Deep Learning Framework for High-Resolution Hydrological Forecasting Using Hydrometric Data

Md Asifur Rahman

A THESIS SUBMITTED TO
THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF ARTS

GRADUATE PROGRAM IN INFORMATION SYSTEMS AND TECHNOLOGY

YORK UNIVERSITY
TORONTO, ONTARIO

April 2026

Abstract

Multi-site multi-horizon hydrological forecasting is gaining increasing attention in watershed management due to spatially distributed hydrological processes, which cannot be represented using single site models. This study develops a deep learning (DL) framework for forecasting hydrological regime in streams at multiple cross-sections using hydrometric observations. It has been shown that multi-site models with a shared static loss function better represent the sites with regular hydrological regimes and moderate flows, while rapid response sites are not modelled accurately. To address this limitation, an Adaptive Weighted Loss strategy was developed to periodically re-assign weights to site losses depending on the validation performance. The framework was assessed on datasets with different hydrological characteristics. The results highlighted that ALF-based DL framework improved performance at response sites by extending horizon of reliable forecasts. Overall, the study shows that combining watershed-level multi-site forecasting with adaptive weighting provides a scalable and reproducible approach for high-resolution hydrological predictions.

Acknowledgement

First of all, I would like to sincerely thank my supervisor, Professor Marina G. Erechtchoukova for her guidance, encouragement and continuous support throughout the course of this research. Her mentorship and feedback greatly contributed to the development of this study and to my growth as a researcher. I wholeheartedly appreciate the chance to carry out this work under her guidance.

Secondly, I am thankful to the faculty of LAPS for awarding the Kitty Lundy Memorial Graduate Fellowship and the faculty of Graduate Studies for the Academic Excellence Fund, which supported this study.

Furthermore, I would also like to extend my appreciation to the committee members, Professor Arik Senderovich and Professor Hany E. Z. Farag, for dedicating their valuable time in reviewing my thesis.

Finally, I would like to thank my wonderful father and mother for their unconditional support and having unwavering belief in me during this journey. I also wish to thank my sister and my brother-in-law for their support during this important phase of my life.

Table of Contents

List of Tables

List of Figures

Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Networks
ML	Machine Learning
DL	Deep Learning
ARIMA	Autoregressive Integrated Moving Average
SARIMA	Seasonal Autoregressive Integrated Moving Average
PARMA	Periodic Autoregressive Moving Average
ARCH	Autoregressive Conditional Heteroscedasticity
GARCH	Generalized Autoregressive Conditional Heteroscedasticity
KNN	K-Nearest Neighbours
DT	Decision Trees
SVM	Support Vector Machines
MLP	Multi-layer Perceptron
CNN	Convolutional Neural Networks
TCN	Temporal Convolutional Networks
DEM	Digital Elevation Model
SWAT	Soil and Water Assessment Tool
RNN	Recurrent Neural Networks
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
TFT	Temporal Fusion Transformer
DNN	Deep Neural Network
BPTT	Backpropagation Through Time
CEC	Constant Error Carousel
ReLU	Rectified Linear Unit
ELU	Exponential Linear Unit
SGD	Stochastic Gradient Descent
RMSprop	Root Mean Square Propagation
Adam	Adaptive Moment Estimation
GradNorm	Gradient Normalization
DAN	Distance-weighted Auto-regularized Neural Network
TRCA	Toronto and Region Conservation Authority

SCN	Spring Creek North
SCS	Spring Creek South
EDD	Etobicoke at Dixie and Derry
LED	Little Etobicoke at Dixie
WSC	West Spring Creek at Dixie
EH	Etobicoke at Hwy 410
EB	Etobicoke at Brampton
HL	Heart Lake
MWY	Mississauga Works Yard
RW	Rathwood
SGF	Sue Grange Farm
EQEW	Etobicoke at QEW
UVMH	Univariate Multi-Horizon
MVMH	Multivariate Multi-Horizon
PMVMH	Peephole LSTM-based Multivariate Multi-Horizon
SLF	Static loss function
ALF	Adaptive Weighted Loss function
NSE	Nash-Sutcliffe Efficiency
MAE	Mean Absolute Error
RMAE	Relative Mean Absolute Error
GIS	Geographic Information System
CYP	Cross-Year Prediction
YAML	Yet Another Markup Language

Chapter 1

Introduction

Managing water resources, mitigating extreme phenomena, e.g., floods or droughts, or conducting an environmental assessment of a policy or undertaking requires hydrological forecasting. While short-term predictions are vital for mitigating adverse extreme events, long-term forecasts support the assessment of the sustainability of a water-related policy or undertakings and general water management. Traditionally, process-based hydrological modeling was utilized to simulate precipitation transformation into flowing water, helping us understand and predict the water trajectories in a watershed. However, these process-based models must undergo site-specific calibration and periodic re-calibration of parameters, which, in turn, requires additional information on the watershed and its land use [1]. Conversely, data-driven machine learning models can extract knowledge and input-output relationships from hydrometric and environmental data without presumptions on physical processes. Additionally, the traditional hydrological models underperformed compared to their counterparts which are based on machine learning (ML) algorithms, due to the scarcity of morphometric data on a watershed.

1.1 Motivation

Hydrological forecasting in Canada, especially for urbanized watersheds, is vital for managing water resources, protecting infrastructures and ensuring public safety against natural phenomena like floods. In Canada, flooding has become the most prevalent occurrence which caused around $800 million per year [2] in insured losses in the past decade and over a few billion dollars when uninsured losses are added. Nearly one and a half million households are positioned in the flood-prone areas, however, only 15% of the homeowners pay a premium for protection against flood damage. Other than economic losses, flooding can trigger severe health concerns like drowning, getting hit by moving objects, or even getting hypothermia due to cold water and it also causes psychosocial impacts in terms of depression and stress disorder. The Greater Toronto Area (GTA) which is the most populous area in Southern Ontario,

experienced the most devastating flooding due to Hurricane Hazel in 1954 which resulted in 81 fatalities and over 7000 people lost their homes [3].

Flood risks in urbanized areas are expected to escalate due to frequent short-term intense precipitation events, eventually causing flash floods which might exceed the capacity of drainage systems in highly impervious watersheds. The most recent flash flood in Toronto was in July 2024, where within only 3 hours around 100 mm of rainfall caused a city-wide flood which accounted for approximately $1 billion in insurance loss [2]. Small urbanized watersheds are susceptible to flash flooding due to their impermeable surface which accelerates runoff causing prominent stream responses to precipitation. Therefore, the propagation of high flow events in natural streams can occur within minutes to a few hours and which underscores that reliable short-term predictions of water levels at high temporal resolution are essential for flood management teams of Canadian conservation authorities.

1.2 Problem Statement

To mitigate flash flood risk and protect water resources, this thesis addressed the primary challenge of developing accurate, reliable and operationally viable data-driven water level forecasting for urbanized watersheds. The reliable forecasts of the models aim to present the flood management and watershed authorities with timely insights from routine monitoring data alone. Existing forecasting approaches are used to generate predictions for a fixed lead time and location using daily and monthly average values of hydrometric characteristics. As land use and infrastructure change over time, a new model is required to accommodate these changes. The study aims to develop a framework for hydrological predictions with variable lead times of fine temporal granularity required for the assessment of hydrological conditions on a watershed, including extreme events, without the necessity to re-train the model for different settings. The model development and application are based solely on data supplied by routine monitoring systems.

Urbanized watersheds pose a far greater challenge for modeling and management because of their impermeable surfaces, which accelerate runoff, causing prominent stream responses to precipitation, quickly converting slow baseflow to stormflow, and steep hydrograph rises that can lead to flash floods. Capturing rapid increases in water level within minutes requires routinely monitored, high–temporal-resolution data that can resolve short response times, improve identification of peak timing, and support reliable real-time forecasting and early warning in small urbanized watersheds.

The core idea is to transform these routinely monitored high-resolution data so that a Deep Learning model can learn patterns to predict high-flow events several lead times ahead. An initial approach is to develop single-site univariate models for each cross-section in order to obtain site-specific error estimates. Due to the well-known ability of ANNs to approximate nonlinear curves and surfaces, the univariate models can learn the spatio-temporal hydrological conditions for each stream gauge effectively.

However, because a watershed may have multiple cross-sections, these single-site models should be applied to each one. In addition, although the univariate multi-horizon models can outperform multivariate multi-horizon models at an individual site because each station has its own local rainfall–runoff response and lag structure. Watershed approach to water resource management requires a clear picture of future watershed conditions to render an informative decision. The results of univariate multi-horizon models from multiple locations on a watershed must be postprocessed which adds an extra layer of the analysis and attracts extra time and computational resources during extreme events. In shared multivariate learning, the model must fit several sites at once, which requires a trade-off. However, the short-term forecasts of the watershed conditions are generated efficiently leaving more time for decision makers. Therefore, the study develops a framework supporting watershed-based approach using a multivariate model in which the entire watershed or interconnected sub-watersheds are treated as the fundamental unit of analysis, accounting for upstream–downstream relationships to capture spatial, temporal, and process interactions within the basin simultaneously across multiple lead times.

Even within a multi-site, multi-horizon framework, a key challenge is that the shared static loss function limits the reliability of models at individual sites. Hydrological regime at some cross-sections can be highly variable or sensitive to water load which largely influences the overall forecasting error, but may not get enough focus during training if all sites are weighted equally in the loss function. To address this challenge, this study introduces an adaptive weighted loss strategy which allocates proportionally estimated predictive errors to improve predictions of all modelled sites.

Building on these observations, this thesis develops a watershed-based data-driven modeling framework which advances from univariate multi-horizon forecasting to a comprehensive multivariate multi-horizon framework and further expands with an adaptive loss strategy to accommodate the diverse spatio-temporal hydrological dynamics across different cross-sections in small, urbanized watersheds.

1.3 Research Aim and Objectives

This study explores high-resolution, multi-site water-level forecasting in small, urbanized watersheds, focusing on capturing spatio-temporal dynamics at multiple sites across watersheds. It aims to develop a novel watershed-scale framework that extends the lead time of reliable forecasts on a watershed. To achieve this goal, the following research question should be addressed:

1. Compare and assess different modeling approaches such as univariate and multivariate multi-horizon models of different architectures, focusing on their accuracy, ability to generalize and scalability.

2. Develop a reproducible, scalable forecasting method utilizing data supplied by routine monitoring

systems to aid in flash flood mitigation and water management decision-making using adaptive loss strategies which enhance prediction accuracy through site-specific benchmarking.

3. Develop a high-resolution, multi-site forecasting framework that integrates water level and rainfall data to effectively capture the spatio-temporal dynamics of small urbanized watersheds.

Chapter 2

Related Works

2.1 Statistical Approaches in Data-driven Hydrological Forecasting

Hydrological processes have always been investigated based on observation data and statistical approaches used to find analytical expressions for dependencies between target variables and forcing functions. Thus, the regression approach was adopted to describe to rainfall-runoff relationships. Following the procedure of Box and Jenkins [4], the autoregressive integrated moving average (ARIMA) model was utilized in the hydrological domain to predict future observations using past data for streamflow and rainfall series [5]. However, the ARIMA model only works with stationary data and often fails when applied to hydrologic processes. To address the limitations of ARIMA, the seasonal ARIMA (SARIMA) accounting for seasonal differences [6], the deseasonalized ARMA model for removing the seasonal component [7], and the periodic ARMA (PARMA) model for varying the model's parameters depending on months [8] were introduced and applied for monthly streamflow prediction. Furthermore, streamflow depicts heteroscedastic variability, which means that it fluctuates with time, particularly during flood and high-flow events. In order to address this issue, models like ARCH (Autoregressive Conditional Heteroscedasticity) and GARCH (Generalized Autoregressive Conditional Heteroscedasticity) [9] were utilized on Yellow River streamflow data and they showcased superior performance against the ARIMA model in addressing the changing variance.

In hydrological forecasting, the variance equation used in ARCH and GARCH models dynamically adjusts forecast uncertainty at each lead time. The ARCH model responds to recent forecast errors, while the GARCH model keeps memory of past volatility, which the ARIMA model cannot do because it assumes constant variance. The advantages of the GARCH model against the constant-variance ARIMA models on daily streamflow data highlight that the GARCH model better captures the time-

varying variance during high-flow events [10].

Despite these advancements, the statistical modeling approach faces significant limitations for water-level forecasting in highly urbanized watersheds [11]. The reason behind this is that the ARIMA, ARCH, or GARCH type models are mainly univariate [12], focusing on a single time series at one gauge only which poses a challenge in a watershed level approach where multiple sites share spatio-temporal dependencies [13]. Therefore, increasing the number of local autoregressive models to cover an entire watershed is computationally expensive and operationally impractical. Another reason is that high-resolution water level forecasting accelerates error growth [14], with time errors compounding at each lead time prediction due to inaccurate peak time forecasting. That is why the focus has shifted towards deep learning-based architectures which learns hydrological dynamics from sequences and enhance performance across diverse regimes and multiple lead times.

2.2 Classical Machine Learning for Water-level Forecasting

Accumulation of large volumes of hydrometric and meteorological data in electronic form opened opportunities for expanding the set of computational techniques for data analysis to include machine learning. The earliest use of machine learning for hydrological forecasting involved a feedforward neural network for rainfall-runoff modeling without the utilization of analytical expressions describing hydrological processes [15]. In the late 1990s [16], applications of traditional machine learning algorithms such as K-Nearest Neighbours (KNN), Decision Trees (DT) for hydrological forecasting were reported in the literature. Around the same time, Support Vector Machines (SVMs) became well accepted in the 2000s for rainfall-runoff classification tasks, while Random Forest was utilized for water quality prediction. Researchers have investigated the use of SVM and kernel-based methods for short-term streamflow forecasting for reservoir inflow during the typhoon alert phase [17] which highlights the superior performance of non-linear kernels compared to linear models.

Early Artificial Neural Networks (ANNs) for hydrological tasks often had a single layer but tens of neurons [18], resulting in high computational cost and underfitting due to scarce data. ANN for streamflow forecasting in the Apure River was presented in [19] and captured hydrological events. The classical artificial neural networks (ANNs), mainly shallow Multi-layer Perceptrons (MLPs), sit between traditional machine learning and deep learning approaches. The researchers applied Multi-layer Perceptron (MLP) for daily streamflow forecasting and monthly runoff prediction, highlighting the ability of MLPs to approximate rainfall–runoff relationships better than linear regression. Further study highlighted that these MLP models generated reliable river flow predictions on daily to monthly timescales [20] but are less suitable for rapidly responding watersheds where flash flood events occur hourly or sub-hourly. Another limitation of machine learning models is that they lack interpretability

compared to physical models, and without the conservation laws, they could defy mass balance [21]. Nevertheless, neural networks attracted the attention of many researchers in the hydrological domain. In the year 2000, researcher have [22, 23] applied an artificial neural network to address real-world hydrological challenges. Both of the studies reflect early yet effective uses of neural network techniques for forecasting runoff and river stage in addition to rainfall data disaggregation. The neural networks represent an area in Artificial Intelligence (AI) that is modeled after the human brain [24].

Similar to traditional ML models, an ANN also requires specified lagged inputs and a memory length, as highlighted [25]. Additionally, the ASCE Task Committee [26] demonstrated that ANN can be very data-demanding and can be very difficult to implement successfully. Blind increase in hidden nodes can cause overfitting which reduces generalization. Lastly, the ANN models cannot explicitly utilize the sequential dependence and spatial interactions, causing the need for sequential modeling approaches that can inherently learn spatio-temporal dependencies from raw time-series data.

2.3 CNN-based Approaches for Hydrological Time Series

In contrast to traditional ML models, sequential deep learning models retain information from earlier time steps and learn temporal sequences and hydrologic memory directly from the data. It only needs the input window size, which refers to how much of the historical data the model can access, and the forecast horizon, which can be one step ahead or a multi-step prediction. Within this group of deep learning models, the Convolutional Neural Networks (CNNs) and Temporal Convolutional Networks (TCNs) have gained popularity for hydrological prediction with their efficiency in capturing temporal dependencies for stormflow generation and routing [29]. A deep CNN was developed to forecast flash flood by utilizing digital elevation model (DEM) precipitation inputs and observing that CNN performed better than SVM [27]. The TCNs are an extension of the 1D CNNs which are designed for time series forecasting that employs causal convolutions, uses dilated convolutions and residual blocks. For runoff forecasting the TCN and encoder–decoder based TCN models [28] demonstrated that the convolutional sequence approach can be effective to capture rainfall–runoff relationships. Another study demonstrated the effectiveness of adopting dilated convolutional sequence models for event-based hydrological prediction by implementing TCN to predict floods [29].

The Convolutional neural network is not able to retain long term memory, therefore, researchers have used CNN with LSTM model which is a hybrid approach for river flow prediction [30]. The study highlighted that the results of the hybrid model were similar to the Soil and Water Assessment Tool (SWAT) model, but in the wet season, it performed superior to SWAT. Further study with the CNN-LSTM-based approach was applied to streamflow forecasting on 35 natural watersheds in South Korea, generating superior Nash-Sutcliffe Efficiency results compared to other non-CNN models [31]. Later on,

the stacking-based ensemble models were used on the basis of different CNNs, such as 1D and 2D CNNs, to generate daily runoff in the Quinebaug River Basin in Connecticut [32]. Similar investigation demonstrated the advantage of using 1D CNN with LSTM, where CNN was able to process the long-term meteorological data, whereas the LSTM part was able to use the extracted features produced by CNN [33].

2.4 Recurrent Sequential Models (RNN, LSTM, Peephole LSTM) and Attention Mechanism

The recurrent neural networks (RNNs) were introduced to explicitly model temporal dependence by maintaining a hidden state that evolves over time. However, traditional RNNs are well known to suffer from the vanishing-gradient problem, which makes learning long-range dependencies difficult. Long Short-Term Memory (LSTM) networks address this limitation through gated memory cells that can preserve and selectively update information over long sequences, enabling them to represent long-term dependencies more effectively [34]. As the use of neural networks and deep learning grew with accessible computing, LSTM architecture was adopted for rainfall-runoff in 241 watersheds [35]. Subsequently, LSTM was utilized for forecasting river flows [36] and a later study [37] showed that the LSTM model performed superior compared to the classical models and was able to address spatio-temporal interdependencies in large datasets.

Further improvements were made in the research [38] where they applied an Encoder-Decoder based LSTM model with an attention mechanism for daily streamflow forecasting over Canadian watersheds, achieving higher performance with NSE (Nash-Sutcliffe Efficiency) and KGF (Kling–Gupta efficiency) for lead times up to 5 days. This study demonstrated that when the attention layers prioritized the most relevant past inputs, performance improved, indicating that the attention mechanism enhances RNN-based models by assigning weights to relevant temporal features, thereby addressing the dimensionality issue of large datasets. Additional study [39] demonstrated that even a single LTSM model is capable of large-scale flow prediction in ungauged basins. They trained the LSTM model on catchment descriptors and meteorological data from 148 gauged basins to forecast flow in ungauged basins. The results showed that LSTM models outperformed the traditional conceptually-based hydrological models due to LSTM's ability to capture spatio-temporal variability, however, they have a substantial drawback due to their black-box nature. As computational resources become available and affordable, the implementations of multi-layer LSTM models for water level forecasting were reported in scientific literature. A stacked-LSTM architecture by using two sequential LSTM layers [40] was implemented in order to achieve higher accuracy compared to a single-layer LSTM model for monthly flow forecasting. However, this superior performance of the multi-layer LSTM model was obtained due to higher computational resources. Now, when it comes to deep convolutional and graph-based models, they

address the spatial and temporal data jointly.

Despite LSTM and GRU (Gated Recurrent Unit) models improving temporal forecasting, their ability to capture long-range dependence and global interactions in the hydrological domain remained constrained. Therefore, attention mechanisms were introduced to overcome these limitations by allowing the models to focus on the most relevant features across large datasets. As self-attention became popular, it was also adopted in the hydrological domain. A transformer-based model is built in a way that it processes all timesteps simultaneously and acquires knowledge to assign importance to a certain part of the history. The first Temporal Fusion Transformer (TFT) model [41] in hydrology was applied to predicted the River Water Quality Index (WQI) using deep recurrent and transformer models, such as TFT, based on the WQI-SA dataset with 10,560 time-series instances for the Bhavani River. Subsequent research [42] highlighted that TFT model by combining LSTM recurrence with multi-head attention and the model was applied to streamflow data over 2,610 distributed catchments, demonstrating that TFT performed superior compared to LSTM and transformer models due to its ability to learn temporal relationships at multiple scales and offer interpretability through attention weights. Following study [43] implemented a Transformer model for 120-hour (5-day) flow prediction on 125 Iowa rivers using 72 hours of past historical flow data and due to the attention importance to capture hydrological patterns, the transformer model achieved higher median NSE and KGE than LSTM, GRU, and encoder–decoder seq2seq models.

A significant improvement of LSTM variation for hydrology is the peephole LSTM, where gates can view the cell state directly. In hydrologic time-series forecasting, this setup is often motivated by the need to align gating decisions with the changes in the internal state. This is particularly useful when the signal is sensitive to timing, such as predicting peak flow times in fast-response watersheds. Research was conducted using a cascade long short-term memory [44] a concrete hydrology study focusing on daily runoff forecasting. The study also highlighted the usefulness of the peephole connections in LSTM which helps the model to retain long term dependencies just by creating a direct connection between the memory cell and the gates.

2.5 Physics-informed Neural Networks and Domain-knowledge Integration

While data-driven models excel at forecasting, one major drawback is their lack of interpretability. In order to address this gap, physics-informed neural networks (PINNs) were introduced to integrate the hydrological domain knowledge directly into the learning process. A physics-informed machine learning is a framework where machine learning models are trained while explicitly integrating physical laws to improve the model accuracy and explainability. There are mainly two ways to integrate the domain knowledge into the deep learning models through their architecture. The first one is to introduce physical

constraints into the loss function of the deep learning model as a penalty term [45, 46] and the other option is to incorporate physical laws in the deep learning model architecture as a layer [47, 48]. Further study [49] developed a hybrid model called P-RNN that incorporates the EXP-HYDRO model as a layer within the RNN. The main purpose of this physics layer is to ensure that the inputs pass through it before the network makes predictions and after that, the output goes into the regular neural network layers. The authors compared four different models which are a physical NN model (P-RNN only), a common NN model (data-driven only), an LSTM model (data-driven), a hybrid DL model (P-RNN + NN layers) and demonstrated that the hybrid DL model combines the best of both worlds. The hybrid model was able to capture spatio-temporal patterns and adapt to diverse catchments, depicting the best performance out of the other three models. In another study, the researchers demonstrated the use of a weighted loss function called lbPINN [50] in order to solve the Navier–Stokes equations. The losses used in vanilla PINNs are mainly a weighted sum of data mismatch, boundary constraints or PDE residuals. The authors observed that manually fixing this loss may be problematic in the training phase, therefore, they propose a self-adaptive loss-balanced lbPINN that uses the Gaussian probabilistic model to create a loss function that is based on the maximum likelihood. The main purpose is that the model is able to know how much importance it can give to each constraint during the training phase. The results demonstrate that lbPINNs achieved better solution accuracy than vanilla PINNs with fixed weights.

2.6 Gaps, Limitations and Opportunities

The literature review reveals a significant gap in addressing flash flood events using high-resolution hydrometric data. Most of the study focused on generating hydrological forecasts on daily, weekly, or monthly timescales in large rivers. However, in small urbanized watersheds where water levels can rise quickly and response times are short, sub-hourly observations can play a key role in detecting flash flood events. Since deep learning models have been leveraged for daily or monthly streamflow prediction at a single gauge or a few gauges, there is a gap in high-resolution multisite forecasting for urbanized watersheds.

The advantage of univariate based LSTM model is that it can approximate well for an individual gauge. However, the model is not able to utilize upstream downstream spatio-temporal relationships since it leverages single-site observation [51, 52]. Multivariate multi-horizon model addresses the issues of univariate models by generating predictions for multiple gauges simultaneously, and only one model of this type can cover the forecasting for the sites available in a watershed, supporting scalability and operational use. The reviewed papers concentrated on medium or large basins with daily or monthly data [38, 53] and overlooked small urbanized watersheds with short travel times. Furthermore, even with multivariate modeling, there is a problem of imbalanced learning due to shared model representation. In

shared multivariate models, sites that exhibit different hydrological dynamics and their gradients conflict with each other [54], making the rapid-response site underrepresented during model training. Due to this issue of shared model learning, the rapid-response sites remained undertrained, posing a challenge for operational use in urbanized watersheds.

These gaps highlight a distinct opportunity for this thesis. To address these challenges, the study focuses on building multivariate LSTM-based models for small urbanized watersheds using sub-hourly water-level and rainfall data rather than daily [10, 19, 20, 38] or monthly [8, 40] data observed in the literature. The multivariate model will predict water levels across all stream gauges at multiple lead times, supporting the watershed approach. Furthermore, an adaptive weighted-loss strategy is introduced that dynamically adjusts site weights based on loss contribution scores during training, so the model learns more evenly.

Chapter 3

Background

3.1 Artificial Neural Networks (ANNs)

The machine learning term was first used in [55] when a program for the game of checkers was implemented by varying the evaluation function. The main idea was how a computer could learn or gain knowledge from experience and this process was referred to as machine learning. Later on, a simple perceptron for pattern recognition was introduced in [56]. Subsequently, the first learning rule was showcased in [57]. However, these earlier works just conceptualized networks of simple units but the multilayer perceptrons (MLPs) with hidden layers was proposed in the 1980s [58]. The neural network are modeled after the human brain and it processes large numbers of data for making predictions [56].

$$z = \sum_{i=1}^{m} w_i x_i + b \qquad (1)$$
$$y = \sigma(z)$$

where x_i denotes the i-th input, w_i is the corresponding weight, b is the bias, z is the weighted sum before activation, σ is the activation function, and y is the neuron output [61].

The neural networks are derived from linear regression and are based on the idea of multiple linear regression models with a non-linear activation function, multiple layers of neurons, or nodes, that are connected in a network. A neural network model consists of input data, neurons, weights, an activation function and an output layer. Typically, an ANN learns the mapping from input variables to output variables with the help of interconnected units which are referred to neurons. All the neurons receive one or more inputs with weights and bias and then pass the result through an activation function (1). A neuron can be activated using an activation function, and model weights are determined through an optimization problem with a loss function used as an optimization goal. The usage of a particular activation function depends on

the specific tasks. If no activation function or a linear activation function is used in a neural network, then the neural network becomes a linear regression model [59]. Similarly, when a linear regression is passed through a sigmoid activation function, then it becomes logistic regression [60].

In a feedforward neural network, the information moves from input layer to output layer in one direction which is called forward propagation. In the training phase the generated prediction values are compared with the observed values with the help of a loss function in which the error is propagated backward in the network in order to update the weights. This procedure is called backpropagation [61]. The limitation of the ANN is that, it generated predictions from input-output relationships while treating them as independent which is a major setback when dealing with sequential data of temporal order.

3.2 Limitations of Neural Networks in Time-Series Data

The feedforward neural network is called a memory-less model as it generally produces outputs from the inputs but does not store memory of the past time steps [62]. Since the neural network is not able to maintain temporal dependencies and assumes all inputs are independent, it is not able to work properly for sequential data. Firstly, the model can only utilize the previous historical data that was manually selected, resulting in a fixed sequence length. Secondly, the variable-length dependencies are hard to handle. Thirdly, the neural network does not inherently determine which specific past events in a sequence are responsible for causing an error observed much later in the sequence. The general deep learning models also exhibit the same problem but the specialized deep learning models [63] that deal with sequential data address this issue explicitly by maintaining a hidden state over time.

In handling large-scale datasets, shallow neural networks were not sufficient, leading to the rise of deep learning (DL). A deep neural network refers to a special type of ANN that consists of more than two hidden layers. The hidden layers in a Deep Neural Network (DNN) are fully connected and the information flows from the input layer through all hidden layers to the output. Whereas a Convolutional Neural Networks (CNN) is different from the typical DNN because it utilizes specialized convolutional layers. Instead of connecting every neuron to every other, it uses a convolution layer and feature mapping by detecting vertical edges, curves, or textures to generate spatial representations from input data.

CNN is used for image classification and object detection where it utilizes kernels to scan input data to identify patterns. The kernel is a matrix of learnable weights used to extract features from image data. The early foundational architecture for the CNN was introduced in [64] and later on, researcher built a convolutional network for detecting handwritten digits [65]. However, the term ‘deep’ was first used in early 2000 on the ‘deep belief networks’ [66], which was mainly based upon a greedy algorithm that can learn deep. Subsequently, the applications of Recurrent Neural Networks were described in [67], which is basically a type of neural network designed for sequences that have feedback loops in the recurrent

layer to maintain information in memory over time and is able to process data sequentially. However, in backpropagation, the gradient of the loss function decays exponentially which affects the network's ability to learn longer dependencies. In order to tackle this vanishing gradient issue, LSTM was developed in [68]. Long Short-Term Memory is a type of RNN which utilizes special units along with the standard units. LSTM uses a memory cell as a special unit that can store information in memory for a longer period than RNN. The Gated Recurrent Unit is a type of RNN has gating mechanism which is an architectural motif for controlling the flow of activation and gradient signals [69].

Despite the success of CNN and RNN-based models, their limitations in handling long-range dependencies led to the development of attention mechanisms. Attention allows models to dynamically focus on the most relevant parts of the input, improving both interpretability and performance. The invention of the attention mechanism leveraged the model's predictive ability by identifying the importance of different features. The main idea was that the decoder would generate the output sequence by dynamically focusing on relevant parts of the input sequence at each time step [70] and later on, the notion of attention mechanism was introduced in [71]. Based on this concept the transformer models work well without recurrence or convolution mechanism. One of the major components of the transformer is the multi-head attention which enables the model to capture global dependencies, focusing on the input sequence simultaneously.

3.3 Recurrent Neural Networks (RNNs) for Time-Series Data

In RNNs, the recurrent connection passes the data from one step to the next step which mainly acts like a hidden state that injects previous information to the model depicted in Figure 1 (a). In the RNN architecture, the neurons are connected in a loop so that the output from the previous block can be used as input for the next block (2).

A standard RNN can be written as

$$h_t = \sigma(W_x x_t + W_h h_{t-1} + b_h) \qquad (2)$$
$$y_t = \text{softmax}\left(W_y h_t + b_y\right)$$

where x_t is the input at time step t, h_{t-1} is the hidden state from the previous time step, h_t is the current hidden state, and y_t is the output. W_x, W_h, and W_y are learnable weights, while b_h and b_y are bias. The function σ is a nonlinear activation, and softmax is an activation function used in the output layer.

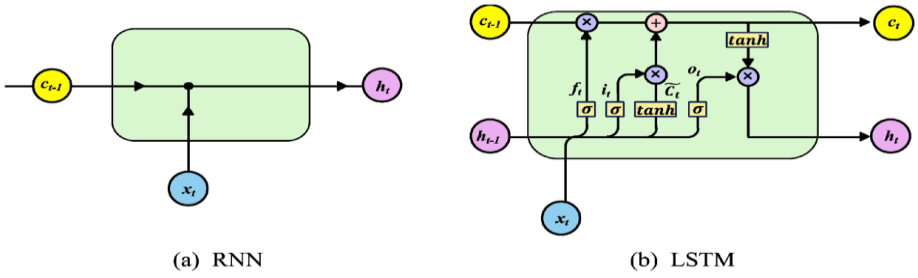
In time series data, the present value can depend on the previous historical data but not only on the current input. As the RNN can handle the sequential data one at a time and when the new information arrives, it updates the internal state, making the model better suited for time series forecasting than the feedforward neural network. However, the traditional RNN has a limitation when it is trained through

backpropagation through time (BPTT) [72], the gradient could shrink or grow drastically, leading to vanishing gradient problem. From the RNN architecture standpoint, with both sigmoid and tanh activation functions, after a range the gradient becomes zero during the backpropagation through time.

Studies have shown that as temporal gap increases the normal gradient descent is not well adapted to long term dependencies [73] and further work [74] justifies the reason behind this and provided solution with gradient clipping method [72]. Due to this motivation, the researcher developed a specialized version of the recurrent neural network called Long Short-Term Memory which retains long term dependencies in the memory.

3.4 Long Short-Term Memory (LSTM) for Time-Series Data

Long Short-Term Memory [68] is a type of recurrent neural network which uses special units called memory cells to store information for longer periods of time. The memory cell (3) allows the model to learn long term dependencies. The LSTM has three gates such as input gate, output gate and forget gate illustrated in Figure 1 (b). The flow of information in the network is controlled by these gates [63]. The gating mechanism helps prevent the vanishing gradient problem. The input gate controls how much information should enter, forget gate decides whether the information from the previous time step should be discarded or retained, and the output gate controls the flow of information from memory cell to the output.



(a) RNN (b) LSTM

Figure 1: RNN and LSTM Architecture

A standard LSTM cell is written as:

$$f_t = \sigma\big(W_f \cdot [h_{t-1}, x_t] + b_f\big)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh\,(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \qquad (3)$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * \tanh(C_t)$$

where i_t, f_t, and o_t are the input, forget, and output gates, c_t is the cell state, $\tilde{c}_t$ is the candidate cell content, h_t is the hidden state, σ is the sigmoid activation, $tanh$ is the hyperbolic tangent activation [63].

One of the limitations of the standard LSTM is that when dealing with non-segmented continuous inputs the network suffers from the internal cells growing indefinitely resulting in performance degradation. This issue can be solved by using an adaptive forget gate which will provide the LSTM cell to reset resulting in freeing up the resources [75]. Another problem with the LSTM is that the gates are not able to see the memory cell which results in forgetting important information. Researchers have solved this by creating the peephole connection [76].

3.5 Peephole LSTM for Time-Series Data

The standard LSTM demonstrates an architectural problem as the cell state known as Constant Error Carousel (CEC) does not have a direct connection to the gates. However, with peephole LSTM, the gates have direct connections to the cell state C_{t-1} (4), which means the gates can take a peek at the memory before deciding which information to keep or forget [76].

A peephole LSTM formulation is

$$\begin{aligned} f_t &= \sigma\big(W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f\big) \\ i_t &= \sigma(W_i \cdot [C_{t-1}, h_{t-1}, x_t] + b_i) \\ o_t &= \sigma(W_o \cdot [C_t, h_{t-1}, x_t] + b_o) \end{aligned} \qquad (4)$$

where f_t, i_t, and o_t are called forget gate, input gate and output gate respectively and from the equations (4) it is observed that the cell state is now accessible to all the. Compared with standard LSTM, all the gates have access to the cells state demonstrated in the equations.

In traditional LSTM, the gates receive information from the inputs and previous block's outputs but does not have a link with the CEC. Therefore, when the output gate is closed none of the gates will be able to access the cells output thus unable to gain control of the CECs [76], depriving the network of crucial information which can lead to undermining performance. Since LSTM can store long term dependencies, due to this limitation, the gates can be unaware of what important information to retain and what to forget. This limitation led to the birth of Peephole LSTM model architecture [76] which involves integrating the direct connections from the cell state to the gates, allowing the gate layers to look at the cell state. Therefore, the gates now do not have to rely only on the input and the past hidden state, as well as the internal state, by establishing the peephole connection. This model improves the network's ability to count timing, and the researchers suggested using this approach in tasks which rely on internal timing to affect gating decision.

3.6 Model Configuration and Concepts

In order to build effective sequence models, focus has been placed on the following best practices for constructing the model architecture and how the forecasting problem is configured. The main task is to determine whether it is univariate or multivariate, and based on the model settings has to be established. A model forecasts future values for only one target in the univariate settings whereas in multivariate approach, the model utilizes multiple inputs to predict several outputs [77]. Furthermore, the time delay embedding is very vital for time series forecasting as it defines the historical or past information provided to the model for forecasting. However, the forecast horizon tells the model how much into the future it can predict [78].

Building an appropriate sequential deep learning model also requires understanding of the model configuration, including the number of layers, number of neurons, learning rates, number of epochs, appropriate activation function, optimizer and the loss function. The first step in making the model is determining the model size. Fewer layers and neurons may lead to model simplicity (underfitting), while too many layers and neurons may result in an overcomplicated model (overfitting). A good practice for any deep learning model is the trial-and-error approach. It is recommended to start with one or two LSTM layers with less than 200 units per layer and only expanding the number of layers and neurons if the model underfits [79]. The next step is to choose appropriate activation functions e.g., ReLU (Rectified Linear Unit) [80], tanh, or sigmoid. However, due to the dying ReLU problem where the negative input signals cause the neurons to become inactive [81], the use of ELU (Exponential Linear Unit) [82] or Swish [83] activation functions is recommended instead. It is also important to prevent the model's overfitting. Dropout is one of the approaches to prevent the model from overfitting [84]. It is done by randomly deactivating between 20% and 50% of neurons. Setting a learning rate is the most challenging task at the training stage, as if it is too high, the training will diverge and if it's too low, then the model will get stuck in local minima. With the Adam optimizer, experimenting normally begins with the learning rate at 0.001 and adjusting within the range of 0.0001–0.01 [84]. Another issue that must be resolved includes the batch size. A small batch size of 8-32 samples may cause noisy updates or may improve model generalization. Conversely, a large batch size of 64-256 samples can help the model train faster but also can cause overfitting [85]. The number of epochs is another important factor. Too few epochs can cause the model to underfit, while too many can make it overfit [85]. Therefore, an early stopping mechanism is utilized to stop the model from underfitting and overfitting. For regression tasks, mean squared error (MSE) and mean absolute error (MAE) are the most popular ones, for classification tasks, the cross-entropy function is preferable.

The utilization of an appropriate optimizer [86] is also crucial for efficient model development. The optimizer mainly determines how weights are updated during the training phase. The popular

Optimizers are SGD, RMSprop and Adam (Adaptive Moment Estimation). The Stochastic Gradient Descent (SGD) stemmed from Robbins–Monro algorithm [87] is a very simple yet effective optimizer that uses the gradient directly but requires careful tuning of the learning-rate [86]. The second one is the Root Mean Square Propagation (RMSprop) which rescales the updates by maintaining a moving average of squared gradient [86]. Finally, the Adam [86] optimizer is the most popular of them due to its adaptive capabilities, while RMSprop can also be utilized for RNNs to prevent the vanishing gradient problem. The Adam optimizer calculates the running estimates of both the gradients and their squared values so that the training coverage is faster.

Recently, the dynamic and adaptive learning paradigms imitating human-like learning were introduced. When a fixed loss is distributed across multiple targets, the optimization is inclined towards the stronger gradients, leading to under-trained performance on some targets [88]. Therefore, dynamic weighted loss strategies can be utilized to mitigate this issue. The Dynamic Weighted Loss methods used in GradNorm (Gradient Normalization for Adaptive Loss Balancing) [88] extend these concepts to the multi-task problem, wherein the adaptive weighting between multiple tasks or targets facilitates balanced learning. In parallel, novel architectures like Distance-weighted Auto-regularized Neural network (DAN) with an adaptive loss function [89] were proposed for long-term forecasting with the ability to adapt to rare extreme events. The purpose of the Distance-weighted Auto-regularized Neural Network (DAN) model is to adjust the loss weights dynamically in order to capture high-flow events like floods. The authors utilized a “polar” representation of the time series by using distance-weighted multi-loss training and in the multi-loss approach, the model assigns weights to different loss terms by using a Gaussian probabilistic model.

Chapter 4

Methodology

4.1 Research Methodology: Machine Learning Life Cycle

Machine Learning (ML) life cycle [90] refers to the interconnected series of processes that help practitioners standardize the development, deployment, and maintenance of models to achieve reliable forecasting for a specific business goal. The ML life cycle is an iterative process that involves developing, testing, and refining a forecasting model through repeated cycles of problem formulation, data preparation, modeling, and evaluation. The research methodology in this study is inspired by the ML life cycle which consists of six stages (Figure 2).

The first step is to understand the problem domain and define the goal in order to properly articulate the importance of the problem. This study addressed the development and investigation of a multi-site forecasting framework that integrates water level and rainfall data for water level forecasting. The second step refers to dataset selection which depends on the availability of the data and also the type of problem. The third step corresponds to data preprocessing, where the data quality checks are done through addressing missing values, temporal gaps, and anomalies caused by sensor issues, aggregation etc. The subsequent step refers to the transformation. In this stage, according to the nature of the problem, different transformation techniques can be used, including label encoding, scaling, time embedding, etc. The next step is the modeling process, where the appropriate machine learning model type is determined. The sixth step is the evaluation step for measuring the model's forecasting reliability and operational trustworthiness using appropriate metrics. This stage is very important because it enables targeted iteration to revisit previous stages whenever the reliability drops. The practitioner can then rethink domain assumptions, adjust feature selection, improve preprocessing, refine embedding choices, or modify model structure and training strategy until a stable forecasting system is achieved.

Figure 2: Methodology

4.2 Study Area and Monitoring Network

A watershed is an area of land which catches precipitation and drains it into a common outlet point, such as a river, wetland, or ocean. Toronto and Region Conservation Authority (TRCA) manages 9 watersheds across the Greater Toronto Area (GTA) including, Etobicoke Creek, Mimico Creek, Humber River, Don River, Highland Creek, Petticoat Creek, Rouge River, Duffins Creek, and Carruthers Creek. Etobicoke Creek watershed was selected for this study due to its highly urbanized watershed with a quick hydrological response to water load, leading occasionally to flash floods during the warm season, with the air temperature consistently above zero. This makes it an ideal case to evaluate short-term, high-resolution water level prediction models. Located on the western side of the Toronto and Region Conservation Authority's (TRCA) jurisdiction, the Etobicoke Creek watershed is nearly 211 km² in size.

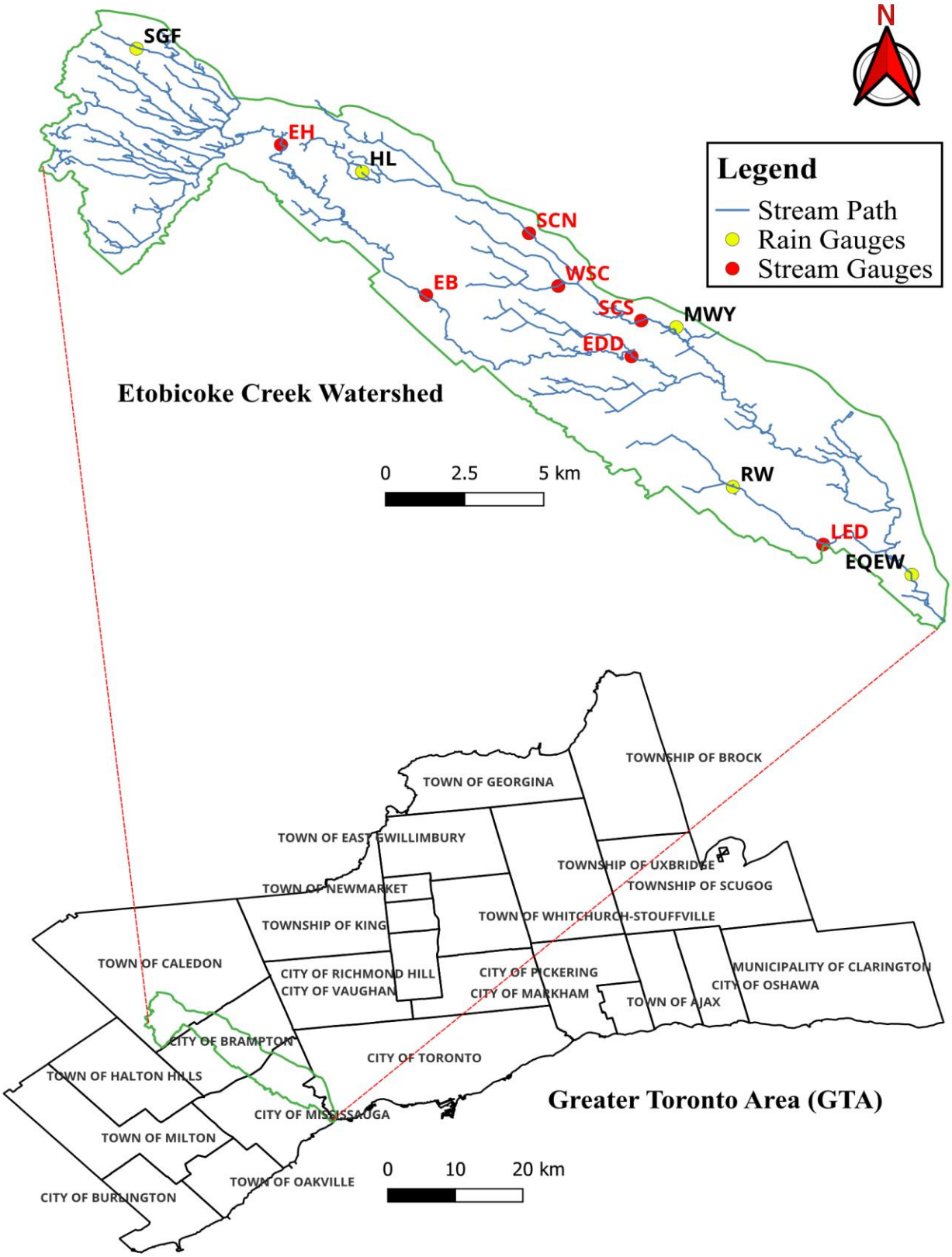


Figure 3: TRCA Stream Network

It begins in Caledon on the south slope of the Oak Ridges Moraine, travels south through Brampton and Mississauga, and drains its water into Lake Ontario. According to [91], the watershed consists of 68% of the urbanized area, 27% of the rural landscape, and 5% transitioning environment. Due to the high level of urbanization, the area is vulnerable to flash floods.

Toronto and Region Conservation Authority (TRCA) manages the routine hydrometric monitoring system and collects data from the watersheds. The focus of this is the Etobicoke Creek watershed, which includes several stream gauges along the main channel and its tributaries (Figure 3). The red markers in the Figure 3 indicate stream gauge locations which include Spring Creek North (SCN), Spring Creek South (SCS), and Etobicoke at Dixie and Derry (EDD), Little Etobicoke at Dixie (LED), West Spring Creek at Dixie (WSC), Etobicoke at Hwy 410 (EH), and Etobicoke at Brampton (EB). The yellow markers represent rain gauge locations for measuring the rainfall data at Heart Lake (HL), Mississauga Works Yard (MWY), Rathwood (RW), Sue Grange Farm (SGF), and Etobicoke at QEW (EQEW). However, the EQEW station's data are not regularly maintained, which is why it is not included in the research.

Table 1: Land Use Change of Etobicoke Creek Watershed

Land Use Category	**2002**	**2012**	**% change (2002–2012)**	**2019**	**% change (2012–2019)**
Urban	53.4%	56.4%	+5.6%	59.5%	+5.4%
Rural	32.5%	30.9%	−5.0%	28.2%	−8.5%
Natural	14.1%	12.7%	−9.6%	12.3%	−3.4%
Impervious Cover	42.9%	45.6%	+6.3%	47.9%	+4.9%

Etobicoke Creek watershed is a heavily urbanized watershed and due to its impermeable surface, accelerates runoff, causing prominent stream responses to precipitation. It is evident from Table 1 that, over the span of 17 years, the Etobicoke Creek watershed became more urbanized as the land use increased from 53.4% to 59.5%, and impervious cover jumped from 42.9% to 47.9% while rural and natural areas declined, reflecting urban development and land conversion, which increases runoff generation [92]. The Etobicoke Creek watershed is divided into eight sub-watersheds (Figure 4), including Headwaters, Spring Creek, West Branch, Little Etobicoke Creek, Lower Etobicoke, Main Branch, Tributary 3, and Tributary 4.

Impervious cover increased between 2002 and 2019 within the sub-watersheds of the Etobicoke Creek watershed and the change is reported in Table 2. The sub-watershed with the lowest impervious

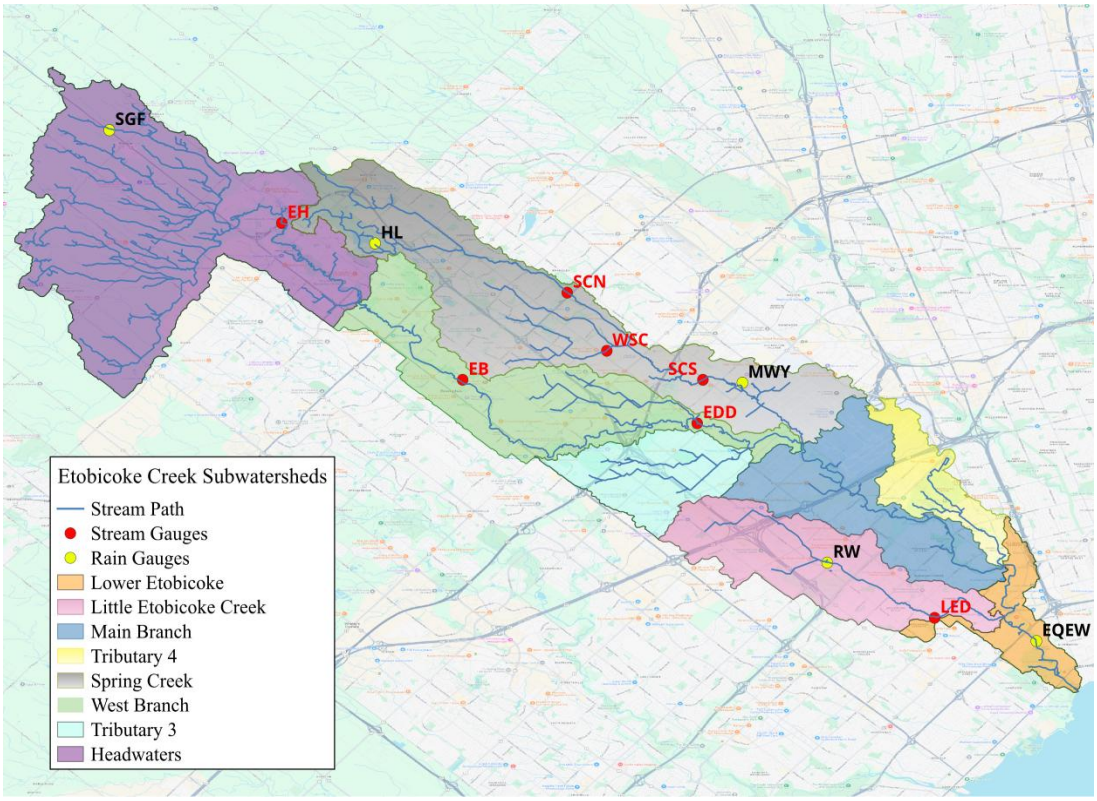


Figure 4: TRCA Sub-watersheds

Table 2: Impervious cover by sub-watershed for 2002, 2012, and 2019

Sub-watershed	Impervious Cover		
	2002	**2012**	**2019**
Headwaters	10.0%	11.9%	14.2%
Spring Creek	46.6%	50.3%	54.1%
West Branch	57.2%	60.0%	61.2%
Tributary 3	57.1%	65.1%	66.8%
Tributary 4	48.4%	50.4%	51.4%
Main Branch	57.5%	59.5%	61.9%
Little Etobicoke	64.5%	66.8%	68.7%
Lower Etobicoke	65.0%	64.9%	65.7%

cover at 14.2%, is Headwaters, where the Little Etobicoke remained the most impervious with 68.7% [92]. However, the largest relative increases in impermeable surfaces occurred in already urbanized sub-watersheds, specifically Tributary 3 (66.8%) and Spring Creek (54.1%). Since these sub-watersheds exhibit heterogeneous characteristics and the stream and rain gauges are distributed across them, a watershed-level modeling approach is more suitable than treating each gauge in isolation.

4.3 Watershed Approach Framework

Since 1946, when Canada adopted the Conservation Authorities Act, managing natural resources, namely water, land, and their ecosystems has been conducted following the watershed approach supporting compromise between preservation of the resources, economic and social needs [93]. This approach requires comprehensive analysis of the watershed's current condition and their potential changes due to natural and anthropogenic factors. Computation techniques became an integral component of such analysis in support of management decisions or policy evaluation. The study is aimed at developing a computational framework in support of the watershed approach. The framework leverages time-delay embedded stormflow water level values from multiple sites to generate reliable multi-horizon forecasts across an entire watershed or a sub-watershed. The workflow of the analysis is presented in Figure 5.

4.3.1 Data sources and Resolution

The location of the stream and rain gauges operated by TRCA provides hydrometric data [94] depicted in Figure 3. Recorded observations from all available monitoring sites were used in the study. Rainfall measurements are conducted every 5 minutes by rain gauges over Heart Lake (HL), Mississauga Works Yard (MWY), Rathwood (RW) and Sue Grange Farm (SGF). Stream gauges estimate water levels every 15 minutes over Spring Creek North (SCN), Spring Creek South (SCS), and Etobicoke at Dixie and Derry (EDD), Little Etobicoke at Dixie (LED), West Spring Creek at Dixie (WSC), Etobicoke at Hwy 410 (EH), and Etobicoke at Brampton (EB). The available stream and rain gauges for each year are reported in Table 3.

Table 3: Year-wise Availability of Hydrometric Monitoring Stations (2013–2020)

Year	**Available Stream Gauges**	**Available Rain Gauges**
2013	SCN, SCS, EDD	HL, MWY, SGF
2014	SCN, SCS, EDD	HL, MWY, SGF
2015	SCN, EB, SCS, EDD	HL, MWY, SGF
2016	SCN, EB, SCS, EDD	HL, MWY, SGF
2017	SCN, EB, SCS, EDD, LED	HL, MWY, RW, SGF
2018	SCN, EB, SCS, EDD, EH, LED	HL, MWY, RW, SGF
2019	SCN, EB, SCS, EDD, EH, WSC, LED	HL, MWY, RW, SGF
2020	SCN, EB, SCS, EDD, EH, WSC, LED	HL, MWY, RW, SGF

4.3.2 Model Development

The Universal Approximation Theorem [95] suggests that there exists a deep neural network which approximates sufficiently well complex non-linear relationships in data. Given the high variability of water

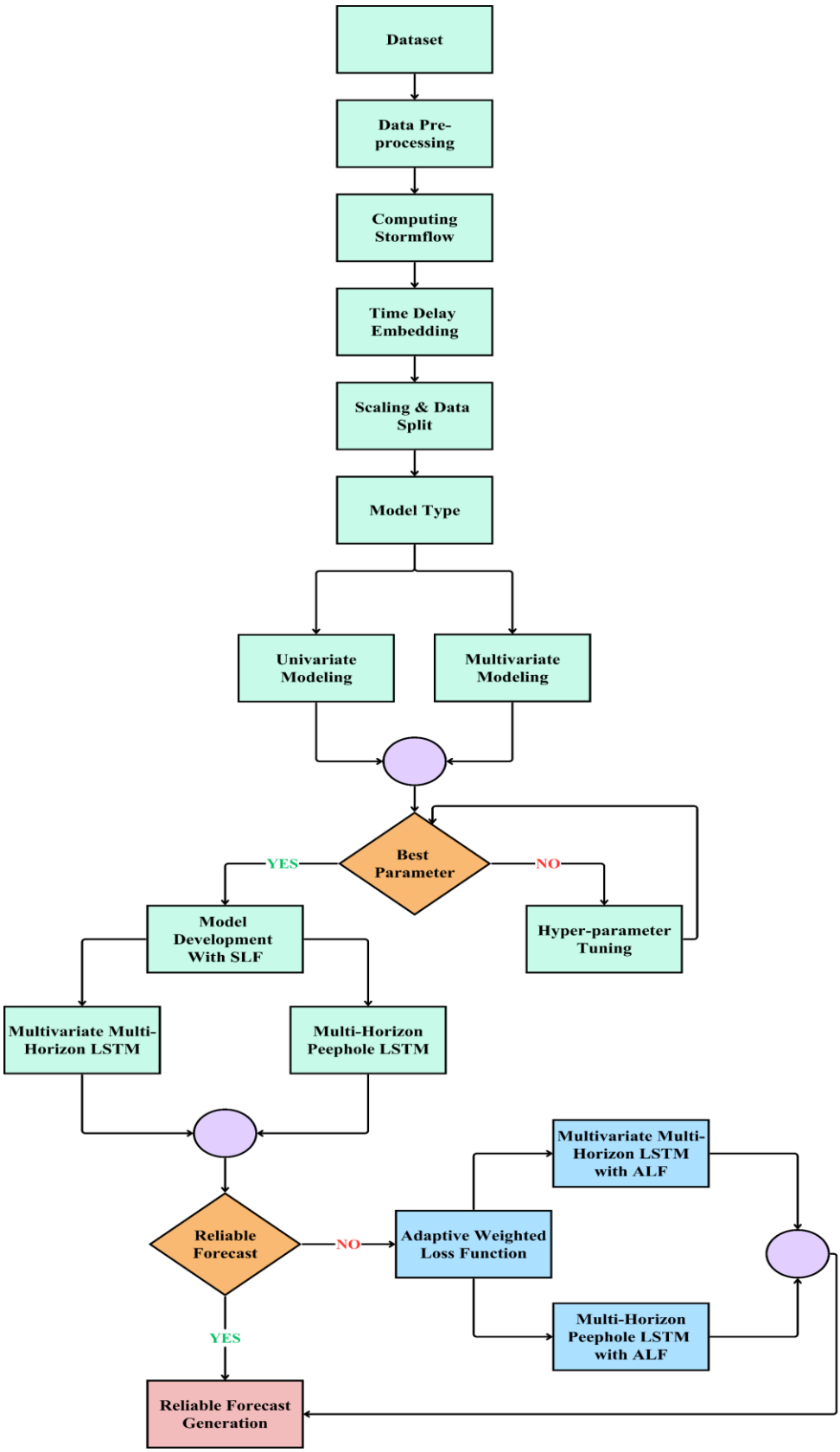


Figure 5: Workflow

level data during flash flood events and the goal of accurate approximation as opposed to explainability of the results, ANNs were expected to outperform classical ML algorithms. LSTM architecture of ANN has proven to be efficient in modeling time series data. This architecture was select for developing deep-learning models. Based on this architecture, the univariate multi-horizon (UVMH) models represent the first type which is mainly developed to obtain site-wise loss contribution which is used to initialize the site-specific model weights for adaptive approach. The model features consist of observation data collected from a single cross-section with all available rain gauges (Table 3), and the model generates a multi-step output vector of stormflow water-level predictions for that cross-section at increasing lead times. The second model type, multivariate multi-horizon (MVMH) was represented by the models that utilize data from all stream and rain gauges (Table 3) and produces a multi-step output vector of stormflow water-level predictions for all the cross-sections at once for all. These two model types were developed using the shared static loss function (SLF), defined as Mean Absolute Error. However, with the introduction of the adaptive loss strategy (ALF), the third model type is constructed which is similar to the second model type, but the loss function is different.

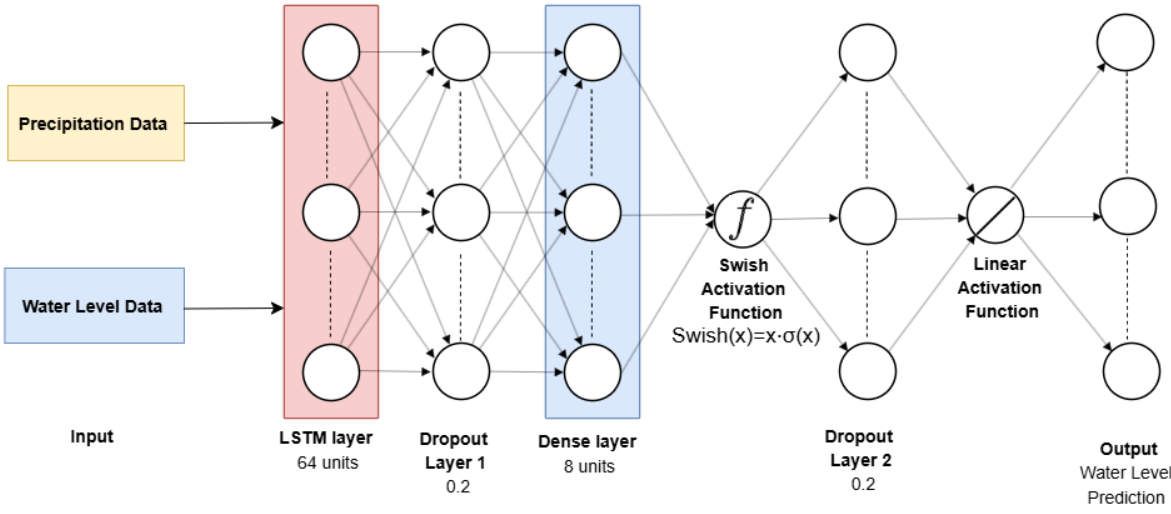


Figure 6: Multi-horizon LSTM-based model architecture

4.3.3 Model Architecture

Two variants of the LSTM model family were developed to model stormflow dynamics in this study. The first one is the standard LSTM model, and the second one is the LSTM with peephole connection, which is called a peephole LSTM model. However, only the standard LSTM-based model was used in the univariate setting while both the LSTM and peephole LSTM were utilized for multivariate multi-horizon model configuration (Figure 6). Both univariate and multivariate models (Figure 6) were designed following a

compact multi-layer architecture with controlled capacity. All models consisted of input, LSTM, dropout, dense layers, and output layers. The input layer represents the values of the model features with integrated time-delay embedding. The input data are processed by a single LSTM layer. The LSTM output is passed through a dropout layer with a drop rate of 20%. A fully connected dense layer and a swish activation function follow. Another dropout layer with 20% regularization is applied to the dense outputs. The network concludes with a linear output layer that contains $N*h$ neurons for multivariate prediction, where N is the number of modeled locations and h is the length of the prediction horizon in time steps. Each output neuron receives input from all dense units. In total, the model contains two hidden layers with trainable parameters, together with a trainable input mapping and a trainable output layer. In the case of multivariate prediction, the output layer contains $N*h$ neurons, whereas for univariate prediction, it contains h neurons.

LSTM is a type of recurrent neural network that works well for sequential data, where long-term dependencies come into play, and its memory cell helps the network to learn those dependencies. The multivariate multi-horizon LSTM model takes the stormflow component of the water level data and precipitation data (Table 3) as input vectors, then in the output, the shared LSTM backbone generates a vector containing future values for multiple lead times across one or more cross-sections. The control flow of the data from the input to memory cells is determined by the input gate. The forget gate decides whether the information from the previous timestep should be retained or forgotten. Finally, the output gate controls the data flowing from the memory cell to the output. The LSTM model was chosen over RNN due to the RNN's vanishing gradient problem. LSTM addresses this issue by having the memory cell and a gating mechanism.

In water level forecasting, where high-flow events play a crucial role, the LSTM is effective at retaining long-term dependencies, such as prior precipitation conditions, thereby producing reliable results. However, in certain cases, the standard LSTM might not produce good performance, since the gates are unaware of the information in the cell state. Since the gates in the standard LSTM cannot see the content in the memory cell directly, the model is unable to decide when to forget, causing it to discard useful information, such as high-flow events. Another aspect is that without observing the information in the current cells, the forget gate might be unable to comprehend if a certain pattern has been completed, which could lead to retention of irrelevant information for too long, like keeping the long baseflow periods. Lastly, the LSTM model may underperform for sites that demonstrate precise counting over long sequences due to the fact that the gates are unable to make a decision based on the current cell state. Therefore, a variant of the LSTM model called peephole LSTM was also utilized to compare its effectiveness with LSTM in cross-section, where the time-sensitive memory retention is vital.

The Peephole LSTM model architecture addresses the limitation of the standard LSTM by establishing a peephole connection from the memory cell to the gates. This model can be very effective in

water level forecasting since it allows the gating mechanisms to directly access the internal cell state, which could lead retain relevant high-flow events. The peephole connection in the network can help in learning more precise timing and maintain memory over longer periods, resulting in improved performance. With the help of peephole connection, the model may leverage the accumulated state of the watershed by peeping at the current cell state, which could lead to a higher chance of detecting peak flow magnitudes during extreme events. Furthermore, the direct connection also helps in stabilizing the gradient flow during backpropagation which makes the model converge quickly. Lastly, by accessing the internal memory the peephole LSTM can better model the sites where water levels can rise quickly within short response times, compared to conventional LSTMs.

4.3.4 Adaptive Weighted Loss function (ALF)

In multivariate modeling, standard loss functions, e.g., mean absolute error, integrate the model's errors from all investigated locations and modeled horizons (5). These overall low error values are achieved due to very small errors at some locations and high errors at others. Since the loss of all sites are combines into a total objective, the shared multivariate LSTMs weight is updated using the sum of site-wise gradients [54, 96]. To balance errors from all sites, a weighted loss function can be employed in the training phase to allocate different weights to different sites:

$$SLF_k = \frac{1}{H}\sum_{h=1}^{H} \left| y_{t,i}^{(h)} - \hat{y}_{t,i}^{(h)} \right|$$

$$SLF = \frac{1}{K}\sum_{k=1}^{K} SLF_k \qquad (5)$$

where $y_{t,i}^{(h)}$ is the true value for site i at lead time h and $\hat{y}_{t,i}^{(h)}$ refers to predicted values, *H* is the size of prediction horizon in timesteps, and *K* is the number of monitoring sites.

At epoch e, the loss-weight vector:

$$\mathbf{w}^{(e)} = \left[w_1^{(e)}, \dots, w_N^{(e)}\right], \qquad w_i^{(e)} \geq 0, \qquad \sum_{i=1}^{N} w_i^{(e)} = 1 \qquad (6)$$

Then the Adaptive weighted Loss at epoch e is:

$$ALF = \sum_{i=1}^{K} w_k^{(e)} SLF_k \qquad (7)$$

where K is the number of investigated sites, SLF_k is the static loss function evaluated at site k, w_i is its learnable weight.

Among the weighted loss strategies, the static approach and adaptive approach are quite popular. The weights are manually set and remain unchanged in the static approach [97] which often leads to decent performance. In the adaptive approach, the weights are dynamically changed for each epoch to balance task learning dynamics [88].

In the proposed epoch-wise adaptive weighted loss strategy, the weights of the components of the loss function are periodically updated based on validation performance. The adaptive weighted loss strategy is a dynamic approach to balance the learning contribution from each site during the training phase (Figure 7). The initial weights of the MVMH model loss function are obtained from the metric values of site-specific, separately trained univariate models. Specifically, the metric values are averaged across all lead times for each cross-section to obtain its loss contribution score. These site-specific average metric values are then normalized (6) so that they sum to 1, supporting proportional learning emphasis across all sites. This process ensures that sites with highly variable dynamics in the univariate setting are given greater priority at the start of MVMH model training.

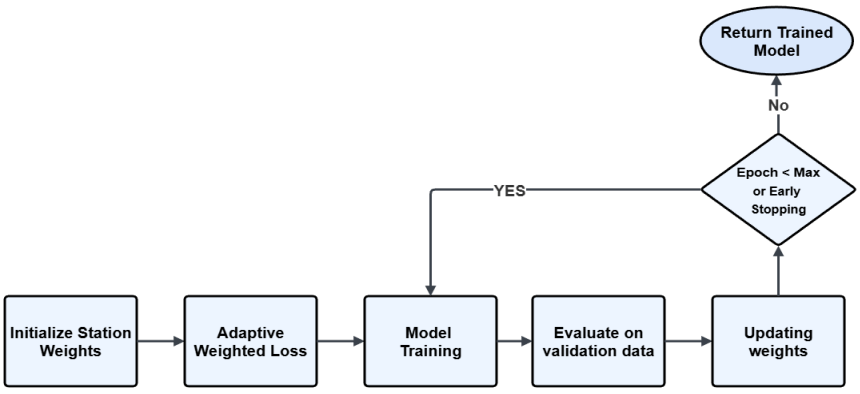


Figure 7: Training with Adaptive Weighted Loss

Adaptive weighting is not limited to the initial assignment as the loss weights are updated after every epoch using validation feedback. In each epoch, the model calculates the SLF (5) for each location and obtains the weights before applying formula (7). In the training phase, the Adam optimizer updates the model parameters on every mini-batch by utilizing the gradient of the weighted loss. After each epoch, the MVMH model performance is evaluated on the validation dataset and the validation performance is computed for each site separately. Subsequently, the validation errors are normalized and then utilized as adjusted sites' weights. Therefore, the site that generated higher errors during validation receives higher weights so that the model provides more attention to minimizing errors there in the next epoch.

This guarantees that no site will be ignored and also prevents model overfitting to sites that are easy to model. As this is not a gradient-based approach, in its feedback loop, the weights are updated externally, allowing the model to adjust errors on rapid-response sites. Over time, this adaptive approach directs the model toward balanced convergence across all sites and it is repeated iteratively until early stopping or the maximum number of epochs are reached. In this study, both the MVMH and PMVMH models with ALF were utilized to assess the performance and the reliability improvement on the rapid-response sites compared to their SLF variants.

4.4 Evaluation Metrics and Reliability Assessment

4.4.1 Nash-Sutcliffe Efficiency (NSE)

The majority of practitioners utilize the Nash-Sutcliffe Efficiency (NSE) index to evaluate overall model performance [96]. The NSE metric mainly calculates the ability of predictions to fit the observed data. Adjusted for its application to the investigated case study, it was evaluated according to the formula (8):

$$NSE = 1 - \frac{\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}{\sum_{t=1}^{n}(y_t - \bar{y})^2} \quad (8)$$

where y_t represents the observed stormflow water level at time *t*, $\hat{y}_t$ denotes the predicted one at time *t*, and $\bar{y}$ is the mean observed stormflow water level over the study period, *n* is the number of elements in an assessed set. The NSE index is oriented towards mean values and is typically less sensitive to extreme values, which are important for flood forecasting.

4.4.2 Mean Absolute Error (MAE)

To complement NSE, the Mean Absolute Error (MAE) was also computed to provide a more robust evaluation:

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t| \quad (9)$$

The MAE metric does not apportion the error to the magnitudes of the target variables. Therefore, a relative error metric was required to provide important information about the model's performance.

4.4.3 Relative Mean Absolute Error (RMAE)

To address this limitation of MAE, the Relative Mean Absolute Error (RMAE) was used in this study. RMAE metric apportions the error according to the scale of the observational data so that a fair comparison is possible when considering all sites with differences in elevations and rates of flow:

$$RMAE = \frac{\sum_{t=1}^{n}|y_t - \hat{y}_t|}{\sum_{t=1}^{n}|y_t|} \quad (10)$$

By incorporating both absolute and relative error metrics, the evaluation ensures that model performance is assessed not only in terms of overall accuracy but also in its ability to capture extreme water level magnitudes that are critical for informative decision-making on a watershed.

4.4.4 Reliability Assessment

Each model's performance was evaluated based on NSE, MAE and RMAE metrics. The NSE metric is used to assess whether the model forecast is able to capture temporal variability or not, while the error metrics assess error magnitude. Since MAE does not account for scale differences, RMAE which normalizes error by observed magnitude, was utilized in this study to compare error magnitude across sites. Model's performance was considered 'very good' if the NSE estimate on the testing set was at or exceeded 0.80 and 'good' when the NSE index was at or above 0.7. The RMAE values were compared with a reliability threshold of 20% used in practice. The predictions generated with RMAE estimates exceeding this threshold were deemed unreliable. In heterogeneous urbanized watersheds, a model can generate good NSE values while having higher error magnitudes, making the model operationally unreliable. Therefore, the objective of this study is to maximize the reliable lead time as much as possible while maintaining RMAE at or below 20% at each cross-section.

4.5 Software Selection

All the computational experiments were conducted on Paperspace with RTX 5000, and the code is written in Python. Various Python libraries, like NumPy and Pandas, were used for data handling and preprocessing, Matplotlib and Seaborn were leveraged to generate visualizations. TensorFlow and Keras were primarily utilized to build the deep learning-based LSTM models. The MLflow tool [99], which is an open-source platform for managing the entire end to end machine learning lifecycle, including experiment configurations, model weights were used to track, log, manage and compare the various model configurations for hyperparameter tuning. Grammarly [100] was used for spelling, grammar checking and writing enhancement during the thesis writing. Lastly, Geographic Information System (GIS) software called QGIS [101] is utilized to create maps and insert maps with TRCA stations (Figures 3 and 4).

Chapter 5

Computational Experiments

5.1 Data Pre-processing

The time series of monitoring data were analyzed. A missing value was set to zero for gaps in the precipitation dataset if the surrounding values were zero. The local mean of non-zero values nearby was used to maintain the hydrograph form for gaps in the water level dataset and precipitation data. Due to the difference in granularity of precipitation and water level readings, the precipitation data were aggregated to 15-minute intervals. All time series for each investigated year were synchronized.

5.1.1 Computing Stormflow

Water level values are measured relatively to an established reference point. Since monitoring sites have different elevations, differences in water level values do not reflect actual water depth at each cross-section making the comparison of water level values across monitoring sites impossible. Therefore, stormflow components of the water level values were derived and used in computations instead of raw water level values. The stormflow values were estimated by subtracting baseflow values from the observed values. The baseflow for each cross-section was estimated as the lowest recorded water level in the absence of precipitation throughout the study period. The stormflow values emphasize the hydrologically relevant high-flow behavior in response to water loads and were further used as features in computational experiments.

5.1.2 Time Delay Embedding

Given the continuity of the rainfall-runoff processes, time delay embedding is required to reconstruct the dynamic state space of a watershed. The time window reflects how much the most recent history may contain useful information for predicting future values. The actual size of the time window depends on the characteristics of a stream and its watershed. Its size can be determined by the analysis of watershed

morphometric characteristics and be based on exploratory computational experiments. Two sets of experiments with two-hour and three-hour embeddings, were tested while maintaining a 3-hour prediction horizon in computational experiments.

5.1.3 Scaling and Data Splitting

Each yearly dataset spans from April 17 at 11:45 am to November 30, 23:45 pm, yielding 21,818 synchronized data samples. The temporal integrity of hydrological processes was preserved by partitioning the data chronologically. Initial 10,000 observations (Apr 17–Jul 30) comprised the training set, the following 4,000 observations (Jul 30–Sep 10) were reserved for the validation set, and the last 7,822 observations (Sep 10–Nov 30) were left for the testing set. Feature scaling using the Min-Max scaling procedure was applied to transform the data to maintain relative magnitudes within each variable, which is critical for stormflow forecasting.

5.2 Experimental settings

According to the univariate multi-horizon (UVMH) approach, separate LSTM based networks were trained for every cross-section from 2013 to 2020. A total of 39 UVMH was developed to cover the entire watershed and provide all site-wise error magnitudes for the Adaptive weighting approach. Conversely, the multivariate multi-horizon LSTM (MVMH) and peephole multivariate multi-horizon LSTM (PMVMH) models under standard loss function were able to generate predictions for the entire watershed, requiring only one model per year. Lastly, the MVMH and PMVMH under Adaptive Weighted Loss (ALF) function was also implemented on a watershed scale basis where only one model can cover the full watershed. For each year the MVMH or PMVMH model takes stormflow component of the water level data as input from all available stream gauges, along with all precipitation inputs (Table 3), generating synchronized 12-step forecasts for all cross-sections simultaneously which demonstrates the watershed approach using multivariate modeling. The input (11) and output sequences (12) for the multivariate models are provided below:

Input sequence at time t (history window):

$$\mathrm{X}_t = [\underbrace{\mathrm{s}(t-L),\ldots,\mathrm{s}(t-1)}_{\textbf{stormflow history}};\underbrace{\mathrm{p}(t-L),\ldots,\mathrm{p}(t-1)}_{\textbf{precipitation history}}] \in \mathrm{R}^{L\times(N+M)} \qquad (11)$$

where $s(t)$ is the water level component of the stormflow at time t, $p(t)$ is the amount of precipitation at time t, L is the size of the time embedding window, H is the forecasting horizon in time steps, N is the number of stream gauges, M is the number of rain gauges. In case of univariate multi-horizon model the N=1 but for the rest of the multivariate model MVMH and PMVMH the $N > 1$.

Multivariate model target for 3-hour horizon:

$$Y_t = [s(t), s(t+1), \ldots, s(t+H-1)] \in R^{H \times N} \qquad (12)$$

5.3 Experiment sets

The proposed approach was evaluated using four experimental setups, all employing MAE as SLF. In the first set of experiments, year-wise multivariate models were developed separately on datasets from 2013 to 2020 using SLF. Yearly datasets were split chronologically in the following way the first 10,000 samples formed training sets, the next 4,000 samples were used in validation, and the remaining 7,818 ones were left for testing. In the second set of experiments, multivariate models were trained based on the ALF on the same datasets. UVMH models MAE values obtained from exploratory computations for all available sites per year (Table 3) were used to define initial adaptive weights. The third set of experiments implemented Cross-Year Prediction (CYP), where the multivariate model was trained on the 2013 dataset with SLF and tested on data from 2014 and 2015 separately to assess its generalization. Specifically, the first 14,000 samples of 2013 were used for training, the next 7,818 for validation. The entire 2014 and 2015 datasets were used to test the developed model. In the last set of experiments, the CYP model was trained with ALF on the same 2013 dataset and tested on 2014 and 2015 datasets.

5.4 Hyperparameter Tuning for Common Model Configuration

Hyperparameter tuning is a crucial step in achieving optimal model configuration settings to prevent both overfitting and underfitting. One of the methods for doing the hyperparameter tuning is the grid search, where all sets of combinations are systematically tested. In this study MLflow [99] tool was utilized to track and log the entire hyperparameter tuning journey. MLflow [99] is an MLOps tool which is used widely in industry as a collaborative platform for maintaining the end to end machine learning lifecycle. Data from 2014 (Figure 8), a dry year, as well as 2019 and 2020, both years with the highest number of stream and rain gauges to reflect hydrological variability and network size, were selected to perform the hyperparameter tuning process. This process aimed to find common configurations for univariate and multivariate models under SLF and multivariate models under ALF. A total of 27 experimental runs for each site per year were conducted using the univariate model under SLF and 81 experimental runs per year were executed using the multivariate model under SLF and ALF (Tables B.1 to B.19).

The common hyperparameters include the number of neurons, learning rate, loss function, number of hidden and LSTM layers, batch size, and the optimization algorithm. In all the experiments, whether it is univariate under SLF and multivariate with SLF or ALF, the swish activation function was used, along with Adam optimization algorithm with a learning rate of 0.0001. Furthermore, the number of epochs was set to 100 with early stopping criteria to prevent overfitting, and before the output layer, a linear activation

function was used. Lastly, the output channel consists of neurons sized as $H \times N$, where H=12 lead times and N equals the number of available stream gauges for that year (Table 3). In the case of univariate multi-horizon model is produces 12 lead time or 3 hours of prediction for a single stream of gauges, so N will be 1 for the univariate model type. The best configuration is chosen based on the most common model that ranked in the top 5 for each site per year.

In order to obtain the ALF's initial weights, the univariate model was developed with the number of LSTM and hidden layers varied from 1 to 3. In each LSTM layer, the number of units should be a power of 2, so it varied from 64 to 256, while the number of neurons per hidden layer was kept constant at 20 units. Furthermore, MAE was used as a loss function for this model type and a fixed dropout rate of 20% was used between layers to maintain regularization. The top-performing common univariate multi-horizon model under SLF configuration for each site over the three investigated years consists of a single LSTM layer with 128 units, a dropout rate of 20%, and a dense layer with 20 units followed by another dropout layer at 20%.

In the case of multivariate multi-horizon models under SLF, a broader search space was investigated to achieve the optimal model configuration. The search explored the number of LSTM and dense layers, which varied from 1 to 3. In each LSTM layer, the number of units should be a power of 2, ranging from 128 to 512, while the number of neurons per dense layer varied from 80, 90, or 100 units with a dropout rate of 20%. Both for the LSTM multivariate (MVMH) and the Peephole multivariate LSTM model (PMVMH), the MAE loss function was utilized in this experiment. The common best model configuration for both the multivariate models involved a two LSTM layers with 512 units each, followed by a 20% dropout, single dense layer with 100 units using swish activation and another dropout of 20%. The output channel consists of neurons accommodating all available stream gauges and rain gauges for that investigated year (Table 3). This model configuration is well suited for years with dry conditions, as well as the years with the maximum number of water level and precipitation stations.

Following the site-wise loss contribution results produced by the univariate multi-horizon model under SLF, multivariate multi-horizon models were used with the adaptive weighted approach. The search space was the same for the MVMH and PMVMH under SLF, however, the training objective was different, as an epoch-wise adaptive weighting approach was leveraged instead of the MAE. Throughout the investigated years, the most common model configuration consists of single LSTM layer with 512 units, followed by a 20% dropout rate, a dense layer with 80 units using swish activation and another 20% dropout. The output size is the same as the multivariate SLF configuration.

A single global random seed was set from the beginning to support consistency across all relevant Python library functions, ensuring the reproducibility of the results. By enabling the determinism function with the specific random seed, TensorFlow avoids non-deterministic algorithms, assuring that identical

results will be produced across repeated runs on the same GPU. Furthermore, the entire hyperparameter search space with the performance metrics is logged, tracked and recorded using the MLflow [99], which captures the exact environment as well as the Python version and package dependencies using a YAML file, allowing the code to be reproduced reliably. Since each run is well documented, along with saving the trained weights and bias, the entire machine learning workflow is reproducible.

Chapter 6

Results and Discussion

Before discussing the forecasting results, it is important to analyze the hydrological characteristics of each study year. Data corresponding to the warm season that runs from April to November for the years 2013 to 2020 were selected and preprocessed separately. These datasets were selected for the investigation because of the availability of time series from all available monitoring sites (Table 3). The years 2013, 2017-2019 were wet years, and 2014-2016, 2020 were hydrologically dry years. Even though four years

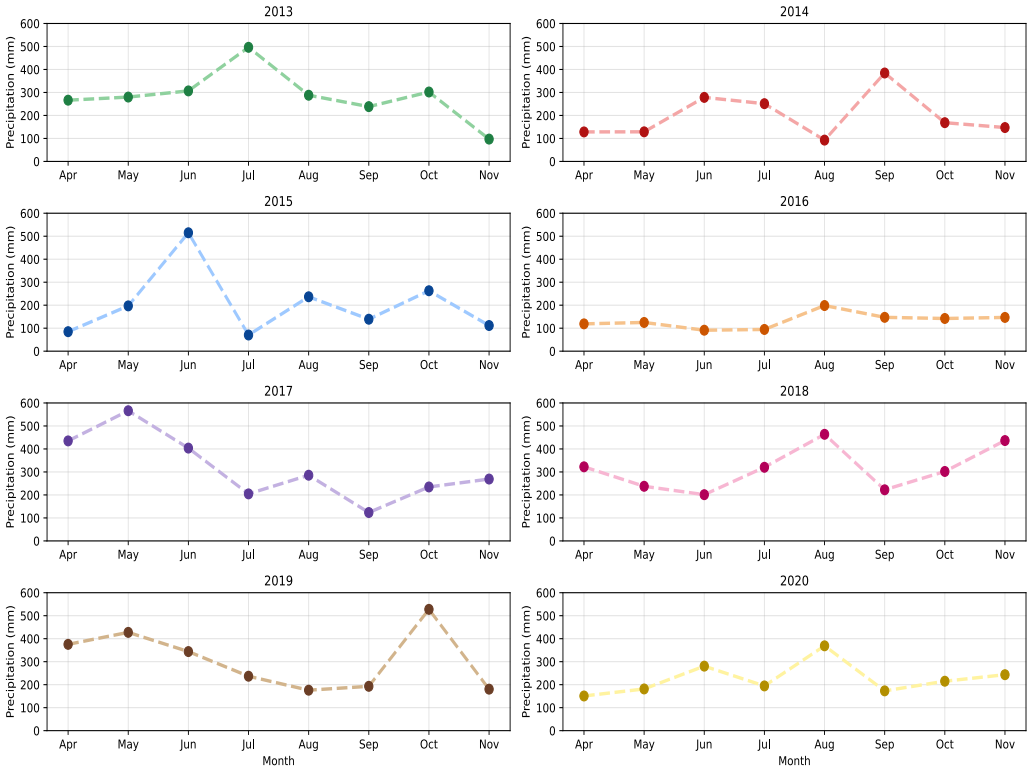


Figure 8: Monthly precipitation amounts in mm during the warm period from 2013 to 2020

are classified as dry and the other four as wet, the spatio-temporal patterns within the respective years' datasets are different in terms of high-flow events, and seasonal distribution. These differences influence watershed response behavior, making it necessary to evaluate forecasting performance under multiple year-wise hydrological settings. Figure 8 clearly illustrates these hydrological characteristics, displaying the total monthly precipitation values in millimeters.

6.1 Exploratory Computations: Time Delay Embedding Selection

The cross sections and rain gauges that have available data from 2013 to 2020 (Table 3) were selected to compare two different historical embedding methods to determine the appropriate time delay embedding for the future watershed approach experiments. The three most common sites were SCN, SCS, EDD and available precipitation gauges HL and MWY. Multivariate model performance was compared with 2-hour and 3-hour time-delay embeddings in Tables 4, 5 and from Table A.1 to A.6. The lead times from 1 to 12 denote successive 15-minute prediction horizons which ranges from 15 minutes ahead to 3 hours ahead and the values reported are the multivariate models' relative error (10). The model architecture used for this exploratory experiment is the same as Figure 6. Based on the RMAE criteria, the model performance was evaluated across years in order to select the appropriate time delay embedding which will be essential to capture upstream-downstream dynamics and relevant historical precipitation records.

Table 4: Comparing Time Embedding of Models in 2016

Lead Time	2 Hour Embedding			3 Hour Embedding		
	MVMH			MVMH		
	SCN	SCS	EDD	SCN	SCS	EDD
1	0.0586	0.2534	0.1113	0.0554	0.2103	0.1235
2	0.0665	0.2582	0.1171	0.0618	0.2101	0.1239
3	0.0835	0.2429	0.1092	0.0598	0.2077	0.1227
4	0.0731	0.2412	0.1211	0.0620	0.1991	0.1267
5	0.0653	0.2367	0.1712	0.0653	0.1941	0.1291
6	0.0726	0.2513	0.1213	0.0706	0.2094	0.1341
7	0.0726	0.2968	0.1284	0.0697	0.2174	0.1361
8	0.0850	0.2574	0.1590	0.0779	0.2173	0.1467
9	0.0833	0.2669	0.1670	0.0784	0.2310	0.1482
10	0.0829	0.2789	0.1464	0.0840	0.2460	0.1599
11	0.0882	0.2832	0.1617	0.0859	0.2556	0.1698
12	0.0902	0.3056	0.1743	0.0916	0.2709	0.1874

The MVMH model performance across the 3 sites underscores the importance of selecting the appropriate travel time for watershed learning. Under the 2 hour embedding, the MVMH model remained below the 20% reliability RMAE for the entire forecast period at SCN in every year. However, modeling for SCS seemed to be challenging with the 2 hour embedding, as the MVMH models were completely unreliable on all lead times from 2014 to 2016 and 2018 to 2020. The developed MVMH model was only

reliable up to one and a half hours in 2013 (Table A.1) and for just 15 minutes in 2017 (Table A.4) at this location. Even at the EDD location, the MVMH model performance was limited, and it was only reliable for 15 minute lead time in 2020.

Table 5: Comparing Time Embedding of Models in 2020

Lead Time	2 Hour Embedding			3 Hour Embedding		
	MVMH			MVMH		
	SCN	SCS	EDD	SCN	SCS	EDD
1	0.1063	0.3555	0.1471	0.0802	0.2620	0.1678
2	0.0734	0.3284	0.2463	0.0816	0.2533	0.1623
3	0.1225	0.3287	0.2020	0.0872	0.2489	0.1731
4	0.1005	0.3791	0.2398	0.0877	0.2472	0.1791
5	0.1037	0.3254	0.2027	0.0916	0.2623	0.1804
6	0.1332	0.3217	0.2106	0.1013	0.2681	0.1860
7	0.1069	0.3129	0.2188	0.1014	0.2778	0.1873
8	0.1109	0.3202	0.3228	0.1112	0.2943	0.1960
9	0.1075	0.3432	0.2086	0.1110	0.3051	0.1987
10	0.1256	0.4064	0.2471	0.1191	0.3184	0.2137
11	0.1216	0.3589	0.2096	0.1249	0.3325	0.2182
12	0.1735	0.3854	0.2331	0.1266	0.3471	0.2325

The model performance of MVMH with 3 hour time delay further justifies the selection of the correct time embedding. RMAE estimates were reliable for the full 3 hour horizon at the SCN site across all years and the EDD location also generated reliable forecasts for most of the years except for 2019 (Table A.6) and 2020 (Table 5). The biggest performance improvement at EDD in 2020 was that, instead of collapsing at 15 minute lead time, the model maintained reliability up to 2 hours and 15 minutes (Table 5). In the case of the most rapid-response station, SCS, the MVMH was below the 20% threshold up to one and a half hours in 2016 (Table 4), and reliable up to one hour and fifteen minutes in 2018 (Table A.5), where the 2 hour embedding variant was completely unreliable in both years. As the 3 hour time embedding provided more reliability across the years for MVMH models and due to the watershed being small in size and vastly urbanized for the three cross sections a 3 hour time delay embedding is appropriate to capture rainfall run-off. However, apart from these three cross sections, more monitoring sites are available (Table 3) from 2015 onward, and the most expanded monitoring network in 2019 and 2020. Therefore, a longer time window become more appropriate for capturing delayed responses within the watershed. A variable sequence of time delay for each site might be helpful. However, LSTM-based sequential models require a consistent input length sequence. Therefore, a three hour time delay embedding was selected. Following

the selection of the three-hour time embedding, further experiments were conducted with the extension to all available stream and precipitation gauges for each year (Table 3).

6.2 Model performance based on SLF and ALF strategy

The experiments across the years 2013 to 2020 highlight that the main contribution of the multivariate multi-horizon framework is that it operationalized the watershed approach as a single multivariate model can generate synchronized 15 minute forecasts up to 3 hours for multiple sites under the same storm focusing. The main challenge is that multivariate models can achieve good average performance overall while still under-training sites with more variable, data-sparse, or harder to learn hydrological responses. The multivariate models with shared static loss function (SLF) depict this limitation, as it compromises excellent model performance at some sites due to poor performance at other sites of the same watershed, leaving the model under-trained for flash-response cross-sections. To address this issue, the second experiment set utilizes Adaptive Weighted Loss Function (ALF) with multivariate models that dynamically adjust site contributions, reallocating optimization pressure towards the sites that exhibit higher errors. The goal of ALF is not to reduce error magnitude for every site, but instead to extend the forecast horizon at rapid-response sites under a single watershed model. The third experiment set leveraged the CYP models which are trained on the entire 2013 dataset and evaluated on subsequent datasets under the SLF, is applied to investigate whether a larger dataset could improve and extend the reliability of models. The last experiment emphasizes applying ALF with CYP models to determine whether ALF further improves performance by leveraging richer data.

Model performance was evaluated on the testing sets using two complementary metrics, which are NSE and Relative MAE. Following commonly used practices, NSE (8) values at or above 0.7 were interpreted as ‘good’ and at or above 0.8 as ‘very good’ model performance. From a practical perspective, a forecast is deemed reliable if its accuracy exceeds 80%. Consequently, generated predictions were evaluated using RMAE (10) to determine their trustworthiness. Those with RMAE at or below 20% were accepted as reliable, illustrated in Figures 9 to 18, where the x-axis denotes the forecast lead times from t+1 to t+12, corresponding from 15 minutes ahead to 3 hours ahead, while the y-axis represents the Relative Mean Absolute Error (10) in percentage. A model performed better than others if its results had higher NSE values, lower RMAE values and remained reliable for a longer lead time. Model performance according to the NSE index (8) for each lead time and monitoring site is presented in Tables 6-14, where the lead times from 1 to 12 represent 15-minute forecasting intervals varied from 15 minutes ahead to 3 hours ahead.

The results from the earlier period (2013, 2014) demonstrate that the proposed Adaptive Weighted Loss Function (ALF) with year-wise models does not uniformly improve all sites but instead redistributes

Table 6: NSE Index of Multivariate Models with SLF & ALF (2013)

Sites	Models	Leadtime											
		1	2	3	4	5	6	7	8	9	10	11	12
SCN	MVMH (SLF)	0.96	0.95	0.95	0.95	0.94	0.94	0.94	0.93	0.93	0.92	0.90	0.89
	PMVMH (SLF)	0.96	0.96	0.96	0.95	0.95	0.94	0.94	0.93	0.93	0.92	0.91	0.89
	MVMH (ALF)	0.97	0.97	0.96	0.95	0.94	0.93	0.93	0.92	0.91	0.90	0.89	0.88
	PMVMH (ALF)	0.98	0.97	0.96	0.95	0.94	0.94	0.93	0.93	0.92	0.91	0.90	0.89
SCS	MVMH (SLF)	0.97	0.96	0.96	0.95	0.93	0.91	0.88	0.84	0.79	0.73	0.68	0.64
	PMVMH (SLF)	0.97	0.96	0.96	0.95	0.93	0.91	0.88	0.84	0.79	0.73	0.69	0.64
	MVMH (ALF)	0.97	0.96	0.95	0.94	0.92	0.90	0.88	0.84	0.79	0.74	0.69	0.65
	PMVMH (ALF)	0.97	0.96	0.95	0.94	0.92	0.91	0.88	0.84	0.79	0.74	0.69	0.65
EDD	MVMH (SLF)	0.99	0.98	0.98	0.97	0.97	0.96	0.95	0.94	0.93	0.91	0.90	0.88
	PMVMH (SLF)	0.99	0.98	0.98	0.97	0.97	0.96	0.95	0.94	0.93	0.91	0.90	0.88
	MVMH (ALF)	0.99	0.98	0.98	0.97	0.97	0.96	0.95	0.94	0.93	0.92	0.90	0.88
	PMVMH (ALF)	0.99	0.98	0.98	0.97	0.97	0.96	0.95	0.94	0.93	0.92	0.90	0.88

Table 7: NSE Index of Multivariate Models with SLF & ALF (2014)

Sites	Models	Leadtime											
		1	2	3	4	5	6	7	8	9	10	11	12
SCN	MVMH (SLF)	0.97	0.97	0.96	0.96	0.96	0.95	0.95	0.95	0.94	0.93	0.92	0.92
	PMVMH (SLF)	0.97	0.97	0.96	0.96	0.96	0.95	0.95	0.95	0.94	0.93	0.92	0.92
	MVMH (ALF)	0.98	0.98	0.98	0.97	0.97	0.97	0.96	0.95	0.95	0.94	0.93	0.93
	PMVMH (ALF)	0.98	0.98	0.97	0.97	0.97	0.96	0.96	0.95	0.95	0.94	0.94	0.93
SCS	MVMH (SLF)	0.93	0.92	0.91	0.90	0.89	0.89	0.86	0.84	0.81	0.76	0.71	0.67
	PMVMH (SLF)	0.93	0.92	0.91	0.90	0.89	0.89	0.86	0.84	0.81	0.76	0.71	0.67
	MVMH (ALF)	0.93	0.93	0.92	0.91	0.90	0.89	0.86	0.84	0.81	0.77	0.72	0.68
	PMVMH (ALF)	0.94	0.94	0.93	0.92	0.91	0.89	0.88	0.85	0.82	0.78	0.74	0.69
EDD	MVMH (SLF)	0.97	0.96	0.95	0.94	0.93	0.92	0.89	0.88	0.86	0.84	0.82	0.79
	PMVMH (SLF)	0.97	0.96	0.95	0.94	0.93	0.92	0.89	0.88	0.86	0.84	0.82	0.79
	MVMH (ALF)	0.99	0.98	0.97	0.96	0.95	0.94	0.93	0.91	0.89	0.87	0.85	0.82
	PMVMH (ALF)	0.99	0.98	0.97	0.96	0.95	0.93	0.92	0.90	0.88	0.86	0.84	0.81

the optimization attention toward the under-trained station. During this period, the SCN cross-section with uniform land cover and the lowest error was well learned by all models, and the EDD site also exhibited stable performance. However, SCS remained the most rapid-response site. Allotted values of NSE index remained above 'very good' threshold up to three-hour horizon at SCN and EDD. Although the NSE index (Table 6) of all the multivariate models remained above 'good' threshold up to two and a half hours at SCS site on 2013 dataset, while RMAE estimates tell a different story. Multivariate models trained with SLF at the SCS site were reliable for only up to 2 hours and 15 minutes on the 2013 dataset, whereas both MVMH-ALF and PMVMH-ALF extended the forecast horizon up to two and a half hours (Figure 9). A similar pattern was observed in 2014, where the challenging spatio-temporal characteristics of the SCS site affected the model performance, which generated limited forecast horizons. Models' forecasting of SCS water level remained 'very good' only up to 2 hours and 15 minutes and 'good' up to 2 hours and 45 minutes (Table 7). The dynamics of water level at SCS cross-section can also be observed from the hydrograph (Figure 11), where the MVMH-ALF model was able to better capture the peak magnitude than MVMH-SLF.

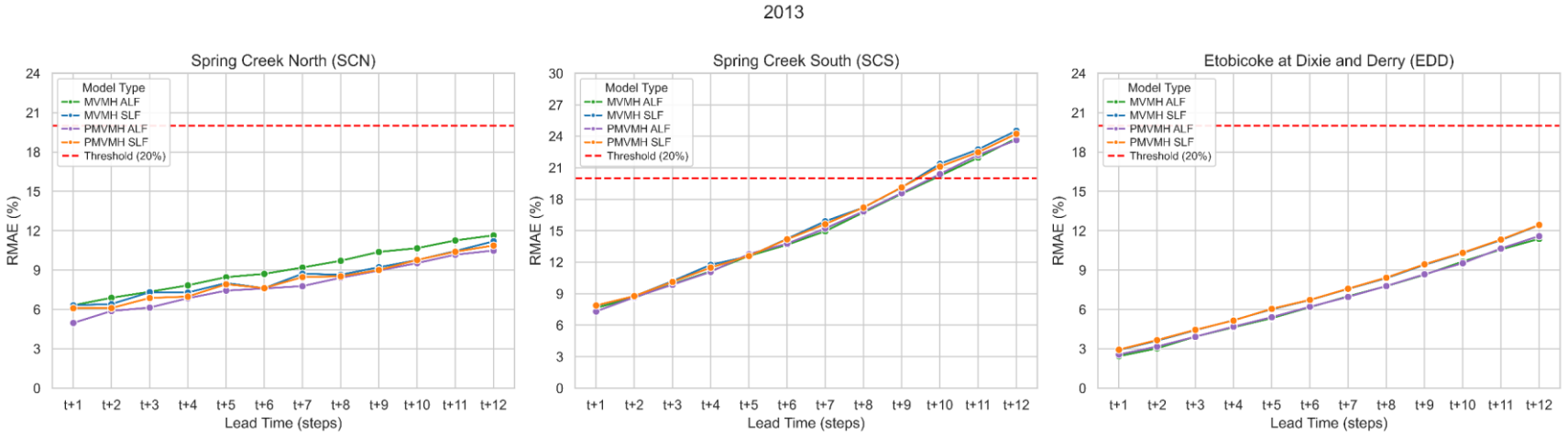


Figure 9: Relative prediction error (%) of multivariate models on 2013 dataset

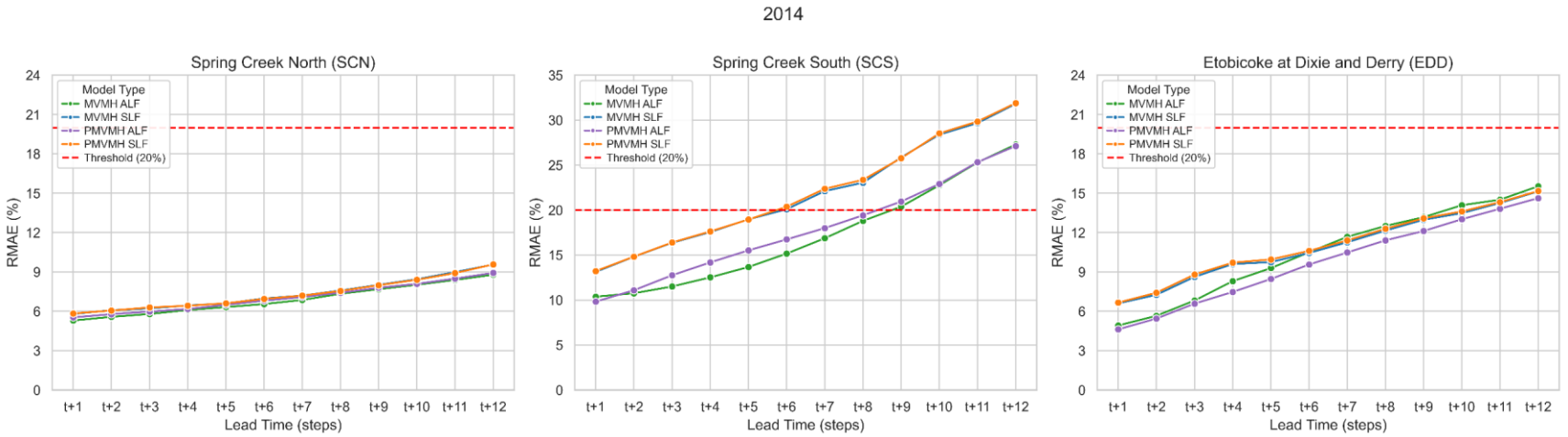


Figure 10: Relative prediction error (%) of multivariate models on 2014 dataset

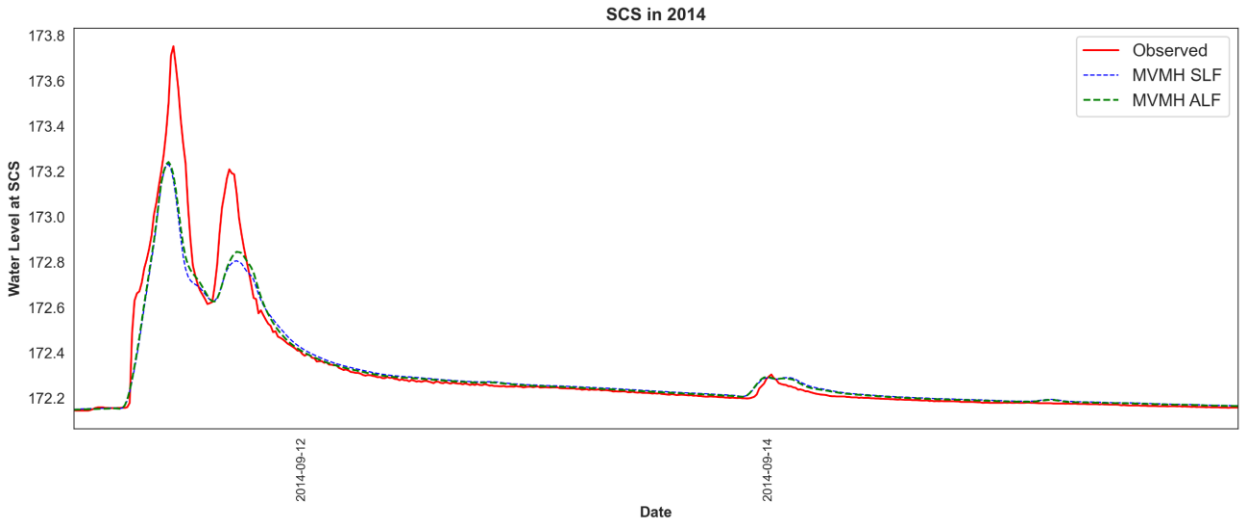


Figure 11: Hydrograph Comparison of MVMH SLF and MVMH ALF at SCS in 2014

Multivariate models with SLF struggled at the rapid-responsive site of SCS, while MVMH–ALF and PMVHMH-ALF extended the reliable forecast horizon by 45 minutes and remained reliable up to 2 hour and 15 minutes (Figure 10). This improvement demonstrates that with a shared static objective, the stable stations lean towards stabilizing the model training, whereas the highly responsive sites receive insufficient corrective updates. The ALF framework solves this issue by adaptively adjusting the site contributions with validation feedback, guiding the gradient updates towards under-trained stations. The cross-year experiments further justify that ALF addresses the static-weight imbalance problem. When the model was trained and validated on the historical 2013 dataset and then tested on the unseen 2014 and 2015 datasets, the objective of the model is to learn cross-year spatio-temporal characteristics across hydrological regimes.

Both the CYP models with SLF and ALF which are trained on the entire 2013 dataset and are evaluated on subsequent years outperformed year-wise MVMH models, highlighting the role of the quality of data in such predictions. The nature of Deep Learning implies that the larger and richer the training set allowing a model to learn, the better the predictions. Training the CYP models on the wet year dataset reflecting multiple high-flow events of various scales allowed the models to depict hydrological spatiotemporal patterns better than on the reduced datasets. These models generalized well, surpassing year-wise models even in the case of the most critical site SCS. Both model variants showcased similar performance but CYP combining with the adaptive weighting mechanism maintained lower RMAE and higher NSE (Table 8) than the CYP-SLF model. After combining the ALF framework with the multivariate cross year model, the reliable forecasting window for SCS increased from 2 hour and 15 minutes to two and a two and a half hour window (Figure 12 and Figure 13).

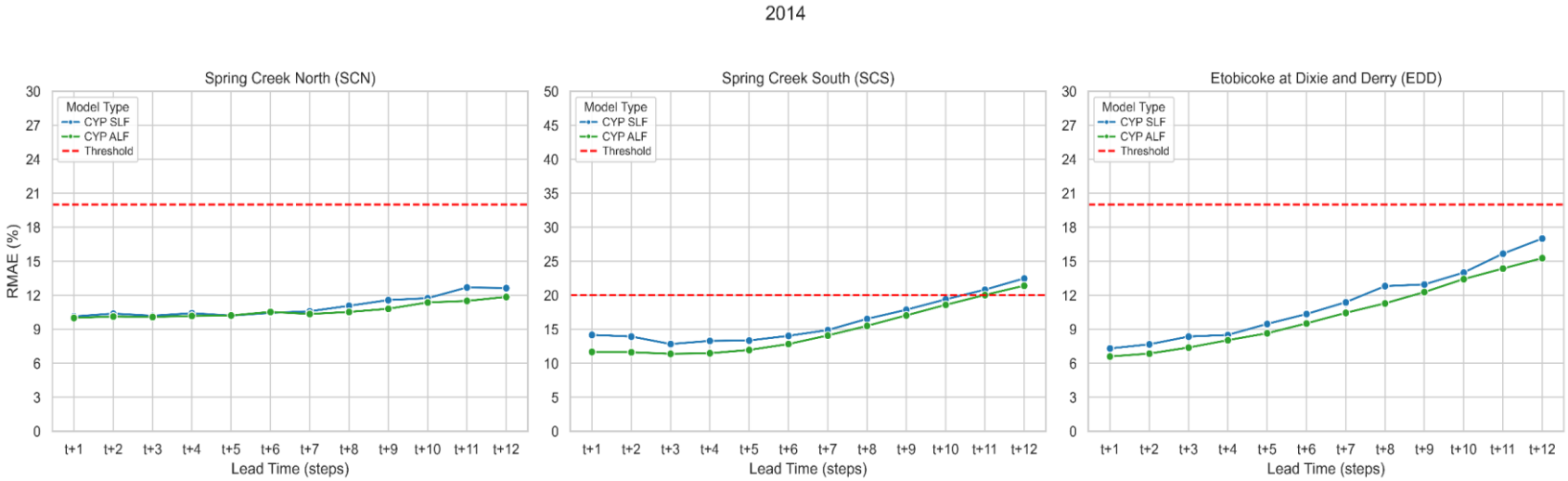


Figure 12: Relative prediction error (%) of cross-year models trained on 2013, tested on 2014 dataset

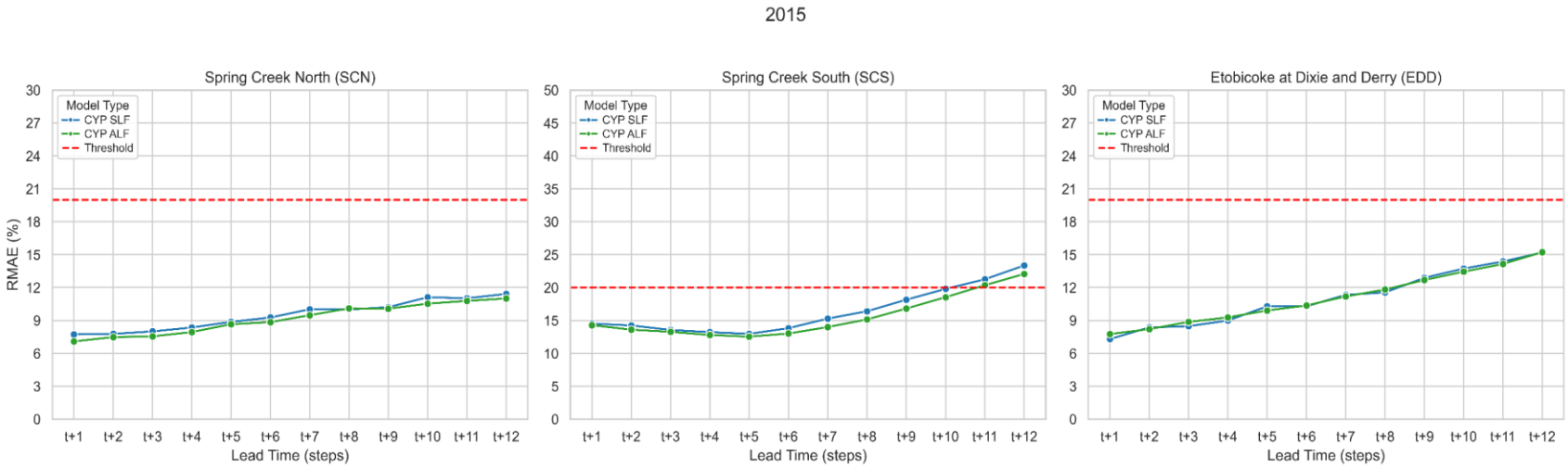


Figure 13: Relative prediction error (%) of cross-year models trained on 2013, tested on 2015 dataset

Table 8: NSE Index of Multivariate LSTM for Cross-Year Predictions with SLF & ALF

Site	Lead Time	CYP SLF		CYP ALF	
		2014	2015	2014	2015
SCN	1	0.942	0.9662	0.9069	0.9723
	2	0.9262	0.9669	0.9052	0.9703
	3	0.9326	0.9657	0.9141	0.9698
	4	0.9312	0.9593	0.9190	0.9632
	5	0.937	0.9497	0.9237	0.9541
	6	0.9305	0.9418	0.9206	0.9492
	7	0.9309	0.9321	0.9234	0.9391
	8	0.9198	0.9285	0.9213	0.9311
	9	0.9139	0.9219	0.9136	0.9287
	10	0.9074	0.912	0.9008	0.9226
	11	0.8872	0.9093	0.8986	0.9176
	12	0.888	0.9021	0.8921	0.9117
SCS	1	0.8346	0.8606	0.8765	0.8727
	2	0.8511	0.8718	0.8796	0.8836
	3	0.8666	0.8912	0.8878	0.8958
	4	0.8775	0.9081	0.8905	0.9113
	5	0.8763	0.9225	0.8827	0.9230
	6	0.87	0.9238	0.8618	0.9200
	7	0.8346	0.9075	0.8267	0.8988
	8	0.7867	0.8832	0.7835	0.8787
	9	0.7318	0.8325	0.7323	0.8424
	10	0.6848	0.7942	0.6791	0.7951
	11	0.6349	0.7586	0.6276	0.7536
	12	0.5834	0.7129	0.5797	0.7097
EDD	1	0.9589	0.9741	0.9687	0.9714
	2	0.9579	0.9677	0.9655	0.9652
	3	0.9529	0.9633	0.9620	0.9611
	4	0.9505	0.9591	0.9558	0.9569
	5	0.9427	0.9515	0.9483	0.9511
	6	0.9326	0.9467	0.9381	0.9450
	7	0.9219	0.9369	0.9237	0.9353
	8	0.8999	0.9301	0.9068	0.9255
	9	0.887	0.914	0.8875	0.9108
	10	0.8605	0.8989	0.8614	0.8963
	11	0.8321	0.8844	0.8352	0.8822
	12	0.7967	0.8665	0.8106	0.8629

Table 9: NSE Index of Multivariate Models with SLF & ALF (2015)

Sites	Models	Leadtime											
		1	2	3	4	5	6	7	8	9	10	11	12
SCN	MVMH (SLF)	0.99	0.98	0.98	0.98	0.97	0.97	0.96	0.95	0.95	0.94	0.93	0.92
	PMVMH (SLF)	0.99	0.98	0.98	0.98	0.97	0.96	0.96	0.95	0.94	0.93	0.92	0.91
	MVMH (ALF)	0.98	0.98	0.98	0.97	0.97	0.96	0.96	0.95	0.94	0.93	0.93	0.91
	PMVMH (ALF)	0.99	0.99	0.98	0.98	0.97	0.97	0.96	0.96	0.95	0.94	0.93	0.92
EB	MVMH (SLF)	0.95	0.94	0.92	0.91	0.88	0.86	0.83	0.80	0.77	0.74	0.70	0.67
	PMVMH (SLF)	0.95	0.93	0.91	0.90	0.87	0.84	0.81	0.78	0.75	0.72	0.68	0.65
	MVMH (ALF)	0.95	0.94	0.92	0.91	0.89	0.86	0.83	0.81	0.78	0.75	0.71	0.67
	PMVMH (ALF)	0.95	0.94	0.93	0.91	0.89	0.87	0.84	0.82	0.78	0.75	0.71	0.68
SCS	MVMH (SLF)	0.94	0.95	0.95	0.94	0.94	0.92	0.91	0.89	0.86	0.82	0.77	0.72
	PMVMH (SLF)	0.96	0.96	0.96	0.95	0.94	0.93	0.91	0.89	0.85	0.81	0.76	0.71
	MVMH (ALF)	0.95	0.95	0.95	0.95	0.94	0.93	0.91	0.89	0.86	0.82	0.77	0.72
	PMVMH (ALF)	0.95	0.95	0.95	0.95	0.94	0.93	0.92	0.89	0.86	0.82	0.77	0.72
EDD	MVMH (SLF)	0.98	0.98	0.98	0.97	0.97	0.96	0.95	0.94	0.93	0.92	0.90	0.88
	PMVMH (SLF)	0.99	0.98	0.98	0.97	0.97	0.96	0.96	0.95	0.94	0.92	0.90	0.88
	MVMH (ALF)	0.99	0.98	0.98	0.97	0.97	0.96	0.96	0.95	0.94	0.92	0.91	0.89
	PMVMH (ALF)	0.99	0.98	0.98	0.97	0.97	0.96	0.95	0.95	0.93	0.92	0.90	0.88

Table 10: NSE Index of Multivariate Models with SLF & ALF (2016)

Sites	Models	Leadtime											
		1	2	3	4	5	6	7	8	9	10	11	12
SCN	MVMH (SLF)	0.98	0.97	0.97	0.96	0.95	0.94	0.93	0.91	0.89	0.87	0.84	0.82
	PMVMH (SLF)	0.98	0.97	0.97	0.96	0.95	0.94	0.93	0.91	0.89	0.87	0.84	0.82
	MVMH (ALF)	0.98	0.98	0.97	0.96	0.96	0.95	0.93	0.92	0.90	0.87	0.85	0.83
	PMVMH (ALF)	0.98	0.98	0.97	0.96	0.96	0.95	0.93	0.92	0.90	0.87	0.85	0.83
EB	MVMH (SLF)	0.98	0.97	0.97	0.96	0.96	0.95	0.94	0.93	0.91	0.88	0.86	0.83
	PMVMH (SLF)	0.98	0.97	0.97	0.96	0.96	0.95	0.94	0.93	0.91	0.88	0.86	0.83
	MVMH (ALF)	0.98	0.97	0.97	0.96	0.96	0.95	0.94	0.93	0.91	0.89	0.86	0.84
	PMVMH (ALF)	0.98	0.97	0.97	0.96	0.96	0.96	0.95	0.93	0.91	0.89	0.87	0.84
SCS	MVMH (SLF)	0.97	0.96	0.96	0.95	0.94	0.93	0.91	0.89	0.85	0.80	0.74	0.69
	PMVMH (SLF)	0.97	0.96	0.96	0.95	0.94	0.93	0.91	0.89	0.85	0.80	0.74	0.69
	MVMH (ALF)	0.97	0.96	0.96	0.95	0.94	0.93	0.92	0.90	0.86	0.82	0.77	0.71
	PMVMH (ALF)	0.97	0.97	0.96	0.95	0.94	0.93	0.92	0.89	0.86	0.82	0.76	0.71
EDD	MVMH (SLF)	0.98	0.97	0.97	0.97	0.96	0.96	0.96	0.95	0.95	0.94	0.93	0.93
	PMVMH (SLF)	0.98	0.97	0.97	0.97	0.96	0.96	0.96	0.95	0.95	0.94	0.93	0.93
	MVMH (ALF)	0.98	0.98	0.97	0.97	0.96	0.96	0.95	0.95	0.95	0.94	0.93	0.92
	PMVMH (ALF)	0.98	0.97	0.97	0.97	0.96	0.96	0.95	0.95	0.95	0.94	0.93	0.93

As the watershed network expanded, the effectiveness of the adaptive strategy became more visible. With the addition of the EB cross-section, which is a heavily urbanized site on the west branch cross-section, the learning problem became more heterogeneous. On the 2015 dataset, the PMVMH with ALF achieved reliable performance up to an hour with the lowest RMAE estimates (10) while the MVMH-ALF was more focused on reducing error at SCS and EDD (Figure 14). This highlights the fact that ALF is correctly identifying the flashy responsive sites, and the peephole gating mechanism is able to retain relevant high-flow events, which improved the forecast reliability. On the contrary, MVMH-SLF outperforms MVMH-

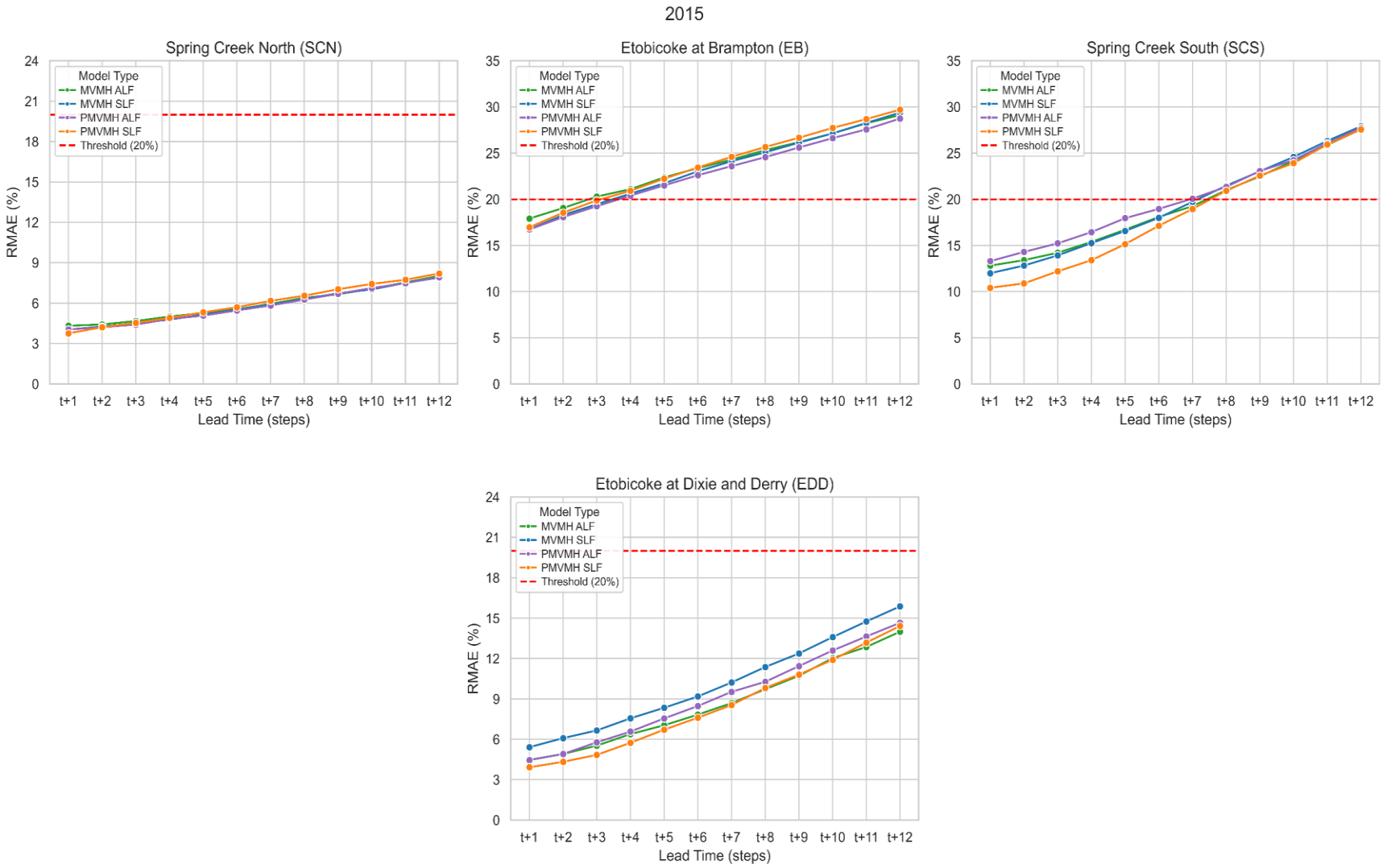


Figure 14: Relative prediction error (%) of multivariate models on 2015 dataset

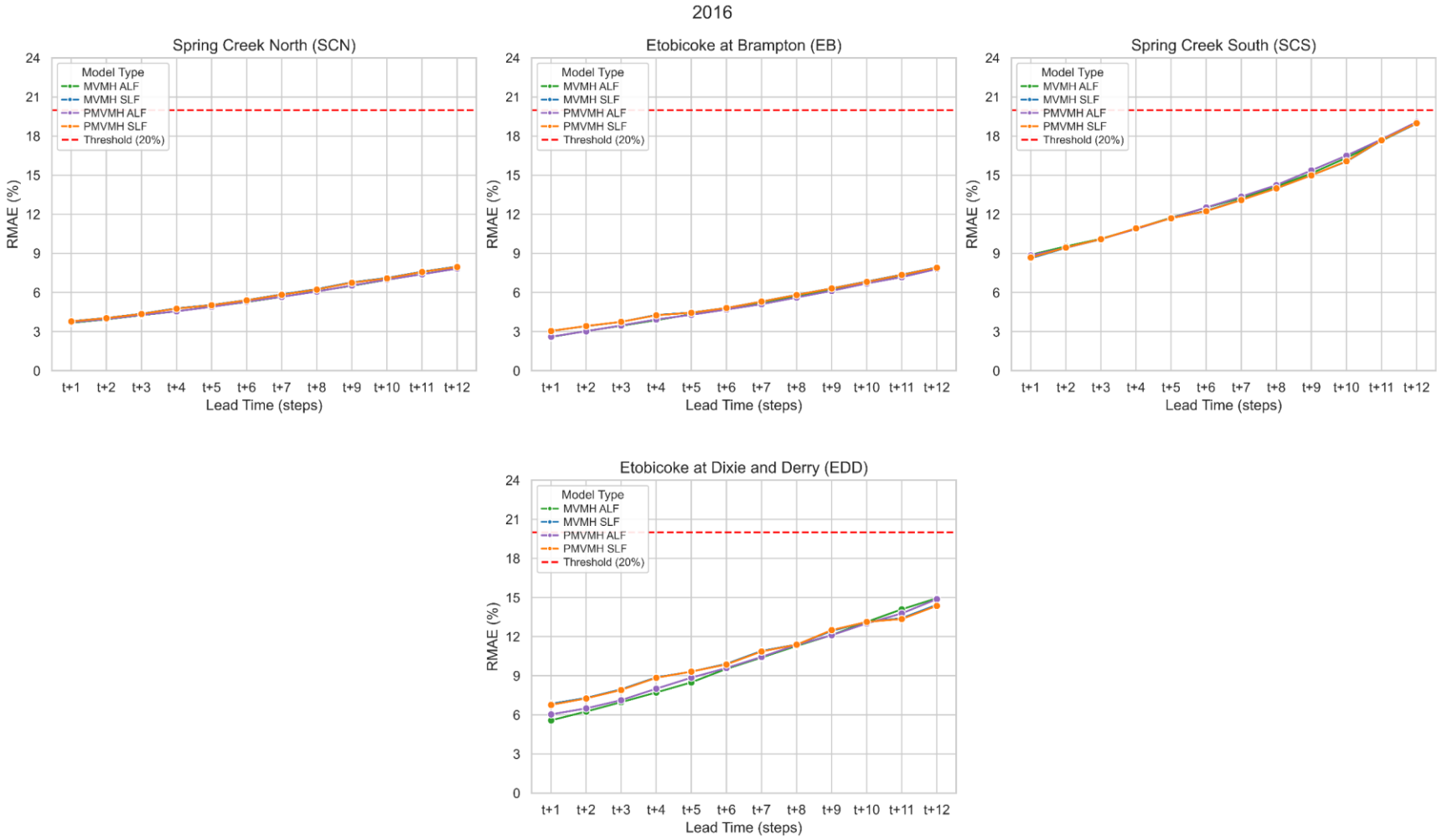


Figure 15: Relative prediction error (%) of multivariate models on 2016 dataset

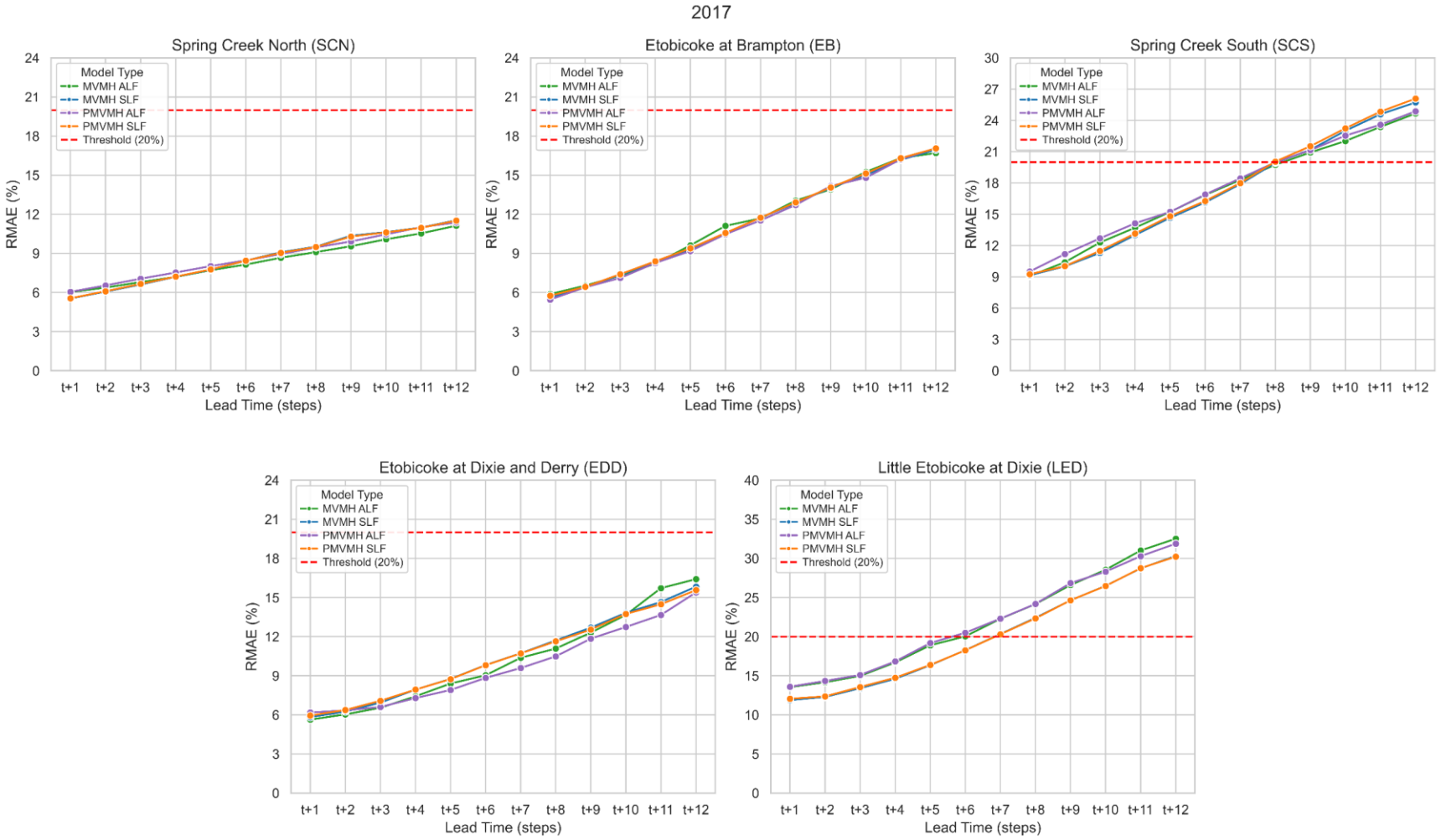


Figure 16: Relative prediction error (%) of multivariate models on 2017 dataset

ALF at SCN because the ALF prioritized SCS and EDD rather than optimizing an already stable site. However, NSE estimates from multivariate models with both SLF and ALF at the SCN site confirmed 'very good' performance across the entire forecast horizon (Table 9). In 2016, all the model types with SLF and ALF were reliable across the entire forecasting window (Figure 15). The model performance exhibited that the SLF model variants outperformed the ALF variants, however, both loss variant models were below the RMAE threshold. As the performance of all sites was already well optimized under a static loss, the impact of the ALF strategy will not further improve performance.

The advantages of the ALF approach between the years 2017 and 2020 depend on the network size and spatio-temporal characteristics of the station, making the trade-offs more explicit. The multivariate models with ALF again improved reliability forecasts at the SCS site. The MVMH with ALF extended reliability from two hours to 2 hours and 15 minutes on the 2017 dataset (Figure 16), whereas both PMVMH and MVMH under SLF performed better at LED. This indicates that the multivariate models with the adaptive approach prioritized stabilizing SCS site while reducing the optimization capacity available for LED site, which is expected in multi-site learning trade-off. To the contrary, both model types generated similar NSE estimates for LED site, as all MVMH models indicated 'very good performance' up to 1 hour and 45 minutes and 'good performance' up to two and a half hours (Table 11). Interestingly, the model performance on 2018 dataset demonstrates opposite behavior for LED site (Figure 18). Being the most rapid-response station, all the models struggled to generate reliable forecasts here, however, MVMH with ALF extended the forecast horizon by 15 minutes compared to SLF variants and generated reliable lead time up to two and a half hours. The performance enhancement at LED indicates that the ALF approach with MVMH shifted its learning focus toward the rapid-response site without undermining watershed-level

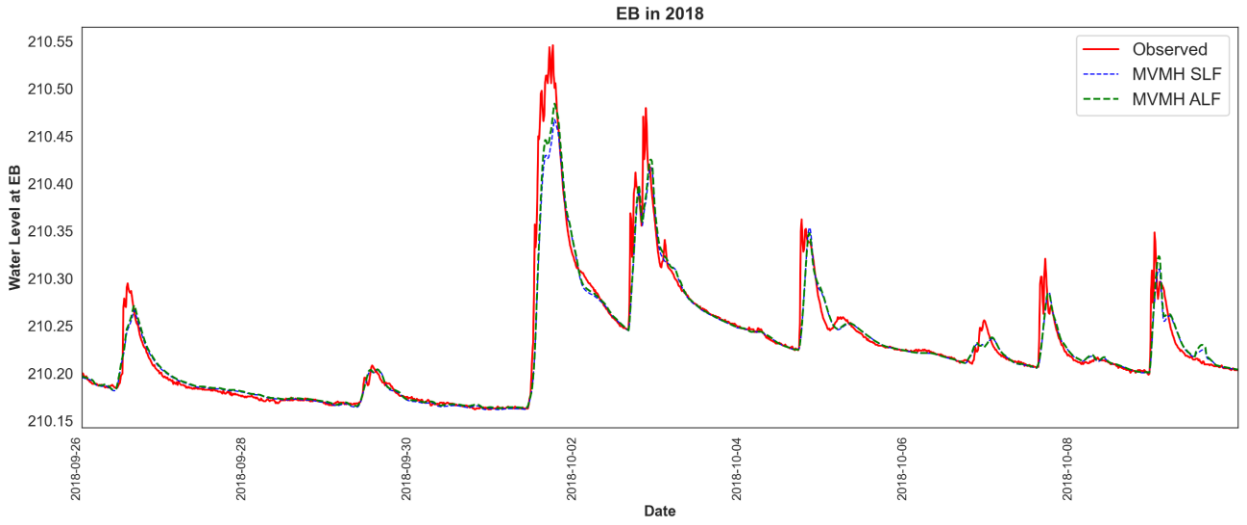


Figure 17: Hydrograph Comparison of MVMH SLF and MVMH ALF at EB in 2018

Table 11: NSE Index of Multivariate Models with SLF & ALF (2017)

Sites	Models	Leadtime											
		1	2	3	4	5	6	7	8	9	10	11	12
SCN	MVMH (SLF)	0.98	0.97	0.96	0.95	0.94	0.93	0.93	0.92	0.91	0.90	0.90	0.89
	PMVMH (SLF)	0.98	0.97	0.96	0.95	0.94	0.93	0.93	0.92	0.91	0.90	0.90	0.89
	MVMH (ALF)	0.97	0.97	0.96	0.95	0.94	0.94	0.93	0.92	0.91	0.91	0.90	0.89
	PMVMH (ALF)	0.97	0.97	0.96	0.95	0.94	0.93	0.93	0.92	0.91	0.91	0.90	0.89
EB	MVMH (SLF)	0.98	0.97	0.96	0.95	0.93	0.91	0.90	0.88	0.85	0.83	0.81	0.79
	PMVMH (SLF)	0.98	0.97	0.96	0.95	0.93	0.91	0.89	0.88	0.85	0.83	0.81	0.79
	MVMH (ALF)	0.98	0.97	0.96	0.95	0.93	0.91	0.89	0.87	0.85	0.83	0.81	0.79
	PMVMH (ALF)	0.98	0.97	0.96	0.95	0.93	0.91	0.89	0.87	0.85	0.83	0.81	0.79
SCS	MVMH (SLF)	0.95	0.93	0.90	0.88	0.85	0.82	0.79	0.75	0.71	0.67	0.64	0.60
	PMVMH (SLF)	0.95	0.93	0.90	0.88	0.85	0.82	0.79	0.75	0.71	0.67	0.64	0.60
	MVMH (ALF)	0.94	0.91	0.89	0.86	0.84	0.80	0.77	0.74	0.70	0.67	0.64	0.60
	PMVMH (ALF)	0.94	0.91	0.89	0.86	0.83	0.80	0.77	0.74	0.70	0.67	0.64	0.60
EDD	MVMH (SLF)	0.99	0.98	0.98	0.98	0.97	0.97	0.96	0.96	0.95	0.94	0.93	0.92
	PMVMH (SLF)	0.99	0.98	0.98	0.98	0.97	0.97	0.96	0.96	0.95	0.94	0.93	0.92
	MVMH (ALF)	0.99	0.98	0.98	0.97	0.97	0.96	0.96	0.95	0.95	0.94	0.93	0.92
	PMVMH (ALF)	0.98	0.98	0.98	0.97	0.97	0.97	0.96	0.96	0.95	0.94	0.93	0.92
LED	MVMH (SLF)	0.96	0.96	0.96	0.94	0.91	0.88	0.83	0.79	0.75	0.71	0.68	0.65
	PMVMH (SLF)	0.96	0.96	0.96	0.94	0.91	0.88	0.83	0.79	0.75	0.71	0.68	0.65
	MVMH (ALF)	0.95	0.96	0.95	0.94	0.91	0.88	0.83	0.79	0.75	0.71	0.67	0.64
	PMVMH (ALF)	0.96	0.96	0.95	0.94	0.91	0.88	0.83	0.79	0.75	0.71	0.68	0.65

Table 12: NSE Index of Multivariate Models with SLF & ALF (2018)

Sites	Models	Leadtime											
		1	2	3	4	5	6	7	8	9	10	11	12
SCN	MVMH (SLF)	0.99	0.98	0.98	0.98	0.97	0.97	0.96	0.95	0.94	0.93	0.92	0.91
	PMVMH (SLF)	0.99	0.98	0.98	0.98	0.97	0.97	0.96	0.95	0.94	0.93	0.92	0.91
	MVMH (ALF)	0.99	0.99	0.98	0.98	0.98	0.97	0.96	0.96	0.95	0.94	0.93	0.92
	PMVMH (ALF)	0.99	0.99	0.98	0.98	0.97	0.97	0.96	0.95	0.94	0.93	0.92	0.91
EB	MVMH (SLF)	0.98	0.98	0.97	0.97	0.96	0.95	0.94	0.93	0.92	0.91	0.90	0.89
	PMVMH (SLF)	0.98	0.98	0.97	0.97	0.96	0.95	0.94	0.93	0.92	0.91	0.90	0.89
	MVMH (ALF)	0.99	0.98	0.98	0.97	0.97	0.96	0.95	0.94	0.93	0.92	0.91	0.90
	PMVMH (ALF)	0.99	0.99	0.98	0.98	0.97	0.97	0.96	0.95	0.94	0.93	0.92	0.91
SCS	MVMH (SLF)	0.97	0.97	0.96	0.96	0.95	0.94	0.94	0.93	0.92	0.91	0.89	0.87
	PMVMH (SLF)	0.97	0.97	0.96	0.96	0.95	0.94	0.94	0.93	0.92	0.91	0.89	0.87
	MVMH (ALF)	0.97	0.97	0.96	0.95	0.95	0.94	0.93	0.93	0.92	0.91	0.89	0.87
	PMVMH (ALF)	0.98	0.97	0.96	0.95	0.95	0.94	0.93	0.93	0.92	0.91	0.89	0.87
EDD	MVMH (SLF)	0.99	0.99	0.99	0.99	0.99	0.99	0.98	0.98	0.97	0.97	0.96	0.95
	PMVMH (SLF)	0.99	0.99	0.99	0.99	0.99	0.99	0.98	0.98	0.97	0.97	0.96	0.95
	MVMH (ALF)	0.99	0.99	0.99	0.99	0.98	0.98	0.98	0.98	0.97	0.97	0.96	0.95
	PMVMH (ALF)	0.99	0.99	0.99	0.99	0.98	0.98	0.98	0.98	0.97	0.97	0.96	0.95
EH	MVMH (SLF)	1.00	1.00	1.00	0.99	0.99	0.99	0.99	0.99	0.99	0.98	0.98	0.98
	PMVMH (SLF)	1.00	1.00	1.00	0.99	0.99	0.99	0.99	0.99	0.99	0.98	0.98	0.98
	MVMH (ALF)	1.00	1.00	1.00	0.99	0.99	0.99	0.99	0.99	0.99	0.98	0.98	0.98
	PMVMH (ALF)	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.98	0.98	0.98	0.98
LED	MVMH (SLF)	0.94	0.94	0.94	0.92	0.90	0.85	0.81	0.76	0.73	0.69	0.66	0.63
	PMVMH (SLF)	0.94	0.94	0.94	0.92	0.90	0.85	0.81	0.76	0.73	0.69	0.66	0.63
	MVMH (ALF)	0.95	0.95	0.95	0.94	0.91	0.87	0.83	0.79	0.75	0.72	0.69	0.66
	PMVMH (ALF)	0.95	0.95	0.95	0.94	0.91	0.88	0.83	0.79	0.75	0.72	0.69	0.65

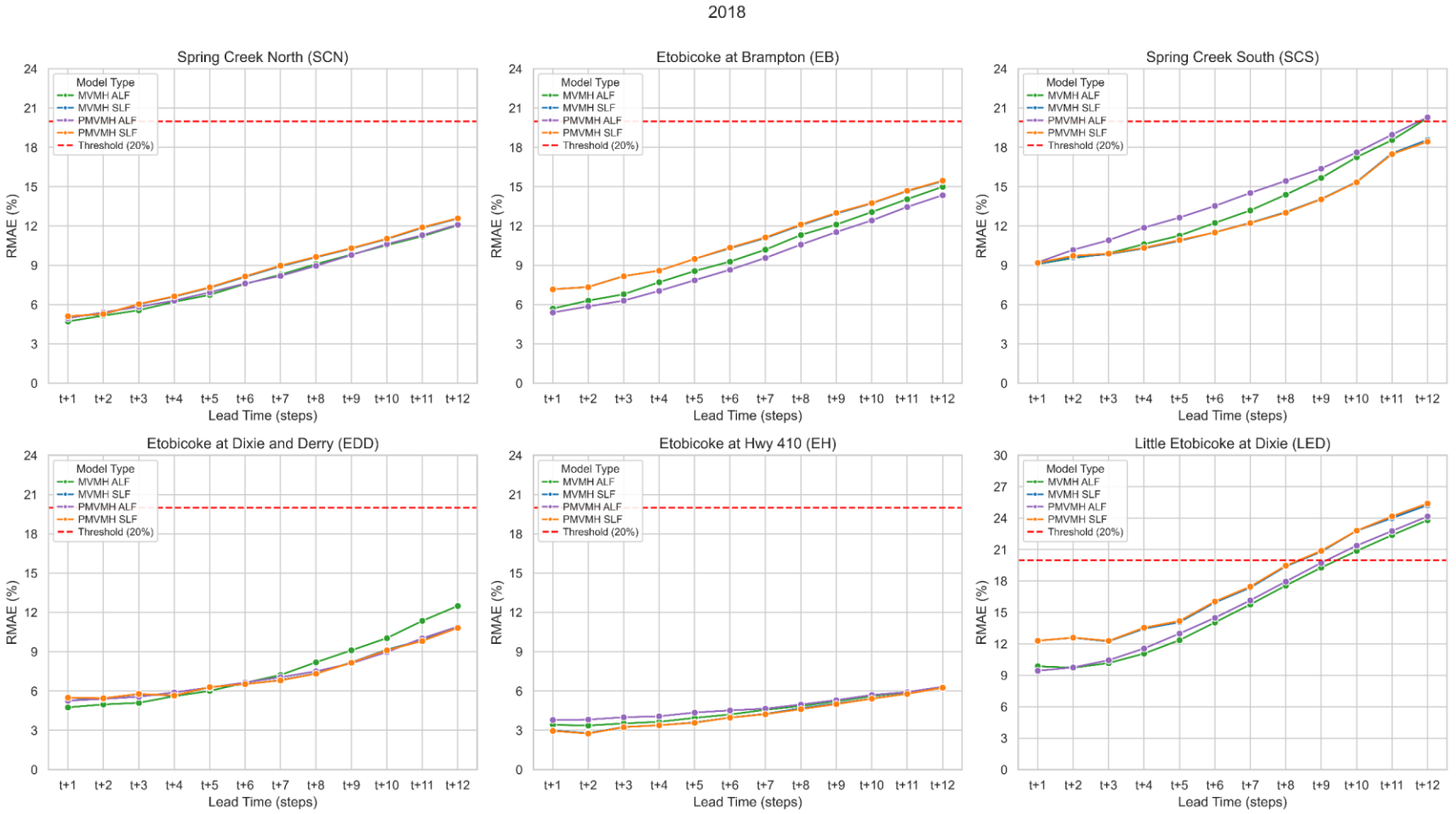


Figure 18: Relative prediction error (%) of multivariate models on 2018 dataset

usability. The NSE index also supports the improvement with the ALF strategy since multivariate models under ALF remained 'good' for two and a half hours, while only up to 2 hours and 15 minutes under SLF (Table 12). Furthermore, Figure 17 also demonstrates that the MVMH with ALF captured the high-flow peaks more effectively than MVMH-SLF around early October, highlighting that ALF strategy directed greater learning emphasis toward the hydrological regime at EB site.

Table 13: NSE Index of Multivariate Models with SLF & ALF (2019)

Sites	Models	Leadtime											
		1	2	3	4	5	6	7	8	9	10	11	12
SCN	MVMH (SLF)	0.98	0.98	0.98	0.98	0.98	0.97	0.97	0.96	0.95	0.94	0.93	0.92
	PMVMH (SLF)	0.98	0.98	0.98	0.98	0.98	0.97	0.97	0.96	0.95	0.94	0.93	0.92
	MVMH (ALF)	0.99	0.99	0.98	0.98	0.98	0.97	0.96	0.95	0.94	0.93	0.92	0.91
	PMVMH (ALF)	0.99	0.99	0.99	0.98	0.98	0.97	0.96	0.96	0.95	0.94	0.92	0.91
EB	MVMH (SLF)	0.98	0.97	0.96	0.95	0.93	0.90	0.87	0.84	0.81	0.78	0.74	0.70
	PMVMH (SLF)	0.98	0.97	0.96	0.95	0.93	0.90	0.87	0.84	0.81	0.78	0.74	0.70
	MVMH (ALF)	0.98	0.98	0.96	0.95	0.93	0.91	0.88	0.85	0.83	0.79	0.76	0.72
	PMVMH (ALF)	0.98	0.98	0.97	0.95	0.93	0.91	0.88	0.85	0.82	0.79	0.75	0.72
SCS	MVMH (SLF)	0.97	0.97	0.98	0.98	0.98	0.97	0.95	0.92	0.89	0.84	0.80	0.75
	PMVMH (SLF)	0.97	0.97	0.98	0.98	0.98	0.97	0.95	0.92	0.88	0.84	0.80	0.75
	MVMH (ALF)	0.96	0.96	0.97	0.97	0.97	0.96	0.94	0.92	0.88	0.84	0.79	0.75
	PMVMH (ALF)	0.96	0.97	0.97	0.97	0.97	0.96	0.94	0.92	0.88	0.84	0.79	0.75
EDD	MVMH (SLF)	0.98	0.98	0.98	0.98	0.98	0.97	0.97	0.97	0.96	0.95	0.94	0.92
	PMVMH (SLF)	0.98	0.98	0.98	0.98	0.98	0.97	0.97	0.97	0.96	0.95	0.94	0.92
	MVMH (ALF)	0.98	0.98	0.98	0.97	0.97	0.96	0.96	0.95	0.95	0.93	0.92	0.91
	PMVMH (ALF)	0.98	0.98	0.98	0.98	0.97	0.97	0.97	0.96	0.95	0.94	0.93	0.91
EH	MVMH (SLF)	0.96	0.95	0.94	0.93	0.91	0.90	0.88	0.87	0.85	0.84	0.83	0.82
	PMVMH (SLF)	0.96	0.95	0.94	0.93	0.91	0.90	0.88	0.87	0.86	0.84	0.83	0.82
	MVMH (ALF)	0.97	0.96	0.96	0.95	0.95	0.94	0.93	0.93	0.92	0.92	0.92	0.91
	PMVMH (ALF)	0.96	0.96	0.96	0.95	0.95	0.94	0.94	0.94	0.93	0.93	0.92	0.92
WSC	MVMH (SLF)	0.95	0.93	0.90	0.87	0.81	0.76	0.70	0.65	0.61	0.57	0.53	0.50
	PMVMH (SLF)	0.95	0.93	0.90	0.87	0.81	0.76	0.70	0.65	0.61	0.57	0.53	0.50
	MVMH (ALF)	0.94	0.93	0.90	0.87	0.82	0.77	0.72	0.67	0.63	0.59	0.55	0.52
	PMVMH (ALF)	0.94	0.92	0.90	0.87	0.82	0.77	0.71	0.67	0.62	0.59	0.55	0.51
LED	MVMH (SLF)	0.95	0.95	0.94	0.92	0.88	0.83	0.77	0.71	0.65	0.59	0.55	0.51
	PMVMH (SLF)	0.95	0.95	0.94	0.92	0.88	0.83	0.77	0.71	0.65	0.59	0.55	0.51
	MVMH (ALF)	0.94	0.94	0.94	0.92	0.88	0.84	0.78	0.71	0.66	0.60	0.56	0.52
	PMVMH (ALF)	0.94	0.94	0.93	0.91	0.88	0.83	0.77	0.71	0.66	0.60	0.56	0.52

The 2019 results indicate that ALF was beneficial for the EB site, where both MVMH and PMVMH with ALF were reliable for the entire forecasting window, extending reliability by 15 minutes compared to their SLF counterparts. This performance upgrade suggests that ALF was successful in reallocating optimization emphasis toward the flashy responsive site. Similar to the 2017 dataset, the SLF models performed better than ALF models at the LED and SCS site (Figure 19), highlighting that the adaptive weighting mechanism prioritized EB and EH sites. The NSE index of all the models remained above its 'good' performance threshold for up to three-hour horizon at SCS and only up to two hour lead time at LED

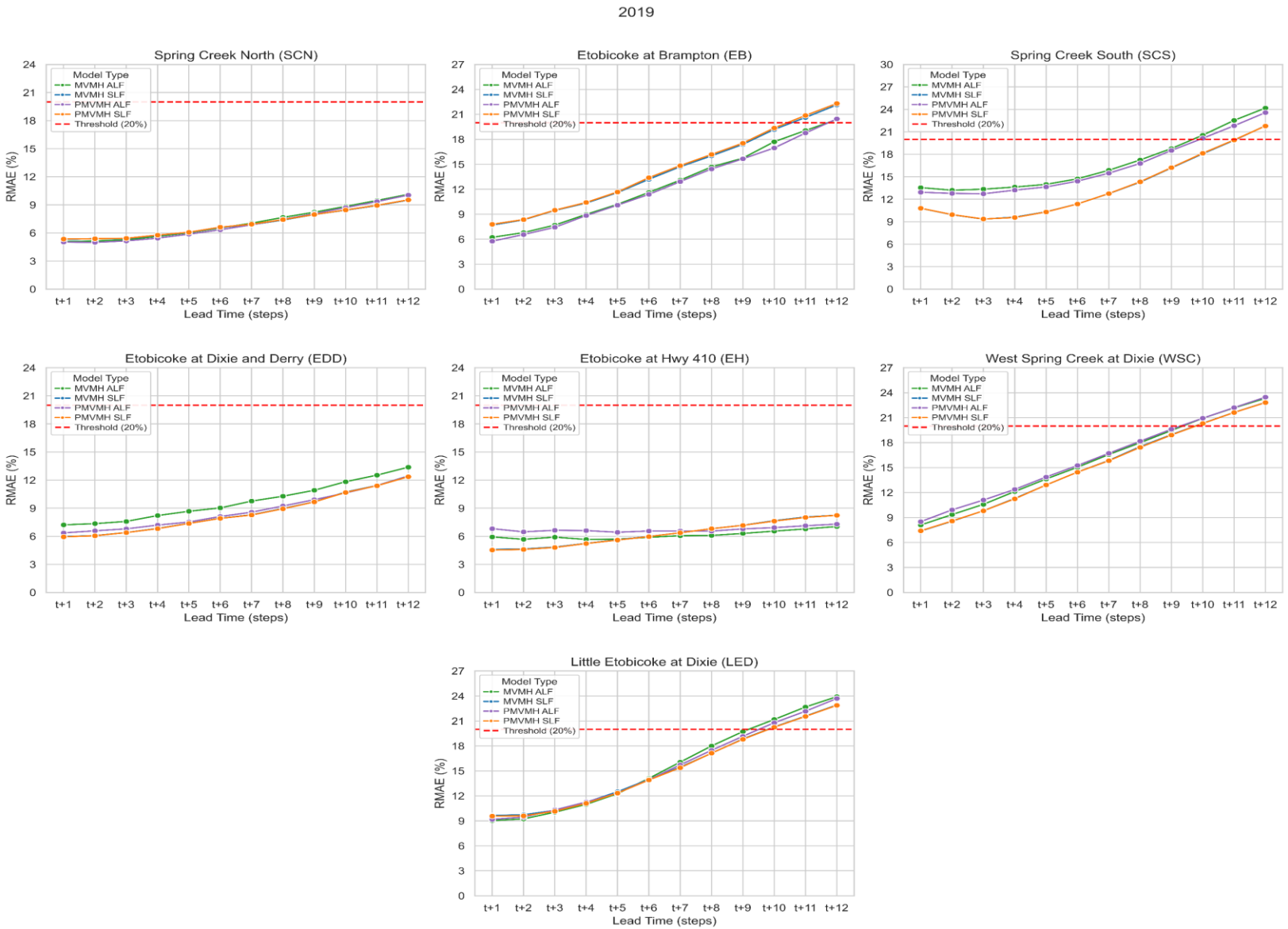


Figure 19: Relative prediction error (%) of multivariate models on 2019 dataset

Table 14: NSE Index of Multivariate Models with SLF & ALF (2020)

Sites	Models	Leadtime											
		1	2	3	4	5	6	7	8	9	10	11	12
SCN	MVMH (SLF)	0.99	0.98	0.98	0.97	0.97	0.96	0.95	0.95	0.94	0.93	0.92	0.91
	PMVMH (SLF)	0.98	0.98	0.98	0.97	0.97	0.96	0.96	0.95	0.94	0.93	0.92	0.91
	MVMH (ALF)	0.98	0.98	0.98	0.97	0.97	0.97	0.96	0.95	0.95	0.94	0.93	0.92
	PMVMH (ALF)	0.98	0.98	0.98	0.97	0.97	0.97	0.96	0.95	0.95	0.94	0.93	0.92
EB	MVMH (SLF)	0.97	0.96	0.95	0.95	0.93	0.92	0.90	0.88	0.86	0.83	0.81	0.79
	PMVMH (SLF)	0.97	0.97	0.95	0.94	0.93	0.92	0.90	0.87	0.85	0.83	0.80	0.78
	MVMH (ALF)	0.97	0.96	0.95	0.94	0.94	0.92	0.90	0.88	0.86	0.84	0.82	0.79
	PMVMH (ALF)	0.96	0.96	0.95	0.94	0.93	0.92	0.90	0.88	0.86	0.84	0.81	0.79
SCS	MVMH (SLF)	0.96	0.95	0.94	0.93	0.92	0.91	0.88	0.85	0.81	0.77	0.73	0.69
	PMVMH (SLF)	0.94	0.94	0.93	0.92	0.91	0.90	0.87	0.84	0.80	0.76	0.72	0.68
	MVMH (ALF)	0.95	0.94	0.94	0.93	0.92	0.91	0.89	0.86	0.82	0.78	0.74	0.70
	PMVMH (ALF)	0.95	0.94	0.94	0.93	0.92	0.90	0.88	0.86	0.82	0.78	0.73	0.69
EDD	MVMH (SLF)	0.98	0.97	0.96	0.96	0.94	0.93	0.92	0.90	0.88	0.85	0.82	0.78
	PMVMH (SLF)	0.96	0.95	0.94	0.93	0.92	0.90	0.88	0.85	0.83	0.81	0.76	0.73
	MVMH (ALF)	0.98	0.97	0.97	0.96	0.94	0.93	0.91	0.90	0.87	0.85	0.81	0.78
	PMVMH (ALF)	0.98	0.97	0.96	0.96	0.95	0.93	0.91	0.89	0.87	0.84	0.80	0.77
EH	MVMH (SLF)	0.99	0.99	0.99	0.99	0.99	0.99	0.98	0.98	0.98	0.98	0.97	0.97
	PMVMH (SLF)	0.99	0.99	0.99	0.99	0.99	0.98	0.98	0.98	0.98	0.97	0.97	0.97
	MVMH (ALF)	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.98	0.98	0.98	0.98	0.97
	PMVMH (ALF)	0.99	0.99	0.99	0.99	0.99	0.99	0.98	0.98	0.98	0.98	0.97	0.97
WSC	MVMH (SLF)	0.92	0.90	0.88	0.86	0.82	0.78	0.74	0.70	0.66	0.62	0.58	0.54
	PMVMH (SLF)	0.91	0.89	0.87	0.84	0.80	0.75	0.71	0.67	0.63	0.59	0.56	0.50
	MVMH (ALF)	0.92	0.90	0.89	0.86	0.82	0.78	0.74	0.70	0.66	0.63	0.59	0.54
	PMVMH (ALF)	0.92	0.90	0.88	0.86	0.82	0.77	0.73	0.69	0.65	0.61	0.58	0.53
LED	MVMH (SLF)	0.93	0.93	0.92	0.92	0.90	0.87	0.84	0.80	0.77	0.73	0.70	0.67
	PMVMH (SLF)	0.92	0.92	0.92	0.91	0.89	0.86	0.84	0.80	0.76	0.73	0.70	0.66
	MVMH (ALF)	0.94	0.93	0.93	0.92	0.91	0.88	0.85	0.82	0.79	0.75	0.72	0.69
	PMVMH (ALF)	0.94	0.94	0.93	0.92	0.91	0.88	0.85	0.82	0.78	0.75	0.71	0.68

site (Table 13). Model results on 2020 dataset reveal another important finding. Allotted values of NSE index for MVMH with SLF and ALF, PMVMH-ALF at EDD, remained above the 'very good' threshold up to 2 hours and 45 minutes in 2020 (Figure 20), and the same models, along with PMVMH-SLF, remained above the 'good' threshold for the 3-hour horizon (Table 14). The NSE results show good performance for longer lead times, but the RMAE reliability window indicates that the forecasting window shortens at the EDD site suggests a rapidly varying response pattern in a highly urbanized setting, which contributed to increased magnitude error. The MVMH with SLF and ALF remains reliable for only one and a half hours, and PMVMH-ALF only for 1 hour and 15 minutes at EDD. Subsequently, the MVMH model under ALF generated the best performance at the WSC site, which is a mid-cross-section between SCN and SCS, extending from 1 hour 15 minutes under MVMH-SLF to one and a half hours on the 2020 dataset. However, the multivariate models under ALF underperformed compared to their SLF counterparts at SCS site. This reflects the intended behavior of ALF, providing importance towards rapid-response sites, WSC, EH, and SCN, and compromised reliability at SCS.

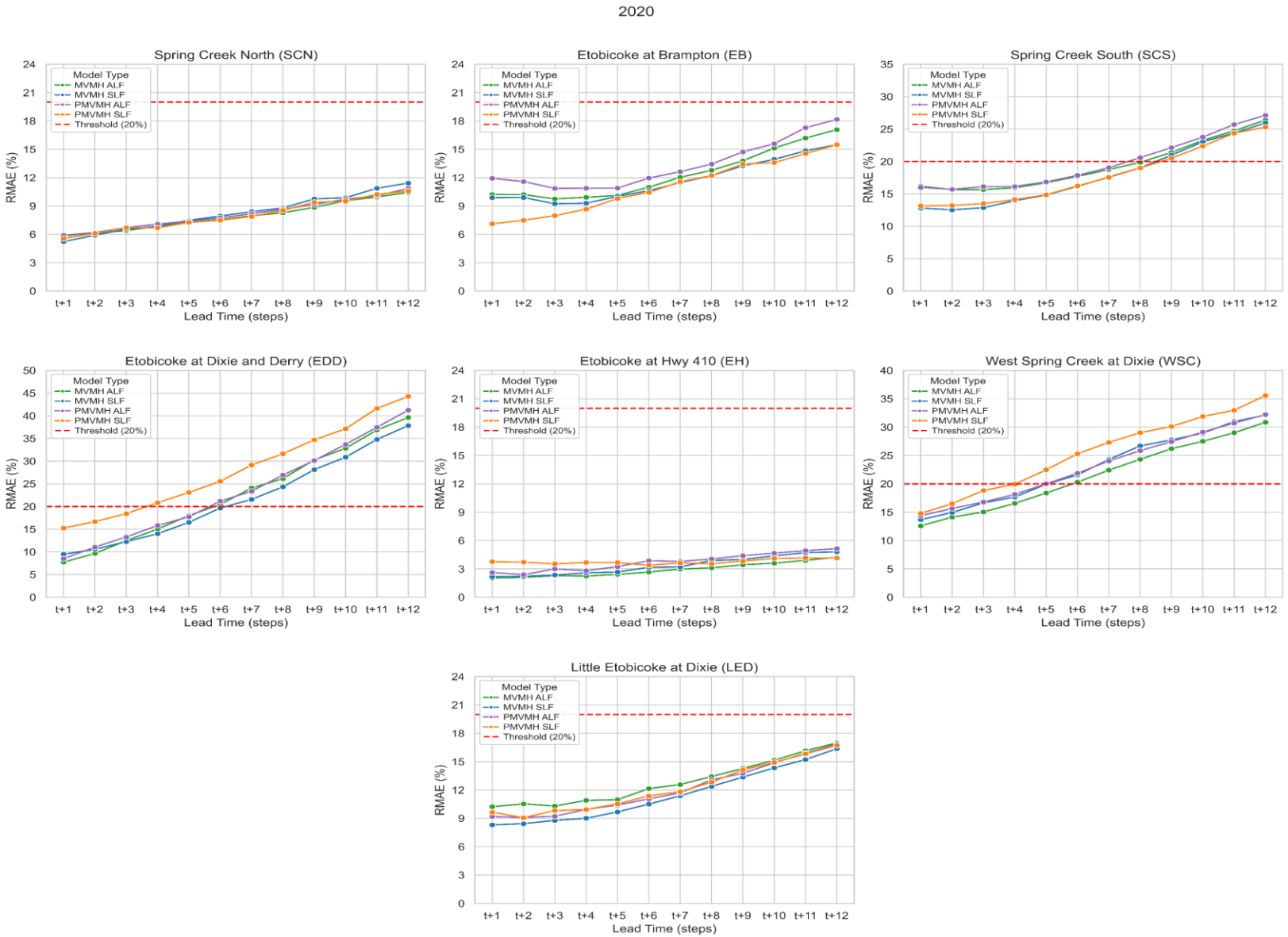


Figure 20: Relative prediction error (%) of multivariate models on 2020 dataset

The comparison between the multivariate LSTM and the peephole multivariate model highlights that the inclusion of peephole connections was beneficial in both the SLF and ALF settings. The sites where the model forecast depends on retaining memory of earlier high-flow events, the peephole model comes into play. The PMVMH-SLF model extended the reliable forecasting horizon from 1 hour and 45 minutes under the MVMH-SLF model to two hours on the 2015 dataset. This suggests that direct gate access to the cell state improved the model's ability to retain long-term runoff information, which was helpful for capturing the delayed translation of flow at this rapid-response site. Similarly, at EB site, the PMVMH-ALF model extended reliability to one hour compared with only 15 min for MVMH-ALF model. Later, PMVMH-ALF further extended the reliable horizon from 2 hours and 15 minutes to two and a half hours on the 2019 dataset. This improvement indicates that the combination of adaptive weighting and peephole connection was particularly effective in this highly urbanized, rapidly responding cross-section. While the peephole setting enhances the timing sensitivity and memory retention, the adaptive loss strategy improved the optimization fairness by providing importance towards flashy responsive stations.

The multivariate models MVMH and PMVMH, requires building only one per year because the model utilizes data from all available gauges for a given year to generate synchronized multi-horizon outputs for all stream gauges at once, supporting watershed approach. This watershed-level approach can effectively capture the spatiotemporal dependencies among various stream branches and the local runoff heterogeneity that single station forecasts often overlook. This multivariate approach also reflects the watershed approach and aligns with the Ontario Conservation Authorities Act framework [93], where the mandate emphasizes managing natural resources and hazards on a watershed basis.

6.3 Deep Learning Framework

Based on the computational experiments and investigated results, the proposed Deep Learning framework (Figure 21) can be generalized beyond the present case study since the core framework is built around steps that are not unique to Etobicoke Creek watershed. It is most directly transferable to other monitored urbanized or natural watersheds for short-term forecasting. The framework begins with data selection, hydrometric and environmental data can be selected, prepared and the time series data can then be transformed into the sequence structure suitable for sequential deep learning models. Based on the problem objective, univariate or multivariate forecasting can be utilized. Subsequently, similar or different variants of deep learning models can be applied under common forecasting settings. Traditional LSTM has been reported to be very successful in hydrological forecasting. Since retaining timing-sensitive event information can also be crucial, peephole LSTM can be used. Following the selection of recurrent models, different weighting loss strategies can be used to compare different model performance and to mitigate task imbalance caused by shared Multi-Task Learning. The univariate multi horizon models are only developed

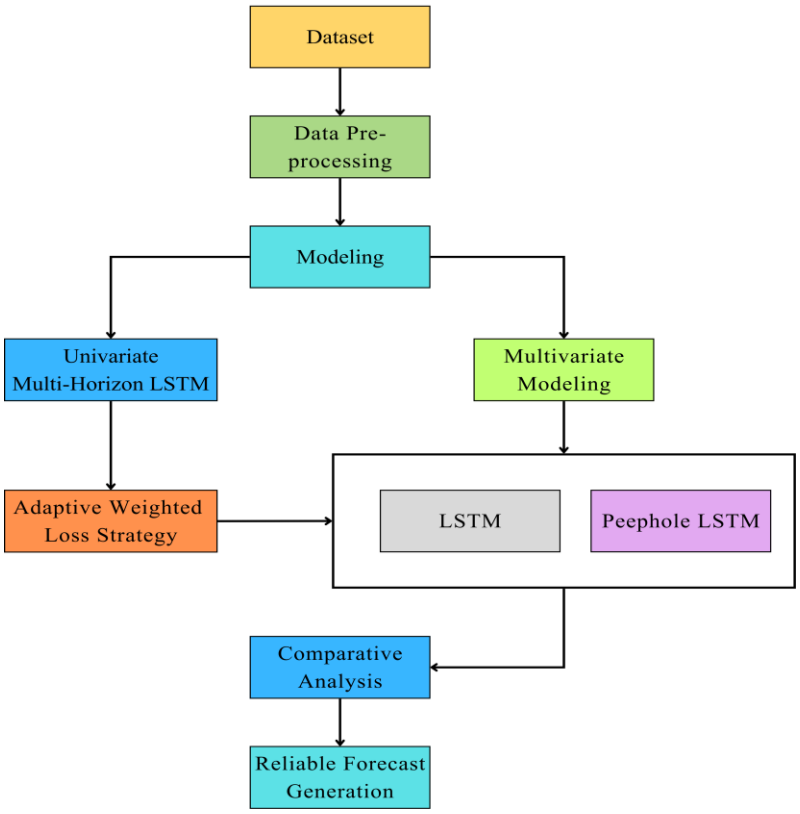


Figure 21: Deep Learning Framework

to investigate site-wise loss contribution to apply the adaptive weighted loss strategy, and the model will be counted as a one-time calibration cost. Multivariate modeling supporting watershed level approach is well-suited for operational usage compared to the univariate setting. The proposed Adaptive weighted loss function uses the site difficulty information from univariate models to train one watershed-level model. According to the results, Multivariate LSTM with the adaptive weighting strategy is useful for reallocating optimization pressure towards more flashy sites and addressing task imbalance problem. Whereas the peephole multivariate with adaptive loss is appropriate for sites demonstrating varying spatio-temporal dynamics that also require maintaining long-term high flow event information. Following the construction of the different model types, the model performance should be evaluated using a reliable forecast threshold suitable on watershed specific features and operational requirements.

This Deep Learning framework supporting watershed approach is not restricted to water level forecasting in small urbanized watersheds, instead it can be used for short-term forecasting of hydrological parameters of any natural stream. Particularly, the steps include data pre-processing, time-delay embedding, scaling, modeling, and reliability-based evaluation can be applied to these forecasting problems. Based on the specific problem domain, minimal adaptation can be required, such as quality control, missing data

handling, or temporal alignment. Therefore, beyond the current case study, the proposed framework can be used for various hydrological forecasting problems and can benefit operational watershed management.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This thesis examined deep learning models of LSTM architecture for generating reliable high-resolution hydrological forecasts in a small, heterogeneous urbanized watershed using data supplied by routine monitoring system. The study evaluated multiple modeling strategies by comparing univariate and multivariate approaches using NSE and Relative MAE metrics. This dual-metric evaluation depicts a clear picture of model predictability and generalization across wet and dry years. The investigated results demonstrate that the LSTM-based models were able to learn the temporal patterns well across sites, which is reflected by NSE index values. The results also uncovered modeling limitations pointing to the utilization of the static loss function on the model training step. A shared static loss function in muti-site forecasting could generate acceptable aggregate performance while concealing site-wise undertraining for sites with flashy responses to water loads.

The proposed watershed-level framework based on adaptive loss strategy using solely data supplied by routine monitoring systems supports the development of scalable deep learning models generating reproducible results with enhancing prediction accuracy. The reproducibility aspect of the framework was achieved by reproducible execution procedures, standardized chronological dataset splits, systematic hyperparameter tuning, and comprehensive experiment logging, making the appropriate model configuration selection traceable. The proposed Adaptive Weighted Loss Function (ALF) was introduced within the framework, which dynamically controls the contribution of each site to the loss function by continuously re-estimating site-wise validation error. The ALF approach reallocates the gradient emphasis toward under-trained sites, which resulted in enhancing prediction reliability of the multivariate learning. Furthermore, the adaptive weighted loss framework, combined with multivariate multi-horizon models, showcased significant improvement in forecast reliability across all investigated sites in year-wise and

cross-year settings. The experiments of using the adaptive approach for year-wise and cross-year forecasting suggest that this strategy is well-suited for multi-site forecasting.

The proposed framework was tested on high-resolution multi-site rainfall and water-level observations to capture spatio-temporal dependence on a watershed. The multivariate multi-horizon models produced synchronized three-hour forecasts across all available cross-sections within a single deployable model for each year, making the framework more scalable and operationally efficient than maintaining multiple site-specific univariate models. Overall, this thesis demonstrates a watershed approach framework which is practical for high-frequency monitoring networks, and the usage of the adaptive loss strategy can extend the forecast horizon for hydrologically difficult sites by explicitly addressing the shared static loss function in multi-site forecasting.

7.2 Future Work

The proposed framework in this study was developed for high-resolution, short-term hydrological forecasting. Therefore, to further improve and extend the applicability of the framework, we suggest the following steps and research directions:

1. In this thesis, a common time-delay embedding was used across all sites, whereas site-specific or variable-length time-delay embedding across different locations should be investigated to better reflect local response times and hydrological regimes.

2. This thesis deals with short-term, precipitation-driven predictions, and the framework was not tested for seasonal floods or flood events developing over weeks or months. Although the proposed ALF is expected to remain useful, further work should extend the framework to long-term hydrological prediction.

Bibliography

[1] P. A. Wilderer, *Treatise on water science*. Newnes, 2010.

[2] Canadian Climate Institute, "Climate Change and Floods," Fact sheet, 2024. [Online]. Available: https://climateinstitute.ca/wp-content/uploads/2024/09/Fact-sheet_-Floods_CanadianClimateInstitute.pdf, 2024. Accessed: Jan. 15, 2026

[3] Public Safety Canada. "Canadian Disaster Database: Hurricane Hazel (Event ID 696)." *Government of Canada.* [Online]. Available: https://cdd.publicsafety.gc.ca/dtpg-eng.aspx?cultureCode=en-Ca&provinces=9&normalizedCostYear=1&dynamic=false&eventId=696. Accessed: Jan. 15, 2026

[4] O. Anderson, "The box-jenkins approach to time series analysis," *RAIRO-Operations Research*, vol. 11, no. 1, pp. 3–29, 1977.

[5] J. Ledolter, "Arima models and their use in modelling hydrologic sequences," 1976.

[6] A. McKerchar and J. Delleur, "Application of seasonal parametric linear stochastic models to monthly flow data," *Water Resources Research*, vol. 10, no. 2, pp. 246–255, 1974.

[7] J. D. Salas, *Applied modeling of hydrologic time series*. Water Resources Publication, 1980.

[8] D. J. Noakes, A. I. McLeod, and K. W. Hipel, "Forecasting monthly riverflow time series," *International Journal of Forecasting*, vol. 1, no. 2, pp. 179–190, 1985.

[9] J. Wu, J. Yin, Y. Hao, Y. Liu, Y. Fan, X. Huo, Y. Liu, and T.-C. J. Yeh, "The role of anthropogenic activities in karst spring discharge volatility," *Hydrological Processes*, vol. 29, no. 13, pp. 2855–2866, 2015.

[10] Modarres, Reza, and Taha BMJ Ouarda. "Modelling heteroscedasticty of streamflow times series." *Hydrological sciences journal* 58, no. 1 (2013): 54-64.

[11] Gegenleithner, Sebastian, Manuel Pirker, Clemens Dorfmann, Roman Kern, and Josef Schneider. "Long short-term memory networks for enhancing real-time flood forecasts: a case study for an underperforming hydrologic model." *Hydrology and Earth System Sciences* 29, no. 7 (2025): 1939-1962.

[12] Wang, Huimin, Songbai Song, Gengxi Zhang, and Olusola O. Ayantoboc. "Predicting daily streamflow with a novel multi-regime switching ARIMA-MS-GARCH model." *Journal of Hydrology: Regional Studies* 47 (2023): 101374.

[13] Yu, Qiutong, Bryan A. Tolson, Hongren Shen, Ming Han, Juliane Mai, and Jimmy Lin. "Enhancing long short-term memory (LSTM)-based streamflow prediction with a spatially distributed approach." *Hydrology and Earth System Sciences* 28, no. 9 (2024): 2107-2122.

[14] Zhang, Lin, Huapeng Qin, Junqi Mao, Xiaoyan Cao, and Guangtao Fu. "High temporal resolution urban flood prediction using attention-based LSTM models." *Journal of Hydrology* 620 (2023): 129499.

[15] A. Minns and M. Hall, "Artificial neural networks as rainfall-runoff models," *Hydrological sciences journal*, vol. 41, no. 3, pp. 399–417, 1996.

[16] K. W. Kang, J. H. Kim, C. Y. Park, and K. J. Ham, "Evaluation of hydrological forecasting system based on neural network model," in *Proc. 25th Congr. Int. Assoc. Hydraul. Res.*, 1993.

[17] Lin, Gwo-Fong, Guo-Rong Chen, Pei-Yu Huang, and Yang-Ching Chou. "Support vector machine-based models for hourly reservoir inflow forecasting during typhoon-warning periods." *Journal of hydrology* 372, no. 1-4 (2009): 17-29.

[18] C. Dawson and R. Wilby, "Hydrological modelling using artificial neural networks," *Progress in physical Geography*, vol. 25, no. 1, pp. 80–108, 2001.

[19] Y. B. Dibike and D. P. Solomatine, "River flow forecasting using artificial neural networks," *Physics and Chemistry of the Earth, Part B: Hydrology, Oceans and Atmosphere*, vol. 26, no. 1, pp. 1–7, 2001.

[20] Wakatsuki, Yuki, Hideaki Nakane, and Tempei Hashino. "River stage modeling with a deep neural network using long-term rainfall time series as input data: application to the Shimanto-River watershed." *Water* 14, no. 3 (2022): 452.

[21] L. Li, Y. Dai, Z. Wei, W. Shangguan, Y. Zhang, N. Wei, and Q. Li, "Enforcing water balance in multitask deep learning models for hydrological forecasting," *Journal of Hydrometeorology*, vol. 25, no. 1, pp. 89–103, 2024.

[22] K. Thirumalaiah and M. C. Deo, "Hydrological forecasting using neural networks," *Journal of Hydrologic Engineering*, vol. 5, no. 2, pp. 180–189, 2000.

[23] S. J. Burian, S. R. Durrans, S. Tomi´c, R. L. Pimmel, and C. N. Wai, "Rainfall disaggregation using artificial neural networks," *Journal of Hydrologic Engineering*, vol. 5, no. 3, pp. 299–307, 2000.

[24] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[25] Bowden, Gavin J., Graeme C. Dandy, and Holger R. Maier. "Input determination for neural network models in water resources applications. Part 1—background and methodology." *Journal of Hydrology* 301, no. 1-4 (2005): 75-92.

[26] ASCE Task Committee on Application of Artificial Neural Networks in Hydrology. "Artificial

neural networks in hydrology. II: Hydrologic applications." *Journal of Hydrologic Engineering* 5, no. 2 (2000): 124-137.

[27] S. Kabir, S. Patidar, X. Xia, Q. Liang, J. Neal, and G. Pender, "A deep convolutional neural network model for rapid prediction of fluvial flood inundation," *Journal of Hydrology*, vol. 590, p. 125481, 2020.

[28] Lin, Kangling, Sheng Sheng, Yanlai Zhou, Feng Liu, Zhiyu Li, Hua Chen, Chong-Yu Xu, Jie Chen, and Shenglian Guo. "The exploration of a temporal convolutional network combined with encoder-decoder framework for runoff forecasting." *Hydrology Research* 51, no. 5 (2020): 1136-1149.

[29] Xu, Yuanhao, Caihong Hu, Qiang Wu, Zhichao Li, Shengqi Jian, and Youqian Chen. "Application of temporal convolutional network for flood forecasting." *Hydrology Research* 52, no. 6 (2021): 1455-1468.

[30] Li, Xia, Wei Xu, Minglei Ren, Yanan Jiang, and Guangtao Fu. "Hybrid CNN-LSTM models for river flow prediction." *Water Supply* 22, no. 5 (2022): 4902-4919.

[31] J. Lee, E.-S. Chung, S. Kim, and D. Kim, "Streamflow forecasting in ungauged basins with cnn-lstm and radar-based precipitation," *Journal of Hydro-environment Research*, vol. 60, p. 100666, 2025.

[32] Xie, Yutong, Wei Sun, Miaomiao Ren, Shu Chen, Zexi Huang, and Xingyou Pan. "Stacking ensemble learning models for daily runoff prediction using 1D and 2D CNNs." *Expert Systems with Applications* 217 (2023): 119469.

[33] Ishida, Kei, Ali Ercan, Takeyoshi Nagasato, Masato Kiyama, and Motoki Amagasaki. "Use of one-dimensional CNN for input data size reduction in LSTM for improved computational efficiency and accuracy in hourly rainfall-runoff modeling." *Journal of Environmental Management* 359 (2024): 120931.

[34] Al-Selwi, Safwan Mahmood, Mohd Fadzil Hassan, Said Jadid Abdulkadir, Amgad Muneer, Ebrahim Hamid Sumiea, Alawi Alqushaibi, and Mohammed Gamal Ragab. "RNN-LSTM: From applications to modeling techniques and beyond—Systematic review." *Journal of King Saud University-Computer and Information Sciences* 36, no. 5 (2024): 102068.

[35] F. Kratzert, D. Klotz, C. Brenner, K. Schulz, and M. Herrnegger, "Rainfall–runoff modelling using long short-term memory (lstm) networks," *Hydrology and Earth System Sciences*, vol. 22, no. 11, pp. 6005–6022, 2018.

[36] W. Xu, Y. Jiang, X. Zhang, Y. Li, R. Zhang, and G. Fu, "Using long short-term memory networks for river flow prediction," *Hydrology Research*, vol. 51, no. 6, pp. 1358–1376, 2020.

[37] E. Rozos, P. Dimitriadis, and V. Bellos, “Machine learning in assessing the performance of hydrological models,” *Hydrology*, vol. 9, no. 1, p. 5, 2021.

[38] L. Girihagama, M. Naveed Khaliq, P. Lamontagne, J. Perdikaris, R. Roy, L. Sushama, and A. Elshorbagy, “Streamflow modelling and forecasting for canadian watersheds using lstm networks with attention mechanism,” *Neural Computing and Applications*, vol. 34, no. 22, pp. 19995–20015, 2022.

[39] R. Arsenault, J.-L. Martel, F. Brunet, F. Brissette, and J. Mai, “Continuous streamflow prediction in ungauged basins: long short-term memory neural networks clearly outperform traditional hydrological models,” *Hydrology and Earth System Sciences*, vol. 27, no. 1, pp. 139–157, 2023.

[40] A. Gharehbaghi, R. Ghasemlounia, S. Daneshvar, and F. Ahmadi, “Developing a novel layer network structure for a lstm model to predict mean monthly river streamflow,” *Applied Water Science*, vol. 15, no. 7, p. 174, 2025.

[41] J. P. Nair and M. Vijaya, “Intelligent systems and applications in engineering,”

[42] S. R. Koya and T. Roy, “Temporal fusion transformers for streamflow prediction: Value of combining attention with recurrence,” *Journal of Hydrology*, vol. 637, p. 131301, 2024.

[43] B. Z. Demiray and I. Demir, “Towards generalized hydrological forecasting using transformer models for 120-hour streamflow prediction,” *arXiv preprint arXiv:2406.07484*, 2024.

[44] Bai, Yun, Nejc Bezak, Bo Zeng, Chuan Li, Klaudija Sapač, and Jin Zhang. "Daily runoff forecasting using a cascade long short-term memory model that considers different variables." *Water Resources Management* 35, no. 4 (2021): 1167-1181.

[45] A. Karpatne, G. Atluri, J. H. Faghmous, M. Steinbach, A. Banerjee, A. Ganguly, S. Shekhar, N. Samatova, and V. Kumar, “Theory-guided data science: A new paradigm for scientific discovery from data,” *IEEE Transactions on knowledge and data engineering*, vol. 29, no. 10, pp. 2318–2331, 2017.

[46] W. L. Zhao, P. Gentine, M. Reichstein, Y. Zhang, S. Zhou, Y. Wen, C. Lin, X. Li, and G. Y. Qiu, “Physics-constrained machine learning of evapotranspiration,” *Geophysical Research Letters*, vol. 46, no. 24, pp. 14496–14507, 2019.

[47] E. De Bézenac, A. Pajot, and P. Gallinari, “Deep learning for physical processes: Incorporating prior scientific knowledge,” *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2019, no. 12, p. 124009, 2019.

[48] N. Wang, D. Zhang, H. Chang, and H. Li, “Deep learning of subsurface flow via theory-guided neural network,” *Journal of Hydrology*, vol. 584, p. 124700, 2020.

[49] S. Jiang, Y. Zheng, and D. Solomatine, “Improving ai system awareness of geoscience knowledge: Symbiotic integration of physical approaches and deep learning,” *Geophysical Research Letters*, vol. 47, no. 13, p. e2020GL088229, 2020.

[50] Z. Xiang, W. Peng, X. Liu, and W. Yao, “Self-adaptive loss balanced physics-informed neural networks,” *Neurocomputing*, vol. 496, pp. 11–34, 2022.

[51] Abdoulhalik, Antoifi, and Ashraf A. Ahmed. "A comparative analysis of advanced machine learning techniques for river streamflow time-series forecasting." *Sustainability* 16, no. 10 (2024): 4005.

[52] Zhang, Yue, Zhaohui Gu, Jesse Van Griensven Thé, Simon X. Yang, and Bahram Gharabaghi. "The discharge forecasting of multiple monitoring station for humber river by hybrid LSTM models." *Water* 14, no. 11 (2022): 1794.

[53] Li, Jiayuan, and Xing Yuan. "Daily streamflow forecasts based on cascade long short-term memory (LSTM) model over the Yangtze River Basin." *Water* 15, no. 6 (2023): 1019.

[54] Yu, Tianhe, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. "Gradient surgery for multi-task learning." *Advances in neural information processing systems* 33 (2020): 5824-5836.

[55] A. L. Samuel, “Some studies in machine learning using the game of checkers,” *IBM Journal of research and development*, vol. 3, no. 3, pp. 210–229, 1959.

[56] F. Rosenblatt, “The perceptron: a probabilistic model for information storage and organization in the brain.,” *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[57] G. Shaw, “Donald hebb: The organization of behavior,” in *Brain Theory: Proceedings of the First Trieste Meeting on Brain Theory, October 1–4, 1984*, pp. 231–233, Springer, 1986.

[58] E. B. Baum, “On the capabilities of multilayer perceptrons,” *Journal of complexity*, vol. 4, no. 3, pp. 193–215, 1988.

[59] M. Khashei, A. Z. Hamadani, and M. Bijari, “A novel hybrid classification model of artificial neural networks and multiple linear regression models,” *Expert Systems with Applications*, vol. 39, no. 3, pp. 2606–2620, 2012.

[60] A. ZAIDI, “Mathematical justification on the origin of the sigmoid in logistic regression,” *Central European Management Journal*, vol. 30, no. 4, pp. 1327–1337, 2022.

[61] Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors." *nature* 323, no. 6088 (1986): 533-536.

[62] Elman, Jeffrey L. "Finding structure in time." *Cognitive science* 14, no. 2 (1990): 179-211.

[63] Graves, Alex. "Long short-term memory." *Supervised sequence labelling with recurrent neural networks* (2012): 37-45.

[64] K. Fukushima, "Neocognitron: A hierarchical neural network capable of visual pattern recognition," *Neural networks*, vol. 1, no. 2, pp. 119–130, 1988.

[65] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, 1998.

[66] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[67] S. Grossberg, "Recurrent neural networks," *Scholarpedia*, vol. 8, no. 2, p. 1888, 2013.

[68] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[69] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[70] D. Bahdanau, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[71] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[72] Werbos, Paul J. "Backpropagation through time: what it does and how to do it." *Proceedings of the IEEE* 78, no. 10 (2002): 1550-1560.

[73] Bengio, Yoshua, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult." *IEEE transactions on neural networks* 5, no. 2 (1994): 157-166.

[74] Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio. "On the difficulty of training recurrent neural networks." In *International conference on machine learning*, pp. 1310-1318. Pmlr, 2013.

[75] Gers, Felix A., Jürgen Schmidhuber, and Fred Cummins. "Learning to forget: Continual prediction with LSTM." *Neural computation* 12, no. 10 (2000): 2451-2471.

[76] Gers, Felix A., Nicol N. Schraudolph, and Jürgen Schmidhuber. "Learning precise timing with LSTM recurrent networks." *Journal of machine learning research* 3, no. Aug (2002): 115-143.

[77] Santos, André AP, Francisco J. Nogales, and Esther Ruiz. "Comparing univariate and multivariate

models to forecast portfolio value-at-risk." *Journal of financial econometrics* 11, no. 2 (2013): 400-441.

[78] Salinas, David, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. "DeepAR: Probabilistic forecasting with autoregressive recurrent networks." *International journal of forecasting* 36, no. 3 (2020): 1181-1191.

[79] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[80] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323, JMLR Workshop and Conference Proceedings, 2011.

[81] L. Lu, Y. Shin, Y. Su, and G. E. Karniadakis, "Dying relu and initialization: Theory and numerical examples," *arXiv preprint arXiv:1903.06733*, 2019.

[82] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, vol. 4, no. 5, p. 11, 2015.

[83] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[84] S. Li, P. Zhao, H. Zhang, X. Sun, H. Wu, D. Jiao, W. Wang, C. Liu, Z. Fang, J. Xue, *et al.*, "Surge phenomenon in optimal learning rate and batch size scaling," *Advances in Neural Information Processing Systems*, vol. 37, pp. 132722–132746, 2024.

[85] H. Li, G. K. Rajbahadur, D. Lin, C.-P. Bezemer, and Z. M. Jiang, "Keeping deep learning models in check: A history-based approach to mitigate overfitting," *IEEE Access*, vol. 12, pp. 70676–70689, 2024.

[86] Goodfellow, Ian, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*. Vol. 1, no. 2. Cambridge: MIT press, 2016.

[87] Robbins, Herbert, and Sutton Monro. "A stochastic approximation method." *The annals of mathematical statistics* (1951): 400-407.

[88] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich, "GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 794–803.

[89] Y. Li, J. Xu, and D. Anastasiu, "Learning from polar representation: An extreme-adaptive model for long-term time series forecasting," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 38, pp. 171–179, 2024.

[90] M. Kalinowski, D. Mendez, G. Giray, A. P. S. Alves, K. Azevedo, T. Escovedo, *et al.*, "Naming the pain in machine learning-enabled systems engineering," *Information and Software Technology*, vol. 187, Art. no. 107866, 2025.

[91] Toronto and Region Conservation Authority, *Etobicoke Creek Watershed Report Card 2018*, 2018. [Online]. Available: https://reportcard.trca.ca/watershed-report-cards/etobicoke-creek/. Accessed: Dec. 15, 2025.

[92] Toronto and Region Conservation Authority, *Etobicoke Creek Watershed Characterization Report*. Toronto, ON, Canada: Toronto and Region Conservation Authority, 2021. [Online]. Available:https://trcaca.s3.ca-central-1.amazonaws.com/app/uploads/2021/06/29173309/AODA-Final-Watershed-Characterization-Report-ECWP-June-24_21.pdf. Accessed: Dec. 15, 2025,

[93] Conservation Ontario, "95% of ontario's population lives in a watershed managed by a conservation authority." [Online]. Available: https://ontarioconservationareas.ca/about/conservation-authorities. Accessed: Feb. 10, 2026

[94] Toronto and Region Conservation Authority, "Dataset," TRCA Data Portal. [Online]. Available: https://data.trca.ca/dataset. Accessed: Mar. 15, 2025

[95] Cybenko G. Approximation by superpositions of a sigmoidal function. Mathematics of control, signals and systems. 1989 Dec;2(4):303-14.

[96] Jeong, Wooseong, and Kuk-Jin Yoon. "Quantifying task priority for multi-task optimization." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 363-372. 2024.

[97] T. Gong, T. Lee, C. Stephenson, V. Renduchintala, S. Padhy, A. Ndirango, G. Keskin, and O. H. Elibol, "A comparison of loss weighting strategies for multi-task learning in deep neural networks," *IEEE Access*, vol. 7, pp. 141627–141632, 2019.

[98] ASCE Task Committee on Definition of Criteria for Evaluation of Watershed Models of the Watershed Management Committee, "Criteria for evaluation of watershed models," *Journal of Irrigation and Drainage Engineering*, vol. 119, no. 3, pp. 429–442, 1993.

[99] MLflow, "MLflow - Open source AI platform for agents, LLMs & models," *MLflow*. [Online]. Available: https://mlflow.org/. Accessed: Mar. 8, 2026.

[100] Grammarly, "Grammarly: Free AI writing assistance," *Grammarly*. [Online]. Available: https://www.grammarly.com/. Accessed: Mar. 8, 2026.

[101] QGIS.org, *QGIS Geographic Information System*, ver. 3.40.11, QGIS Association. [Online].

Available: QGIS.org. Accessed: Feb. 1, 2026.

Appendices

Appendix A

Exploratory Computational Analysis with Time Delay Embedding

Table A.1: Comparing Time Embedding of Models in 2013

Lead Time	2 Hour Embedding			3 Hour Embedding		
	MVMH			MVMH		
	SCN	SCS	EDD	SCN	SCS	EDD
1	0.1252	0.1395	0.0942	0.1249	0.1705	0.0924
2	0.1269	0.1414	0.0902	0.1197	0.1624	0.0961
3	0.1307	0.1516	0.0975	0.1358	0.1663	0.1311
4	0.1366	0.1657	0.1092	0.1256	0.1710	0.1296
5	0.1410	0.1730	0.1073	0.1292	0.1817	0.1203
6	0.1446	0.1918	0.1179	0.1328	0.1834	0.1253
7	0.1466	0.2150	0.1193	0.1369	0.1991	0.1403
8	0.1469	0.2304	0.1361	0.1421	0.2159	0.1355
9	0.1547	0.2417	0.1421	0.1524	0.2302	0.1748
10	0.1736	0.2540	0.1510	0.1532	0.2520	0.1608
11	0.1693	0.2799	0.1575	0.1568	0.2766	0.1717
12	0.1814	0.2979	0.1646	0.1648	0.2899	0.1736

Table A.2: Comparing Time Embedding of Models in 2014

Lead Time	2 Hour Embedding			3 Hour Embedding		
	MVMH			MVMH		
	SCN	SCS	EDD	SCN	SCS	EDD
1	0.0944	0.3613	0.1410	0.0868	0.3533	0.1562
2	0.0867	0.3628	0.1506	0.0978	0.3393	0.1459
3	0.0856	0.3423	0.1430	0.1087	0.3740	0.1356
4	0.0898	0.3294	0.1420	0.1043	0.3804	0.1385
5	0.0908	0.3408	0.1477	0.0979	0.3662	0.1388
6	0.0975	0.3462	0.1499	0.0941	0.3743	0.1470
7	0.0953	0.3575	0.1633	0.0956	0.4018	0.1580
8	0.0995	0.3489	0.1717	0.1044	0.3955	0.1627
9	0.1002	0.3626	0.1827	0.1036	0.4003	0.1576
10	0.1126	0.3777	0.2005	0.1121	0.3997	0.1636
11	0.1141	0.3841	0.1992	0.1151	0.4315	0.1758
12	0.1117	0.3918	0.2138	0.1194	0.4256	0.1805

Table A.3: Comparing Time Embedding of Models in 2015

Lead Time	2 Hour Embedding			3 Hour Embedding		
	MVMH			MVMH		
	SCN	SCS	EDD	SCN	SCS	EDD
1	0.0821	0.3582	0.1076	0.0760	0.2427	0.1336
2	0.0789	0.3528	0.1192	0.0751	0.2166	0.1234
3	0.0869	0.3281	0.1142	0.0751	0.2212	0.1372
4	0.0850	0.3241	0.1233	0.0802	0.2086	0.1342
5	0.0863	0.3404	0.1236	0.0873	0.2126	0.1278
6	0.0886	0.3154	0.1345	0.0973	0.2156	0.1425
7	0.1058	0.3234	0.1474	0.0873	0.2273	0.1495
8	0.0934	0.3334	0.1578	0.0917	0.2506	0.1572
9	0.0948	0.3389	0.1674	0.1024	0.2648	0.1771
10	0.0983	0.3440	0.1847	0.1098	0.2753	0.1791
11	0.1084	0.3643	0.1927	0.1238	0.2861	0.1958
12	0.1078	0.3687	0.2026	0.1045	0.3075	0.2051

Table A.4: Comparing Time Embedding of Models of Models in 2017

Lead Time	2 Hour Embedding			3 Hour Embedding		
	MVMH			MVMH		
	SCN	SCS	EDD	SCN	SCS	EDD
1	0.1066	0.1932	0.1026	0.1550	0.4851	0.1190
2	0.1180	0.2018	0.1226	0.1487	0.4168	0.1057
3	0.1177	0.2245	0.1443	0.1402	0.4128	0.1286
4	0.1418	0.2017	0.1493	0.1525	0.3885	0.1193
5	0.1550	0.2116	0.1482	0.1764	0.3916	0.1400
6	0.1144	0.2422	0.1732	0.1601	0.4117	0.1408
7	0.1269	0.2511	0.1923	0.1690	0.4152	0.1703
8	0.1316	0.2513	0.1960	0.1796	0.3960	0.1646
9	0.1347	0.2774	0.2000	0.1500	0.4434	0.1641
10	0.1601	0.2995	0.2379	0.1668	0.4453	0.1841
11	0.1418	0.3163	0.2735	0.2070	0.4382	0.1907
12	0.1470	0.3075	0.2503	0.2038	0.4264	0.2186

Table A.5: Comparing Time Embedding of Models in 2018

Lead Time	2 Hour Embedding			3 Hour Embedding		
	MVMH			MVMH		
	SCN	SCS	EDD	SCN	SCS	EDD
1	0.0994	0.2391	0.0882	0.0742	0.1884	0.1183
2	0.0975	0.2176	0.0931	0.0933	0.1848	0.1066
3	0.0876	0.2154	0.0990	0.0781	0.1780	0.1179
4	0.0868	0.2332	0.1072	0.0831	0.2006	0.1115
5	0.0998	0.2418	0.1189	0.0894	0.1905	0.1100
6	0.1024	0.2327	0.1264	0.0888	0.2123	0.1184
7	0.0972	0.2450	0.1341	0.0943	0.2139	0.1190
8	0.1102	0.2276	0.1448	0.1069	0.2129	0.1329
9	0.1163	0.2495	0.1478	0.1075	0.2321	0.1426
10	0.1239	0.2510	0.1576	0.1103	0.2518	0.1491
11	0.1404	0.2613	0.1646	0.1215	0.2657	0.1741
12	0.1276	0.2765	0.1827	0.1317	0.2734	0.1639

Table A.6: Comparing Time Embedding of Models in 2019

Lead Time	2 Hour Embedding			3 Hour Embedding		
	MVMH			MVMH		
	SCN	SCS	EDD	SCN	SCS	EDD
1	0.0880	0.2505	0.0779	0.0922	0.2523	0.1028
2	0.0833	0.2245	0.0775	0.1119	0.2495	0.1432
3	0.0927	0.2188	0.0993	0.1238	0.2247	0.1106
4	0.0785	0.2224	0.1040	0.1202	0.2520	0.0962
5	0.0885	0.2124	0.1119	0.1131	0.2481	0.1554
6	0.0980	0.2260	0.1215	0.1000	0.2350	0.1330
7	0.1042	0.2333	0.1611	0.1006	0.2392	0.1629
8	0.1169	0.2404	0.1279	0.1264	0.2461	0.1654
9	0.1074	0.2498	0.1492	0.1347	0.2562	0.1345
10	0.1153	0.2786	0.1552	0.1116	0.2757	0.2320
11	0.1153	0.2726	0.1728	0.1535	0.2739	0.1851
12	0.1271	0.2892	0.1805	0.1327	0.3100	0.1766

Appendix B

Hyperparameter Configuration Space

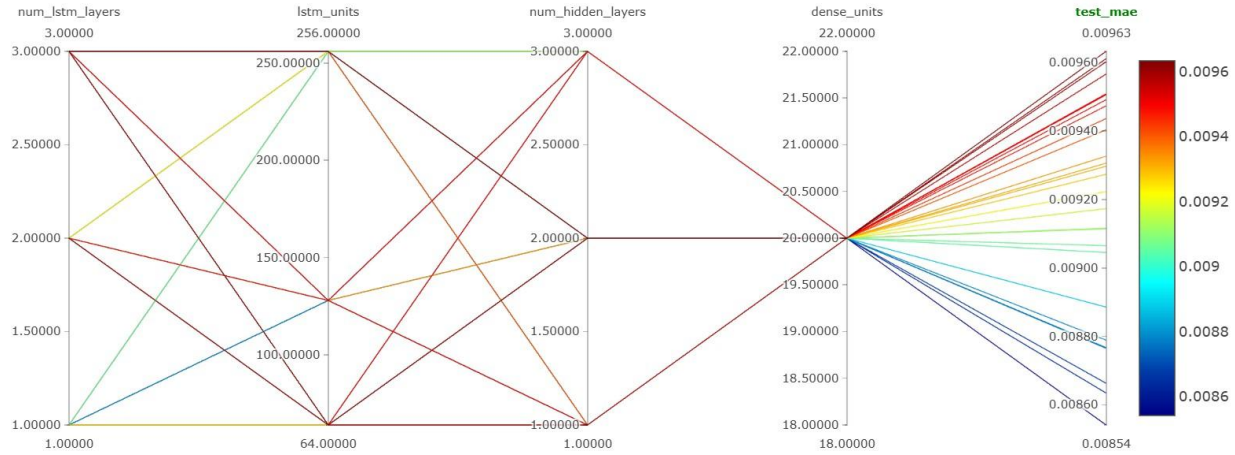


Figure B.1: Hyperparameter configuration space of UVMH at SCN on 2014 dataset with MLflow

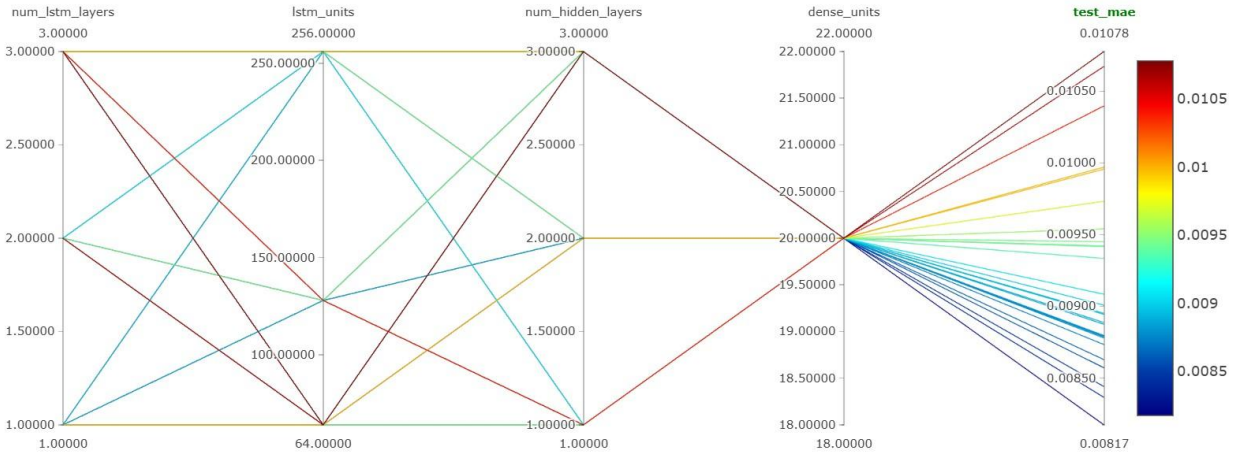


Figure B.2: Hyperparameter configuration space of UVMH at EDD on 2014 dataset with MLflow

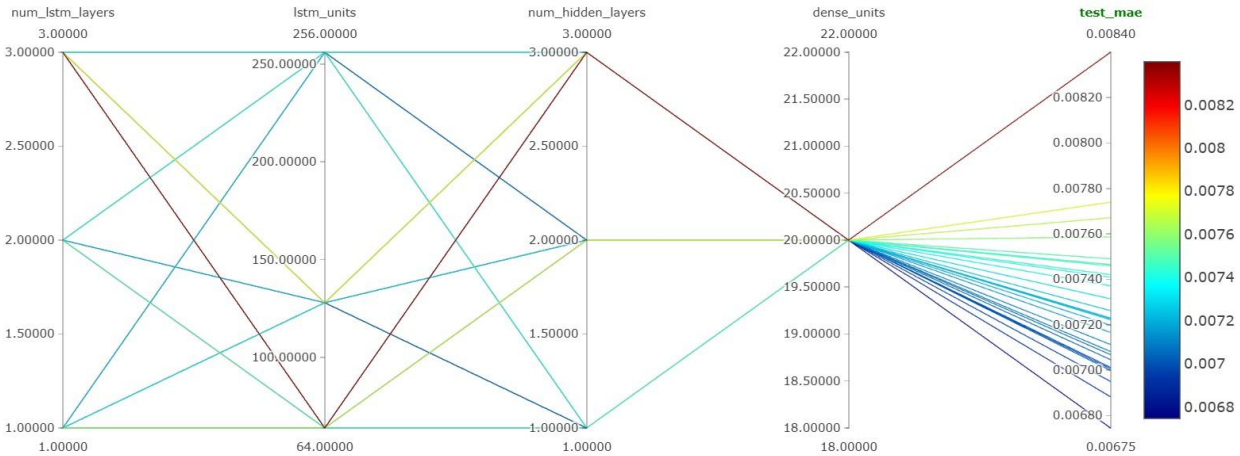


Figure B.3: Hyperparameter configuration space of UVMH at SCS on 2014 dataset with MLflow

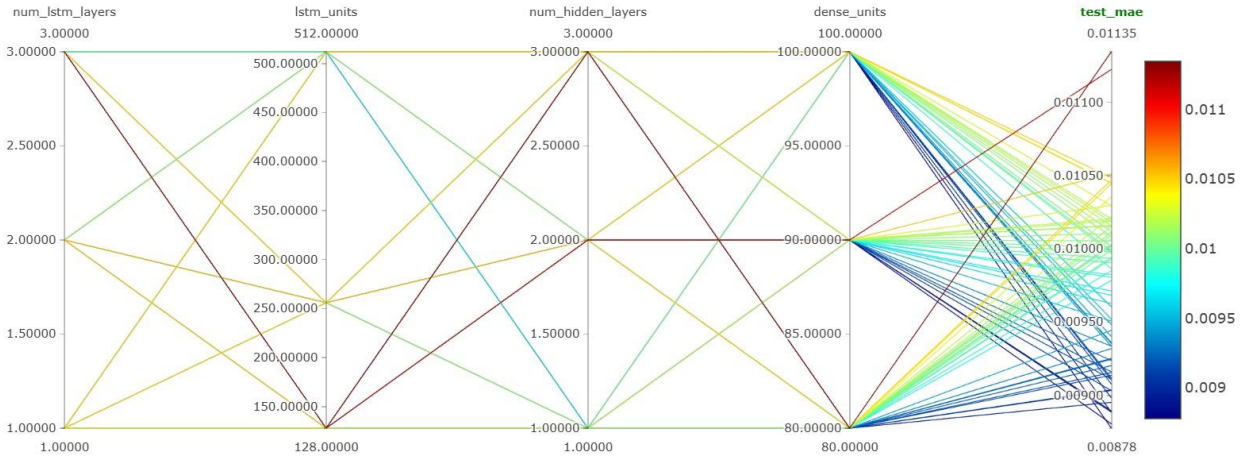


Figure B.4: Hyperparameter configuration space of MVMH on 2014 dataset with MLflow

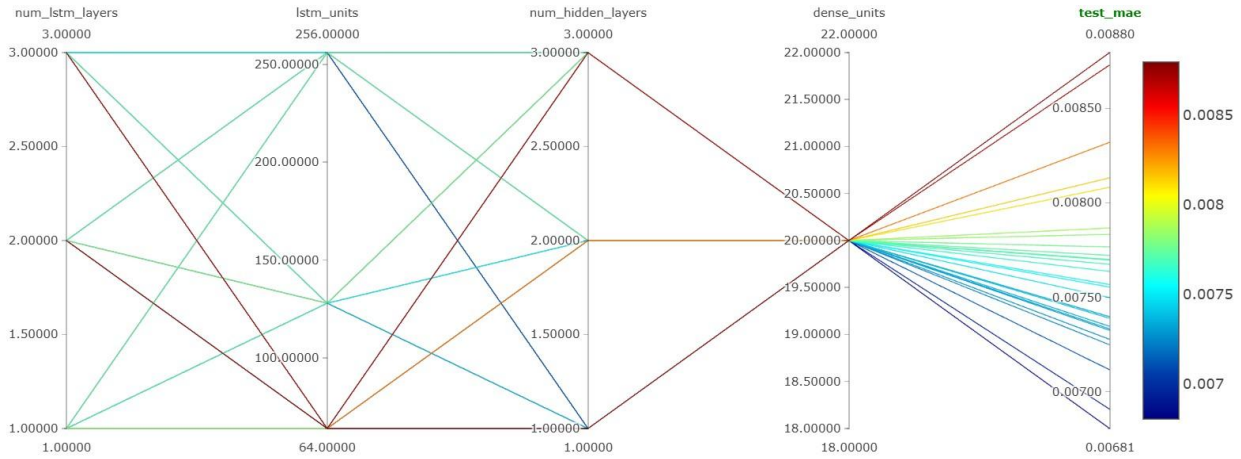


Figure B.5: Hyperparameter configuration space of UVMH at SCN on 2019 dataset with MLflow

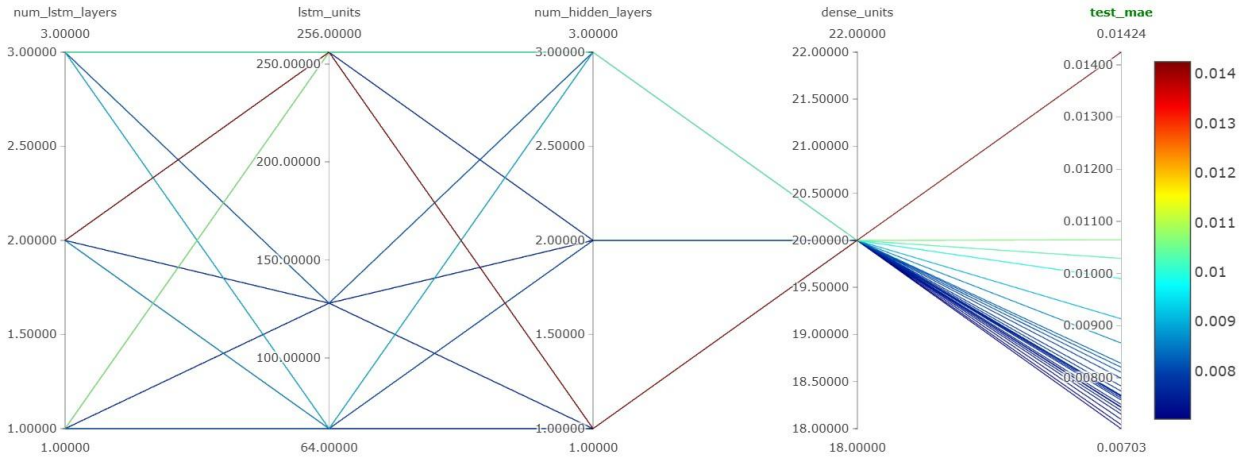


Figure B.6: Hyperparameter configuration space of UVMH at EDD on 2019 dataset with MLflow

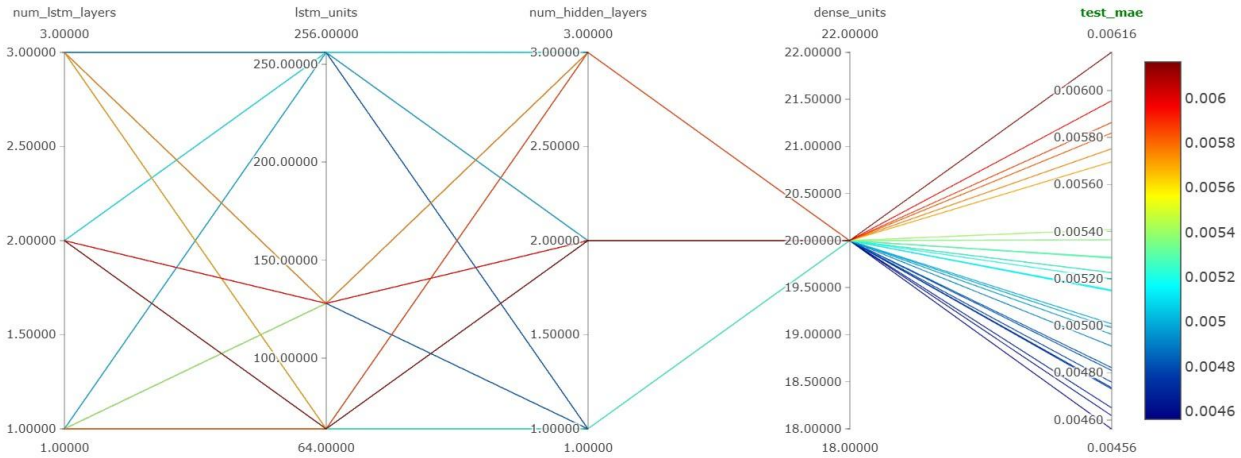


Figure B.7: Hyperparameter configuration space of UVMH at SCS on 2019 dataset with MLflow

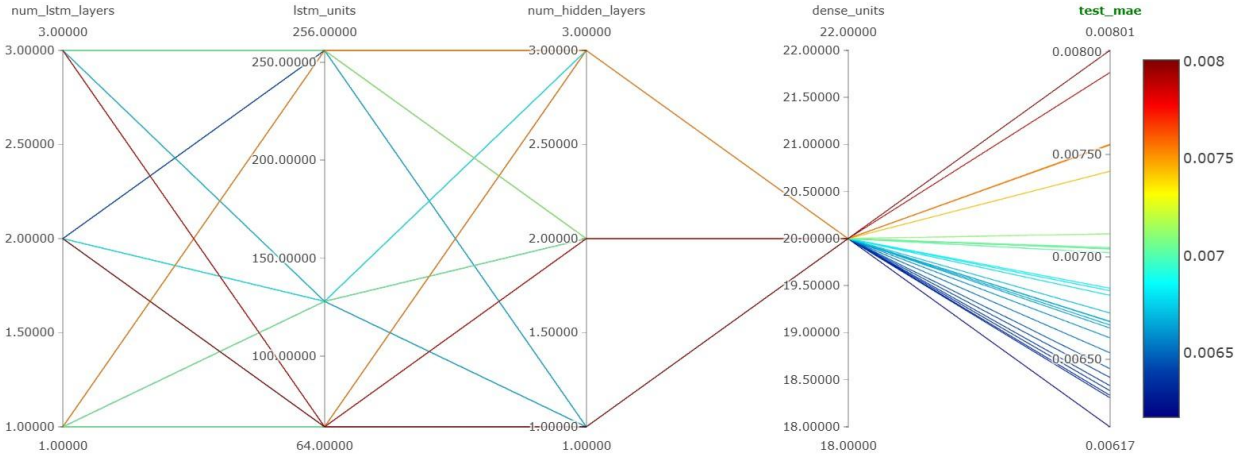


Figure B.8: Hyperparameter configuration space of UVMH at EB on 2019 dataset with MLflow

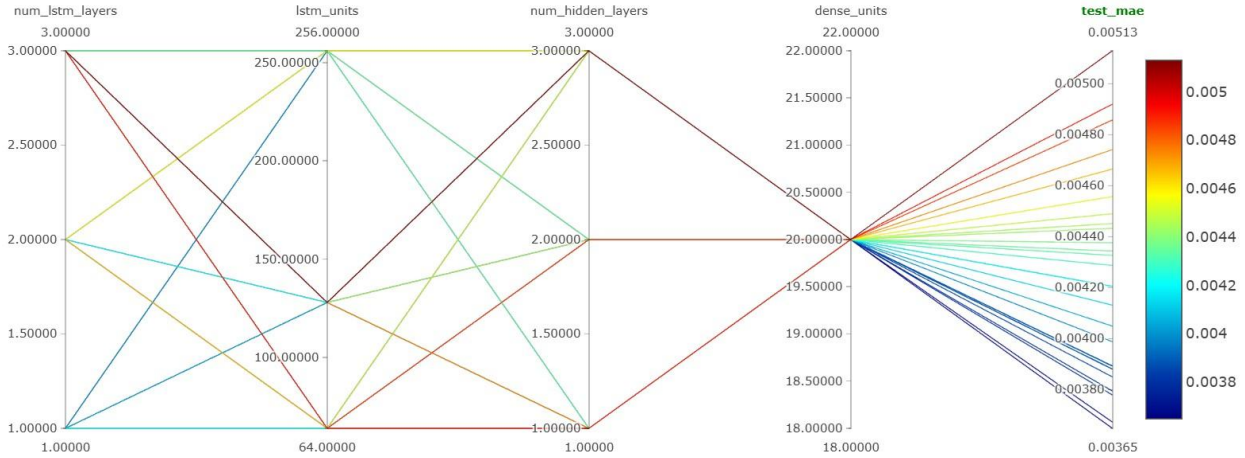


Figure B.9: Hyperparameter configuration space of UVMH at EH on 2019 dataset with MLflow

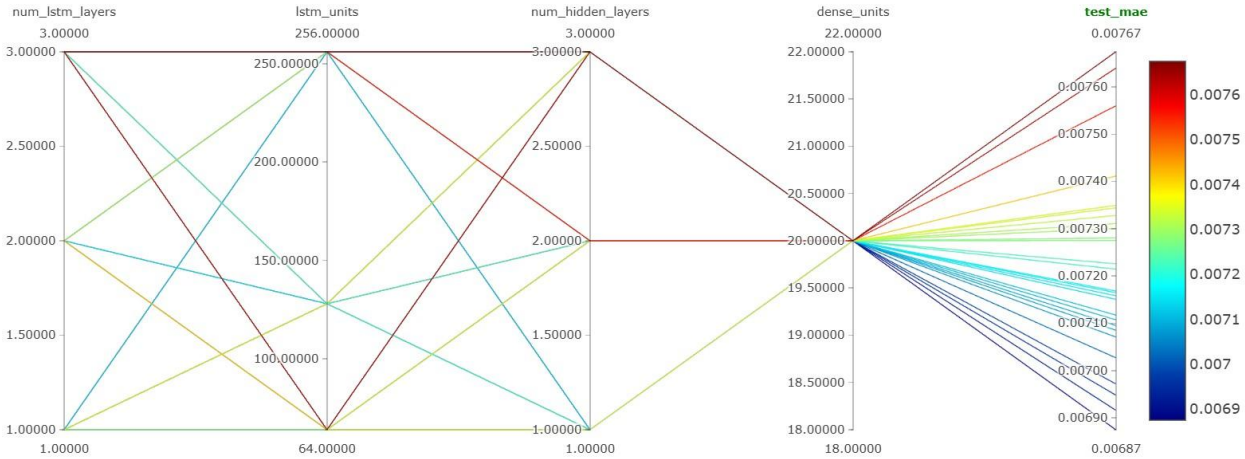


Figure B.10: Hyperparameter configuration space of UVMH at WSC on 2019 dataset with MLflow

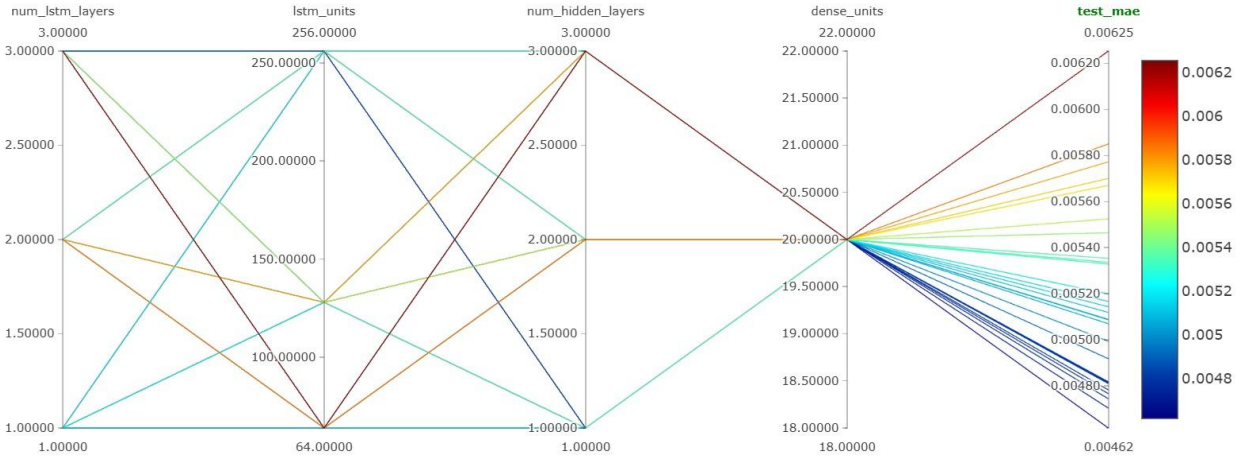


Figure B.11: Hyperparameter configuration space of UVMH at LED on 2019 dataset with MLflow

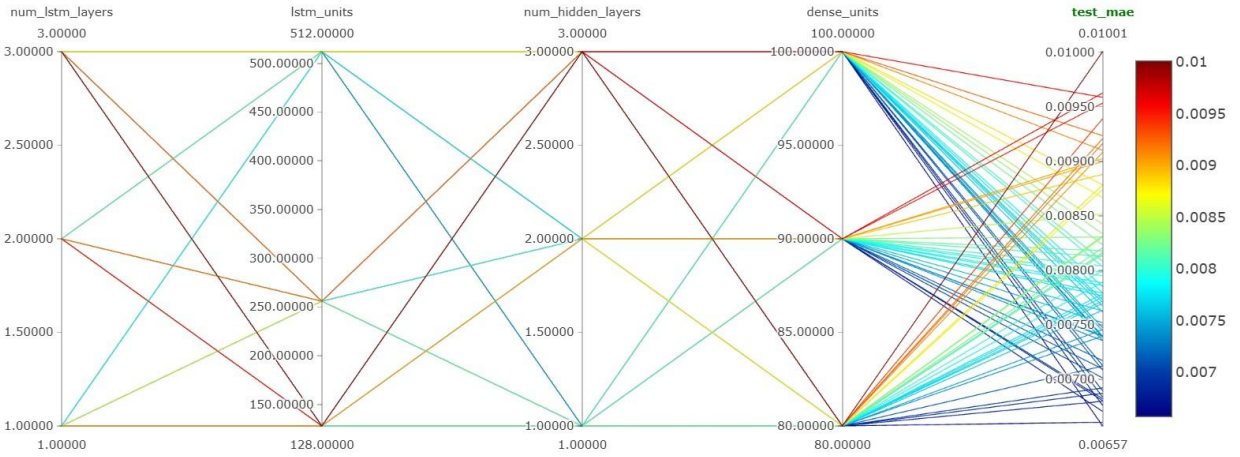


Figure B.12: Hyperparameter configuration space of MVMH on 2019 dataset with MLflow

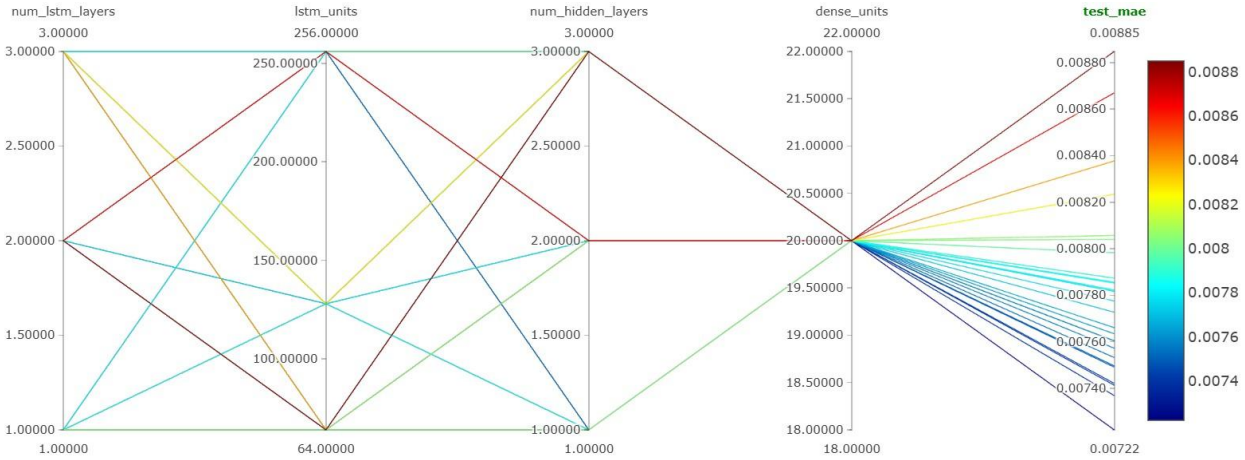


Figure B.13: Hyperparameter configuration space of UVMH at SCN on 2020 dataset with MLflow

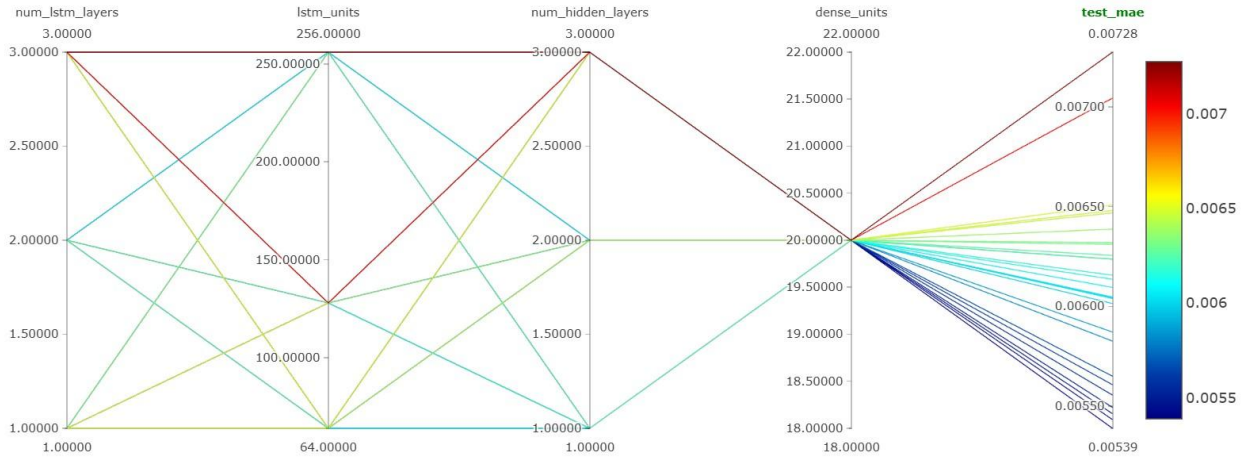


Figure B.14: Hyperparameter configuration space of UVMH at EDD on 2020 dataset with MLflow

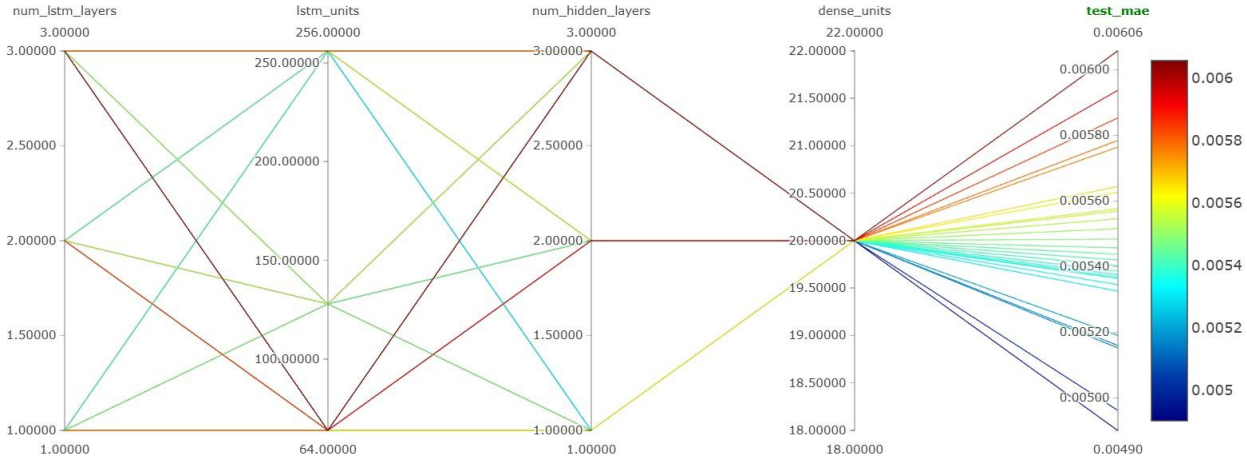


Figure B.15: Hyperparameter configuration space of UVMH at SCS on 2020 dataset with MLflow

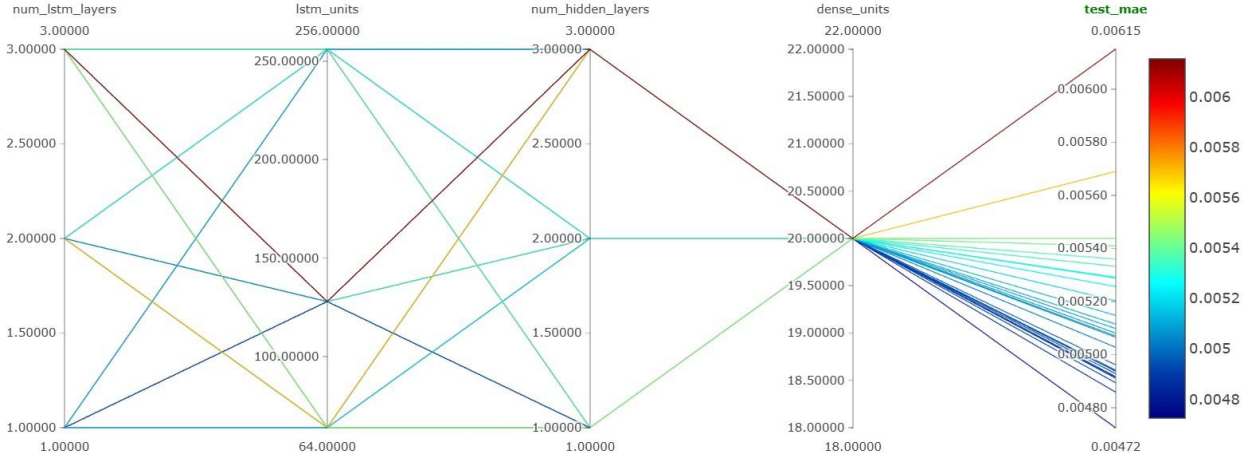


Figure B.16: Hyperparameter configuration space of UVMH at EB on 2020 dataset with MLflow

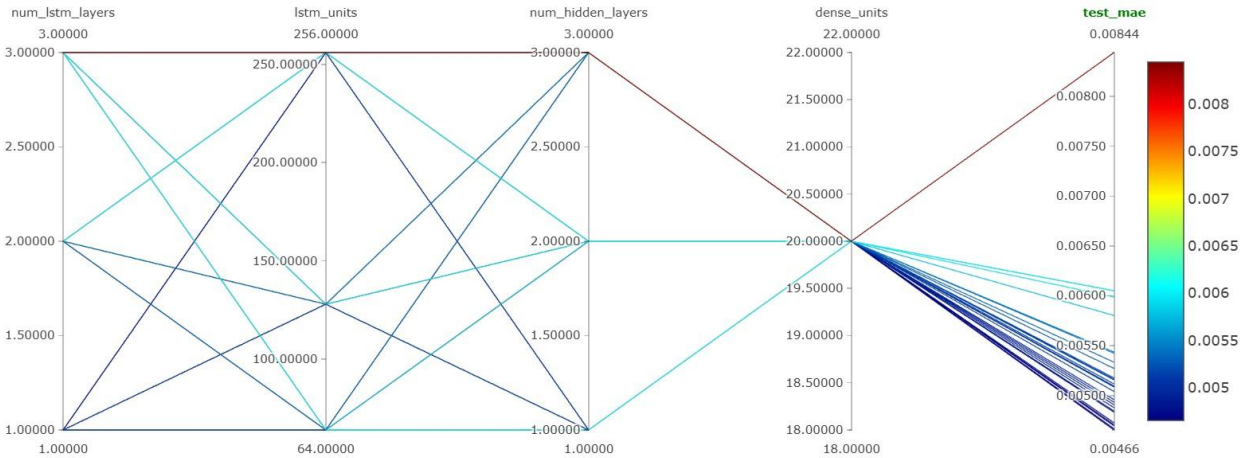


Figure B.17: Hyperparameter configuration space of UVMH at EH on 2020 dataset with MLflow

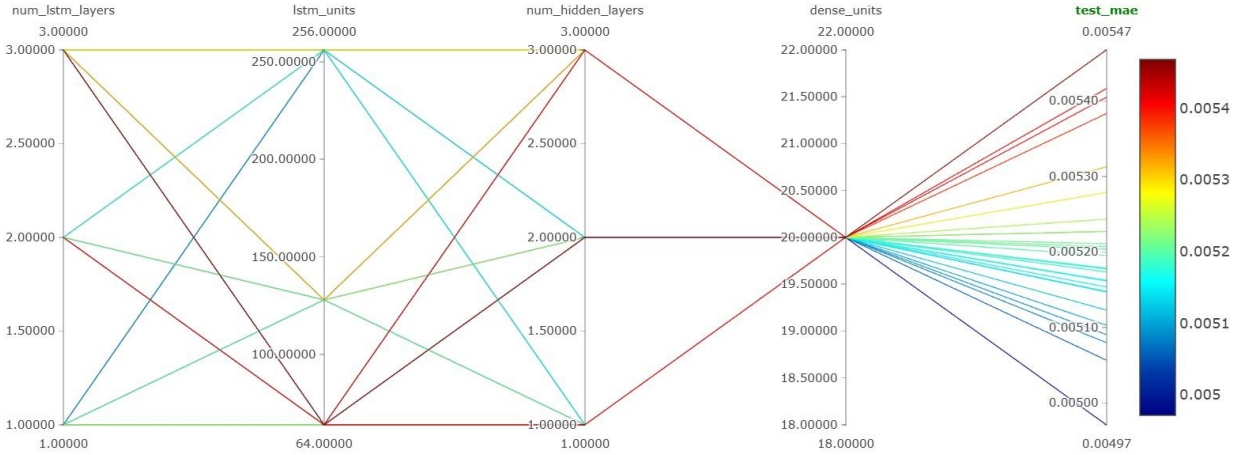


Figure B.18: Hyperparameter configuration space of UVMH at WSC on 2020 dataset with MLflow

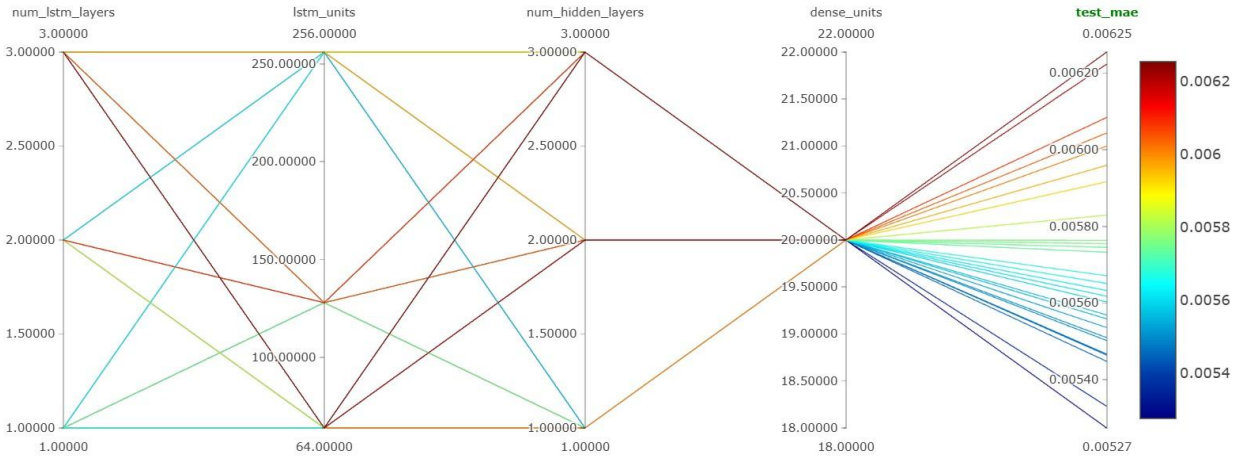


Figure B.19: Hyperparameter configuration space of UVMH at LED on 2020 dataset with MLflow

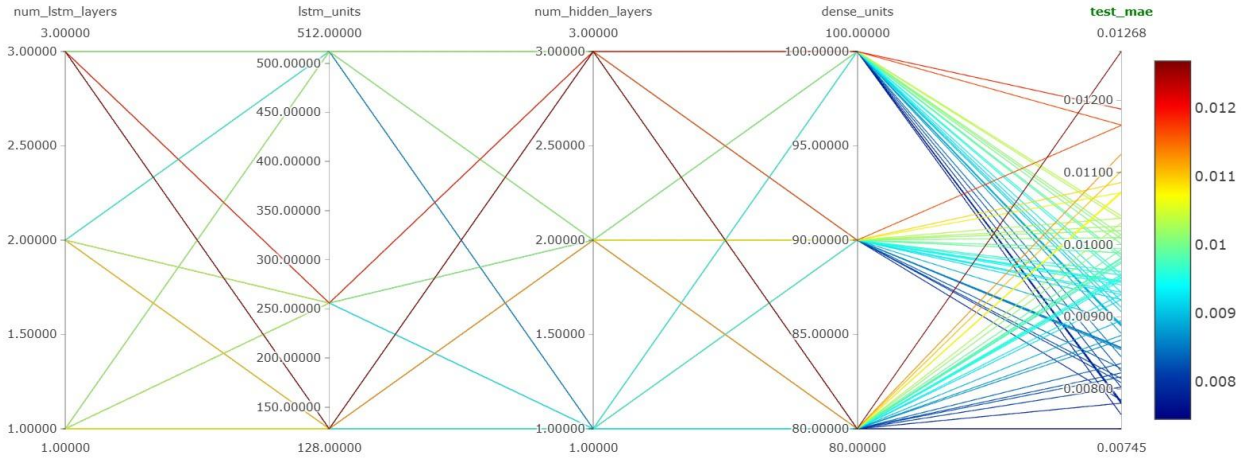


Figure B.20: Hyperparameter configuration space of MVMH on 2020 dataset with MLflow