Simulating Reservoir Operation using Deep Learning (DL) Models

Masal Vinod Kundlik

AGRICULTURAL AND FOOD ENGINEERING DEPARTMENT
INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR
KHARAGPUR – 721 302

Simulating Reservoir Operation using Deep Learning (DL) Models

Thesis submitted to
Indian Institute of Technology, Kharagpur
for the award of the degree

of

Master of Technology

in

Land and Water Resources Engineering

by

Masal Vinod Kundlik
(Roll No.: 20AG62R14)

Under the guidance of

Prof. Ashok Mishra

AGRICULTURAL AND FOOD ENGINEERING DEPARTMENT
INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR
KHARAGPUR – 721302
MAY 2022

Simulating Reservoir Operation using Deep Learning (DL) Models

Thesis submitted by

Masal Vinod Kundlik

(Roll No.: 20AG62R14)

Approved by

Prof. Ashok Mishra
(Supervisor)

Prof.Rintu Banerjee
(Head of the Department)

AGRICULTURAL AND FOOD ENGINEERING DEPARTMENT
INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR
KHARAGPUR – 721 302
MAY 2022

ACKNOWLEDGEMENTS

Every effort in this world comes to its fruitful culmination, not because of the sincere work of one but only due to the combined support and endeavor of many. First of all, thanks to the Almighty God and my parents for blessing me with endurance and strength.

I wish to place on record my profound sense of gratitude and indebtedness to my project supervisor Prof. Ashok Mishra, Agricultural and Food Engineering Department, Indian Institute of Technology Kharagpur for his unfailing guidance, prolific encouragement, constructive suggestions, and continuous involvement during each and every phase of this research work. His moral support and love inspired me to complete this research work in time.

I am very much thankful to Prof. Rintu Banerjee, Head of the Department, Agricultural and Food Engineering, IIT Kharagpur extending the necessary facilities during this work.

It gives me pleasure to express my gratitude to Prof. R. Singh, Prof. K. N. Tiwari, Prof. M. K. Jha, Prof. D. R. Mailapalli, Prof. A. Pathak, Prof. P. Ganguli, and other faculty members of the Agricultural and Food Engineering Department for their encouragement and kind cooperation.

It is my pleasure in expressing thanks to my seniors Mr. Uday Bhadoriya, Mr. Rajesh Ranjan, and Mr. Shashi Bhushan for their co-operation throughout the research tenure.

Everything I am today is because of my wonderful parents. No words of thanks to them will be enough for their infinite love and motivation.

I express my hearty thanks to my friends, batch mates, seniors, and juniors for their moral support and a great company.

Lastly, thanks to all the respected people around me who directly and indirectly helped me during my degree programme.

Date:

Place: IIT Kharagpur (Masal Vinod Kundlik)

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

.

LIST OF SYMBOLS AND ABBREVIATIONS

Abbreviation	**Description**
%	Percentage
ºC	Degree Celsius
RMSE	Root Mean Square Error
MSE	Mean Square Error
MAE	Mean Absolute Error
NSE	Nash Sutcliffe Model Efficiency
DVC	Damodar Valley Corporation
IMD	India Meteorological Department
ML	Machine Learning
DL	Deep Learning
RNN	Recurrent Neural Networks
CNN	Convolutional Neural Networks
LSTM	Long Short-Term Memory

Abstract

Dams are vital human-built infrastructures that play essential roles in flood control, hydroelectric power generation, water supply and navigation. Over the last decade, deep learning (DL) techniques have become popular in the field of streamflow forecasts, reservoir operation planning and scheduling approaches. In this study, Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models are developed to simulate daily reservoir operations of Maithon reservoir by using 36 years of historical reservoir operation records. This study aims to explore the influence of the hyperparameter settings on model performance and to explore the applicability of the deep learning models for reservoir operation simulation. Results show that the batch size, neurons and epoch have a great influence on the fitting effect and running time of models. The performance of the GRU model improves as the number of hidden layers increases, whereas the RNN and LSTM models perform better with two hidden layers. LSTM and GRU model simulate outflow based on large time step as compared to RNN model. The experiments and their evaluation sow that LSTM performs better among RNN and GRU networks with analysed coefficient of determination as 0.736 and 0.729 during training and testing period, respectively. Applicability of LSTM model, having the best evaluation statistics, could be further tested in other reservoirs. The findings of this study will help the water resource managers to apply DL models for reservoir operation planning.

Keywords: Deep learning, Recurrent Neural Network, Long Short-Term Memory model, Gated Recurrent Unit, hyperparameters tuning, reservoir operation

CHAPTER 1

INTRODUCTION

Dams/Reservoirs play an important role in water resources management system. Human beings, in half of the major global river systems, manage and utilize water resources through reservoirs for power generation, water supply, navigation, disaster prevention, flood control and drought relief (Dynesius and Nilsso, 1994; Shang et al., 2018). A competent reservoir operating plan is a key to avail the comprehensive benefits of the reservoir during the operation and management period. Any shortcoming in operation plan may lead to failure of dams and result to losses of life and property. Also, a reasonable and effective reservoir operating plan is essential for proper reservoir functioning and coordination among multiple contradicting goals such as flood control and water supply (Sasireka et al., 2018). Simulation and prediction of reservoir storage or release are required prior to the development of proper reservoir operation plans, thus, that becomes important to achieve all reservoir functions and to avoid danger to humans and river ecology (Loucks and Sigvaldason, 1981). Further, in recent years, many countries have adopted dynamic reservoir operations plans to mitigate the adverse effects of reservoirs construction and maintain the health of river ecosystems.

Reservoir operation depends on multiple factors with strong nonlinear interactions, which are influenced by natural conditions such as precipitation, runoff, land surface cover, and human needs such as irrigation demand, industrial production and domestic water consumption. Conceptual or physical-based models (such as HEC-ResSim, WEAP, RiverWare) have been developed and are widely used in reservoir operation simulation (Klipsch and Hurst, 2003; Yates et al., 2005). Physical-based models provide a more practical, physical and mathematical basis for the calculation of controlled releases or storage. However, the practical application scenarios of reservoir operation are extremely complex and involve multiple time scales and multiflow regimes, often accompanied by occasional emergencies (Zhang et al., 2019). Reservoir operations incorporate medium and long-term (seasonal and monthly scale) operation task of managing downstream water supply and optimization of economic benefit. Further, these also undertake short-term (daily and hourly scale) operation tasks of managing power grid load, water demand, navigation and stimulation of fish breeding, disaster prevention, emergency operations during floods, droughts etc. These various scheduling scenarios illustrate that the actual operation process of a reservoir is rapidly changing and often deviates from the operation plan. These deviations often make it difficult for operation rule based physical models to accurately simulate reservoir operation and predict the reservoir-controlled releases (Johnson et al., 1991; Oliveira and Loucks, 1997). In addition, when the physical model needs to be rebuilt with a new scheduling rule, the demand for the professional expertise of the reservoir operator is high, and the overall time of the modelling may not meet the requirements of emergency operation. Moreover, the above-mentioned complex hydrologic factors introduce uncertainty in the river-reservoir systems and increase the difficulty of using physical models (Zhang et al., 2018).

In recent years, with the development of artificial intelligence (AI) and big data mining technology, machine learning (ML) models have become popular and are widely used in various science applications. These models do not heavily rely on physical meaning, but are good at solving nonlinear simulation and prediction problems influenced by multiple complex factors. Some studies used AI or ML algorithms to simulate reservoir storage or release (Chang et al., 2005). Currently, among various ML algorithms available, artificial neural network (ANN), support vector machine and regression (SVM and SVR), and decision trees (DTs) are the most widely used models in reservoir operation (Chaves and Chang, 2008). In contrast to physical-based models, ML models have the ability to autonomously learn the various reservoir operation rules from a large amount of hydrological data and real-time reservoir operation data.

ML algorithms, however, still have some shortcomings, such as insufficient feature extraction capability and longer time consumption. Therefore, recently, a new branch of machine learning (ML), i.e., deep learning (DL) is used to overcome the conventional limitations. DL, derived from ANN, has unique capabilities to solve tasks that ML models could never be solved**.** DL algorithms are an abstract, high-level representation of attribute categories or characteristics through the combination of low-level features and can significantly improve recognition accuracy (Neelakantan and Pundarikanthan, 2000; Yang et al., 2016). Thus, DL can be utilized for the evaluation of the complex and nonlinear association between streamflows, basin properties and meteorological variables. There are various deep learning algorithms used for simulating reservoir operations such as Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU)**.**

Out of all models, RNN, LSTM and GRU model are widely used DL models, which are used for hydrological forecasting because of their ability to solve complex scheduling problems. RNNs have a loop like structure and can persist short-term past information. Due to RNNs have the limitations of the vanishing/exploding gradient problem, and thus, can work with short-term temporal dependencies only. LSTMs were introduced to overcames these limitations with the help of memory cells in their structure. LSTM can, by default, retain the information for long period of time for processing, predicting and classifying data. Much later, a decade and half after LSTM, GRU was introduced (Cho et al., 2014). They are similar to LSTM networks but with a simpler architecture, suitable to work on long-term dependencies and sequential data (Zaytar and Amrani, 2016, Zhang et al., 2018).

There is lack of comprehensive studies related to the simulation of reservoir operation using deep learning models. Therefore, this study attempts to explore the application and performance of multiple deep learning models to simulate the operation of a multipurpose reservoir in India. Thus, keeping the above-mentioned conditions in view, the following research objectives are formulated:

Objectives:

1. To perform hyperparameters tuning of the RNN, LSTM and GRU models for simulating the operation of Maithon reservoir.
2. To train and test the hyperparameters tuned models (RNN, LSTM and GRU) for Maithon reservoir operation.
3. To compare the performance of tested RNN, LSTM and GRU models to simulate the Maithon reservoir operation.

CHAPTER 2
REVIEW OF LITERATURE

This chapter presents a brief review of the various studies regarding the application of various machine learning models in water resources systems.

Kumar et al. (2004) utilized a feedforward network and recurrent neural network (RNN) to estimate the monthly flow of the Hemavathi River in India. The feed forward network was trained using the conventional back propagation algorithm with many improvements and the recurrent neural network was trained using the method of ordered partial derivatives. The selection of architecture and training procedure for both the networks were presented. The selected ANN models were used to train and forecast the monthly flows of a river in India, with a catchment area of 5189 km^2 up to the gauging site. The trained networks were used for both single step ahead and multiple steps ahead forecasting. A comparative study of both networks indicated that the recurrent neural networks perform better than the feed forward networks.

Pan et al. (2007) used a deterministic linearized RNN (DLRNN) that deals with the system’s nonlinearity by recalibration at each time interval, and relates the weights of DLRNN to unit hydrographs in order to describe the transition of the rainfall-runoff processes. Case studies of 38 events, from 1966 to 1997, were implemented in the Wu-Tu watershed of Taiwan, where the runoff path-lines are short and steep. A comparison between the DLRNN and a feed-forward neural network demonstrated the advantage of DLRNN as a dynamic system model. DLRNN showed superiority in the performance of rainfall–runoff simulations and the ability to recognize transitions in hydrological processes.

Firat (2008) used rainfall data and seven lags of flow to predict daily streamflow in the Seyhan Stream and the Cine Stream in Turkey. For this purpose, ANN, adaptive neuro-fuzzy inference system (ANFIS), generalized regression neural networks (GRNN), feedforward neural networks, and auto-regressive methods were developed. For the Seyhan River, the forecasting models were established using combination of antecedent daily river flow records. On the other hand, for the Cine River, daily river flow and rainfall records were used in input layer. For both stations, the data sets were divided into three subsets training, testing and verification data set. The river flow forecasting models having various input structures were trained and tested to investigate the applicability of ANFIS, ANN and AR methods. The results demonstrated that ANFIS model is superior to the GRNN and FFNN forecasting models. ANFIS can be successfully applied and provide high accuracy and reliability for daily river flow forecasting.

Hassaballah et al. (2011) developed a methodology based on coupled simulation-optimization approach for determining filling rules for the proposed Mandaya Reservoir in Ethiopia. The combined system of reservoirs is set in MIKE BASIN

Simulation model, which is then used for simulation of a limited set of feasible filling rules of the Mandaya reservoir according to the current storage level, the inflow, and the time of the year. The simulation model is coupled with multi-objective optimization Non-dominated Sorting Genetic Algorithm (NSGA-II), which is adopted for determining optimial filling rules of the Mandaya Reservoir. Results demonstrate that optimal release- (and correspondingly filling-) rules for Mandaya Reservoir which maximize the hydropower generation in both Mandaya and Roseires reservoirs.

Sattari et al. (2012) used time-lagged RNNs and backpropagation neural network methods to estimate daily flow into the Elevian Dam reservoir in Iran by using the daily inflow (2004–2007). To compare the performance of TLRN, a back propagation neural network was used. The TLRN model with gamma memory structure, eight input layer nodes, two hidden layer and one output layer (8-2-1) was found performing best out of three different models used in forecasting daily inflow. Comparison of results with back propagation neural network suggested that neither TLRN nor back propagation approaches were good in forecasting high inflow. However, both approaches performed well when used to forecast low inflow values. Statistical test suggested that both TLRN and back propagation neural network models were able to reproduce similar basic statistics as that of the actual data.

Sattari et al. (2013) used support vector machines (SVM) and an M5 tree model to estimate daily flow in the Sohu river in Turkey. Results of the M5 model tree was compared with support vector machines. Both modelling approaches were used to forecast up to 7-day ahead streamflow. A comparison of correlation coefficient and root mean square value indicated that M5 model tree approach works equally well to the SVM for same day discharge prediction. The M5 model tree also worked well up to 7-day ahead discharge forecasting in comparison of SVM with the data set. An advantage of using M5 model tree approach was the availability of simple linear models to predict the discharge as well use of less computational time.

Chang et al. (2014) simulated real-time forecast of urban flood in Taipei City of Taiwan using back propagation (BP) ANN and dynamic ANNs (Elman NN; nonlinear autoregressive network with an exogenous inputs-NARX network). Results showed that the GT can efficiently identify the effective rainfall stations as important inputs to the three ANNs. Recurrent connections from the output layer (NARX network) imposed more effects on the output than those of the hidden layer (Elman NN). NARX network performed the best in real-time forecasting. The NARX network produces coefficients of efficiency within 0.9–0.7 (scenario I) and 0.7–0.5 (scenario II) in the testing stages for 10–60-min-ahead forecasts accordingly. This study suggested that the proposed NARX models can be valuable and beneficial to the government authority for urban flood control.

Fang et al. (2014) proposed a new storage allocation rule based on target storage curves. Joint operating rules were also proposed to solve the operation problems of a multi-reservoir system with joint demands and water transfer-supply projects. The joint operating rules included a water diversion rule to determine the amount of diverted water in a period, a hedging rule based on an aggregated reservoir to determine the total release from the system, and a storage allocation rule to specify the release from each reservoir. The multi-reservoir water supply system located in Liaoning Province, China, including a water transfer-supply project, was employed as a case study to verify the effectiveness of the proposed joint operating rules and target storage curves. Results indicated that the proposed operating rules were suitable for the complex system. The storage allocation rule based on target storage curves showed an improved performance with regard to system storage distribution.

Teegavarapu and Simonovic (2014) developed a simulation model to understand the dynamic behavior of a hydraulically coupled multiple reservoir system when operated in real-time. An object-oriented simulation environment conceived on the principles of system dynamics (SD) was used for the development of the model. The modeling process involved developing causal loop, stock and flow diagrams, and carrying out simulations using difference equations to integrate stocks. A simulation model based on a real-life hydropower reservoir system in the Province of Manitoba, Canada was developed. The behavior of the reservoir system for extreme events was observed and conclusions relevant to real-time operation of the systems were drawn. The simulation models developed in this study provide global description of a hydraulically coupled reservoir system with ability to easily model dynamic processes to obtain operational release decisions that can be adopted in real-time.

Esmaeilzadeh et al. (2017) estimated daily flow to the Sattarkhan Dam in Iran by using data-mining methods, including ANNs, the M5 tree method, support vector regression, and hybrid Wavelet-ANN methods. Twelve scenarios consisting of input variables with different combinations of flow and climatological data by different lag times were created to study the role played by input variables in the accuracy of the model. Results of the various scenarios showed that when the inputs were consisted of the amount of temperature, precipitation and previous discharges, the next-day discharge could be estimated with higher accuracy by the WANN model. Conclusively, the obtained results revealed that the wavelet transformation created significant effect for increasing the prediction accuracies of the ANN model.

Chiang et al. (2018) integrated ensemble techniques into artificial neural networks to reduce model uncertainty in hourly streamflow predictions in Longquan Creek and Jinhua River watersheds in China. The results showed that the ensemble neural networks greatly improved the accuracy of streamflow predictions as compared to a single neural network, and the improvement made by the ensemble neural network is about 19–37% and 20–30% in Longquan Creek and Jinhua River watersheds, respectively, for 1–3 h ahead streamflow prediction. Moreover, the results obtained from different ensemble strategies were quite consistent in both watersheds, indicated that the ensemble strategies are insensitive to hydrological and physiographical factors. Finally, the output intervals of ensemble streamflow prediction also reflected the possible peak flow.

Sasireka et al. (2018) formulated a new operating rule for multipurpose reservoir using hedging rules and the developed model was applied to a case study of Bargi reservoir in the Narmada basin in India. In order to increase the reliability of water supply for municipal, irrigation and average annual power production, the new operating rule was developed using Standard Operation Policy (SOP) and hedging rule according to the priority of release for different purposes. The hedging rule based simulation model satisfied 97.5% of municipal water supply which was more than 8.25% of the present operational policy. The spill of the reservoir was decreased by 57 % compared to present policy.

Zhang et al. (2018) used backpropagation (BP) neural network, SVR and LSTM model to operate the Gezhouba Dam reservoir in China in hourly, daily, and monthly time scales. The results showed that in BP neural network and LSTM model, the effects of the number of maximum iterations on model performance should be prioritized, for the SVR model, the simulation performance is directly related to the selection of the kernel function, and sigmoid and RBF kernel functions should be prioritized. The BP neural network and SVR were suitable for the model to learn the operation rules of a reservoir from a small amount of data and the LSTM model was able to effectively reduce the time consumption and memory storage required by other AI models. Further, it showed good capability in simulating low-flow conditions and the outflow curve for the peak operation period.

Zhang et al. (2019) used three deep learning algorithms i.e., RNN, LSTM, and gated recurrent unit (GRU) to predict outflows in the Xiluodu reservoir. The number of iterations and hidden nodes mainly influenced the model precision, and the former had more effect than the latter. Batch size mainly affected the calculated speed. All three models predicted the reservoir outflow accurately and efficiently. The operating decision generated by three models could implement the flood control and power generation goal of the reservoir and meet the operating regulation and under different hydrological periods, the influence factors of reservoir operation and their importance are different.

CHAPTER 3
MATERIALS AND METHODS

This section presents the details of the study area, data acquired and the methodology followed to achieve the objectives of the study.

3.1. Location of the Study Area

Maithon dam is located at Latitude 23°51′1″N and Longitude 86°46′40″E in Maithon city in Jharkhand, India. It is situated on the Barakar River, approximately 12.9 km above its junction with the Damodar river. Maithon dam, with length of 4,789 m (15,712 ft) long and height of 50 m (165 ft), was mainly designed for flood control. However, it also serve other purposes such as water supply for industrial and domestic purposes, navigation and hydropower generation. It is the first underground power generating station in Southeast Asia having electric power generation capacity of around 60,000 KW.

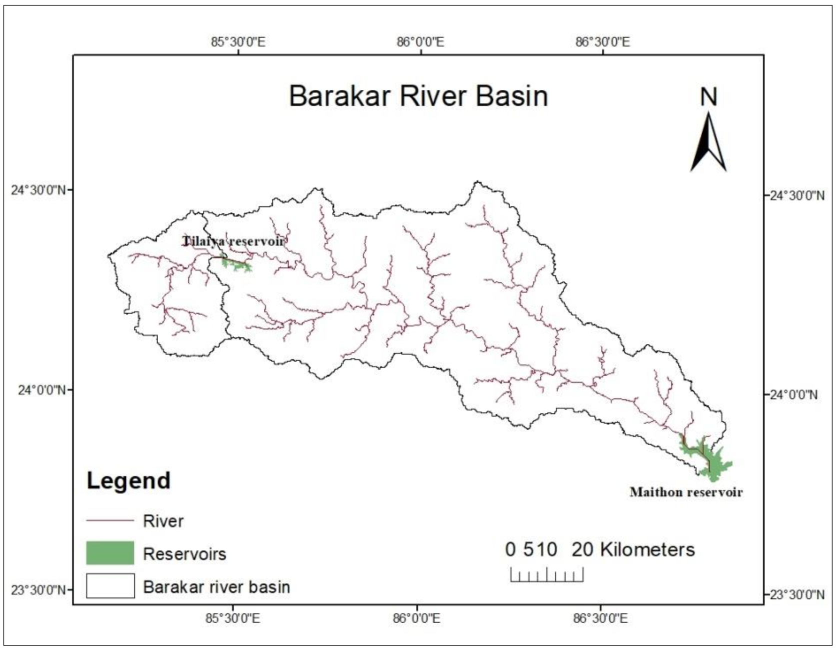

Fig. 3.1 Index map of Barakar river basin showing the location of Maithon Reservoir

3.2 RNN

Recurrent Neural Networks (RNN) are the type of neural networks with loops, which allow them to persist information from the past in the network model. Fig. 3.2 shows the representation of basic loop structure in RNN. Here, the center square represents a neural network, which takes input x_t at the current time slice (t) and gives the value h_t as an output. The loop shown in the structure enables it to use information from past time slices (t-1) to produce output for the current time slice (t). Thus, the decision made at previous time slice (t-1) affects the decision to be made at the current time slice (t). Therefore, the response of the network to the new data depends on the current input as well as the output from the recent past data. The RNN output calculation is based on iteratively calculating the output of the following two equations.

$$h_t = H(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{3.1}$$

$$y_t = W_{hy}h_t + b_y \tag{3.2}$$

Where, x_t is the input sequence at the current time slice (t), y_t is the output sequence at time slice (t), and h represents the hidden vector sequence from time slice 1 to T. W and b represents weight matrices and biases, respectively. Lastly, H is an activation function used for the hidden layer.

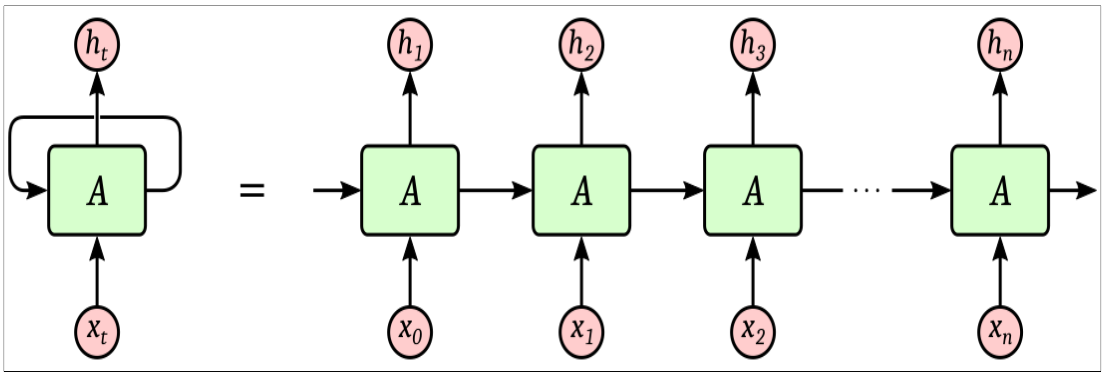

Fig. 3.2 Basic Recurrent Neural Network (RNN) Structure

3.3 LSTM

LSTM is a complex recurrent model to address the deficiencies of RNNs. RNN can effectively deal with nonlinear time series, however, there are still some shortcomings such as gradient disappearance and gradient explosion. Thus, an RNN cannot capture the temporal features especially the long-term dependencies. To solve these problems, the LSTM model was invented. The difference between the LSTM model and RNN model is that the RNN block in the hidden layer is replaced by the LSTM block as shown in Fig. 3.2. RNN enables modeling a long-term memory capability. The LSTM block consists of an input gate, a memory cell, a forget gate, and an output gate as shown in Fig. 3.3.

The corresponding forward propagation steps and equations are presented below (Apaydin et al., 2020). The first step in LSTM is to decide what data should be removed from the cell state. This decision is made by a sigmoid layer called the forget gate. The forget gate will generate an $\boldsymbol{f_t}$ value between 0 to 1, based on the previous moment output $\boldsymbol{h_{t-1}}$ and current input $\boldsymbol{x_t}$.

$$f_t = \sigma(w_{xf}x_t + w_{hf}h_{t-1} + b_f) \tag{3.3}$$

The second step is to decide what new data will be saved in the cell state. This has two parts: first, the input gate determines which values will be updated. Next, the memory cell creates a vector of new candidate values C_t which can be added to the state. Later, the values produced by these two parts will be combined.

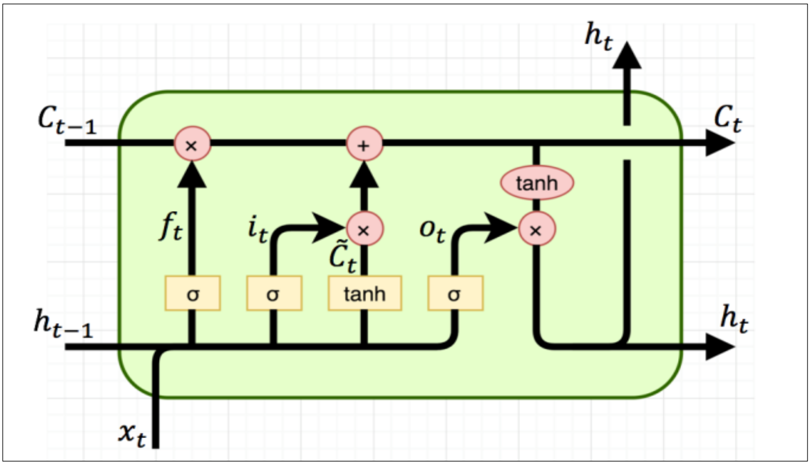

Fig. 3.3 Long-Short Term Memory (LSTM)

$$i_t = \sigma(w_{xi}x_t + w_{hf}h_{t-1} + b_i) \tag{3.4}$$

$$C_t = f_tC_{t-1} + i_t \tanh(w_{xc}x_t + w_{hc}h_{t-1} + b_c) \tag{3.5}$$

The final step is to determine the output of the model. First, an initial output is obtained through the sigmoid layer (output gate), and then the C_t value is resized to -1 to 1 through tanh activation function and then, multiplied by the output of the sigmoid gate to only output the target parts.

$$o_t = \sigma(w_{xo}x_t + w_{ho}h_{t-1} + b_o) \tag{3.6}$$

$$h_t = o_t \tanh(C_t) \tag{3.7}$$

Where, f, i, C, o and h are the forget gate, input gate, cell, output gate, and hidden output, respectively, and σ and tanh represent the gate activation function (the logistic sigmoid) and the hyperbolic tangent activation function, respectively, W represents the corresponding weight coefficient matrix.

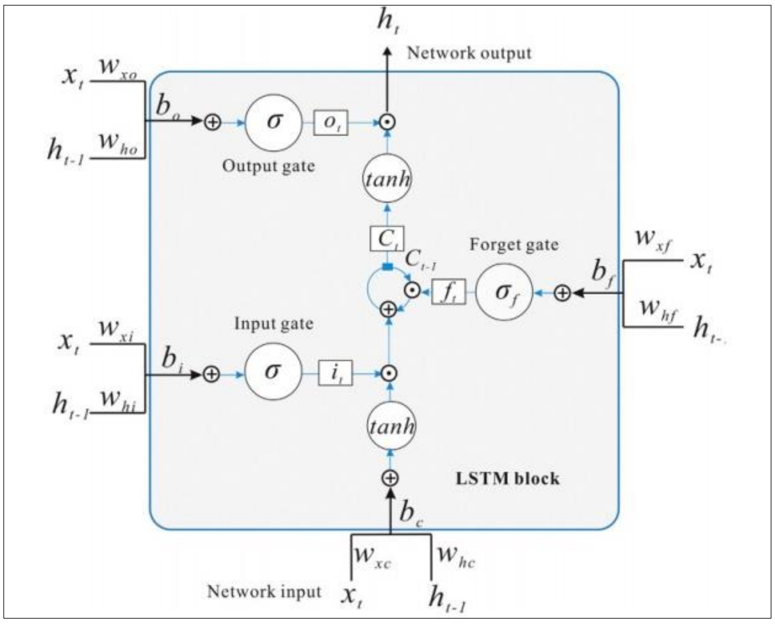

Fig. 3.4 LSTM block structure. (Zhang *et.al., 2018*)

3.4 GRU

GRU model is another type of modified RNNs with memory cells. They are similar to LSTM but with simpler cell architecture. GRU also has gating mechanism to control the flow of information through cell state but has fewer parameters and does not contain an output gate.

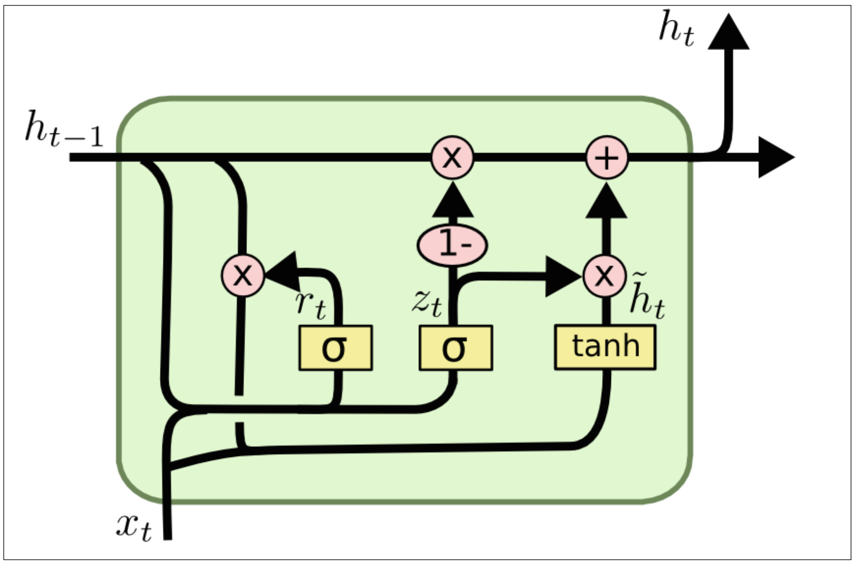

Fig. 3.5 Gate Recurrent Unit (GRU)

Fig. 3.5 shows a particular single cell structure of GRU. It consists of two gates: ‘r’ and ‘z’ are reset gate and update gate, respectively. The reset gate regulates the flow of new input to the previous memory, and the update gate determines how much of the previous memory to keep. In comparison to LSTM, in GRU, the n of the input and forget gate and the previous hidden state is connected to the reset gate directly. Another difference is in the exposure of memory content. As GRUs do not have an output gate, it exposes all of its memory content, whereas in LSTM the memory content to be used or seen by other units/cells in the network is managed by the output gate. The following equations are used in the GRU output calculations:

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \tag{3.8}$$

$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z) \tag{3.9}$$

$$\bar{h}_t = \tanh(W_{xh}x_t + W_{hh}(r_t \times h_{t-1}) + b_h) \tag{3.10}$$

$$h_t = (z_t \times h_{t-1}) + (1 - z_t) \times \bar{h}_t \tag{3.11}$$

In Equations (3.8)-(3.11), x_t , h_t , r_t , z_t represent the input vector, output vector, reset gate and an update gate, respectively. All W variables denote the weight matrices, and b are biases. Activation functions used are the same as LSTM, sigmoid function and hyperbolic tangent function.

3.5 Data used

In this study, daily reservoir operating data and meteorological data of Maithon reservoir from 1980 to 2016 are collected from Damodar Valley Corporation (DVC) and India Meteorological Department (IMD), respectively.

Table 3.1 Details of the data used

	Data	**Sources**	**Resolution**
Reservoir operating data	Reservoir inflow	Damodar Valley Corporation (DVC)	-
	Reservoir outflow		-
	Water level		-
	Irrigation demand		
	Municipal and Industrial (M&I) demand		-
Meteorological data	Rainfall	India Meteorologica l Department (IMD)	$0.25^o \times 0.25^o$
	Max. Air temperature		$1^o \times 1^o$
	Min. Air temperature		$1^o \times 1^o$

3.6 Model Building

Fig. 3.6 shows the overall methodology of the analysis carried out in the study. The process starts with data collection which includes inflow and outflow of the reservoir, water level upstream and downstream of the dam and meteorological data such as rainfall, maximum air temperature, minimum air temperature. Pre-processing of data is performed, thereafter based on the commonly accepted "80/20" split rule, the daily scale data from 1 January 1980 to 2 April 2010 is used for training models and the remainder is used for testing.

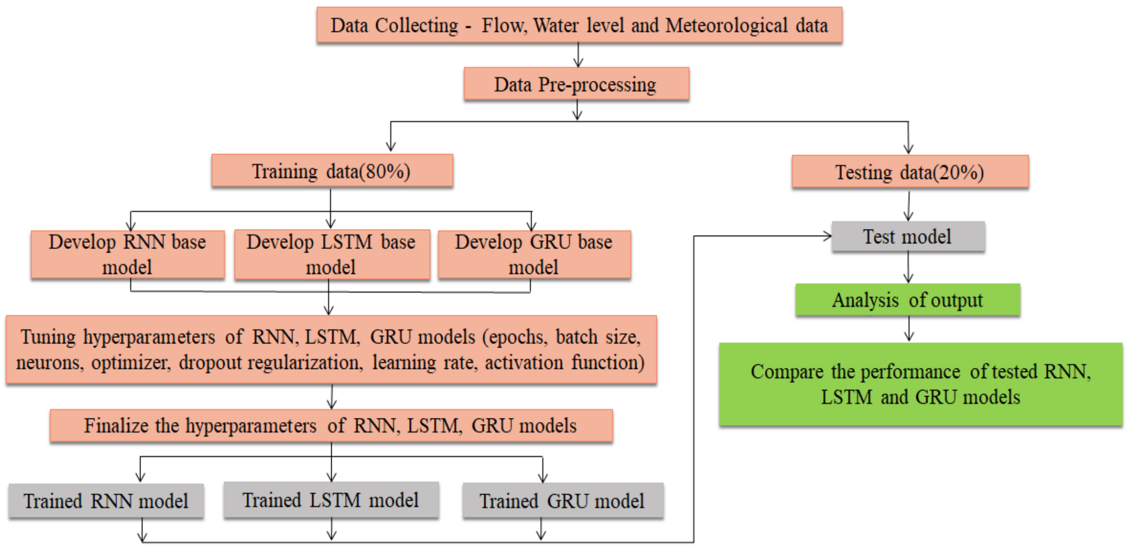
Data Collecting - Flow, Water level and Meteorological data
Data Pre-processing
Training data(80%)
Testing data(20%)
Develop RNN base model
Develop LSTM base model
Develop GRU base model
Test model
Analysis of output
Compare the performance of tested RNN, LSTM and GRU models
Tuning hyperparameters of RNN, LSTM, GRU models (epochs, batch size, neurons, optimizer, dropout regularization, learning rate, activation function)
Finalize the hyperparameters of RNN, LSTM, GRU models
Trained RNN model
Trained LSTM model
Trained GRU model

Fig 3.6 Work flow process of the methodology

The reservoir operating data and meteorological data are used as the model inputs (decision variables) and output (target variable). Specifically, the model inputs are current inflow, previous inflow (1-moment lag), previous outflow (1-moment lag), previous water level of reservoir, current and previous irrigation and M&I demand of the downstream region, the current and previous metrological data such as rainfall, maximum air temperature, minimum air temperature, ET and the current time. Here, previous data variable refers to as the 1-moment lag data series. Therefore, seventeen input variables are included in the model. The model output is the current outflow of the reservoir. To match the consistency of the model, all source data were normalized in the range 0.0–1.0 by using MinMaxScaler and after the model simulation, output is transformed back to original values.

3.7 Hyperparameter tuning

Deep learning algorithms require choosing a set of optimal parameters or hyperparameters, and tuning process is also known as hyperparameter tuning or hyperparameter optimization. Hyperparameters are ML algorithm's parameters, which control the learning process. Deep learning algorithms have different types of hyperapremeters such as epochs, batch size, neurons, optimizer, dropout regularization and weight constraint, learning rate, activation functions. It is necessary to tune hyperparameters so that the problem can be solved optimally. There are several approaches for hyperparameter tuning. In this study, we have selected the grid search method, which consists of exhaustive searching through a subset of hyperparameter space of the algorithm, followed by a performance metric. In the grid searching process, data is scanned to configure optimal parameters of a specific model. Parameters are specific to the type of model considered. Grid searching is not confined to one type of model, but can be applied throughout deep learning, such that the best parameters can be identified for the model. The process of grid searching builds a model on each parameter combination possible, leading to multiple iterations. These model combinations for every parameter are stored, and thus, grid searching is computationally expensive.

Table 3.2 Hyperparameters of deep learning models considered in the study.

Hyperparameters	**Values**
Batch Size	32, 64, 128, 256, 512
Epochs	10, 50, 100, 150, 200
Optimizer	SGD, Nadam, Adamax, RMSprop, Adagrad, Adadelta, Adam
Learning Rate	0.0001, 0.001, 0.01, 0.1
Weight Initialization	Uniform, lecun_uniform, normal, zero, glorot_normal, glorot_uniform, he_normal, he_uniform
Activation Function	relu, tanh, sigmoid, hard_sigmoid, linear
Dropout	0, 0.1, 0.2, 0.3, 0.4, 0.5
Weight constraints	1, 2, 3, 4, 5
Neurons	10, 20, 30, 40, 50, 60, 70, 80, 90, 100
Hidden layers	1, 2, 3

3.7.1 Batch Size

The batch size is a hyperparameter that gives an idea about the number of samples to be considered for tuning before the internal model parameters are updated. Batch size represents the length of the input data sequence that model can utilize for one time.

3.7.2 Epochs

The number of epochs is a hyperparameter that gives an estimate about the number of times the learning algorithm will work for the training dataset. One epoch refers to each sample in the training dataset having a chance for updating the internal model parameters. An epoch may incorporate one or more batches. The number of epochs defines the time taken for the entire set of input training data to be passed through the neural network while training occurs. Multiple epochs during training will ensure that the network changes its weights during training. Lesser or higher epochs may lead to either underfitting or overfitting of model.

3.7.3 Optimizer

The optimization algorithm is another important hyperparameter considered for tuning. It is possible for weights and biases of the network to change, while the algorithm trains, affecting the overall model performance. If network predictions based on the model are poor, the output of the loss function is high. An optimizer is responsible for bridging the gap between updating model parameters and the loss function. Here, loss function is used to indicate tuning quality. Stochastic Gradient Descent (SGD), Root Mean Square Propagation (RMSprop), Adagrad, Adadelta, Adam, Adamax, and Nadam are some optimizers used in hyperparameter tuning.

3.7.4 Learning Rate

Learning rate is the most important hyperparameter therefore, it is very important to understand how to set it correctly in order to achieve good performance. The learning rate hyperparameter controls the rate or speed at which the model learns. Specifically, it controls the amount of apportioned error that the weights of the model are updated with each time they are updated, such as at the end of each batch of training. Given a perfectly configured learning rate, the model will learn to best approximate the function given available resources (the number of layers and the number of nodes per layer) in a given number of training epochs (passes through the training data). If the learning rate (LR) is too small, overfitting can occur. Large learning rates help to regularize the training but if the learning rate is too large, the training will diverge.

3.7.5 Weight Initialization

Weight initialization is a procedure to set the weights of a neural network to small random values that define the starting point for the optimization (learning or training) of the neural network model. Weight Initialization is a very imperative concept in deep neural networks and using the right initialization technique can heavily affect the accuracy of the deep learning model. Thus, an appropriate weight initialization technique must be employed into consideration.

3.7.6 Activation Function

Activation function decides, whether a neuron should be activated or not by calculating weighted sum and further adding bias with it. The purpose of the activation function is to introduce non-linearity into the output of a neuron. Neural network has neurons that work in correspondence of weight, bias and their respective activation function. In a neural network, we would update the weights and biases of the neurons on the basis of the error at the output. This process is known as back-propagation. Activation functions make the back-propagation possible since the gradients are supplied along with the error to update the weights and biases. A neural network without an activation function is essentially just a linear regression model.

The activation function does the non-linear transformation to the input making it capable to learn and perform more complex tasks.

3.7.7 Dropout

Dropout is a technique that drops neurons from the neural network or ‘ignores’ them during training, in other words, different neurons are removed from the network on a temporary basis**.** During training, dropout modifies the idea of learning all the weights in the network to learning just a fraction of the weights in the network. During the standard training phase, all neurons are involved and during dropout, only a few select neurons are involved with the rest 'turned off'. So after every iteration, different sets of neurons are activated, to prevent some neurons from dominating the process. This, therefore, helps us reduce the chances of overfitting, and allows for the rise of deeper and bigger network architectures that can make good predictions on data.

3.7.8 Weight constraints

Weight constraints provide an approach to reduce the overfitting of a deep learning neural network model on the training data and improve the performance of the model on new data.

3.7.9 Neurons

Neurons in a hidden layer may be defined as the number of units representing the number of neurons of a layer. Neurons are the building components of neural networks in the deep learning model. In a deep learning model, neurons can form synapses with many neurons from the preceding layer. Each synapse has a weight, which determines the importance of the previous neuron in the larger neural network. A larger network requires more training and at least the batch size and number of epochs should ideally be optimized with the number of neurons. The overall capacity of the network is controlled by the number of neurons in a layer.

3.7.10 Hidden layer

A hidden layer in a neural network is a layer in between input layers and output layers, where artificial neurons take in a set of weighted inputs and produce an output through an activation function. The number of network layers will affect the learning ability, training time, and test time of the model. Higher model layers lead to strong learning ability of model. However, the deeper the model, the higher the complexity of the model, the more difficult it is to converge, and the more difficult and time consuming is the training. Hidden layers enable neural networks to capture very complex relationships and achieve exciting performance in many tasks.

3.8 Time Step

Time steps means how much behind (in time) model looks into to make a prediction. It can be referred as number of past data points considered in addition to the current data point to make the model prediction. Time step helps to understanding the long dependence and pattern of data. High value of time step increases the complexity as well as the execution time of model.

3.9 Model evaluation statistics

To mathematically quantify the performance of the model, three statistical measures are calculated: mean absolute error (MAE), root mean square error (RMSE) and R^2. RMSE is valuable because it indicates error in the units (or squared units) of the constituent of interest which contributes to result analysis, and 0 indicates a perfect fit. The R^2 is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model. R^2 values range from negative infinity to 1, with 1 indicating an exact match between simulated and observed values. An RMSE value less than half the standard deviation of the observed data may be considered low and $R^2 > 0.5$ model performance can be considered satisfactory. The equations of the selected statistics are shown below.

$$MAE = \frac{1}{n}\sum_{i=0}^{n}|(o_i - s_i)| \tag{3.12}$$

$$RMSE = \sqrt{\frac{\sum_{i=0}^{n}(s_i - o_i)^2}{n}} \tag{3.13}$$

$$R^2 = 1 - \frac{\sum_{i=0}^{n}(o_i - s_i)^2}{\sum_{i=0}^{n}(o_i - \bar{o_i})^2} \tag{3.14}$$

Where o_i and s_i are the respective observed and simulated values for outflows and n is the total number of observations.

CHAPTER 4
RESULTS AND DISCUSSION

This chapter presents the results obtained in this study by following the procedure discussed in chapter 3. First, hyperparameters of all three models are tuned and finalized. Thereafter, all models are trained and tested. Finally, model performances are compared to each other.

4.1 Tuning of Hyperparameters

Hyperparameters are ML model parameters that control the learning process. Deep learning algorithms may have different types of epochs, batch size, neurons, optimizer, dropout regularization, weight constraint, learning rate and activation functions for generalizing data patterns. These hyperparameters are tunned for RNN, LSTM and GRU models.

4.1.1 Tuning of batch size and epochs

To account for the effect of batch size and epochs on the model performance, batch size (32, 64, 128, 256, 512) and the number of epochs (10, 50, 100, 150, 200) are varied as shown in Table 4.1. Fig. 4.1 shows the performance statistics for different combinations of batch sizes and epochs for the RNN, LSTM, and GRU models. RNN model performs better (R^2- 0.7282) with a combination of 32 batch sizes and 100 epochs. LSTM model gives better statistics with a combination of 64 batch sizes and 100 epochs. GRU model performs better with the same batch size as the LSTM model. Best epochs are 150 for GRU model which is greater than RNN and LSTM models. The batch size of the model does not have much effect on model performance whereas epochs directly affect model performance as shown in Fig. 4.1. Batch size only affects the execution time of the model. In the case of epochs, a general increase in the number of epochs improves the model's performance. However, high epochs increase execution time and may cause model overfitting.

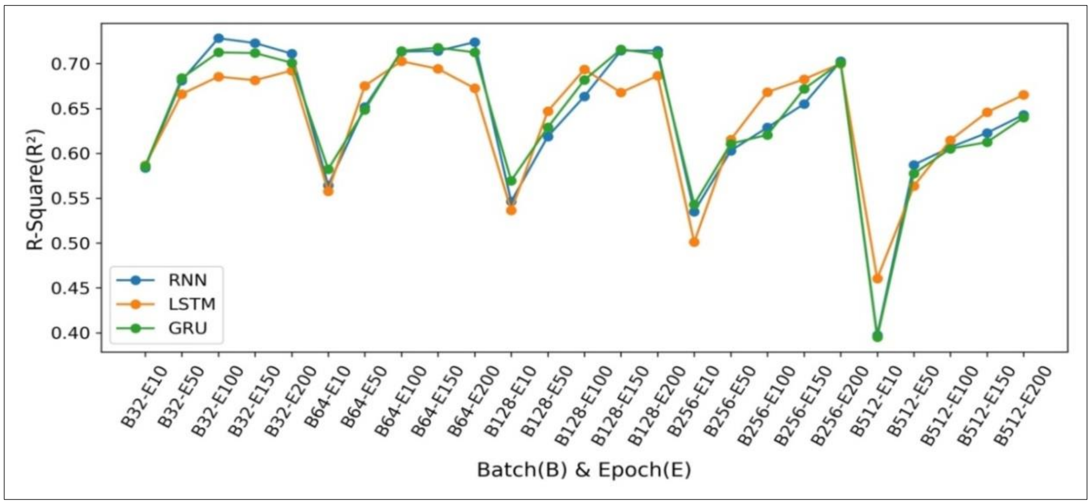

Fig. 4.1 Comparison of coefficient of determination (R^2) for different combinations of batch size and epochs for RNN, LSTM and GRU models

Table 4.1 Performance evaluation statistics of RNN, LSTM, and GRU models for various combinations of batch size and epochs

Batch size and Epochs	RNN			LSTM			GRU		
	MAE (HM/ day)	RMSE (HM/ day)	R^2	MAE (HM/ day)	RMSE (HM/ day)	R^2	MAE (HM/ day)	RMSE (HM/ /day)	R^2
B=32, E=10	276.2	879.5	0.583	283.4	879.5	0.586	288.2	879.5	0.585
B=32, E=50	236.2	770.4	0.681	243.5	788.6	0.666	239.8	766.8	0.683
B=32, E=100	**232.6**	**712.3**	**0.728**	232.6	766.8	0.685	225.3	734.1	0.712
B=32, E=150	232.6	719.6	0.722	232.6	770.4	0.681	228.9	734.1	0.711
B=32, E=200	236.2	734.1	0.711	225.3	759.5	0.692	236.2	748.6	0.700
B=64, E=10	305.2	901.3	0.564	290.7	908.6	0.557	319.8	883.1	0.582
B=64, E=50	243.5	806.8	0.651	243.5	777.7	0.675	254.4	810.4	0.648
B=64, E=100	228.9	730.5	0.713	**232.6**	**745.0**	**0.702**	236.2	730.5	0.714
B=64, E=150	228.9	730.5	0.714	232.6	755.9	0.694	**228.9**	**726.8**	**0.717**
B=64, E=200	232.6	719.6	0.723	228.9	781.4	0.672	232.6	734.1	0.712
B=128, E=10	327.1	919.5	0.546	319.8	930.4	0.536	308.9	894.0	0.569
B=128, E=50	254.4	843.1	0.618	254.4	810.4	0.646	261.6	832.2	0.628
B=128, E=100	243.5	792.3	0.663	239.8	755.9	0.693	243.5	770.4	0.681
B=128, E=150	225.3	730.5	0.714	236.2	788.6	0.667	228.9	726.8	0.715
B=128, E=200	228.9	730.5	0.714	228.9	766.8	0.686	228.9	734.1	0.710
B=256, E=10	356.1	930.4	0.535	323.4	963.1	0.500	327.1	923.1	0.543
B=256, E=50	261.6	861.3	0.603	268.9	846.8	0.615	265.3	850.4	0.610
B=256, E=100	258.0	832.2	0.628	250.7	785.0	0.668	265.3	839.5	0.620
B=256, E=150	243.5	803.2	0.654	239.8	770.4	0.682	232.6	781.4	0.671
B=256, E=200	232.6	745.0	0.702	228.9	748.6	0.699	236.2	748.6	0.701
B=512, E=10	519.7	1057.6	0.397	410.6	1003.0	0.460	566.9	1057.6	0.395
B=512, E=50	268.9	875.8	0.587	283.4	901.3	0.563	287.1	886.7	0.577
B=512, E=100	261.6	857.7	0.606	272.5	846.8	0.614	265.3	857.7	0.605
B=512, E=150	258.0	839.5	0.622	258.0	814.1	0.645	265.3	850.4	0.612
B=512, E=200	250.7	817.7	0.642	243.5	792.3	0.665	250.7	810.4	0.639

Note: Bold values represents the best case

4.1.2 Optimizer

An optimizer is a hyperparameter responsible for bridging the gap between updating model parameters and the loss function. There are various optimizers used for tunning in ML models as shown in Table 4.2. Fig 4.2 shows the coefficient of determination for various optimizers in RNN, LSTM, and GRU models. RNN model shows best performance statistics with NAdam optimizer (R^2- 0.662) and relatively poor statistics with Adagrad optimizer as compared to other optimizers. Adam optimizer shows the best results for the LSTM model (R^2- 0.709). In the case of the GRU model, the Adamax optimizer is more suitable with an R^2 of 0.680. Adadelta optimizer in both LSTM and GRU models performs poorly.

Table 4.2 Performance evaluation statistics of RNN, LSTM, and GRU models for various optimizers

Optimizers	RNN			LSTM			GRU		
	MAE (HM/ day)	RMSE (HM/ day)	R^2	MAE (HM/ day)	RMSE (HM/ day)	R^2	MAE (HM/ day)	RMSE (HM/ day)	R^2
SGD	430.6	1248.2	0.167	375.6	1038.4	0.424	461.7	1287.6	0.112
RMSprop	291.0	797.5	0.660	238.4	751.8	0.699	272.8	834.8	0.629
Adagrad	503.8	1370.6	0.004	383.0	1040.8	0.422	517.1	1360.6	0.008
Adadelta	548.3	1362.0	0.008	767.1	1418.1	0.085	885.7	1463.5	0.150
Adam	290.8	817.9	0.642	**228.3**	**737.6**	**0.709**	265.1	794.7	0.664
Adamax	309.2	812.3	0.647	244.7	812.6	0.647	**277.4**	**773.7**	**0.680**
Nadam	**272.7**	**795.0**	**0.662**	229.2	741.2	0.707	273.8	807.5	0.652

Note: Bold values represents the best case.

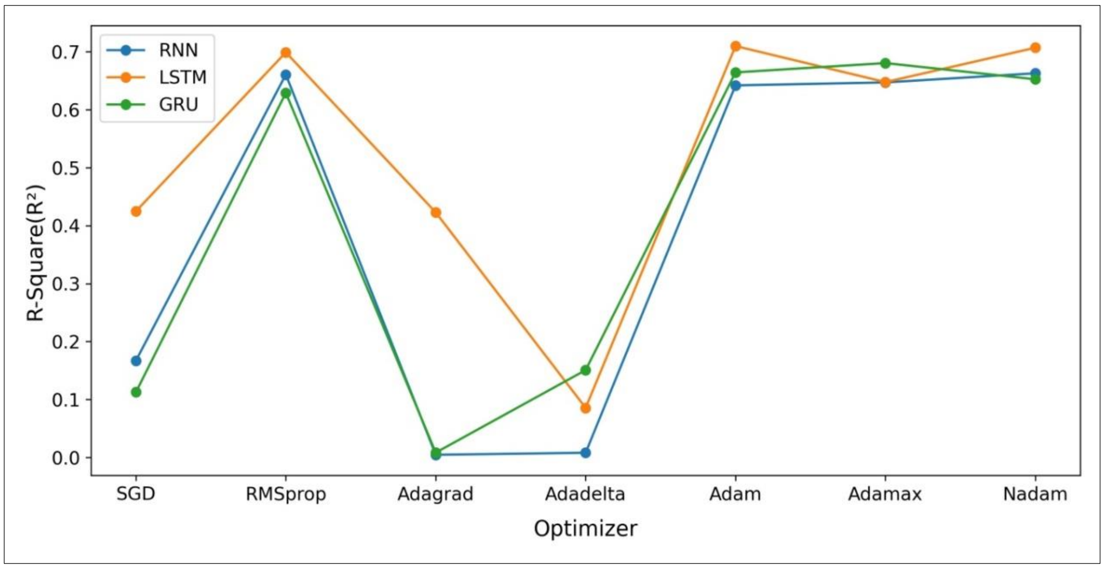

Fig. 4.2 Comparison of coefficient of determination (R^2) for various optimizers of RNN, LSTM and GRU models

4.1.3 Learning Rate

Learning rate controls the rate or speed at which the model learns during the training of the model. To study the effect of learning rate on model performance, learning rate (0.0001, 0.001, 0.01, 0.1) is varied as shown in Table 4.3. RNN model performs better (R^2-0.589) with a learning rate of 0.01. In the case of LSTM and GRU, models show the best performance statistics with a learning rate of 0.001. The best coefficient of determination for LSTM and GRU models are 0.703 and 0.713, respectively. Fig. 4.3 shows that when learning rate of RNN model is greater than 0.01 then, the performance of the model starts to decrease. Similarly, when the learning rate is more than 0.001, the performance of LSTM and GRU models begins to decrease because in case of large learning rate, gradient descent may inadvertently increase the training error rather than decreasing it. When the learning rate is too small, model training is slower. Large learning rates help to regularize the training but if the learning rate is too large, the training will diverge, especially for RNN model, as shown in Fig. 4.3.

Table 4.3 Performance evaluation statistics of RNN, LSTM, and GRU models for various learning rate

Learning rate	RNN			LSTM			GRU		
	MAE (HM/ day)	RMSE (HM/ day)	R^2	MAE (HM/ day)	RMSE (HM/ day)	R^2	MAE (HM/ day)	RMSE (HM/ day)	R^2
0.0001	288.46	936.81	0.531	268.83	868.94	0.596	272.06	883.33	0.583
0.001	253.27	887.45	0.579	**234.98**	**745.57**	**0.703**	**237.68**	**732.66**	**0.713**
0.01	**262.43**	**875.92**	**0.589**	234.50	745.80	0.703	245.68	749.81	0.699
0.1	500.79	1120.98	0.08	439.82	1003.22	0.447	368.08	1042.20	0.241

Note: Bold values represents the best case.

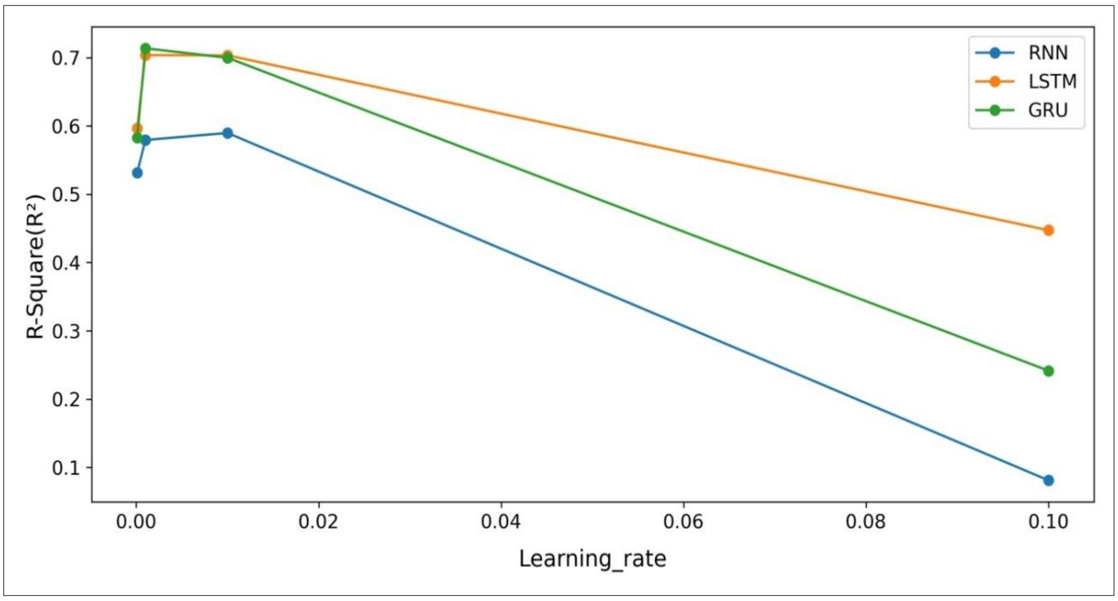

Fig. 4.3 Comparison of coefficient of determination (R^2) for various learning rate of RNN, LSTM and GRU models

4.1.4 Weight Initialization Methods

To account for the effect of weight initialization on the model performance, various weight initialization methods as shown in Table 4.4. Fig. 4.4 shows the performance statistics for different weight initialization methods for the RNN, LSTM, and GRU models. RNN model performs better (R^2- 0.667) for the glorot_uniform weight initialization method. Both LSTM and GRU models show better performance statistics with the he_normal weight initialization method, with 0.713 and 0.685 coefficient of determination respectively. For the zero weight initialization approach, RNN, LSTM, and GRU models are poorly trained cause of model training like a linear model.

Table 4.4 Performance evaluation statistics of RNN, LSTM, and GRU models for various weight initialization methods

Weight Initialization	RNN			LSTM			GRU		
	MAE (HM/ day)	RMSE (HM/ day)	R^2	MAE (HM/ day)	RMSE (HM/ day)	R^2	MAE (HM/ day)	RMSE (HM/ day)	R^2
uniform	277.5	797.8	0.661	266.8	768.5	0.685	275.0	783.3	0.672
lecun_uniform	277.8	807.6	0.652	259.8	745.1	0.703	279.1	799.8	0.658
normal	280.6	796.9	0.661	268.0	772.3	0.681	264.7	775.5	0.679
zero	500.7	1120.9	0.08	500.7	1120.9	0.08	500.7	1119.9	0.08
glorot_normal	283.0	805.8	0.653	261.3	758.1	0.693	268.7	803.9	0.655
glorot_uniform	**282.7**	**789.8**	**0.667**	264.6	751.4	0.698	264.0	801.9	0.657
he_normal	284.2	794.1	0.663	**260.3**	**731.5**	**0.713**	**262.1**	**767.1**	**0.685**
he_uniform	282.2	817.6	0.643	267.4	741.6	0.705	265.6	798.3	0.660

Note: Bold values represents the best case.

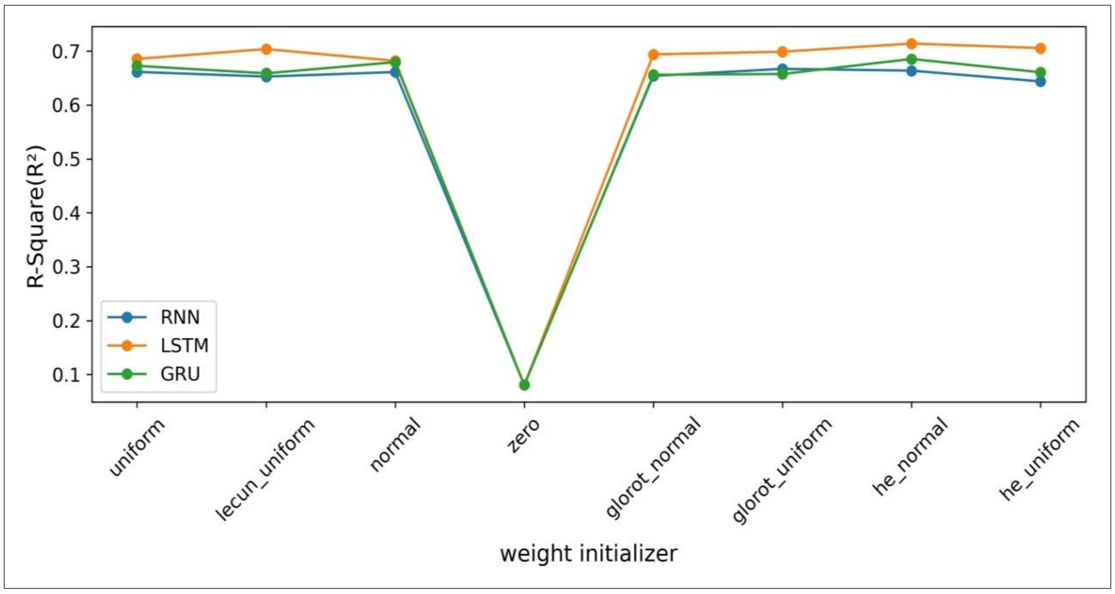

Fig. 4.4 Comparison of coefficient of determination (R^2) for various weight initialization methods of RNN, LSTM and GRU models

4.1.5 Neuron Activation Function

The activation function makes the model capable to learn and perform more complex tasks. To study the effect of various neuron activation functions on RNN, LSTM and GRU models are shown in Table 4.5. Sigmoid activation function shows best results for RNN and LSTM models with 0.691 and 0.708 R^2 values respectively. GRU model shows the best performance statistics with linear activation function (R^2-0.714). Fig 4.5 illustrates that the performance of the RNN model is poor with a linear activation function. In comparison to the RNN model, the LSTM and GRU models perform better for all activation functions.

Table 4.5 Performance evaluation statistics of RNN, LSTM, and GRU models for various neuron activation functions

Activation Function	RNN			LSTM			GRU		
	MAE (HM/ day)	RMSE (HM/ day)	R^2	MAE (HM/ day)	RMSE (HM/ day)	R^2	MAE (HM/ day)	RMSE (HM/ day)	R^2
relu	267.69	885.98	0.643	234.99	744.74	0.703	237.55	763.45	0.689
tanh	253.16	881.12	0.650	230.82	742.25	0.706	231.69	745.47	0.703
sigmoid	**254.71**	**888.83**	**0.691**	**236.33**	**739.26**	**0.708**	238.79	749.98	0.700
hard_sigmoid	259.06	896.03	0.641	234.17	750.95	0.699	238.31	737.06	0.710
linear	257.98	879.62	0.630	232.04	745.62	0.703	**235.86**	**732.25**	**0.714**

Note: Bold values represents the best case.

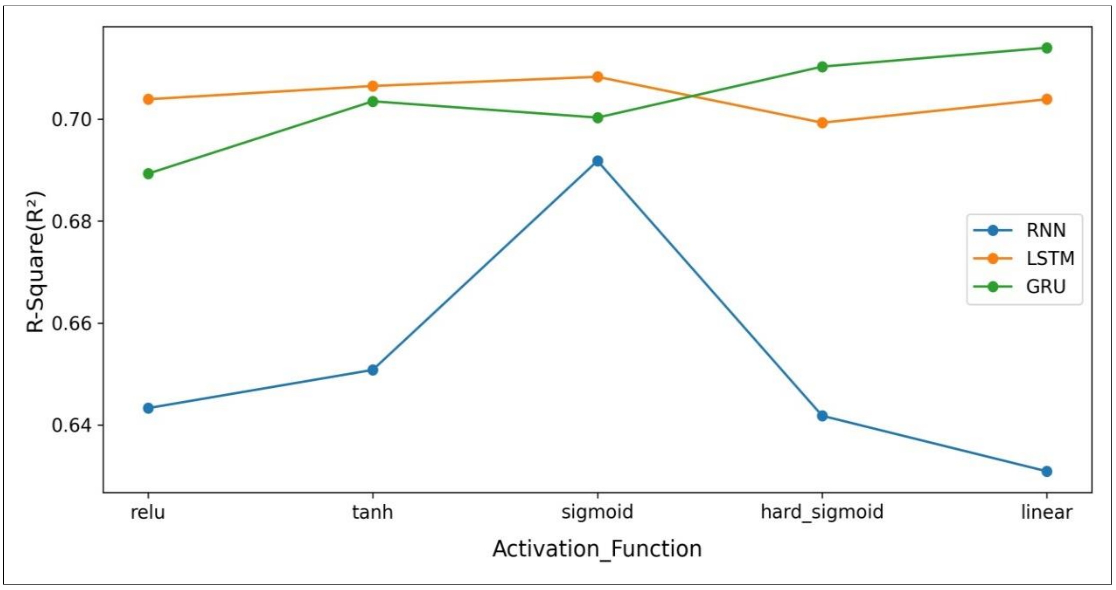

Fig. 4.5 Comparison of coefficient of determination (R^2) for various neuron activation functions of RNN, LSTM and GRU models

4.1.6 Dropout Regularization and weight constraints

To account for the effect of dropout regularization and weight constraints on the model performance, dropout regularization (0. 0.1, 0.2, 0.3, 0.4, 0.5) and weight constraints (1, 2, 3, 4, 5) are varied as shown in Table 4.6. RNN model performs better (R^2- 0.678) with the combination of zero dropouts and 1 weight constraint. LSTM and GRU models give better statistics with a combination of 0.1 dropouts and 2 weight constraints. Fig 4.2 shows the coefficient of determination for various dropout regularization and weight constraints. When the dropout value exceeds 0.1, the performance of all models begins to deteriorate. RNN model performs poorly with increasing dropout regularization and weight constraints as compared to LSTM and GRU models.

Table 4.6 Performance evaluation statistics of RNN, LSTM, and GRU models for various dropout rate and weight constraint

Dropout(D)	RNN			LSTM			GRU		
Weight constraint(W)	MAE (HM/day)	RMSE (HM/day)	R^2	MAE (HM/day)	RMSE (HM/day)	R^2	MAE (HM/day)	RMSE (HM/day)	R^2
D 0-W 1	**267.80**	**776.81**	**0.678**	241.51	751.85	0.699	252.49	746.12	0.703
D 0-W 2	316.82	824.61	0.638	242.17	729.13	0.71	238.73	766.63	0.687
D 0-W 3	291.53	811.46	0.649	232.49	748.03	0.702	273.73	758.44	0.693
D 0-W 4	284.10	786.55	0.670	237.25	749.46	0.701	244.09	743.13	0.705
D 0-W 5	286.04	781.33	0.674	244.07	737.32	0.710	266.80	761.06	0.691
D 0.1-W 1	253.04	887.27	0.579	235.50	744.3	0.704	232.56	749.08	0.700
D 0.1-W 2	255.56	886.83	0.580	**232.07**	**734.26**	**0.712**	231.94	742.10	0.706
D 0.1-W 3	259.69	883.92	0.582	236.19	746.34	0.703	234.59	743.50	0.705
D 0.1-W 4	265.03	883.57	0.583	234.18	742.71	0.706	231.80	746.96	0.702
D 0.1-W 5	258.58	883.35	0.583	232.84	755.92	0.695	**242.02**	**737.70**	**0.710**
D 0.2-W 1	253.82	888.56	0.578	234.33	761.68	0.690	232.10	741.69	0.706
D 0.2-W 2	256.97	884.29	0.583	235.43	750.01	0.700	236.76	746.83	0.702
D 0.2-W 3	254.12	885.59	0.581	236.86	748.36	0.701	237.84	752.25	0.698
D 0.2-W 4	260.07	886.41	0.580	234.87	751.76	0.698	234.33	757.83	0.693
D 0.2-W 5	261.34	884.64	0.582	233.33	748.68	0.701	236.98	751.55	0.698
D 0.3-W 1	266.29	892.57	0.573	240.42	758.58	0.693	241.29	760.35	0.691
D 0.3-W 2	259.94	893.68	0.573	241.83	768.12	0.685	238.99	758.16	0.693
D 0.3-W 3	259.94	888.25	0.579	235.93	749.69	0.700	237.54	754.32	0.696
D 0.3-W 4	258.15	887.94	0.579	235.99	755.15	0.696	240.65	756.85	0.694
D 0.3-W 5	256.55	887.28	0.578	237.08	766.14	0.687	235.02	747.89	0.702
D 0.4-W 1	260.50	894.27	0.573	239.36	750.68	0.699	232.90	765.65	0.686
D 0.4-W 2	254.82	892.98	0.573	236.47	765.04	0.688	240.72	778.19	0.676
D 0.4-W 3	261.22	898.37	0.568	245.60	758.26	0.693	245.53	796.57	0.661
D 0.4-W 4	264.48	893.42	0.574	239.26	777.51	0.678	235.54	758.45	0.692
D 0.4-W 5	258.80	898.54	0.569	240.07	758.23	0.692	241.40	776.73	0.677
D 0.5-W 1	259.11	887.98	0.579	243.94	764.79	0.687	234.74	765.80	0.687
D 0.5-W 2	259.42	899.35	0.567	243.25	763.71	0.689	240.95	780.79	0.674
D 0.5-W 3	261.26	896.55	0.571	241.72	778.18	0.677	241.30	769.47	0.683
D 0.5-W 4	261.16	904.07	0.563	240.39	765.60	0.687	240.24	787.44	0.669
D 0.5-W 5	259.91	903.73	0.564	236.68	758.70	0.693	242.45	778.77	0.676

Note: Bold values represents the best case.

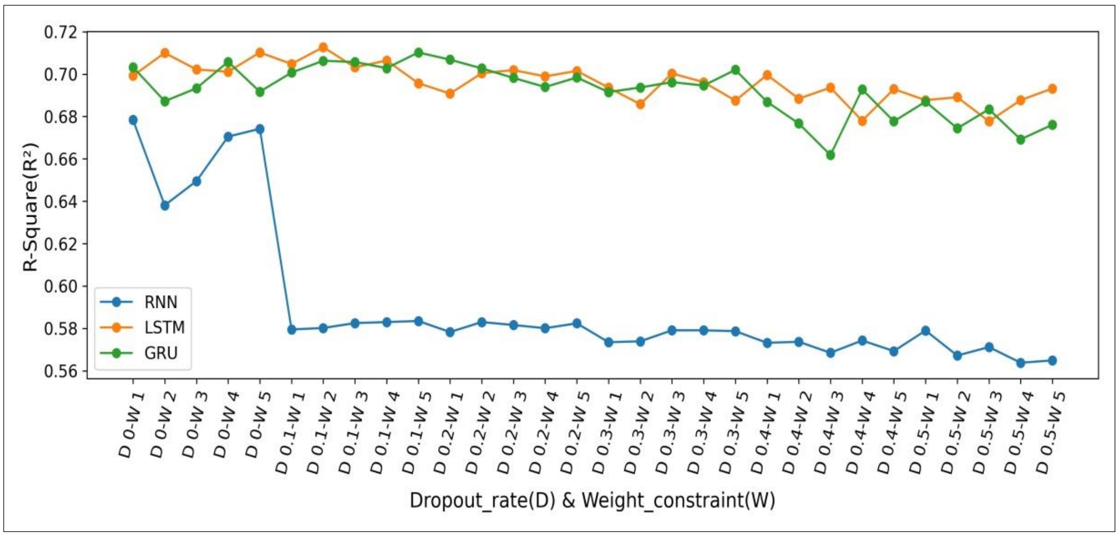

Fig. 4.6 Comparison of coefficient of determination (R^2) for various dropout rate and weight constraint of RNN, LSTM and GRU models

4.1.7 Number of neurons

Neurons are varied from 10 to 100 at a 10 neurons interval for analyzing the effect neurons on model performance of RNN, LSTM, and GRU models as shown in Table 4.7. RNN model performs better (R^2-0.6745) with 70 neurons. In the case of LSTM and GRU models, 60 neurons perform better with an R^2 value of 0.701 and 0.716, respectively. Fig. 4.7 illustrates that the performance of the model improves with increasing neurons value but after 60 neurons for the RNN model and 70 neurons for LSTM and GRU models performance starts to decline. RNN model required high neurons as compared to LSTM and GRU models. In the case of neurons, generally, an increase in the number of neurons improves the model's performance. However, high neurons increase execution time and may cause model overfitting.

Table 4.7 Performance evaluation statistics of RNN, LSTM, and GRU models for various neurons

Neurons	RNN			LSTM			GRU		
	MAE (HM/ day)	RMSE (HM/ day)	R^2	MAE (HM/ day)	RMSE (HM/ day)	R^2	MAE (HM/ day)	RMSE (HM/ day)	R^2
10	255.97	798.31	0.658	246.99	769.24	0.684	240.90	800.19	0.658
20	267.00	801.33	0.658	236.24	759.15	0.692	239.76	785.98	0.670
30	301.23	804.99	0.654	233.79	758.56	0.692	237.89	749.94	0.699
40	274.04	782.82	0.673	233.48	754.09	0.697	231.40	737.36	0.710
50	361.81	813.47	0.645	238.18	765.90	0.687	239.27	758.54	0.693
60	278.28	798.55	0.660	**232.97**	**748.18**	**0.701**	**233.23**	**728.21**	**0.716**
70	**260.38**	**781.57**	**0.674**	239.73	754.94	0.696	231.28	750.22	0.700
80	287.84	788.96	0.667	232.73	768.75	0.685	261.80	734.62	0.712
90	268.18	794.09	0.663	238.72	801.75	0.657	243.65	741.99	0.706
100	291.93	826.66	0.636	235.65	762.85	0.690	242.96	740.65	0.707

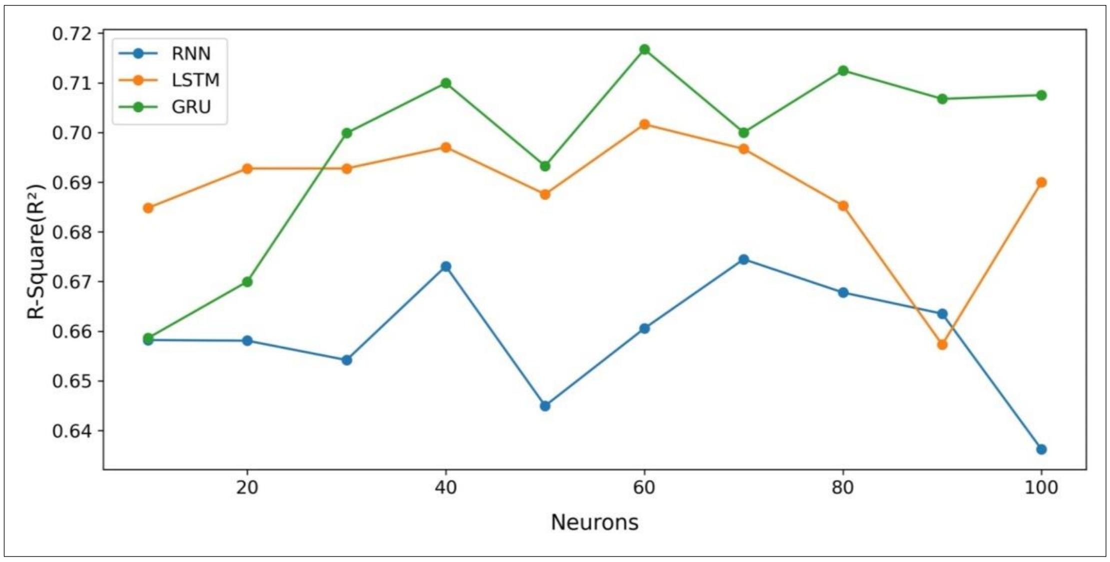

Fig. 4.7 Comparison of coefficient of determination (R^2) for various neurons of RNN, LSTM and GRU models

4.1.8 Finalized hyperparameters for RNN, LSTM and GRU models

Select the optimum hyperparameters for a model based on statistical performance as shown in Table 4.8. These finalized hyperparameters are used for training RNN, LSTM and GRU models for the maithon reservoir.

Table 4.8 Finalized hyperparameter for RNN, LSTM and GRU models.

Hyperparameter	**RNN**	**LSTM**	**GRU**
Batch Size	32	32	64
Epochs	100	100	150
Optimizer	Nadam	Adam	Adamax
Learning Rate	0.01	0.001	0.001
Weight Initialization	glorot_uniform	he_normal	he_normal
Activation Function	sigmoid	sigmoid	linear
Dropout	0	0.1	0.1
Neurons	70	60	60

4.2 Training and testing of the hyper-tuned model

The finalized hyperparameters are used to develop the basic model structures. Thereafter, RNN, LSTM and GRU models for simulation of Maithon reservoir operation are trained and tested for the different number of hidden layers and various time steps.

4.2.1 RNN model performance with one hidden layer and varying time step

The effect of multiple time steps (7, 30, 180, and 365 days) on the RNN model with one hidden layer is shown in Table 4.8. RNN model performs better with a time step of 180 days, having R^2 values of 0.736 and 0.572 in the training and testing period, respectively. A high difference between the training and testing statistics for all time steps signifies more overfitting in the model. Fig. 4.8 shows the comparison of observed outflow and simulated outflow for the testing period. RNN model with one layer showed poor statistics for 30 days time step as compared to other time steps.

Table 4.9 Performance evaluation statistics of RNN model with one hidden layer for training and testing period

Time step (T) days	**Training**			**Testing**		
	MAE (HM/day)	**RMSE (HM/day)**	**R^2**	**MAE (HM/day)**	**RMSE (HM/day)**	**R^2**
T-7	245.45	636.24	0.649	290.65	904.85	0.564
T-30	320.29	678.98	0.603	375.77	940.29	0.529
T-180	**216.81**	**480.88**	**0.736**	**273.23**	**870.78**	**0.572**
T-365	241.49	506.06	0.727	285.02	890.91	0.571

Note: Bold values represents the best case.

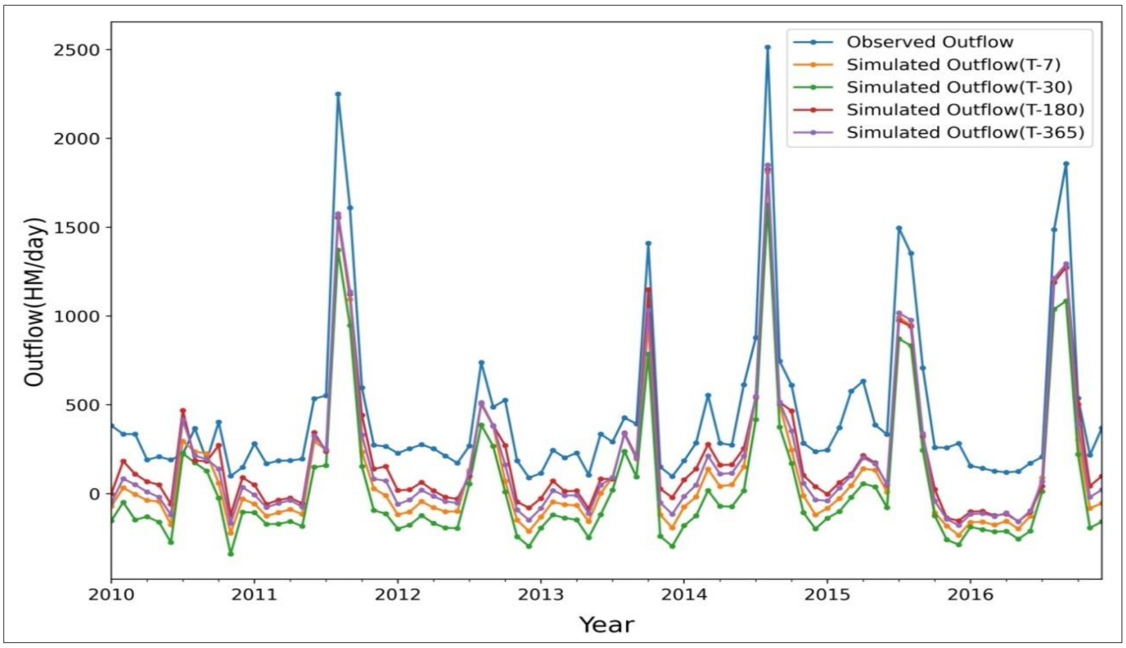

Fig. 4.8 Comparison of observed and simulated outflow at monthly scale for different time steps of one hidden layer RNN model

4.2.2 RNN model performance with two hidden layers and varying time step

Table 4.9 presents the performance evaluation statistics of the RNN model for different time steps. RNN model performs better for of 180 days' time step, having R^2 values of 0.738 and 0.584 in the training and testing period, respectively. Fig. 4.9 shows the comparison of observed and simulated outflows for the testing period. RNN model performance with time step 30 days is worst as compared to other time steps. Performance of RNN model is not much improved with increasing hidden layers, may be because of vanishing gradient problem.

Table 4.10 Performance evaluation statistics of RNN model with two hidden layers for training and testing period

Time Step(T) days	**Training**			**Testing**		
	MAE (HM/day)	**RMSE (HM/day)**	**R^2**	**MAE (HM/day)**	**RMSE (HM/day)**	**R^2**
T-7	245.00	628.35	0.658	284.25	897.00	0.572
T-30	280.05	645.15	0.642	327.10	911.71	0.558
T-180	**221.78**	**479.53**	**0.738**	**275.64**	**901.20**	**0.584**
T-365	231.98	510.51	0.723	274.59	890.84	0.571

Note: Bold values represents the best case.

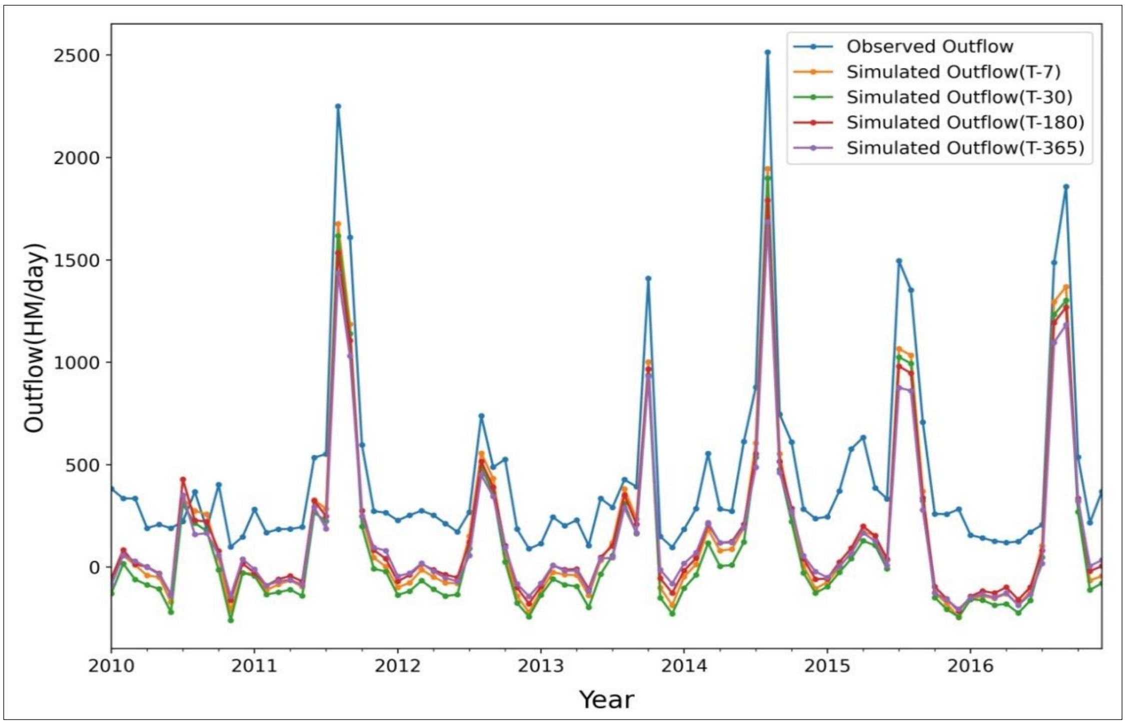

Fig. 4.9 Comparison of observed and simulated outflow at monthly scale for different time steps of two hidden layers RNN model

4.2.3 RNN model performance with three hidden layers and varying time step

The performance evaluation statistics of the RNN model with different time steps are shown in Table 4.10. The RNN model comparatively performs better with a 180 days time step, with the coefficient of determination of 0.7411 and 0.5827 in the training and testing period, respectively. A comparison of observed outflow and simulated outflow for the testing period shown in Fig. 4.10. RNN model with three layers showed poor statistics for 30 days time step as compared to other time steps. The performance statistics of an RNN model with three hidden layers are lower than with two hidden layers.

Table 4.11 Performance evaluation statistics of RNN model with three hidden layers for training and testing period

Time Step(T) days	**Training**			**Testing**		
	MAE (HM/day)	**RMSE (HM/day)**	**R^2**	**MAE (HM/day)**	**RMSE (HM/day)**	**R^2**
T-7	243.17	626.98	0.659	282.22	893.46	0.575
T-30	256.26	632.62	0.655	304.57	897.83	0.571
T-180	**216.25**	**476.85**	**0.741**	**274.86**	**903.26**	**0.582**
T-365	251.68	521.37	0.711	290.73	892.98	0.569

Note: Bold values represents the best case.

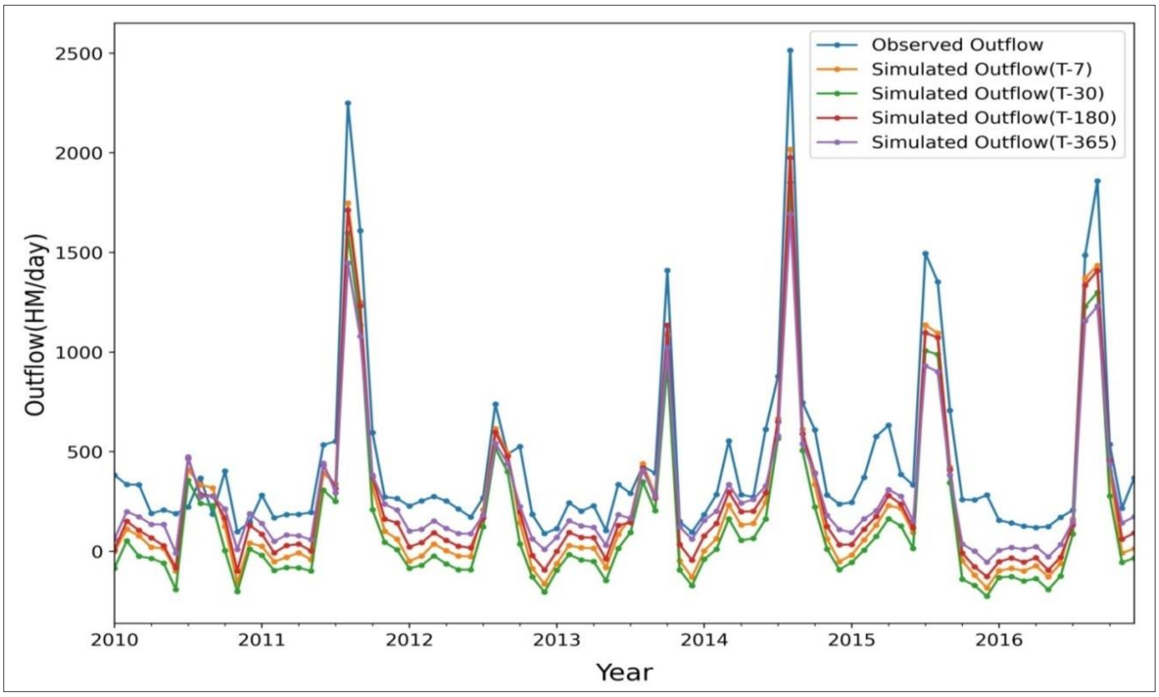

Fig. 4.10 Comparison of observed and simulated outflow at monthly scale for different time steps of three hidden layers RNN model

4.2.4 Best RNN model for maithon reservoir

RNN model shows the best performance statistics with 2 hidden layers and a time step of 180 days. The training period's MAE, RMSE, and R^2 are 221.78 HM/day, 479.53 HM/day, and 0.738, respectively, whereas the testing period's performance statistics are 275.64 HM/day, 901.20HM/day, and 0.584. RNN model overpredicts the outflow for some of the high flows, but for most other part, it underpredicts the outflow in both the training and testing period (Fig. 4.11). Fig. 4.12 more clearly shows the performance of the model on a daily scale for the testing period.

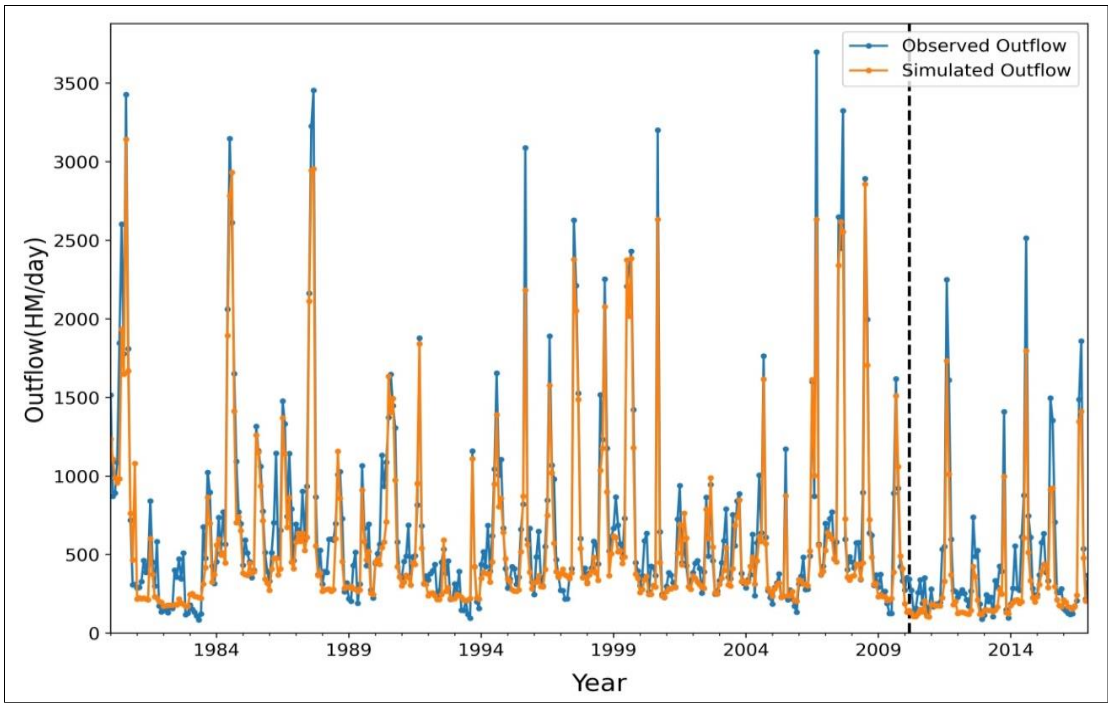

Fig. 4.11 Comparison of observed and simulated outflows by using RNN model for training and testing period at monthly scale

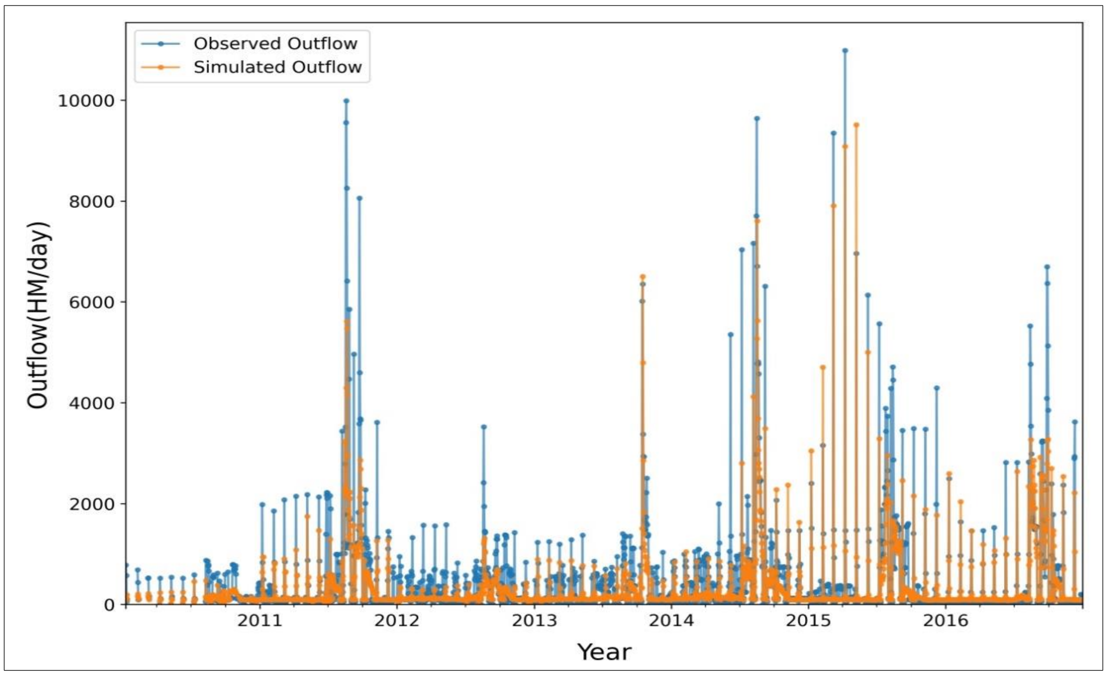

Fig. 4.12 Comparison of observed and simulated outflows for testing period by using RNN model at daily scale

4.2.5 LSTM model performance with one hidden layer and varying time step

LSTM model performance evaluation statistics with one hidden layer and varying time step (7, 30, 180, and 365 days) is shown in Table 4.12. The LSTM model performs better with a 180-day time step, with R^2 values of 0.666 and 0.641 in the training and testing period, respectively. The comparison between observed and simulated outflows for testing data is shown in Fig. 4.13. LSTM model performs poorly for time step 30 days as compared to other time steps. One hidden layer LSTM model performance is not much improved with increasing time step but perform better as compared to RNN model.

Table 4.12 Performance evaluation statistics of LSTM model with one hidden layers for training and testing period

Time step (T) days	Training			Testing		
	MAE (HM/day)	RMSE (HM/day)	R^2	MAE (HM/day)	RMSE (HM/day)	R^2
T-7	267.21	707.69	0.733	250.80	645.27	0.639
T-30	277.61	737.66	0.710	253.84	661.51	0.623
T-180	**312.75**	**807.02**	**0.666**	**244.81**	**560.83**	**0.641**
T-365	313.98	776.47	0.674	254.05	583.10	0.638

Note: Bold values represents the best case.

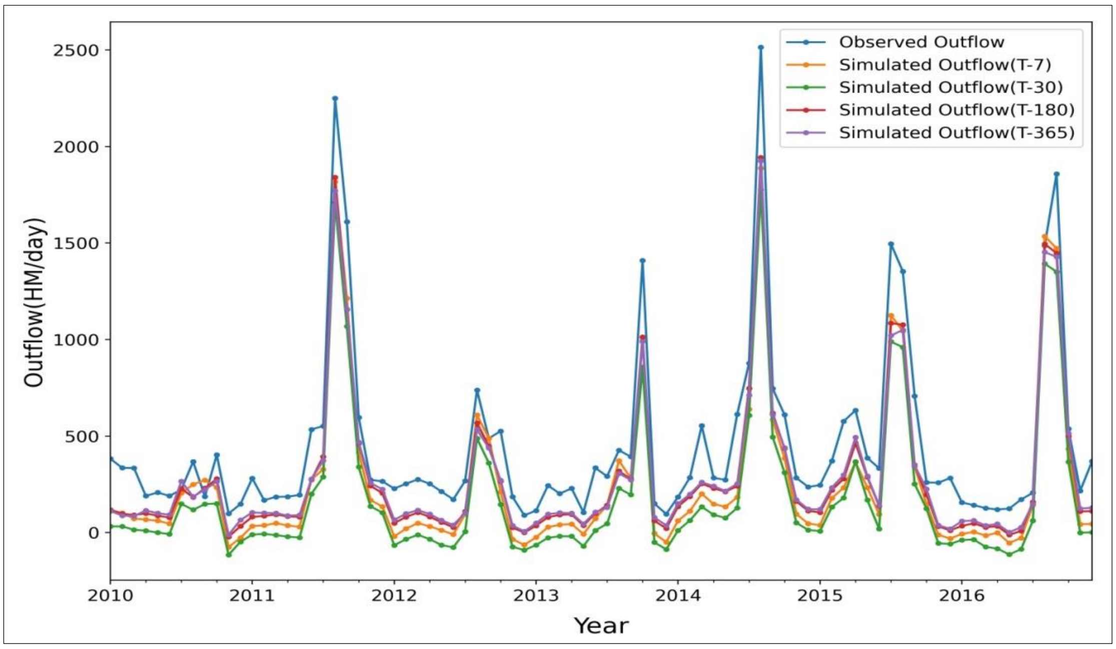

Fig. 4.13 Comparison of observed and simulated outflow at monthly scale for different time steps of one hidden layer LSTM model

4.2.6 LSTM model performance with two hidden layers and varying time step

The performance of the LSTM model with varied time steps (7, 30, 180, and 365 days) is shown in Table 4.13. LSTM model performs better with 365 days time step, having coefficient of determination as 0.736 and 0.729 in training and testing period. LSTM model performs better with increasing time step and hidden layer (Fig. 4.14). LSTM model with two layers showed poor statistics for 30 days time step as compared to other time steps. When the time step in the LSTM model is increased, the model performs better and the gap between training and testing of the LSTM model decreased.

Table 4.13 Performance evaluation statistics of LSTM model with two hidden layers for training and testing period

Time step (T) day	Training			Testing		
	MAE (HM/day)	RMSE (HM/day)	R^2	MAE (HM/day)	RMSE (HM/day)	R^2
T-7	259.08	756.61	0.695	254.86	626.21	0.660
T-30	257.83	751.19	0.699	236.21	633.45	0.655
T-180	277.19	791.44	0.719	227.17	558.66	0.706
T-365	**242.63**	**727.38**	**0.736**	**221.77**	**509.99**	**0.729**

Note: Bold values represents the best case.

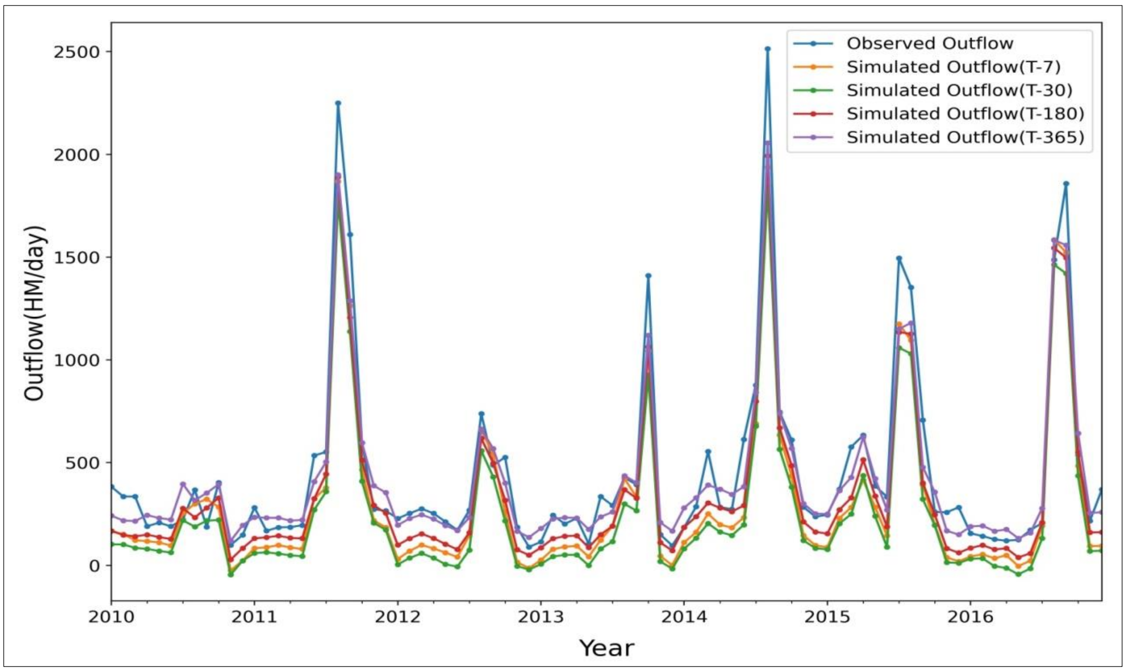

Fig. 4.14 Comparison of observed and simulated outflow at monthly scale for different time steps of two hidden layers LSTM model

4.2.7 LSTM model performance with three hidden layers and varying time step

The performance of the three hidden layers LSTM model with different time steps (7, 30, 180, and 365 days) is shown in Table 4.14. The three hidden layers LSTM model better statistics with 180-days time step, having 0.739 and 0.680 R^2 values in the training and testing period, respectively. A comparison of observed outflow and simulated outflow for the testing period is shown in Fig. 4.15. Increasing hidden layers raises the disparity between training and testing of the LSTM model that may cause model overfitting.

Table 4.14 Performance evaluation statistics of LSTM model with three hidden layers for training and testing period

Time step (T) days	Training			Testing		
	MAE (HM/day)	**RMSE (HM/day)**	**R^2**	**MAE (HM/day)**	**RMSE (HM/day)**	**R^2**
T-7	258.47	700.06	0.709	251.83	649.78	0.634
T-30	265.95	700.40	0.699	265.24	641.57	0.646
T-180	**273.66**	**713.51**	**0.739**	**234.40**	**530.02**	**0.680**
T-365	283.48	665.99	0.760	243.86	559.44	0.667

Note: Bold values represents the best case.

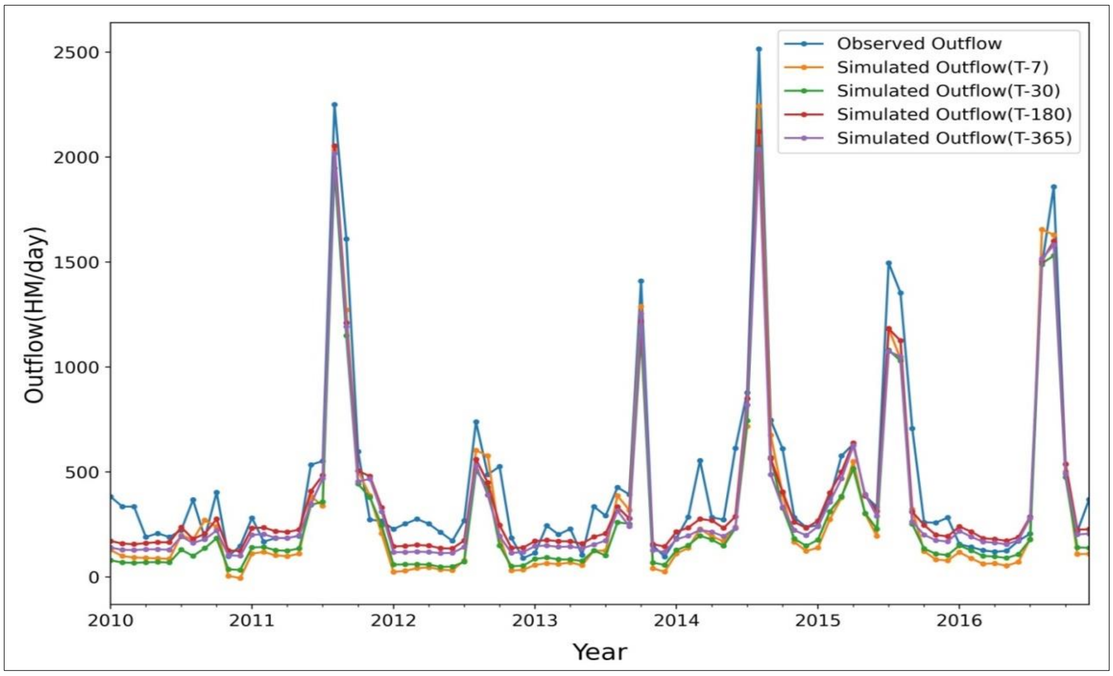

Fig. 4.15 Comparison of observed and simulated outflow at monthly scale for different time steps of three hidden layers LSTM model

4.2.8 Best LSTM model for maithon reservoir

LSTM model better performs with 2 hidden layers and 180 days time step, having the MAE, RMSE, and R^2 are 221.78 HM/day, 479.53 HM/day, and 0.738, respectively in training period, whereas the testing period performance statistics, are 275.64 HM/day, 901.20HM/day, and 0.584 respectively. LSTM model performs better in the training period as compared to the testing period (Fig. 4.16). LSTM model Performance at daily scale to testing period is shown in Fig. 4.17. LSTM model predicts the outflow better for high flow in comparison to low flows.

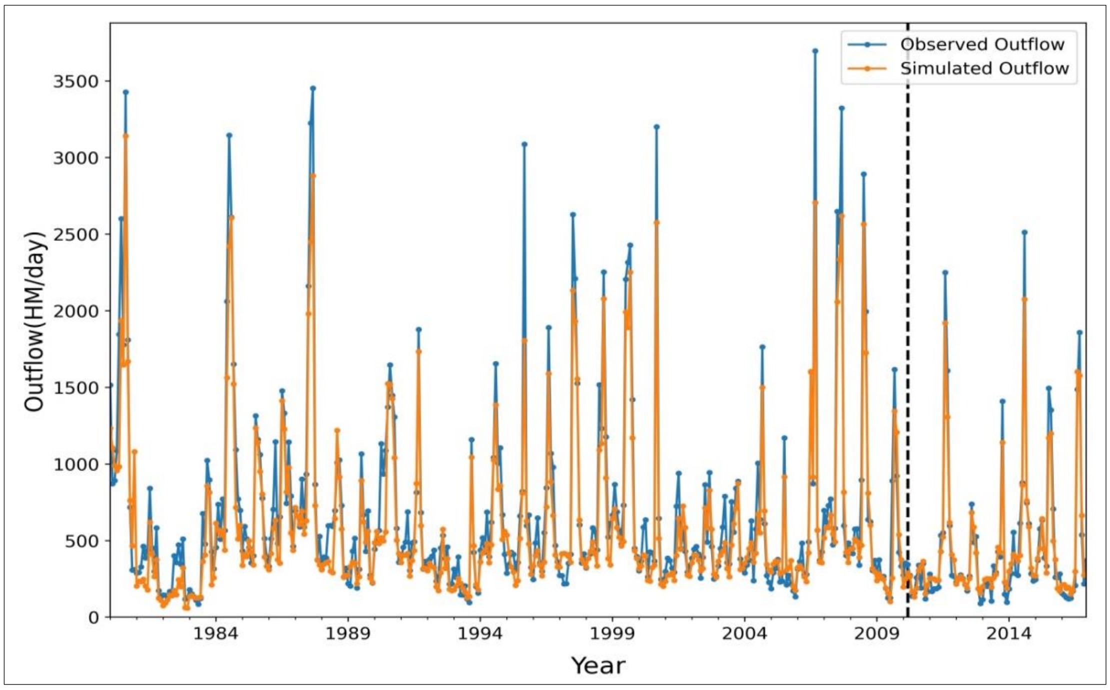

Fig. 4.16 Comparison of observed and simulated outflows by using LSTM model for training and testing period at monthly scale

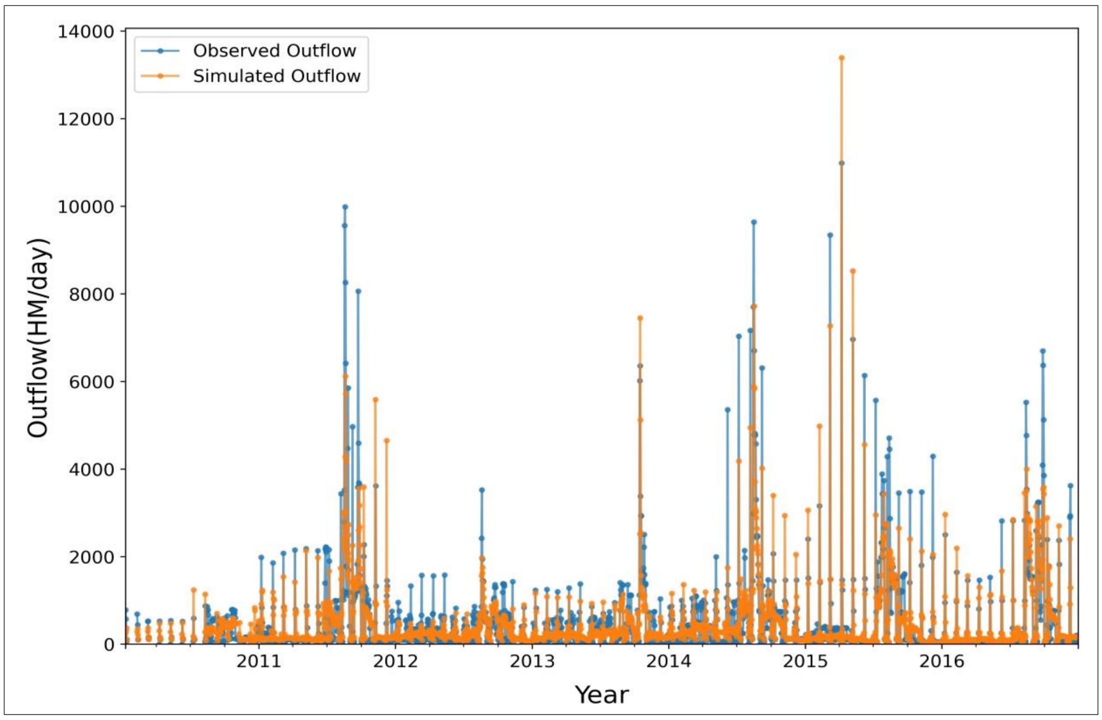

Fig. 4.17 Comparison of observed and simulated outflows by using LSTM model for testing period at daily scale

4.2.9 GRU model performance with one hidden layer and varying time step

The effect of multiple time steps (7, 30, 180, and 365 days) on the GRU model with one hidden layer is shown in Table 4.15. GRU model performs comparatively better with a time step of 30 days, having R^2 values of 0.689 and 0.669 in the training and testing period, respectively. Time step less than 30 days then, the difference between performance statistics of GRU model to training and testing period is high. When the time step is greater than 30 days, the GRU model performs better in the testing period as compared to the training period. GRU model overpredicts the low outflows and underpredicts the high outflows (Fig. 4.18).

Table 4.15 Performance evaluation statistics of GRU model with one hidden layers for training and testing period

Time step (T) days	Training			Testing		
	MAE (HM/day)	RMSE (HM/day)	R^2	MAE (HM/day)	RMSE (HM/day)	R^2
T-7	238.99	723.94	0.721	239.20	621.09	0.665
T-30	**234.10**	**763.77**	**0.689**	**218.95**	**619.66**	**0.669**
T-180	266.34	824.97	0.651	215.57	498.09	0.717
T-365	238.79	749.92	0.696	218.37	485.09	0.749

Note: Bold values represents the best case.

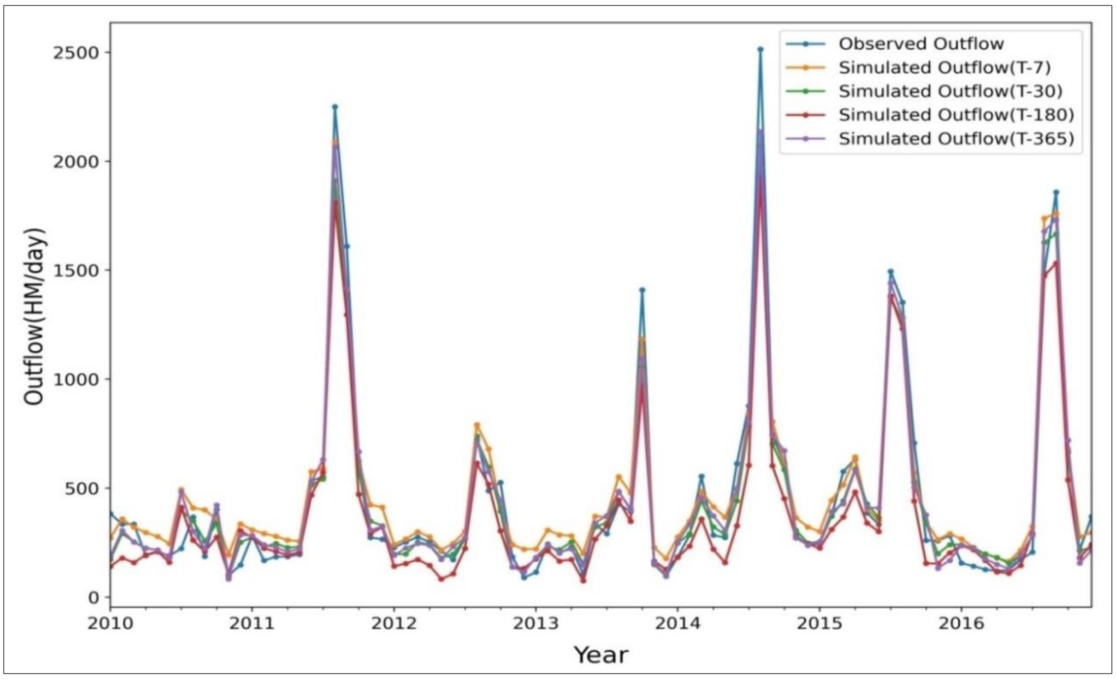

Fig. 4.18 Comparison of observed and simulated outflow at monthly scale for different time steps of one hidden layer GRU model

4.2.10 GRU model performance with two hidden layers and varying time step

Table 4.16 shows the performance evaluation statistics of two hidden layer GRU models with varied time steps (7, 30, 180, and 365 days). GRU model performs comparatively better to time step 180 days, having 0.731 and 0.722 R^2 values in the training and testing period, respectively. The difference between the model's statistical performance in the training and testing period decreases with increasing time steps

(Fig. 4.19). GRU model overpredicts the outflow in low flow and underpredicts in high flow.

Table 4.16 Performance evaluation statistics of GRU model with two hidden layers for training and testing period

Time step (T) days	Training			Testing		
	MAE (HM/day)	RMSE (HM/day)	R^2	MAE (HM/day)	RMSE (HM/day)	R^2
T-7	252.82	684.58	0.750	255.31	638.27	0.647
T-30	244.09	689.28	0.747	250.10	663.92	0.621
T-180	**244.41**	**724.03**	**0.731**	**221.04**	**493.38**	**0.702**
T-365	230.89	684.26	0.747	227.59	515.99	0.707

Note: Bold values represents the best case.

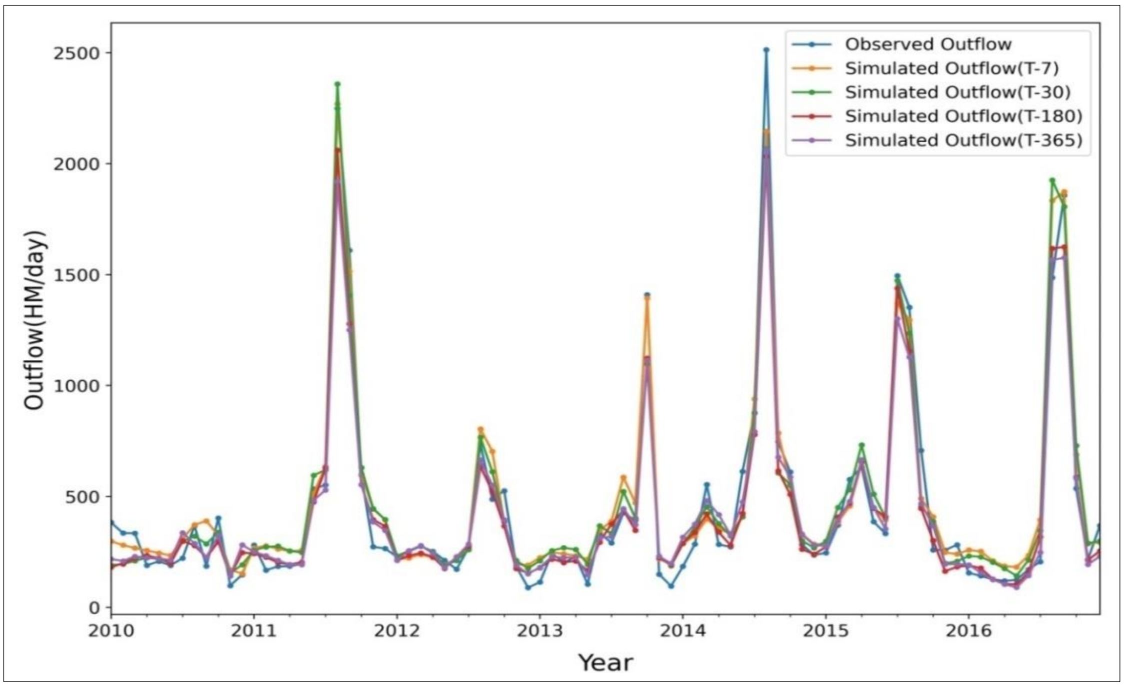

Fig. 4.19 Comparison of observed and simulated outflow at monthly scale for different time steps of two hidden layers GRU model

4.2.11 GRU model performance with three hidden layers and varying time step

The performance of the GRU model with varied time steps (7, 30, 180, and 365 days) is shown in Table 4.17. GRU model performs better with 365 days time step, having 0.768 and 0.711 R^2 in the training and testing period. GRU model performs better with increasing time step and hidden layer (Fig. 4.20). GRU model with three hidden layers showed poor statistics for 30 days time step as compared to other time steps. When the time step in the GRU model is increased, the model performs better and the gap between training and testing of the GRU model decreased.

Table 4.17 Performance evaluation statistics of GRU model with three hidden layers for training and testing period

Time step (T) days	Training			Testing		
	MAE (HM/day)	**RMSE (HM/day)**	**R^2**	**MAE (HM/day)**	**RMSE (HM/day)**	**R^2**
T-7	249.76	625.82	0.791	255.89	585.55	0.703
T-30	248.25	683.53	0.751	251.15	652.24	0.634
T-180	260.92	701.01	0.748	235.68	524.71	0.686
T-365	**242.29**	**655.66**	**0.768**	**238.22**	**520.70**	**0.711**

Note: Bold values represents the best case.

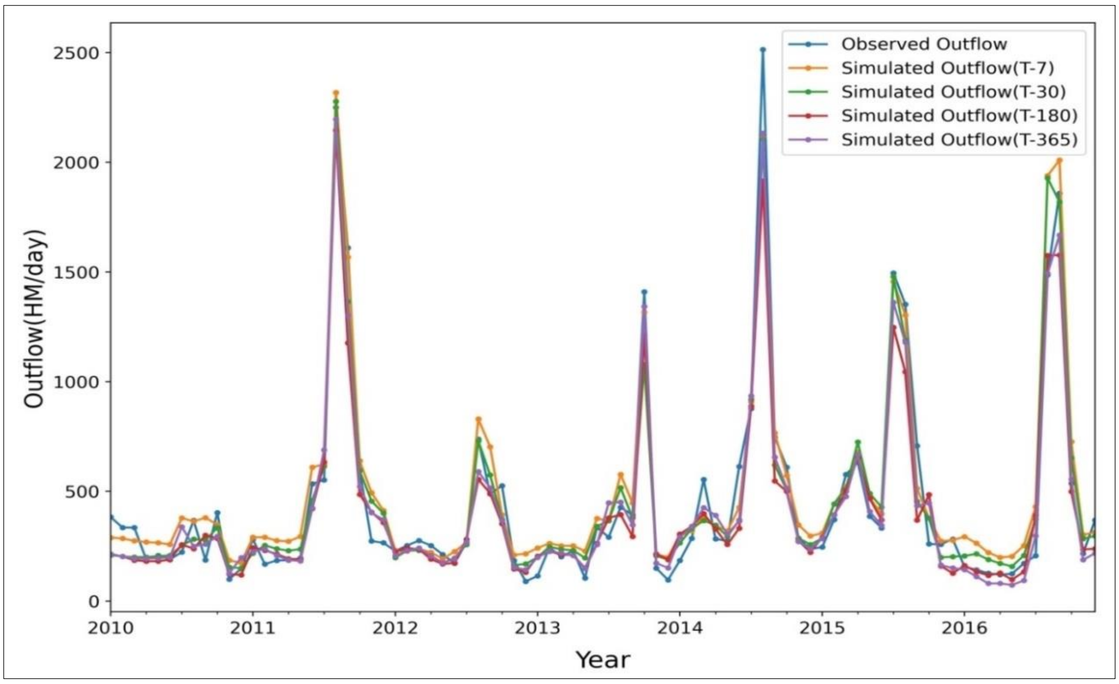

Fig. 4.20 Comparison of observed and simulated outflow at monthly scale for different time steps of three hidden layers GRU model

4.2.12 Best GRU model for maithon reservoir

GRU model performs better with three hidden layers and a 365-day time step, with MAE, RMSE, and R^2 of 242.29 HM/day, 655.66 HM/day, and 0.768, respectively, for the training period, and 238.22 HM/day, 520.70 HM/day, and 0.711, respectively, for the testing period. In comparison to the testing phase, GRU model performance statistics are better in the training period (Fig. 4.21). In high flow, the GRU model overpredicts the outflow, while in low flow, it underpredicts (Fig. 4.22). With more hidden layers and time steps, the performance of the GRU model improves, but so does the complexity and execution time.

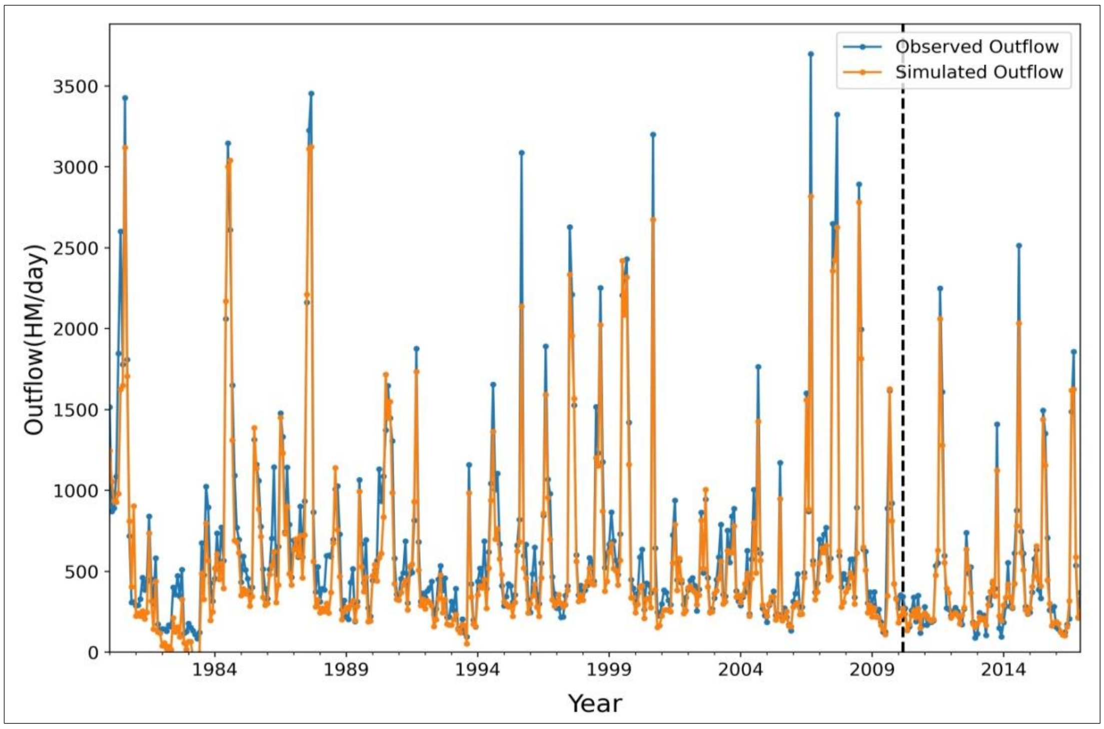

Fig. 4.21 Comparison of observed and simulated outflows by using GRU model for training and testing period at monthly scale

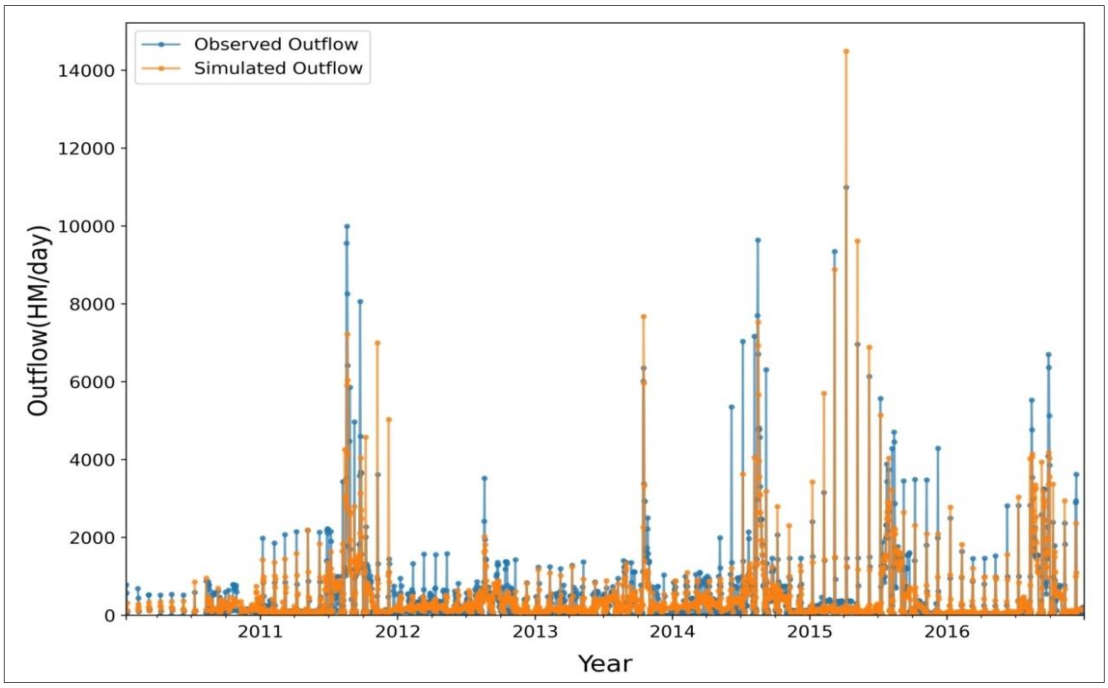

Fig. 4.22 Comparison of observed and simulated outflows by using GRU model for testing period at daily scale

4.3 Model comparison

The observed Maithon reservoir outflows are compared with simulated results based on the three different DL models, i.e., RNN, LSTM and GRU models. Prediction accuracy and residual analysis are used to compare the performance of models.

4.3.1 Comparison of RNN, LSTM and GRU models

The performance of RNN, LSTM and GRU models under the various combinations are evaluated by examining the model evaluation statistics of the respective best models. LSTM model performs better in the training and testing period as compared to RNN and GRU models, having the MAE, RMSE, and R^2 are 221.78 HM/day, 479.53 HM/day, and 0.738, respectively in training period, whereas the testing period performance statistics are 275.64 HM/day, 901.20HM/day, and 0.584 respectively (Fig. 4.23). However, among the methods applied, the LSTM network is better than other methods and can simulate peak flow periods fairly. LSTM model performs better with large time steps as compared to the RNN model. As noted above, the greatest feature of this network is its skill to learn long-term dependencies, and the forget gate makes the network keep or forget the desired amount of previous memory and thus helps to improve the modeling. The GRU method is similar to the LSTM method, and its results are close to those of LSTM.

The performance of an RNN, LSTM and GRU model for the testing period is more clearly shown on a daily scale (Fig 4.24). RNN model performs poorly as compared to LSTM and GRU models. RNN and LSTM models perform better with 2 hidden layers, while the GRU model performs better with 3 hidden layers. LSTM model predicts the low outflow more accurately, whereas the GRU model performs better at high outflows as shown in Fig 4.25. Thus, the performance ranking of models based on better performance statistics accuracy is LSTM > GRU > RNN.

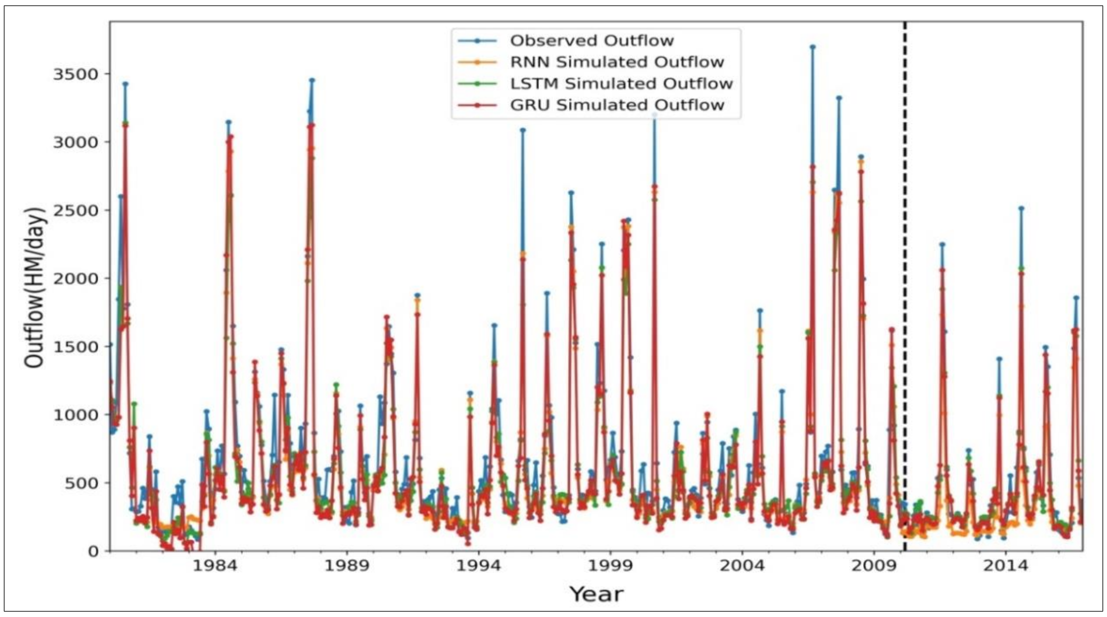

Fig. 4.23 Comparison of observed and simulated outflow for training and testing period using RNN, LSTM and GRU model at monthly scale

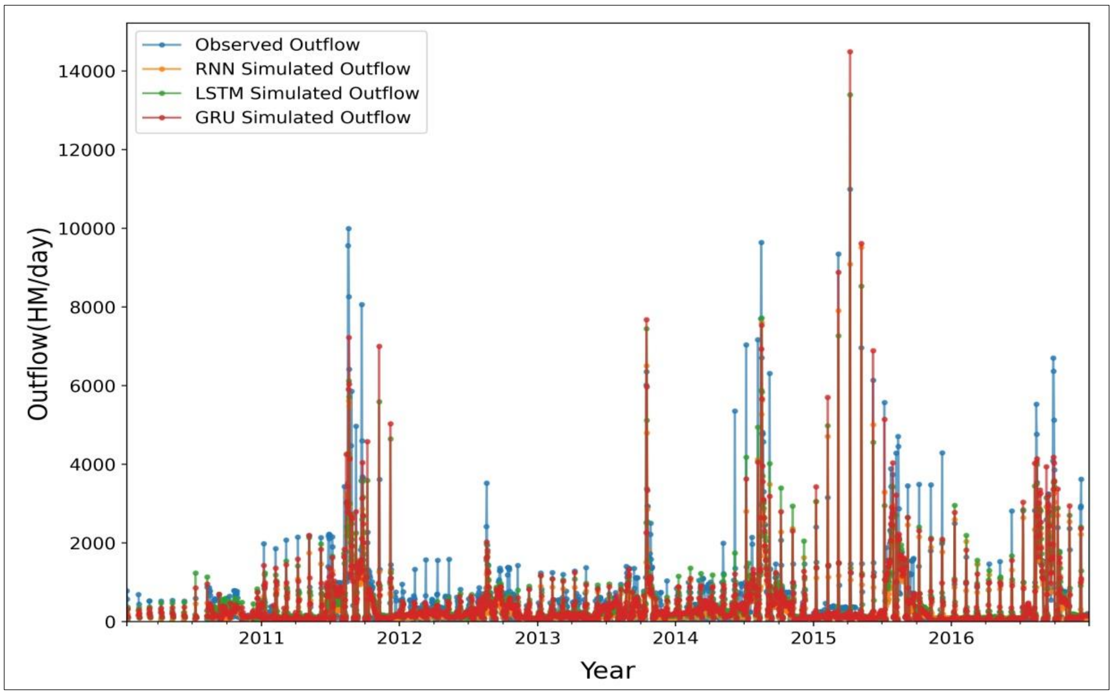

Fig. 4.24 Comparison of observed and simulated outflow for testing period using RNN, LSTM and GRU model at daily scale

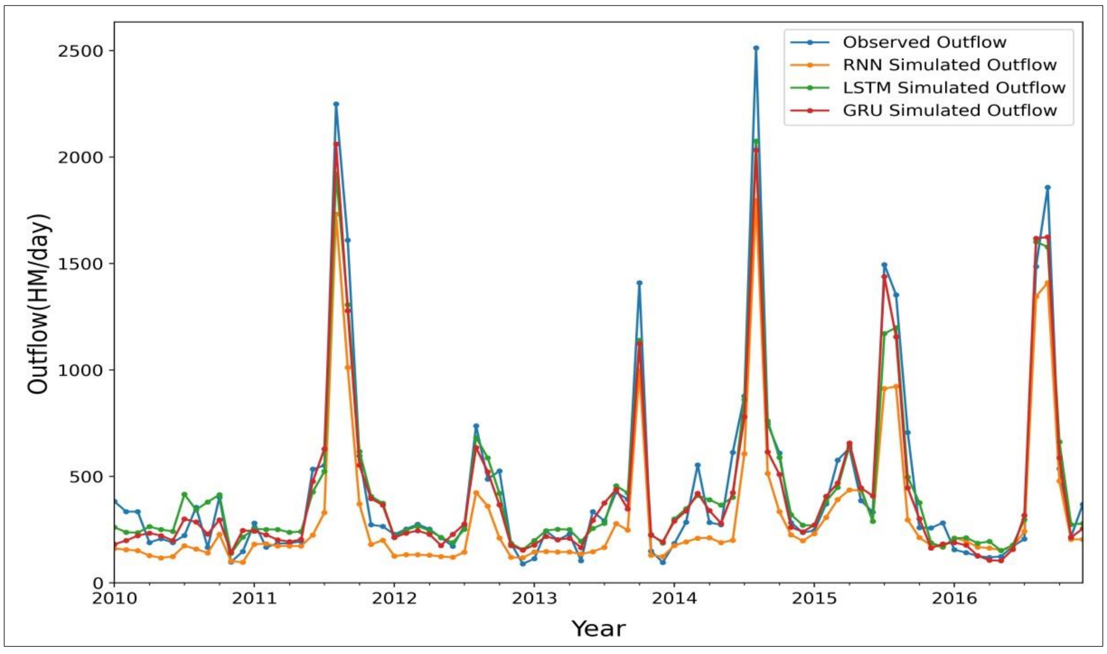

Fig. 4.25 Comparison of observed and simulated outflow for testing period using RNN, LSTM and GRU model at monthly scale

4.3.2 Residuals analysis

Fig. 4.26 shows the residual (r_s) analysis to evaluate the uncertainty of the models, performed on the statistical results of the best three models. In Fig. 4.26 (a), (c), (e) show the scatter points of r_s as a function of observed reservoir outflows. It is clear that the r_s values do not appear to be randomly distributed over the flow interval for each model, and the r_s of the all three model shows an obvious increasing trend with an increase in outflow. In Fig. 4.26 (b), (d), (f) displays the probability density distribution of r_s for the three models. The results show that the probability density distribution curve of r_s for all models presents a unimodal distribution with a sharp peak. For RNN values of r_s are mainly distributed between −1 and 1.5 (Fig. 26 (b)). For LSTM model values of r_s are mainly ranging between −1.5 and 1.5 (Fig. 26 (d)). The r_s of the GRU model are mainly distributed between −1.5 and 1.5 (Fig. 26 (f)).

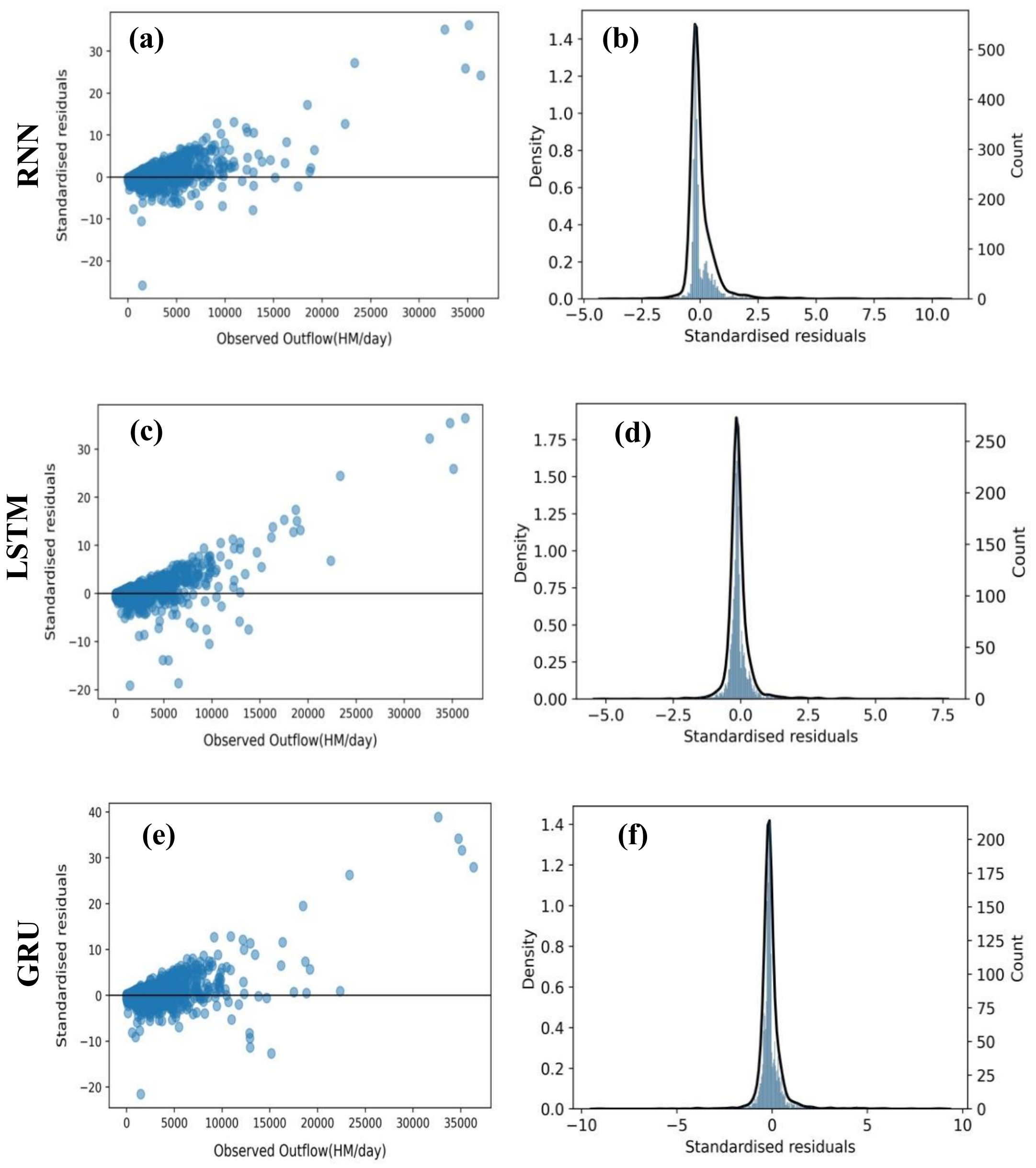

Fig. 4.26 Investigation of residuals of RNN (row1), LSTM (row2), and GRU (row3) at daily scale ((a), (c), (e)) residuals as a function of observed reservoir outflows. ((b), (d), (f)) Fitted (solid line) and actual (bars) probability density function (PDF) of the residuals

CHAPTER 5

SUMMARY AND CONCLUSION

Reservoir operation is an important part of reservoir management, and the theory and methods of reservoir operation are gradually developed with the construction of reservoirs and hydropower stations in the early 20th century. Reservoir operation models are divided into two main categories: models based on physical concepts and data-driven models. However, a physical-based model is useful only if the operating rules incorporated in the simulation can realistically reflect the actual operation. In practice, the operation of a reservoir is affected by many uncertain factors, such as natural conditions and artificial demand, and the operation often deviates from the operating rules, which limits the application of such models. AI models or data-driven models, are able to autonomously learn the operating rules from the historical operation data of a reservoir and thus have more robustness and are good at dealing with complex factors.

This project evaluated the three networks RNN, LSTM and GRU, and compared their performances. The developed simulation models are trained and tested using the historical flow and meteorological data of Maithon reservoir.

Based on the results obtained, the following conclusions were drawn:

1. RNN and LSTM models perform better with less batch size and epoch as compared to GRU model, having 32 and 64 batch sizes, 100 and 150 epochs respectively.
2. Best optimizers of RNN, LSTM and GRU models are Nadam, Adam, Adamax optimizers, respectively.
3. The RNN model has lower learning rate and dropout as compared to LSTM and GRU models.
4. RNN and LSTM showed better performance statistics with sigmoid activation function, while GRU model performance is comparatively better with linear activation function.
5. The glorot uniform weight initialization approach is suitable for RNN model whereas the he normal method gives better results for LSTM and GRU models.
6. The performance of the GRU model improves as the number of hidden layers increases, whereas the RNN and LSTM models perform better with two hidden layers.
7. In comparison to the RNN model, the LSTM and GRU models simulate outflow based on a large time step of 365 and 180 days, respectively, which help in the better simulation of outflow.
8. Overall, LSTM model's performance statistics are better with R^2 values of 0.736 and 0.729 in training and testing period, respectively.

Apaydin, H., Feizi, H., Sattari, M. T., Colak, M. S., Shamshirband, S., & Chau, K. W. (2020). Comparative analysis of recurrent neural network architectures for reservoir inflow forecasting. *Water*, *12*(5), 1500.

Chang, F. J., Chen, P. A., Lu, Y. R., Huang, E., & Chang, K. Y. (2014). Real-time multi-step-ahead water level forecasting by recurrent neural networks for urban flood control. *Journal of Hydrology*, *517*, 836-846.

Chang, Y. T., Chang, L. C., & Chang, F. J. (2005). Intelligent control for modeling of real-time reservoir operation, part II: artificial neural network with operating rule curves. *Hydrological Processes: An International Journal*, *19*(7), 1431-1444.

Chaves, P., & Chang, F. J. (2008). Intelligent reservoir operation system based on evolving artificial neural networks. *Advances in Water Resources*, *31*(6), 926-936.

Chiang, Y. M., Hao, R. N., Zhang, J. Q., Lin, Y. T., & Tsai, W. P. (2018). Identifying the sensitivity of ensemble streamflow prediction by artificial intelligence. *Water*, *10*(10), 1341.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Dynesius, M., & Nilsson, C. (1994). Fragmentation and flow regulation of river systems in the northern third of the world. *Science*, *266*(5186), 753-762.

Esmaeilzadeh, B., Sattari, M. T., & Samadianfard, S. (2017). Performance evaluation of ANNs and an M5 model tree in Sattarkhan Reservoir inflow prediction. *ISH Journal of Hydraulic Engineering*, *23*(3), 283-292.

Firat, M. A. H. M. U. T. (2008). Comparison of artificial intelligence techniques for river flow forecasting. *Hydrology and Earth System Sciences*, *12*(1), 123-139.

Hassaballah, K., Jonoski, A., Popescu, I., & Solomatine, D. P. (2012). Model-based optimization of downstream impact during filling of a new reservoir: case study of Mandaya/Roseires Reservoirs on the Blue Nile River. *Water resources management*, *26*(2), 273-293.

Heydari, M., Othman, F., & Qaderi, K. (2015). Developing optimal reservoir operation for multiple and multipurpose reservoirs using mathematical programming. *Mathematical Problems in Engineering*, *2015*.

Johnson, S. A., Stedinger, J. R., & Staschus, K. (1991). Heuristic operating policies for reservoir system simulation. *Water Resources Research*, *27*(5), 673-685.

Klipsch, J.D., Hurst, M.B., (2003). HEC- ResSim: Reservoir System Simulation 2003 User's manual US Army Corps of Engineers. Hydrologic Engineering Centre, Davis, CA.

Kumar, D. N., Raju, K. S., & Sathish, T. (2004). River Flow Forecasting using Recurrent Neural Networks. *Water Resourses Management*, 18, 143–161.

Oliveira, R., & Loucks, D. P. (1997). Operating rules for multireservoir systems. *Water resources research*, *33*(4), 839-852.

Pan, T. Y., Wang, R. Y., & Lai, J. S. (2007). A deterministic linearized recurrent neural network for recognizing the transition of rainfall–runoff processes. *Advances in Water Resources*, *30*(8), 1797-1814.

Sasireka, K., Neelakantan, T. R., & Suriyanarayanan, S. (2018). Simulation of Multipurpose Reservoir Operation with Hedging Rules.

Sattari, M. T., Yurekli, K., & Pal, M. (2012). Performance evaluation of artificial neural network approaches in forecasting reservoir inflow. *Applied Mathematical Modelling*, *36*(6), 2649-2657.

Shang, Y., Lu, S., Ye, Y., Liu, R., Shang, L., Liu, C., ... & Fan, Q. (2018). China'energy-water nexus: Hydropower generation potential of joint operation of the Three Gorges and Qingjiang cascade reservoirs. *Energy*, *142*, 14-32.

Taghi Sattari, M., Pal, M., Apaydin, H., & Ozturk, F. (2013). M5 model tree application in daily river flow forecasting in Sohu Stream, Turkey. *Water Resources*, *40*(3), 233-242.

Teegavarapu, R. S., & Simonovic, S. P. (2014). Simulation of multiple hydropower reservoir operations using system dynamics approach. *Water resources management*, *28*(7), 1937-1958.

Yang, T., Gao, X., Sorooshian, S., & Li, X. (2016). Simulating California reservoir operation using the classification and regression-tree algorithm combined with a shuffled cross-validation scheme. *Water Resources Research*, *52*(3), 1626-1651.

Yates, D., Sieber, J., Purkey, D., & Huber-Lee, A. (2005). WEAP21—A demand-, priority-, and preference-driven water planning model: part 1: model characteristics. *Water international*, *30*(4), 487-500.

Zhang, D., Lin, J., Peng, Q., Wang, D., Yang, T., Sorooshian, S., & Zhuang, J. (2018). Modeling and simulating of reservoir operation using the artificial neural network, support vector regression, deep learning algorithm. *Journal of Hydrology*, *565*, 720-736.

Zhang, D., Lin, J., Peng, Q., Wang, D., Yang, T., Sorooshian, S., & Zhuang, J. (2018). Modeling and simulating of reservoir operation using the artificial neural network, support vector regression, deep learning algorithm. *Journal of Hydrology*, *565*, 720-736.

Zhang, D., Peng, Q., Lin, J., Wang, D., Liu, X., & Zhuang, J. (2019). Simulating reservoir operation using a recurrent neural network algorithm. *Water*, *11*(4), 865.