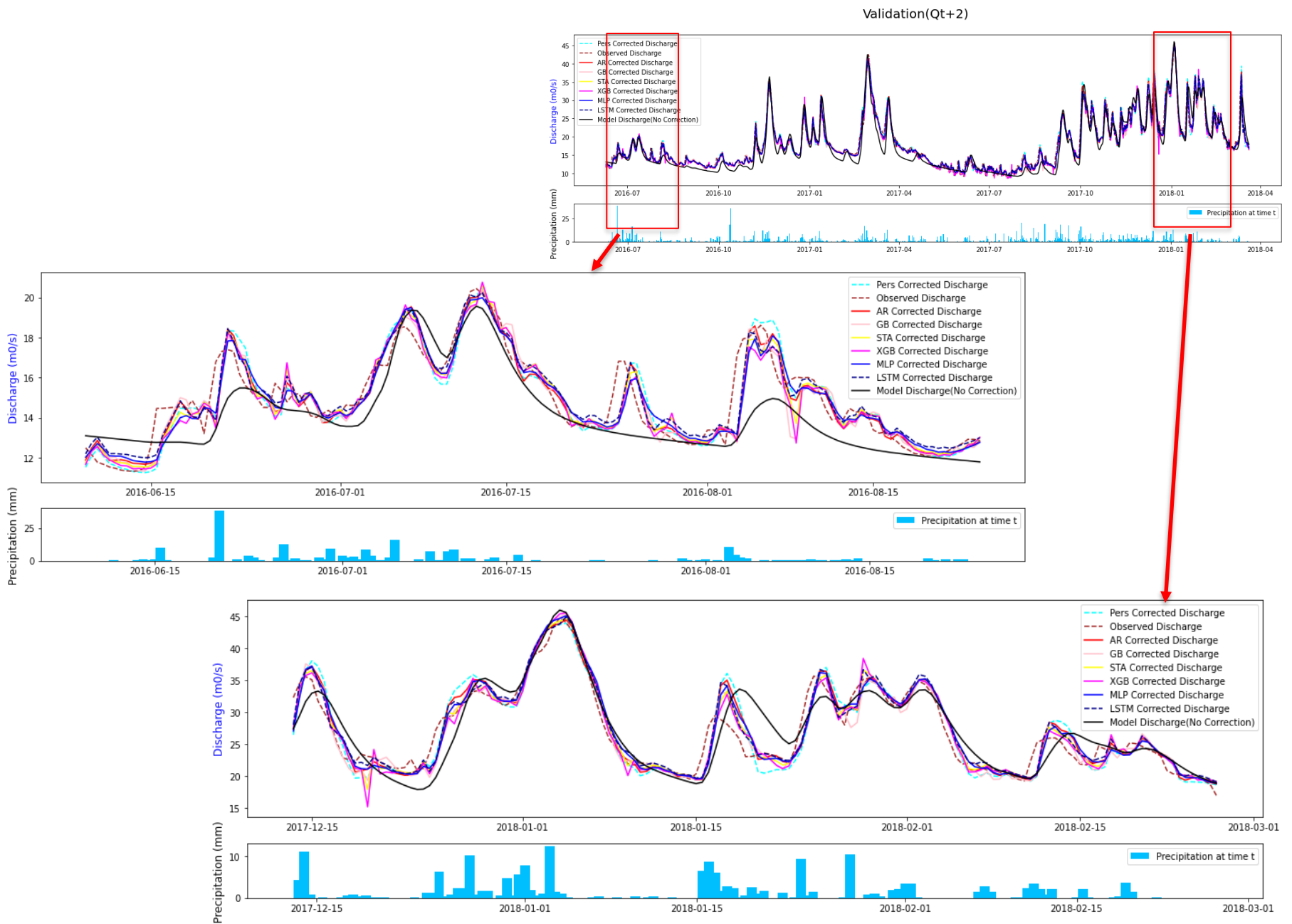


Figure 23: Comparison plots of corrected discharge time series at 24 hours lead time on the validation set.

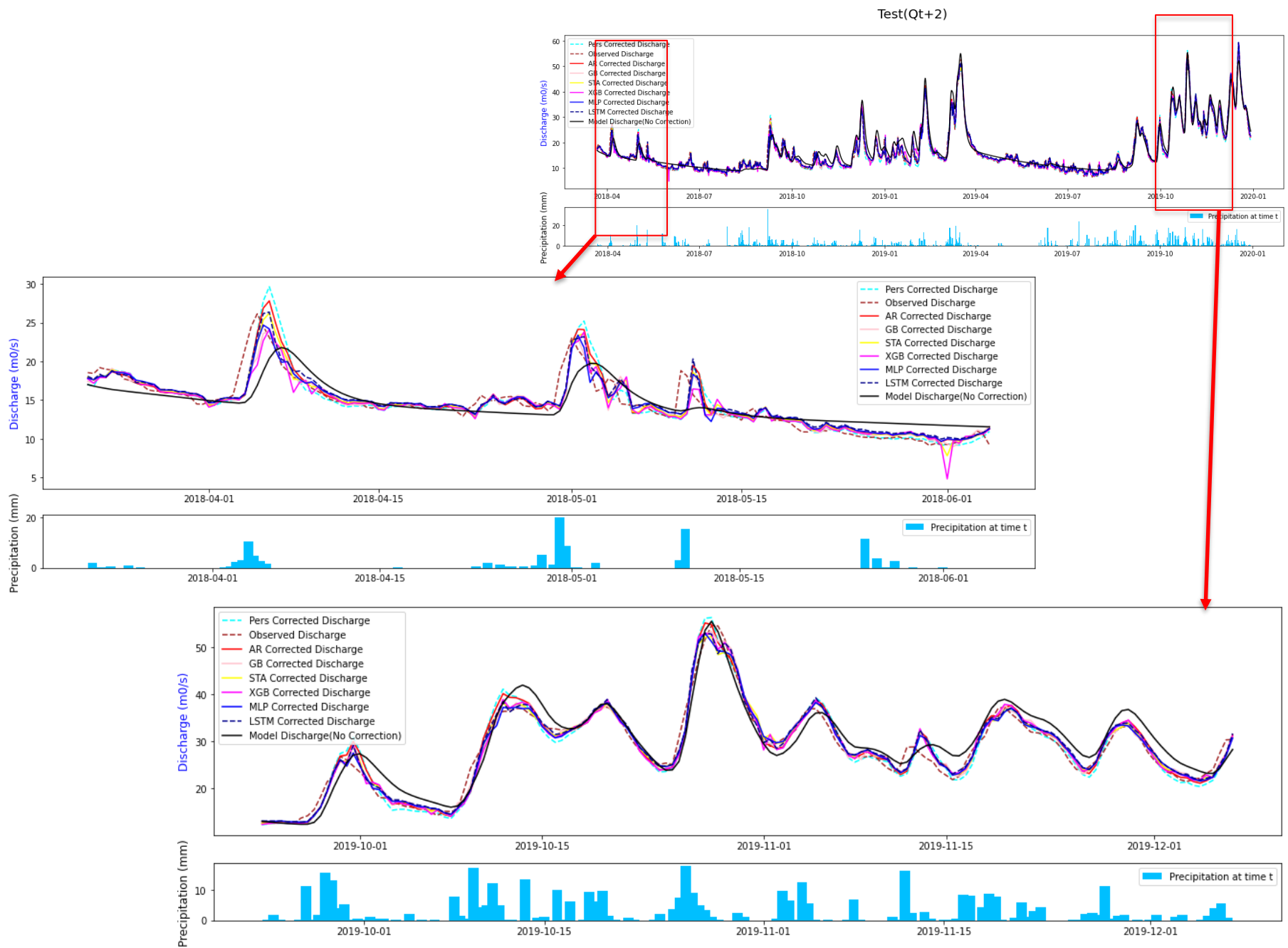


Figure 24: Comparison plots of corrected discharge time series at 24 hours lead time on the test set.

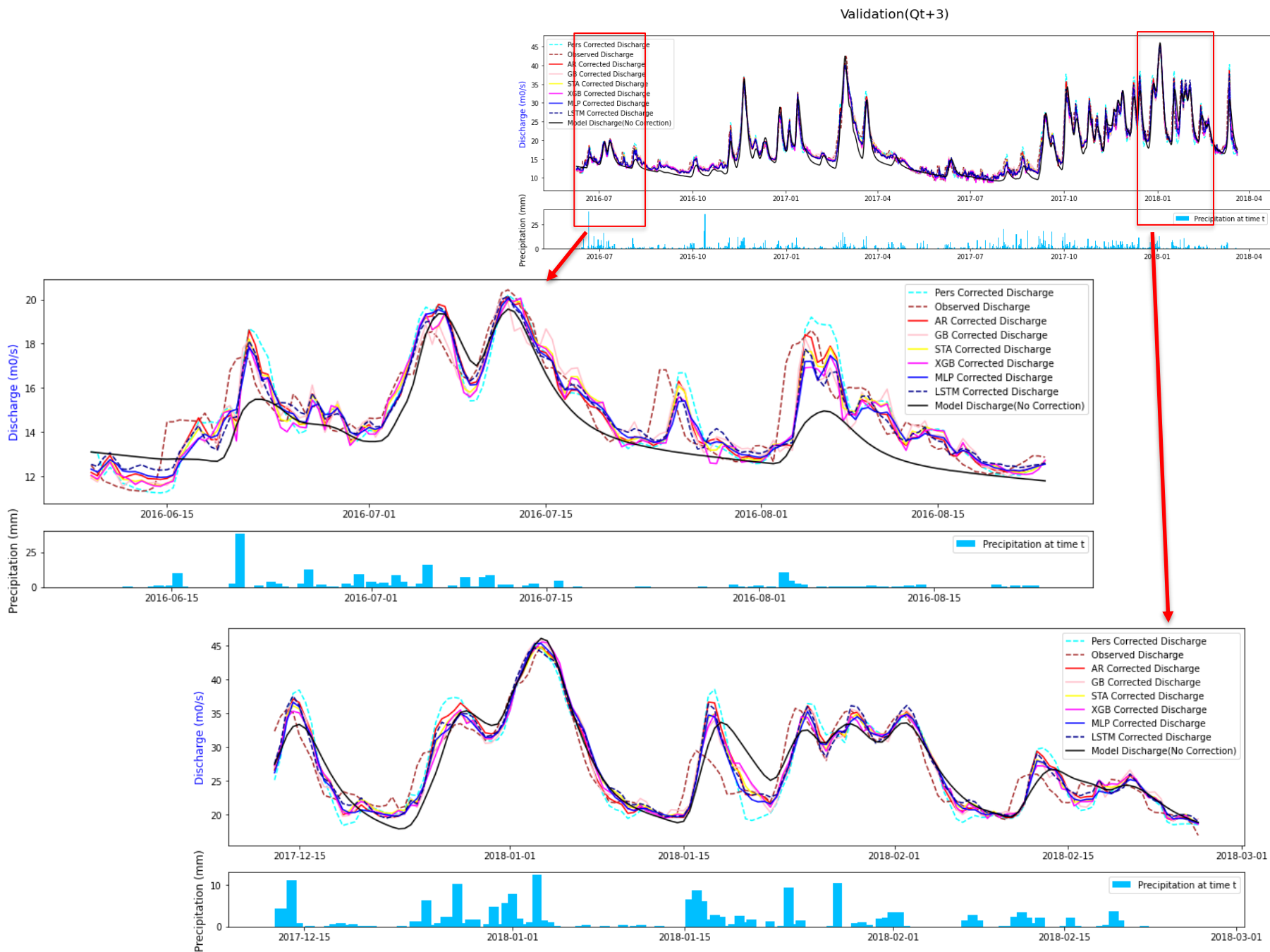


Figure 25: Comparison plots of corrected discharge time series at 36 hours lead time on the validation set.

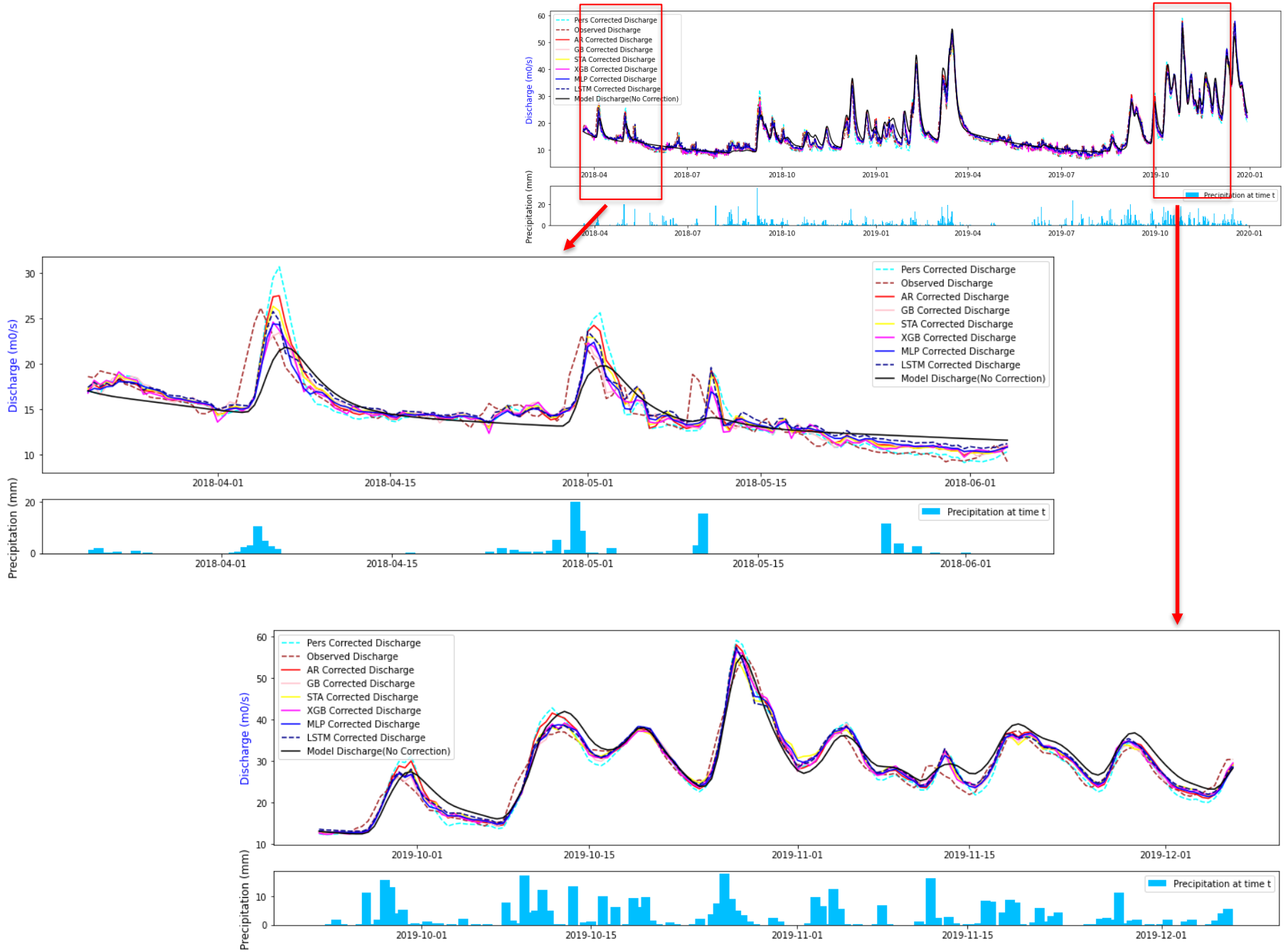


Figure 26: Comparison plots of corrected discharge time series at 36 hours lead time on the test set.