A Comparative Study on Forecasting of Long-term Daily Streamflow using ANN, ANFIS, BiLSTM, and CNN-GRU-LSTM

Sajjad M. Vatanchi
Ferdowsi University of Mashhad Faculty of Engineering
Hossein Etemadfard (etemadfard@um.ac.ir)
Ferdowsi University of Mashhad Faculty of Engineering
Mahmoud Faghfour Maghrebi
Ferdowsi University of Mashhad Faculty of Engineering
Rouzbeh Shad
Ferdowsi University of Mashhad Faculty of Engineering

Research Article

Keywords: Streamflow prediction, ANN, ANFIS, Bidirectional LSTM, deep learning, CNN

Posted Date: March 31st, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1443377/v1

Title Page

A Comparative Study on Forecasting of Long-term Daily Streamflow using ANN, ANFIS, BiLSTM, and CNN-GRU-LSTM

Sajjad M. Vatanchi[1], Hossein Etemadfard[2] Mahmoud Faghfour Maghrebi[3], and Rouzbeh Shad[4]*

[1] PhD Student in Civil Engineering Department, Ferdowsi University of Mashhad, Mashhad, Iran. ORCID: 0000-0001-5170-4620

[2] Assistant Professor in Civil Department, Ferdowsi University of Mashhad, Mashhad, Iran. ORCID: 0000-0003-3805-1711

Email: etemadfard@um.ac.ir

[3] Professor in Civil Department, Ferdowsi University of Mashhad, Mashhad, Iran. ORCID: 0000-0002-0082-0020

[4] Associate Professor in Civil Department, Ferdowsi University of Mashhad, Mashhad, Iran. ORCID: 0000-0003-2078-8571

Abstract

One of the essential phases in water resource planning and management is streamflow forecast. It is necessary for the functioning of hydropower plants, agricultural planning, and flood control. The present study applied Artificial Neural Network (ANN) model, Adaptive Neuro-Fuzzy Inference System (ANFIS), Bidirectional LSTM (BiLSTM), and hybrid Convolutional Neural Network (CNN) Gated Recurrent Unit (GRU) Long-Short Term Memory (LSTM) model to predict the long-term daily streamflow in the Colorado River, USA. 60% of the data (1921–1981) was used for training, while 40% of the data (1981–2021) was utilized for testing the model's performance. The obtained outcomes of the suggested models were assessed using four assessment indices, including by Normalized Root Mean Square Error (NRMSE), Mean Absolute Error (MAE), Correlation Coefficient (r), and Nash–Sutcliffe Coefficient (E_{NS}). Based on the comparison of outputs, in the testing phase, it was determined that the ANFIS model with NRMSE = 0.116, MAE = 24.66, r = 0.968, and E_{NS} = 0.936 outperformed the other studied models in terms of reliability and accuracy. While the CNN-GRU-LSTM and BiLSTM models are complex, they do not perform better. The comparison demonstrates that the performance of their respective models is not much better than the two standard models-ANN and ANFIS.

Keywords: Streamflow prediction, ANN, ANFIS, Bidirectional LSTM, deep learning, CNN.

1. Introduction

Accurate streamflow forecasting is critical for water management activities, including enhancing hydroelectricity generating efficiency, irrigation planning, reservoir operation schedule, and flood management. However, due to the non-linear essence of streamflow time series, streamflow prediction remains among the most challenging issues in hydrology.

Various hydrological models and techniques have been investigated in recent decades to obtain accurate streamflow predictions. Conceptual and empirical models are the two types of approaches used. The conceptual or physically-based models are used to model the physical mechanics of different hydrological processes. The empirical model, which incorporates a data-driven model, is constructed from historical data in hydrological time series (Peng et al., 2017). A wide variety of time series methods, including autoregressive (AR) methods, autoregressive moving average (ARMA) methods, linear regression (LR), wavelet transform (WT), artificial neural networks (ANNs), support vector machines (SVMs), chaos theory, genetic programming (GP), as well as their combinations (e.g., hybrid and integrated models) are commonly used for streamflow forecasting (Sun et al. 2019). By conducting large-scale computer simulations using stochastic and machine learning methods, Papacharalampous et al. (2019) indicated that stochastic and ML methods could produce equally valuable forecasts.

Many data-driven approaches for hydrological time series forecasting have been presented in the last few decades to improve streamflow forecasting accuracy. For example, streamflow prediction was achieved using AR and ARMA methods (stochastic models) based on the time series (Mehdizadeh and Kozekalani Sales 2018). These statistical models examine the time series dataset to establish a viable method for simulating streamflow using conventional statistics. However, such models have revealed difficulties in capturing the streamflow's non-linear properties.

In comparison to the stochastic models, AI-based data-driven models such as the Artificial Neural Network (ANN), Support Vector Machine (SVM), and Fuzzy Neural Network (FNN) are superior in modelling processes (Ghimire et al. 2021). Due to its potential in addressing the related non-linearity and non-stationarity in hydrological processes, these AI models have proven an exceptional capability in hydrology and have reported a successful application for streamflow process modelling (Tao et al. 2021). One of the most apparent benefits of the ANN approach is that no well-defined mechanism for algorithmically turning inputs into outputs is required (Peng et al., 2017).

The ANN model for streamflow prediction is one of the widely used AI models in hydrology. It imitates the action of biological neurons and can solve the related non-linearity of time-series data. In one of the first investigations, Zealand et al. (1999) used the ANN model to simulate streamflow in a section of the Winnipeg River system in Northwest Ontario, Canada. The authors determined that the used ANN model outperforms the traditional Winnipeg Flow Forecasting model (WIFFS) in prediction capability. In another study, Demirel et al. (2009) investigated flow prediction using the soil and water assessment tool (SWAT) and ANN models, finding that ANN outperforms SWAT in peak flow prediction.

Several AI models have been proposed for streamflow modellings, such as support vector regression (SVR), adaptive neuro-fuzzy inference system (ANFIS), extreme learning machine (ELM), random forest (RF), and their hybridized versions with various optimization algorithms (Yaseen et al. 2020). Meshram et al. (2021) employed three AI algorithms to forecast streamflow in the Shakkar watershed (Narmada Basin), India: ANFIS, GP, and ANN.

According to the results, the ANFIS has the best performance in projecting streamflow time series for the Shakkar watershed according to the results. The implementation of ANFIS models has received much attention in the previous decade because of various advantages. One of the significant benefits of ANFIS for modelling is that it allows all causes not included in an ideal model while excluding particular causes taken into account in physically-based models (Samanataray and Sahoo 2021). Sanikhani and Kisi (2012) used two types of ANFIS models for monthly streamflow forecasting in Turkey's Firat-Dicle basin: ANFIS with grid partition (ANFIS-GP) and ANFIS with sub clustering (ANFIS-SC). According to their findings, the ANFIS-SC model performed better in streamflow forecasting than the ANFIS-GP model. An evaluation of the performance of ANN, wavelet neural network (WNN), and ANFIS models has been done for the prediction of daily streamflow for Büyük Menderes River (Dalkiliç and Hashimi 2020). The outcomes showed that the WNN model demonstrated the best performance than the ANN and ANFIS models. Riahi-Madvar et al. (2021) used the Fire-Fly Algorithm (FFA), Genetic Algorithm (GA), Grey Wolf Optimization (GWO), Particle Swarm Optimization (PSO), and Differential Evolution (DE) hybridized with ANFIS to train and assess the performance of numerous evolutionary algorithms. The results showed that all created hybrid algorithms significantly outperformed the classic ANFIS model for all prediction horizons (Daily, weekly, monthly, and annually).

New AI models represented by deep learning (DL) models have been established in a recent study for streamflow modelling. In the time-series prediction of streamflow, many DL architectures, for example, Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN), have been developed and widely employed (Ghimire et al. 2021). According to the categorization of DL, RNN is one of the usual designs that belongs to the supervised learning category and is designed to handle sequential input effectively. Due to their comparable design, the Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are two common designs in RNN-derived architectures. Both are RNNs that have evolved in different ways. For example, Zhu et al. (2020) propose a probabilistic Long Short-Term Memory network coupled with the Gaussian process (GP) to forecast daily streamflow. The proposed model performed satisfactorily, according to the results.

Additionally, several complex DL models built on LSTM and GRU architectures have also been proposed to improve the predictive performance of the models, including CNN-GRU (Le et al., 2021) and CNN-LSTM (Ni et al. 2020). The complexity of DL models is noticeable in most of the research described above because the proposed models are frequently a mix of LSTM (or GRU) with one or more other architectures. For example, the Bidirectional LSTM (BiLSTM) model is a complicated model that competes with LSTM and GRU models (with just one hidden layer). The CNN model is used to extract the inherent characteristics of the streamflow time series in this study, based on the integration of CNN, GRU, and LSTM. The GRU and LSTM models, on the other hand, use the feature derived by CNN to forecast streamflow.

The present study examines the behaviour of ANN, ANFIS, BiLSTM, and CNN-GRU-LSTM models to forecast long-term streamflow with different combinations of input discharge data sets. The CNN model is used to extract the inherent characteristics of the streamflow time series based on the integration of CNN, GRU, and LSTM. The GRU and LSTM models, on the other hand, use the feature derived by CNN to forecast streamflow.

2. Materials and Methods

2.1. ANN

The adaptive nature, learning ability, easy identification, and rapid operation of ANNs make them one of contemporary science's most popular topics. They are a cutting-edge data processing system proven to successfully cope with dynamic non-linear hydrological occurrences (Jimmy et al., 2021). It is based on the brain and nervous system's functioning, making it a reliable computing tool.

ANN architectures based on MLP are probably the most popular. The network is composed of simple neurons called perceptrons. As shown schematically in Fig. 1, the perceptron computes a single output from many real-valued inputs by creating combinations of linear connections based on input weights and even non-linear transfer functions.

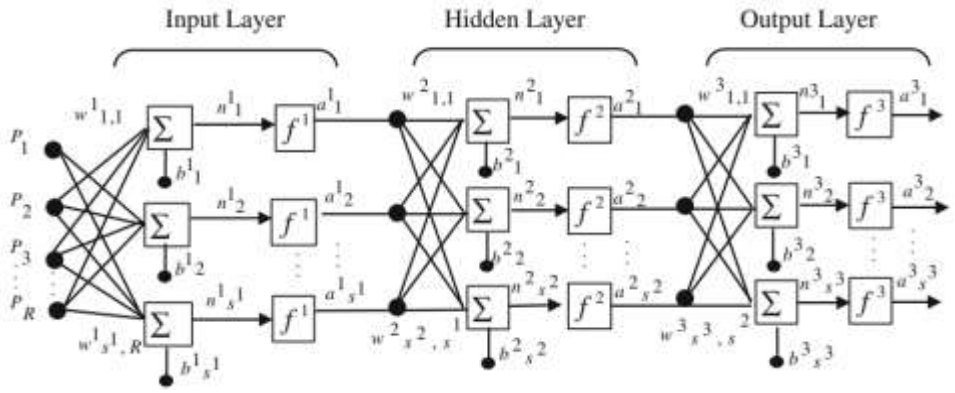

Fig. 1 Schematic view of a typical MLP (Rezaeianzadeh et al. 2014)

The MLP may be expressed mathematically as:

$$y = f\left(\sum_{i=1}^{n} w_i p_i + b\right) \tag{1}$$

where w_i represents the weight vector; p_i is the input vector ($i = 1, \ldots, n$); b is the bias, f is the transfer function; and y is the output. The current work employs an MLP structure in which one hidden layer with a configurable number of neurons is examined. The hidden layers' optimal value of neurons should be computed.

Yonaba et al. (2010) showed that using a tangent sigmoid as a transfer function was more efficient for streamflow forecasting in previous studies. As a result, in this study, we employed the tangent sigmoid transfer function, which is defined as follows for any variable of s:

$$f(s) = \frac{2}{\left(1 + e^{-2s}\right)} - 1 \tag{2}$$

It is worth noting that the functions used in neurons in the middle and output layers are of the tangent sigmoid (Eq. (2)) and linear ($f(s) = s$) types, respectively.

MLPs are often trained using the back error propagation approach, which involves iteratively changing the network's interconnecting weights to minimize a preset error, such as mean square error (MSE). The training algorithm's goal is to reduce the global error E, which is defined as:

$$E = \frac{1}{M}\sum_{m=1}^{M} E_m \tag{3}$$

where M is the total number of training patterns, and E_m defines the error for training pattern M. E_m is derived as:

$$E_m = \frac{1}{2}\sum_{k=1}^{n}\left(o_k - t_k\right)^2 \tag{4}$$

where n is the total number of output nodes, o_k is the network output at the kth output node and t_k is the target output at the kth output node. The weights and biases are changed in the training procedure to lower the global error. According to previous research (Kişi 2007), the Levenberg–Marquardt algorithm delivers satisfactory accuracy for most ANN applications; hence it was chosen for this study. Without constructing the Hessian matrix, this algorithm is aimed to approach second-order training speed (Kişi 2007).

In this study, a neural network model with one hidden layer has been proposed. The number of neurons has been chosen based on the number of inputs.

2.2. Adaptive Neuro-Fuzzy Inference System (ANFIS)

ANN is a robust approach for simulating a variety of real-world issues, but it is not without flaws. A fuzzy system like ANFIS may be a preferable alternative if the input data is ambiguous or susceptible to much uncertainty (Moghaddamnia et al., 2009). Jang (1993) was the first to propose the ANFIS approach, and he successfully applied its concepts to a variety of issues.

ANFIS combines fuzzy systems with the learning ability of neural networks (Ebtehaj and Bonakdari 2014). The ANFIS models are classed as Mamdani, Sugeno, and Tsumoto. However, Sugeno's approach is the most often utilized (Takagi and Sugeno 1985). Membership functions are used in fuzzy logic models to turn input data into fuzzy values ranging from 0 to 1 (Çekmiş et al., 2014). An ANFIS model consists of both nodes and rules. The rules allow one to represent the relationships between a predictor (input) and the predictand (output) by using nodes as membership functions (MFs).

ANFIS requires the application of feature extraction rules to input target data stored in a fuzzy-based rule system (i.e., the IF-THEN rule). According to their antecedents (If part) and consequences (Then part), the rules are defined. A rule in a Sugeno system is a linear combination of crisp weighted inputs. Two inputs (x and y) and one output (f) are presented in Eqs. (5) and (6):

Rule 1: IF x is P_1 and y is Q_1; then $f_1 = p_1x + q_1y + r_1$ (5)

Rule 2: IF x is P_2 and y is Q_2; then $f_2 = p_2x + q_2y + r_2$ (6)

P_i and Q_i are the MFs of the inputs x and y; f_i represents the weighted mean of the single rule outputs, and p_i, q_i, and r_i are the parameters of the output function (i = 1, 2). Fig. 2 depicts the ANFIS model's architecture, with two inputs (x and y) and one output (f).

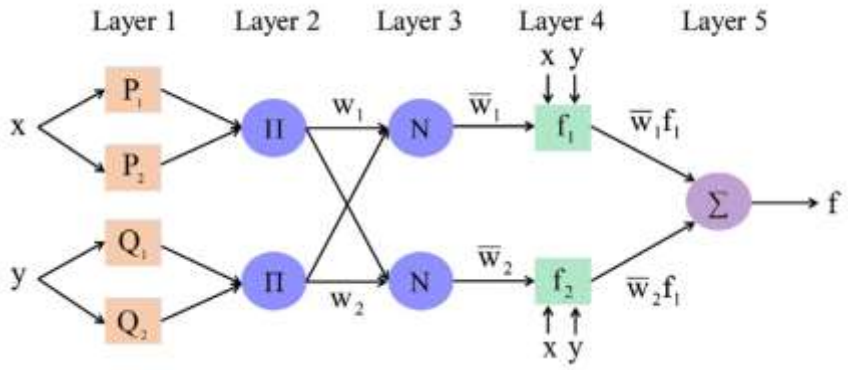

Fig. 2 Architecture of an ANFIS with 2 inputs and 5 layers. Layer 1 (Input Fuzzy Rules); Layer 2 (Input MF); Layer 3 (Fuzzy Neurons); Layer 4 (Output MF); Layer 5 (Summation and Weights); w = weights; x, y = inputs; P_1 and P_2 = fuzzy rules; Q_1, Q_2 = fuzzy rules; x, y (with arrows = target-input in training phase. f = output (Yaseen et al. 2017)

The following are the functions of each layer in the ANFIS:

Layer 1: variables of the input layer are converted to a fuzzy membership function (MF) as the output of a node. Generalized bell functions generally describe the MFs for P_i and Q_i as:

$$\mu_{Ni} = \frac{1}{1+\left[\left(x - c_i\right)/a_i\right]^{2b_i}} \tag{7}$$

where $\{a_i, b_i, c_i\}$ is the parameter set.

Layer 2: Every node calculates the degree of rule activation. In this layer, the membership functions are multiplied:

$$w_i = \mu_{Ai}\left(x\right)\mu_{Bi}\left(y\right)\left(i = 1,2\right) \tag{8}$$

where μ_{Ai} is the membership degree of x in the A_i set, μ_{Bi} is the membership degree of y in the B_i set. Each node output indicates the firing strength of a rule.

Layer 3: The ith node calculates the firing strength (activity degree) ratio of the i rule to the sum of activation degrees of all rules. $\bar{w}_i$ is the normalized membership degree of i rule.

$$\bar{w_i} = \frac{w_i}{\left(w_1 + w_2\right)}\ \left(i = 1,2\right) \tag{9}$$

Layer 4: The output of any node is computed:

$$\bar{w_i} f_i = \bar{w_i}\left(p_i x + q_i y + r_i\right) \tag{10}$$

where p, q and r are changeable consequent parameters.

Layer 5: The node of the fifth layer produces the final network output as a summation of all incoming signals.

$$\sum \bar{w_i} f_i = \frac{\sum w_i f_i}{\sum w_i} \tag{11}$$

Fast training and adjusting the network parameters are done in two steps. In the first step, premise parameters are defined, and then information is propagated to layer 4 in the network. The significant parameters are identified at layer 4 using a least-squares estimator. The selected parameters are fixed in the next step or the backward pass while

propagating the error. The gradient descent approach is then used to modify the premise parameters. The only user-specific information necessary is the number of membership functions, aside from training patterns.

The Fuzzy C-means (FCM) clustering algorithm gives some information for building the fuzzy inference system, such as initial rules and fuzzy structure (Cobaner 2011). Then, based on this clustering technique, fuzzy inference systems may give a sophisticated model for establishing links between inputs and output space. The FCM-ANFIS model that was used in this investigation is detailed below.

Bezdek (1973) was the first to introduce the FCM clustering method. The cluster centres in data point spaces are defined in the FCM method by having a set number of clusters in hand, and then each data point is allocated to a cluster by a predetermined membership degree. The method employs an iterative algorithm to minimize the function:

$$Min\ J_{FCM} = \sum_{c=1}^{C}\sum_{i=1}^{N} w_{ic}^{\ p} \left\| x_i - v_c \right\|^2 \ s.t.\ \sum_{c=1}^{C} w_{ic} = 1,\ i = 1, 2, ..., N \tag{12}$$

in which p ($1 < p$) is known as fuzzifier portion, and N is the number of data points; C, the number of clusters; w_{ic}, the number of belongings of the ith data point to the cth cluster; v, is the center of the cluster and x is the data point. The following formula is used to calculate the amount of w_{ic}:

$$w_{ic} = \frac{1}{\sum_{L=1}^{C}\left(\left(x_i - v_c\right)^2 / \left(x_i - v_L\right)^2\right)^{1/(p-1)}} \ for\ i = 1, 2, ..., N \ \ and \ \ c = 1, 2, ..., C \tag{13}$$

Centres are calculated at the start of the centre vectors by:

$$v_c = \frac{\sum_{j=1}^{N} w_{jc}^{\ p} x_j}{\sum_{j=1}^{N} w_{jc}^{\ p}} \tag{14}$$

The FCM process continues until it reaches a convergence condition.

The ANFIS-FCM model is used in this work primarily because of its capacity to learn, construct, expense, and categorize input-target data. In the streamflow forecasting problem that is generally complicated due to the chaotic nature of the data itself, an ANFIS-FCM model can intelligently extract information and convert it to fuzzy systems. Still, a more considerable time expended in training the model is necessary for accurate estimation.

2.3. Convolutional neural networks (CNN)

Convolutional neural networks (CNNs) are neural networks that employ convolution as a replacement for common matrix multiplication in at least one layer. By enabling multivariate inputs and multivariate outputs, CNNs have the advantage of the multi-layer perceptron (MLP) in time series prediction. Without needing the model to learn directly from lagged observations, it may also learn arbitrary yet complicated functional connections. As a result, the CNN model may learn the most relevant representation to the prediction issue from a wide series of inputs (Fawaz et al., 2019). CNN employs convolution layers to encode information instead of conventional neural networks, which use densely connected computational units (neurons) in hidden layers. Convolution, pooling, and fully-connected layers are the three primary layers of a CNN model. Depending on the aim, the number of layers or layer types for a basic CNN model can be layered dynamically. The convolutional layer uses several convolution kernels to learn feature

representations of inputs. The idea of using CNNs for time series prediction comes from the filters that function for particular recurring patterns in sequences and are utilized for future value prediction. CNNs perform better in noise sequences because of their hierarchical nature. The noise can be eliminated from each successive layer to maintain the important patterns (Miau and Hung 2020). One of the key reasons to utilize the convolutional layer instead of the fully connected layer is that the fully connected layer would result in many parameters needing many resources. Before being fed into the next layer, the outputs of a convolutional layer are processed via a non-linear activation function. When the input is a 1-D signal, a convolutional layer h is built using a series of $k = 1,..., N_k$ small filters (L×1) as follows:

$$h_i^k = f\left(\sum_{l=1}^{L} w_l^k X_{i+l} + b^k\right) \tag{15}$$

where a common choice for $f(\cdot)$ is the rectified linear unit (ReLU). These layers can be combined with other architectures for more complex tasks, such as LSTM (Ghimire et al., 2021) or GRU (Miau and Hung, 2020).

2.4. Long short-term memory (LSTM)

Hochreiter and Schmidhuber (1997) introduced the LSTM network architecture, which is considered a development of the RNN. A typical LSTM network comprises a series of memory blocks organized in a pattern known as the LSTM cells. These memory blocks play the same role as neurons in the hidden layers. The memory block diagram in the LSTM network architecture is shown in Fig. 3.

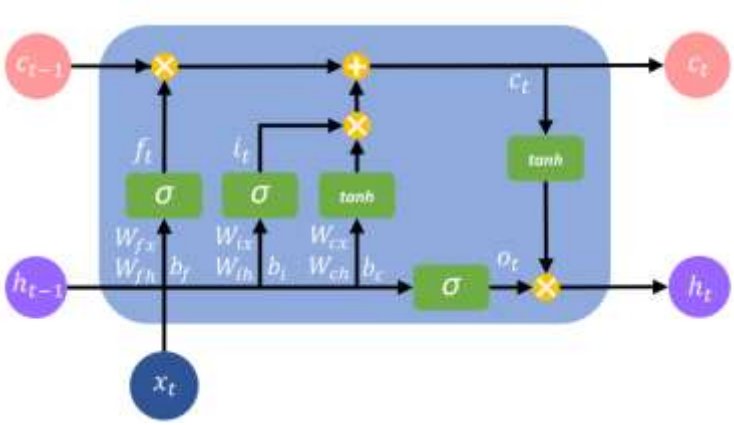

Fig. 3 The typical structure of an LSTM cell (Song et al. 2020)

In the LSTM network, each memory cell is interconnected by two vital components; the hidden state (h_t) is known as short-term memory, and the cell state (c_t) is known as long-term memory. The LSTM network can store essential data for a long time (Le et al., 2019). The following equations reflect the cell state and hidden state values for an LSTM cell (at time step t):

$$f_t = \sigma\left(W_{fx} x_t + W_{fh} h_{t-1} + b_f\right) \tag{16}$$

$$i_t = \sigma\left(W_{ix} x_t + W_{ih} h_{t-1} + b_i\right) \tag{17}$$

$$o_t = \sigma\left(W_{ox} x_t + W_{oh} h_{t-1} + b_o\right) \tag{18}$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes \tanh\left(W_{cx} x_t + W_{ch} h_{t-1} + b_c\right) \tag{19}$$

$$h_t = o_t \otimes \tanh\left(c_t\right) \tag{20}$$

where W stands for weight parameters, b for bias, σ for sigmoid function, and $\otimes$ the element-wise multiplication. Each subscript of W and b indicates the weight and bias of a different gate. For example, W_{fx} is the weight of input x_t at the gate f_t. c is the cell state, h the hidden state, x the input, and tanh the hyperbolic tangent function. As indicated in Eq. (16), the first gate is the forget gate f_t, which controls the information from the previous cell state. It determines how much data should be retained or passed on to the next stage. The input gate is the second gate, determining how much new data should be utilized. Meanwhile, the current cell state can be updated by combining the output from the forget gate with the input gate, as stated in Eq. (19). Finally, the latest cell state and input data are utilized to update the most current hidden state and act as the LSTM output (Eq. (20)). Backward propagation was used to update the parameters of the LSTM.

2.5. Gated Recurrent Unit (GRU)

Cho et al. (2014) introduced GRU as a simple variant of LSTM due to their similarity in design and equally excellent performance. Instead of three gating layers, like in the LSTM design, it only contains two gating layers for processing information: the update gate (z_t) and the reset gate (r_t). The architecture of a GRU memory cell is shown in Fig. 4.

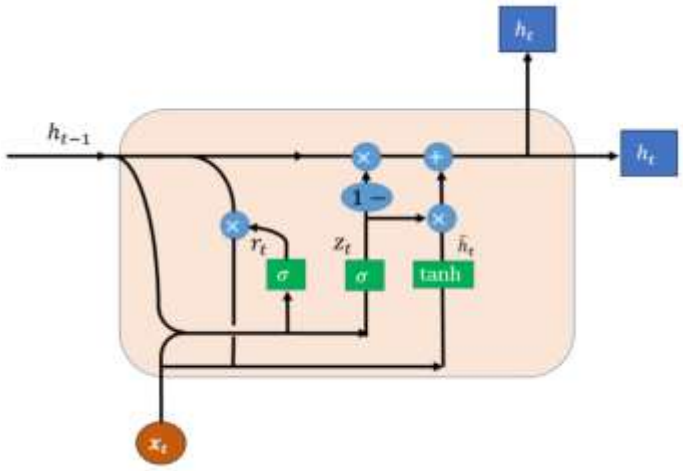

Fig. 4 The typical structure of a GRU cell (Huang et al. 2019)

The output of the memory cell (h_t-hidden layer) at the *t-th* time step is used for both decision-making and input data for succeeding cells in the GRU architecture. The following equations are used to calculate this value:

$$z_t = \sigma\left(U_z x_t + W_z h_{t-1}\right) \tag{21}$$

$$r_t = \sigma\left(U_r x_t + W_r h_{t-1}\right) \tag{22}$$

$$\tilde{h}_t = \tanh\left(U_h x_t + r_t \otimes W_h h_{t-1}\right) \tag{23}$$

$$h_t = \left(1 - z_t\right) \otimes \tilde{h}_t + z_t \otimes h_{t-1} \tag{24}$$

where $\tilde{h}_t$ is the candidate of the hidden state at time step t; U_j and W_j are weight matrices; and $\otimes$ denotes the element-wise multiplication.

The CNN-GRU-LSTM architecture is employed in this study. The first half is CNN for feature extraction, and the second half is GRU and LSTM prediction for feature analysis and next-point streamflow prediction. The CNN part of the proposed CNN-GRU-LSTM architecture has five 1D convolution layers, one pooling layer (MaxPooling1D), and one fully-connected layer. Also, one layer and two layers are used for GRU and LSTM, respectively.

2.6. Bidirectional LSTM (BiLSTM)

Data processing is performed in one direction (forward) for conventional LSTM models, and crucial information is stored (or forecasted) based on previous data. Because future input information cannot be obtained from the current state, this is one of the weaknesses of the conventional LSTM network (Le et al., 2021). BiLSTM is a more advanced version of the LSTM architecture that simultaneously analyses data forward and backward. As a result, BiLSTM is intended to capture the input sequence's future and past information better than conventional LSTM (Graves et al., 2013). An example framework for a BiLSTM model is shown in Fig. 5.

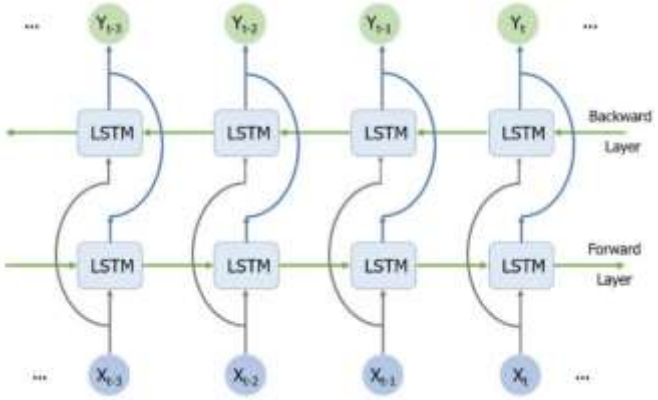

Fig. 5 The structure of a BiLSTM neural network (Le et al. 2021)

The number of hidden layers used for the BiLSTM neural network is 2. Also, in this study, all models were developed and objectively evaluated using Matlab. Training parameters (hyperparameters) such as the optimization technique, loss function, and learning rate are critical components of DL models and the internal parameters of the models. The Adam optimization algorithm with a default-learning rate of 0.001 is applied throughout this investigation. Furthermore, the suggested number of epochs for DL models to collect the crucial information during the training phase is 200.

2.7. Study area and model development

This study used lagged daily data combinations of streamflow from 10/1/1921 to 10/1/2021 to create a prediction model based on ANN, ANFIS, BiLSTM, and CNN-GRU-LSTM. Figure 6 shows a map of the current research site, the Colorado River (at Lees Ferry reach). Lees Ferry Gage is upstream from the gravel bar at the mouth of the Paria River on the left bank. Every discharge measurement has been made since December 1966 from the Modern

Cableway, located around 15 meters upstream from the Lees Ferry Gage. Up to this date, discharge measurements were taken from cableways farther upstream. The daily streamflow data were acquired from the USGS (https://waterdata.usgs.gov/nwis/nwis).

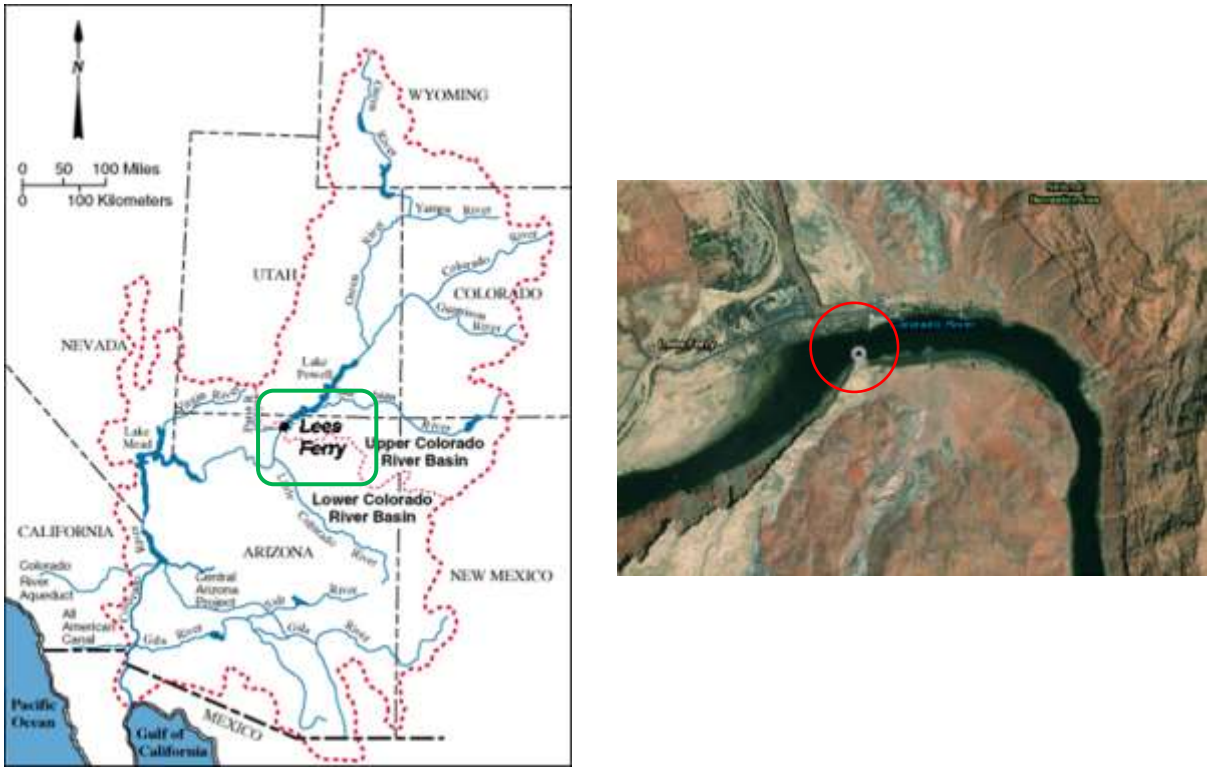

Fig. 6 Location of Lees Ferry, Colorado River study site in the USA, where experiments are carried out to validate the ANN and ANFIS model to predict daily streamflow (Q, m^3/s) (https://waterdata.usgs.gov/nwis/nwis)

The first stage in data preparation is establishing the streamflow (Q) time-series stationarity. To do this, the Dicky–Fuller (DF) and KPSS tests were used in this study. The null hypothesis that the Q time series is non-stationary will be rejected if the DF test is used. Applying the KPSS test implies that the null hypothesis, which suggests that the Q time series is trend stationery, will be valid. The correlation analysis phase follows, intending to determine the model's input combination order. The desired models use correlated lags computed using the correlation statistics approach (i.e., autocorrelation and partial autocorrelation functions) to detect appropriate input attributes for the target attribute (Fig. 7).

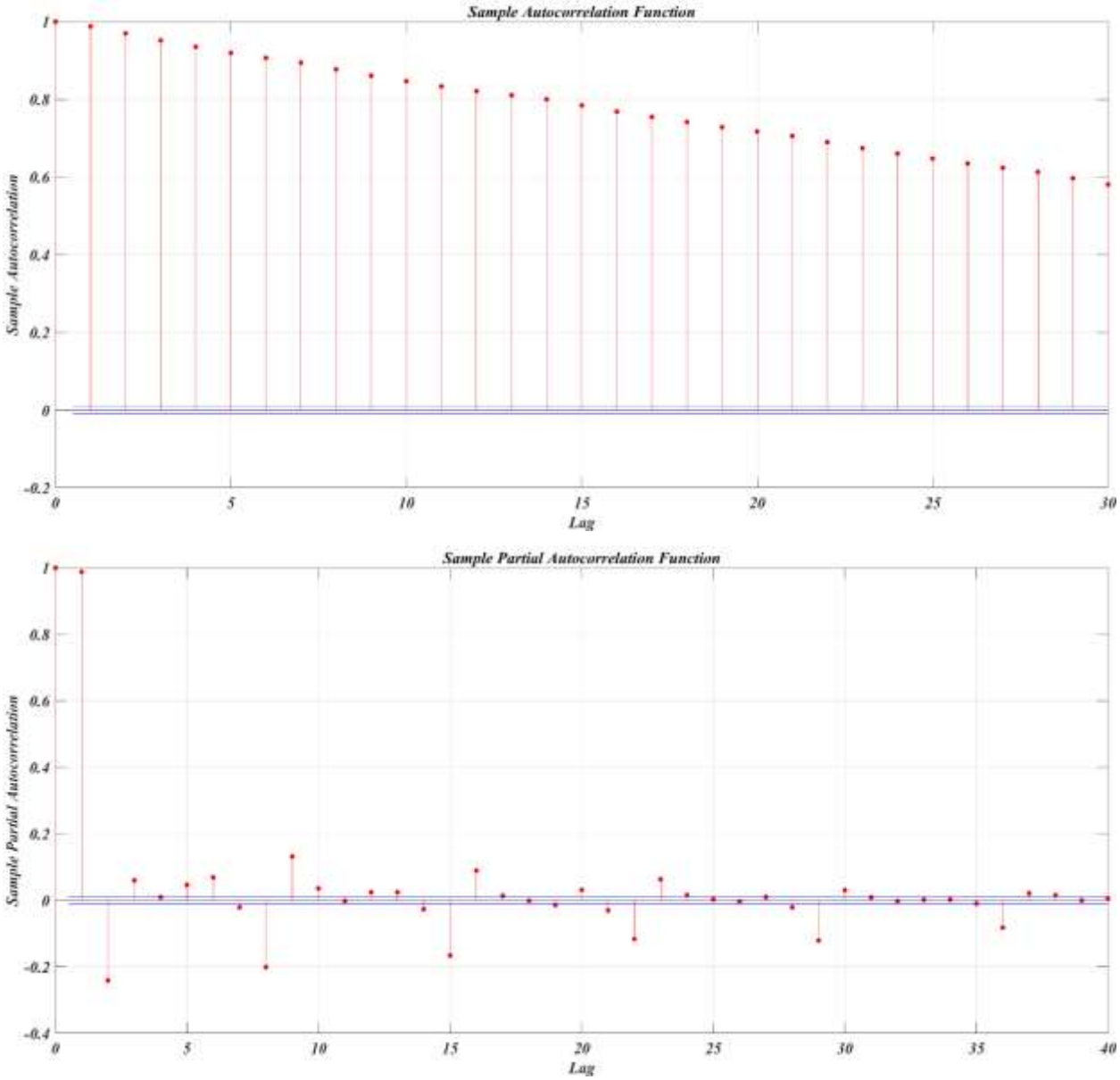

Fig. 7 The statistical correlation process function of the investigated daily scale river flow

According to Fig. 7, the number of inputs in the training period for input combinations varies from 1 to 5, and there are five different input combinations reported in this study (Table 1). All forecasting models have the same output.

Table 1 Different input combinations of the models

Input combination	Output	Scenario	Model name			
Q_t	Q_{t+1}	1	ANN1	ANFIS1	BiLSTM1	CNN-GRU-LSTM1
Q_t, Q_{t-1}		2	ANN2	ANFIS2	BiLSTM2	CNN-GRU-LSTM2
Q_t, Q_{t-1}, Q_{t-2}		3	ANN3	ANFIS3	BiLSTM3	CNN-GRU-LSTM3
Q_t, Q_{t-1}, Q_{t-2}, Q_{t-3}, Q_{t-4}, Q_{t-5}		4	ANN4	ANFIS4	BiLSTM4	CNN-GRU-LSTM4
Q_t, Q_{t-1}, Q_{t-2}, Q_{t-3}, Q_{t-4}, Q_{t-5}, Q_{t-6}, Q_{t-7}, Q_{t-14}, Q_{t-21}		5	ANN5	ANFIS5	BiLSTM5	CNN-GRU-LSTM5

The modelled dataset was scaled between 0 and 1 using Eq. (25) to avoid high values of variables in the dataset for more accessible simulation. Then converted to its original scale using Eq. (26), where Q, Q_{min}, and Q_{max} represent the input data value and its overall minimum and maximum values, respectively.

$$Q_n = \frac{Q_{actual} - Q_{\min}}{Q_{\max} - Q_{\min}} \tag{25}$$

$$Q_{actual} = Q_n \left(Q_{\max} - Q_{\min}\right) + Q_{\min} \tag{26}$$

After normalization, the data is randomly split into two portions for training and testing. The training datasets spanned 60% of the time series (from October 1, 1921, to October 1, 1981), while the testing datasets covered the remaining 40% (from October 2, 1981, to October 1, 2021). Figure 8 illustrates a graph of the measured daily river discharge through time, including the training and testing periods.

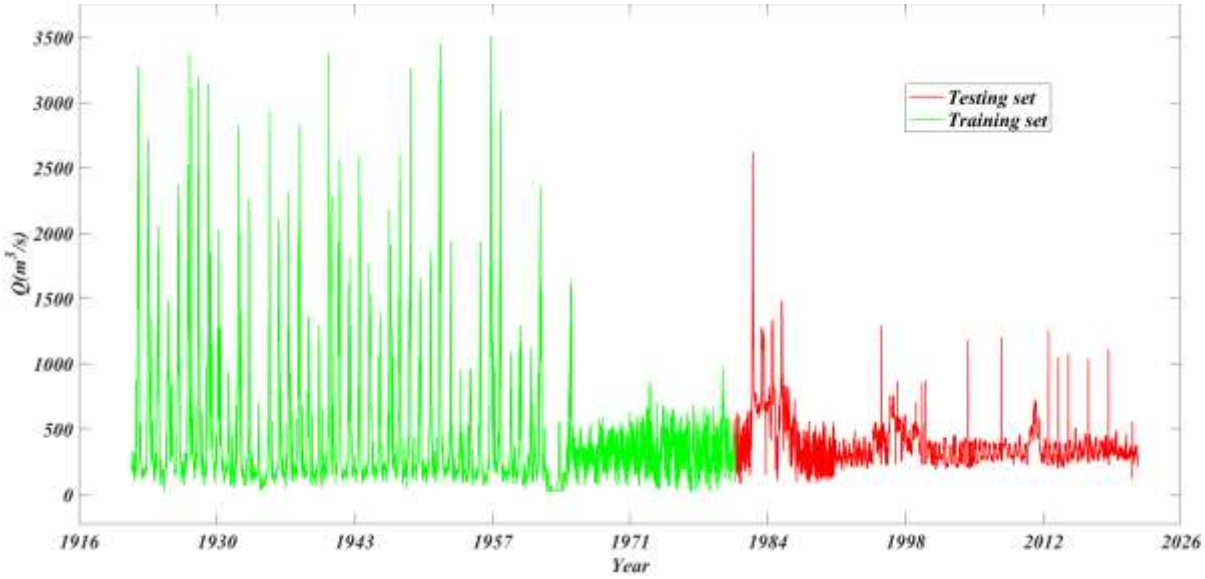

Fig. 8 Time series plot for the observed discharge data with a period of 10/01/1921 to 10/01/2021 in Colorado River

2.8. Performance metrics

Diagnostic plots and statistical score measures were used to assess the models' performance in the testing phase. These metrics are described as:

1. Relative Root Mean Square Error (MAE, m^3/s):

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|Q_p - Q_m\right| \tag{27}$$

2. Normalized Root Mean Square Error (NRMSE):

$$NRMSE = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(Q_p - Q_m\right)^2}}{\bar{Q}_m} \tag{28}$$

3. Nash–Sutcliffe coefficient (E_{NS}):

$$E_{NS} = 1 - \frac{\sum_{i=1}^{N}\left(Q_m - Q_p\right)^2}{\sum_{i=1}^{N}\left(Q_m - \bar{Q}_m\right)^2} \tag{29}$$

4. Correlation Coefficient (r):

$$r = \frac{\sum_{i=1}^{N}\left(Q_m - \bar{Q}_m\right)\left(Q_p - \bar{Q}_p\right)}{\sqrt{\sum_{i=1}^{N}\left(Q_m - \bar{Q}_m\right)^2}\sqrt{\sum_{i=1}^{N}\left(Q_p - \bar{Q}_p\right)^2}} \tag{30}$$

3. Results and discussion

This section only presents the quantitative comparison between the observed and forecasted value series corresponding to the testing phase. This study focuses on the performance of ANN, ANFIS, BiLSTM, and CNN-GRU-LSTM models for streamflow prediction. According to Table 2, the Colorado River is analyzed in five possible scenarios based on input combinations. The efficiency of models was obtained for each input combination in terms of MAE, NRMSE, E_{NS}, and r indices during the testing period, as shown in Table 2. It should be noted that in Table 2, for the ANN models, the mean value of 100 runs was reported. As shown in Table 2, in most cases, increasing the number of inputs will improve the performance of all studied models. The highest and lowest value of the E_{NS} and r is related to the ANFIS and ANN model, respectively. The BiLSTM and CNN-GRU-LSTM models perform nearly identically in different combinations. On the other hand, the output of the CNN-GRU-LSTM model is superior to that of BiLSTM. However, according to all performance criteria, the ANFIS model performs slightly better than others.

Table 2 Performance of ANN, ANFIS, BiLSTM, and CNN-GRU-LSTM Model

Model name	Testing			
	MAE	NRMSE	E_{NS}	r
ANN1	71.71	0.221	0.750	0.928
ANN2	57.65	0.208	0.724	0.907
ANN3	46.95	0.181	0.822	0.933
ANN4	38.68	0.149	0.885	0.957
ANN5	**35.29**	**0.139**	**0.903**	**0.957**
ANFIS1	26.92	0.125	0.926	0.963
ANFIS2	27.28	0.124	0.928	0.964
ANFIS3	26.87	0.122	0.929	0.964
ANFIS4	26.04	0.120	0.931	0.965
ANFIS5	**24.66**	**0.116**	**0.936**	**0.968**
BiLSTM1	57.50	0.176	0.853	0.962
BiLSTM2	28.20	0.126	0.924	0.962
BiLSTM3	30.59	0.128	0.922	0.962
BiLSTM4	27.95	0.126	0.925	0.962
BiLSTM5	**26.42**	**0.123**	**0.928**	**0.964**
CNN-GRU-LSTM1	26.47	0.124	0.927	0.963
CNN-GRU-LSTM2	26.88	0.125	0.925	0.963
CNN-GRU-LSTM3	26.88	0.126	0.924	0.962
CNN-GRU-LSTM4	26.70	0.124	0.927	0.963
CNN-GRU-LSTM5	**26.09**	**0.123**	**0.928**	**0.964**

Figure 9 shows a scatter plot of ANN, ANFIS, BiLSTM, and CNN-GRU-LSTM models for the Colorado River in the testing phase, with similar observed data. When used to forecast low and medium flows, the ANFIS model outperforms others, demonstrating its superiority. In addition, the box plot (Fig. 10) also visualizes additional statistical comparisons to evaluate the performance of examined models compared to observed data. As shown in Fig. 10, the BiLSTM and CNN-GRU-LSTM models performed better and more consistently than the others in forecasting peak discharge. However, the average of all models is in the same range.

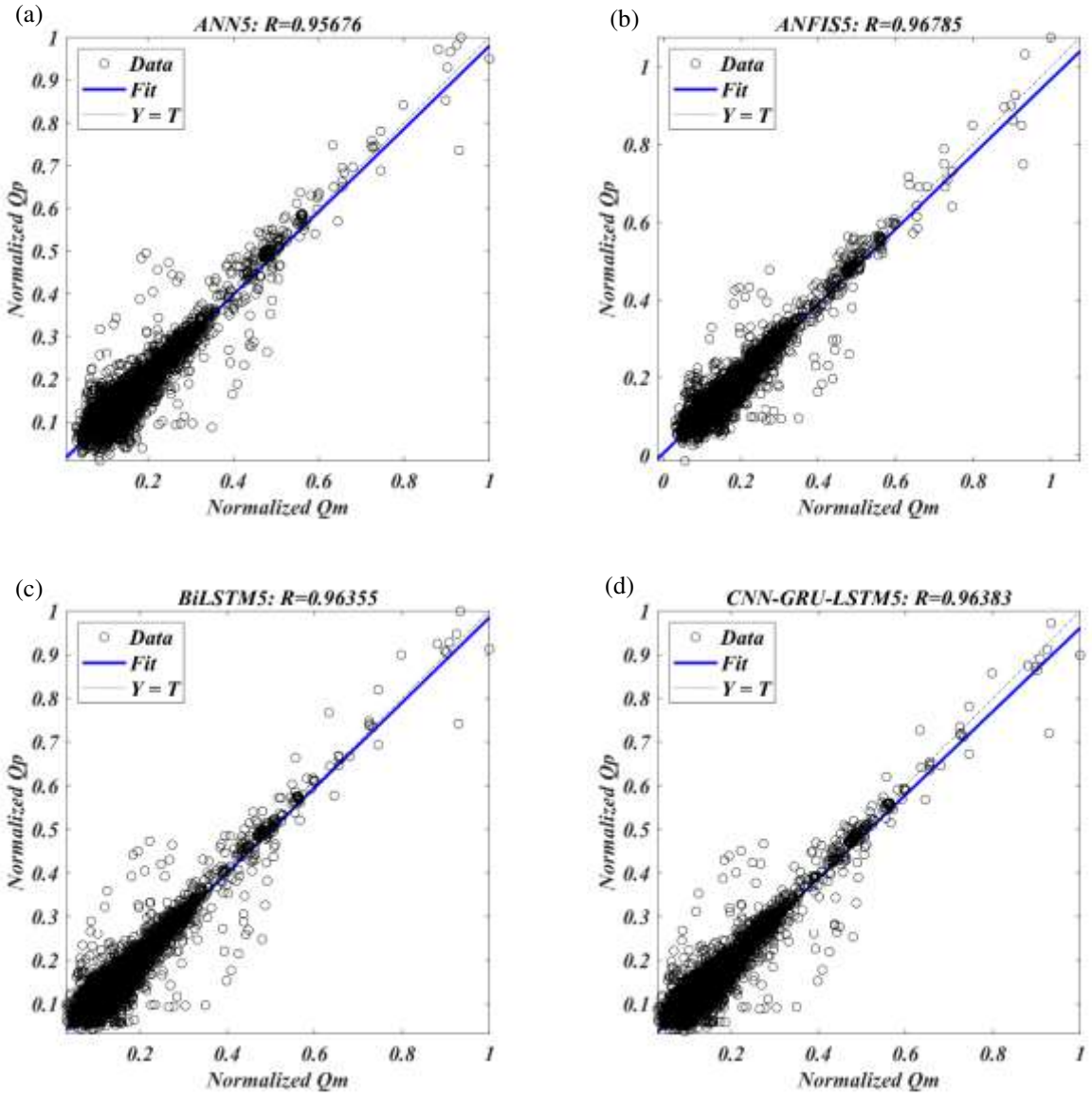

Fig. 9 Scatter Plots of Observed and Predicted Streamflow (a) ANN, (b) ANFIS, (c) BiLSTM, and (d) CNN-GRU-LSTM Model

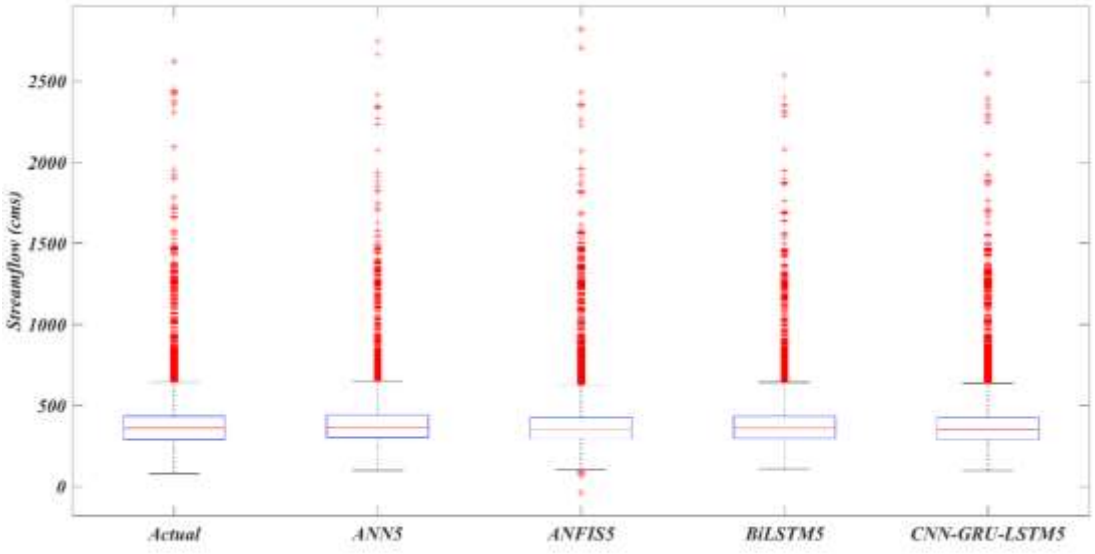

Fig. 10 Box plot of daily streamflow forecasting in the testing step

Figures 11-14 show the actual vs expected flow discharge for the ANN, ANFIS, BiLSTM, and CNN-GRU-LSTM models in the Colorado River, respectively. Also, the errors and histogram of prediction errors are presented in different time indexes. The figure reveals the outperformance of the ANFIS model against the other models. The comparison of histograms reveals the outperformance of the ANFIS model against the other models. In a more quantitative term, the ANFIS model shows the highest frequency of error (7879) in the bin ($-11.2 < \text{error} \leq 15$) followed by the CNN-GRU-LSTM (7283), BiLSTM (7028), and finally ANN (5404).

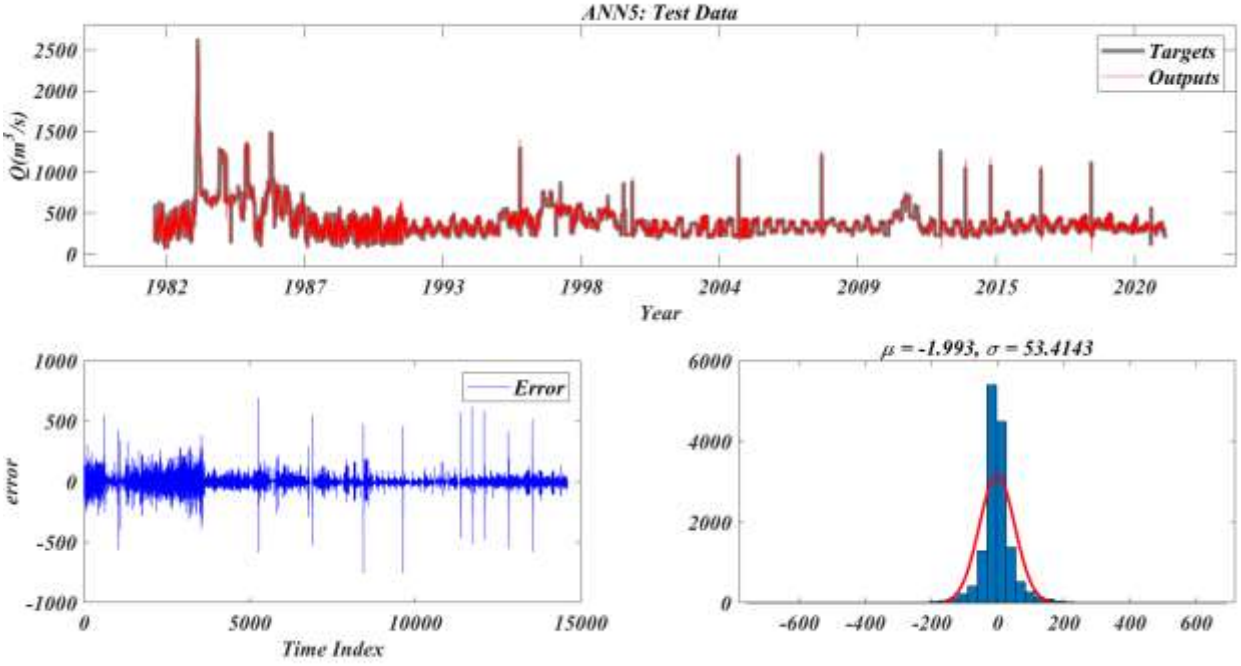

Fig. 11 Comparison of measured and predicted streamflow (Q/m^3) using ANN

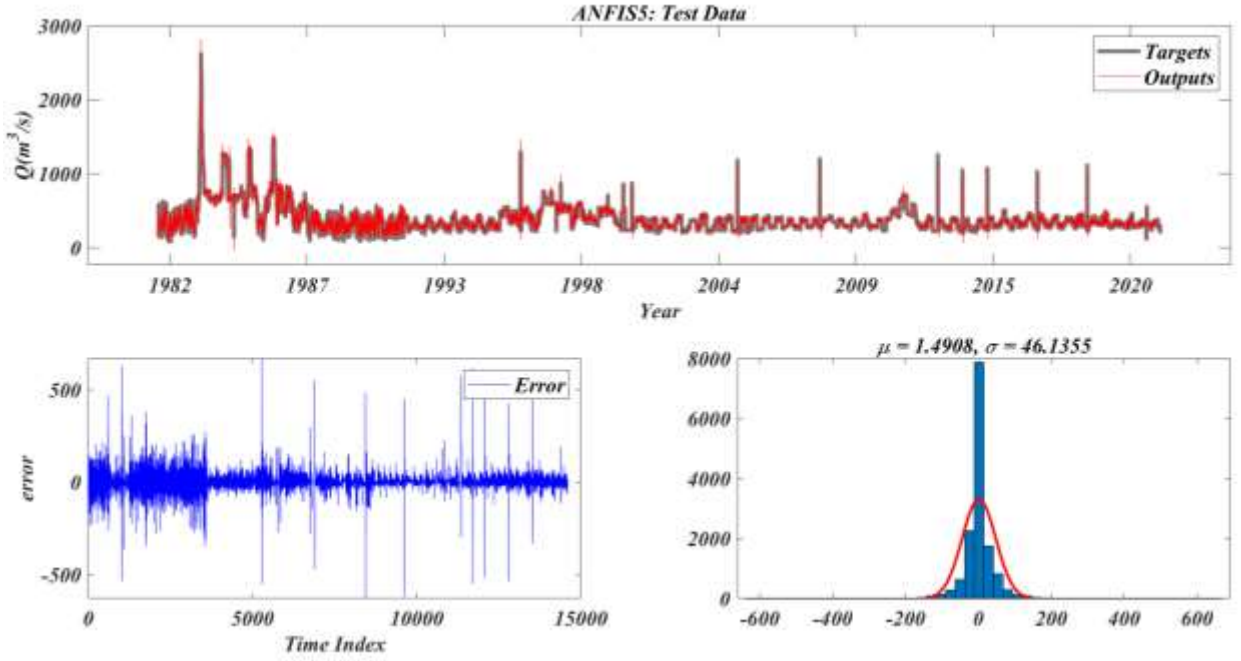

Fig. 12 Comparison of measured and predicted streamflow (Q/m^3) using ANFIS

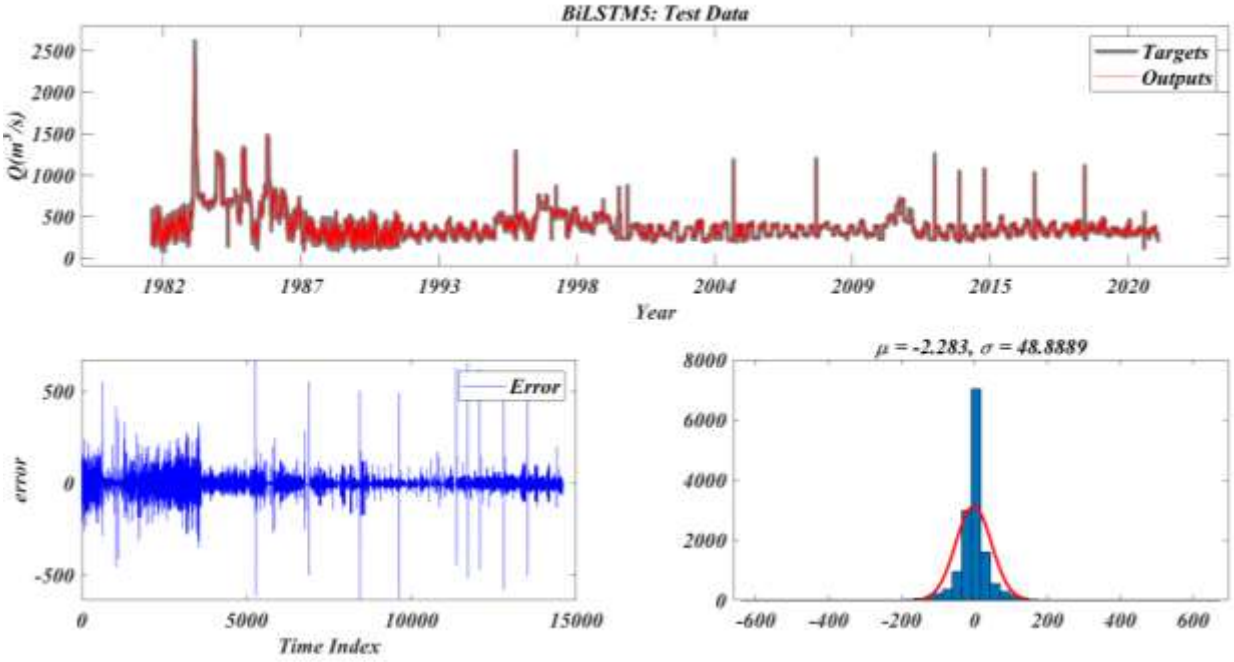

Fig. 13 Comparison of measured and predicted streamflow (Q/m^3) using BiLSTM

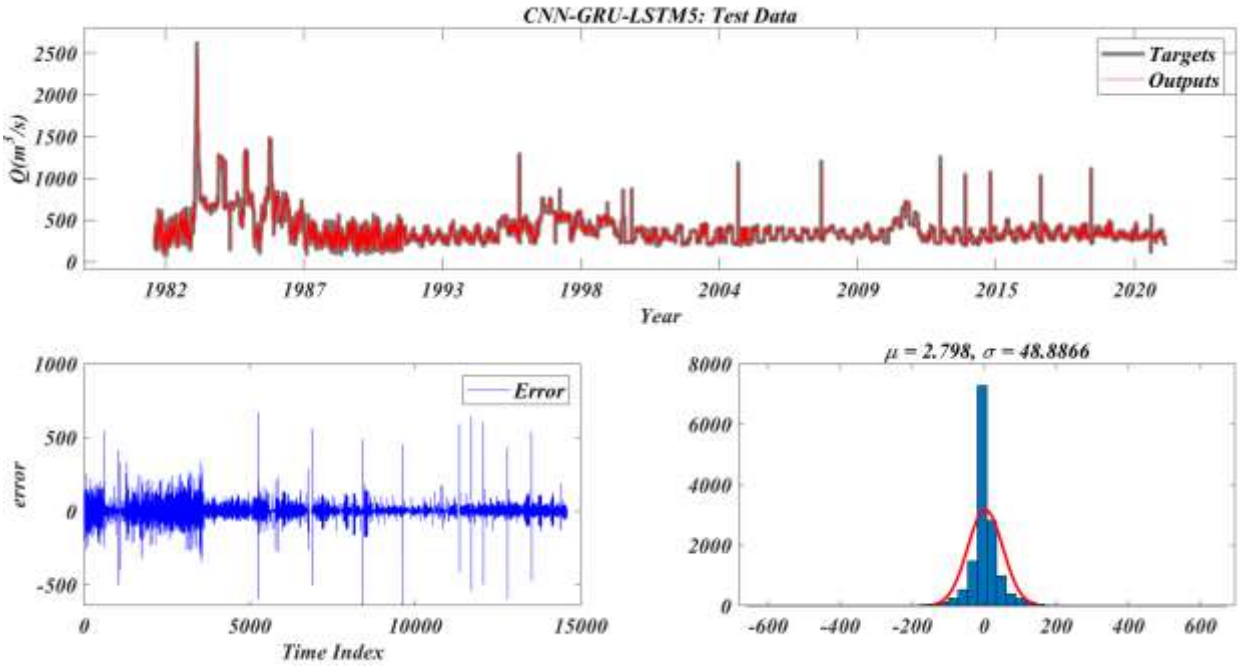

Fig. 14 Comparison of measured and predicted streamflow (Q/m^3) using CNN-GRU-LSTM

Table 2 shows the estimated peak discharge for several models on June 29, 1983. For the BiLSTM5 and CNN-GRU-LSTM5 models, these relative errors are 3.28% and 2.78%, respectively, with absolute errors of about 86 and 73 m^3/s. This amount is an acceptable tolerance in the event of a flood.

Table 2 Results of flood peak forecasting

Case	Projected peak	Actual peak	Absolute error (m^3/s)	Relative error (%)
ANN5	2749	2622	127	4.84
ANFIS5	2821		199	7.59
BiLSTM5	2536		86	3.28
CNN-GRU-LSTM5	2549		73	2.78

In general, according to the relative error of Table 2, there are no significant differences between the studied models. The complexity of the CNN-GRU-LSTM and BiLSTM models is not accompanied by performance improvement because the comparison results illustrate that their respective performance is not higher than the two standard models—ANN and ANFIS. Therefore, it is proposed to apply simpler models like ANN and ANFIS.

4. Conclusions

In the present study, streamflow magnitudes were forecasted using long-period flow data from Lees Ferry, Colorado River hydrometric stations in the United States. Three essential outcomes were found, which are briefly stated as follows:

Firstly, the ANFIS model outperforms the other models in accuracy and reliability for all input combinations.

Secondly, the model with ten inputs produced the best and most accurate results, with high E_{NS}, r, and low NRMSE and MAE, among the possible input combinations. When using artificial intelligence models, choosing the proper input parameters is crucial.

Thirdly, the performance of the BiLSTM and CNN-GRU-LSTM for forecasting peak discharge in the testing phase is better than ANFIS and ANN models. Additionally, in general, CNN-GRU-LSTM and BiLSTM models do not exhibit performance improvement compared to ANN and ANFIS models due to their complexity.

Acknowledgements

This study did not receive any external funding.

Funding

The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Competing Interests

The authors have no relevant financial or non-financial interests to disclose.

Author Contributions

All authors read and approved the final manuscript.

Reference

Bezdek JC (1973) Cluster validity with fuzzy sets.

Çekmiş A, Hacıhasanoğlu I, Ostwald MJ (2014) A computational model for accommodating spatial uncertainty: Predicting inhabitation patterns in open-planned spaces. Build Environ 73:115–126.

Cho K, Van Merriënboer B, Gulcehre C, et al. (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv Prepr arXiv14061078.

Cobaner M (2011) Evapotranspiration estimation by two different neuro-fuzzy inference systems. J Hydrol 398:292–302.

Dalkiliç HY, Hashimi SA (2020) Prediction of daily streamflow using artificial neural networks (ANNs), wavelet neural networks (WNNs), and adaptive neuro-fuzzy inference system (ANFIS) models. Water Sci Technol Water Supply 20:1396–1408. doi: 10.2166/ws.2020.062.

Demirel MC, Venancio A, Kahya E (2009) Flow forecast by SWAT model and ANN in Pracana basin, Portugal. Adv Eng Softw 40:467–473. doi: 10.1016/j.advengsoft.2008.08.002.

Ebtehaj I, Bonakdari H (2014) Performance evaluation of adaptive neural fuzzy inference system for sediment transport in sewers. Water Resour Manag 28:4765–4779.

Fawaz HI, Forestier G, Weber J, et al. (2019) Deep learning for time series classification: a review. Data Min Knowl Discov 33:917–963.

Ghimire S, Yaseen ZM, Farooque AA, et al. (2021) Streamflow prediction using an integrated methodology based on convolutional neural network and long short-term memory networks. Sci Rep 11:1–26. doi: 10.1038/s41598-021-96751-4.

Graves A, Mohamed A, Hinton G (2013) Speech recognition with deep recurrent neural networks. In: 2013 IEEE international conference on acoustics, speech and signal processing. Ieee, pp 6645–6649.

Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9:1735–1780.

Huang Z, Yang F, Xu F, et al. (2019) Convolutional gated recurrent unit–recurrent neural network for state-of-charge estimation of lithium-ion batteries. IEEE Access 7:93139–93149.

Jang J-S (1993) ANFIS: adaptive-network-based fuzzy inference system. IEEE Trans Syst Man Cybern 23:665–685

Jimmy SR, Sahoo A, Samantaray S, Ghose DK (2021) Prophecy of Runoff in a River Basin Using Various Neural Networks. In: Communication Software and Networks. Springer, pp 709–718.

Kişi Ö (2007) Streamflow Forecasting Using Different Artificial Neural Network Algorithms. J Hydrol Eng 12:532–539. doi: 10.1061/(ASCE)1084-0699(2007)12:5(532).

Le XH, Ho HV, Lee G, Jung S (2019) Application of Long Short-Term Memory (LSTM) neural network for flood forecasting. Water (Switzerland) 11: doi: 10.3390/w11071387.

Le XH, Nguyen DH, Jung S, et al. (2021) Comparison of Deep Learning Techniques for River Streamflow Forecasting. IEEE Access 9:71805–71820. doi: 10.1109/ACCESS.2021.3077703.

Mehdizadeh S, Kozekalani Sales A (2018) A Comparative Study of Autoregressive, Autoregressive Moving Average, Gene Expression Programming and Bayesian Networks for Estimating Monthly Streamflow. Water Resour Manag 32:3001–3022. doi: 10.1007/s11269-018-1970-0.

Meshram SG, Meshram C, Santos CAG, et al. (2021) Streamflow Prediction Based on Artificial Intelligence Techniques. Iran J Sci Technol - Trans Civ Eng. doi: 10.1007/s40996-021-00696-7.

Miau S, Hung WH (2020) River flooding forecasting and anomaly detection based on deep learning. IEEE Access 8:198384–198402. doi: 10.1109/ACCESS.2020.3034875.

Moghaddamnia A, Gousheh MG, Piri J, et al. (2009) Evaporation estimation using artificial neural networks and adaptive neuro-fuzzy inference system techniques. Adv Water Resour 32:88–97.

Ni L, Wang D, Singh VP, et al. (2020) Streamflow and rainfall forecasting by two long short-term memory-based models. J Hydrol 583:124296.

Papacharalampous G, Tyralis H, Koutsoyiannis D (2019) Comparison of stochastic and machine learning methods for multi-step ahead forecasting of hydrological processes. Stoch Environ Res Risk Assess 33:481-514.

Peng T, Zhou J, Zhang C, Fu W (2017) Streamflow forecasting using empirical wavelet transform and artificial neural networks. Water (Switzerland) 9:1–20. doi: 10.3390/w9060406.

Rezaeianzadeh M, Tabari H, Arabi Yazdi A, et al. (2014) Flood flow forecasting using ANN, ANFIS and regression models. Neural Comput Appl 25:25–37. doi: 10.1007/s00521-013-1443-6.

Riahi-Madvar H, Dehghani M, Memarzadeh R, Gharabaghi B (2021) Short to Long-Term Forecasting of River Flows by Heuristic Optimization Algorithms Hybridized with ANFIS. Water Resour Manag 35:1149–1166. doi: 10.1007/s11269-020-02756-5.

Samanataray S, Sahoo A (2021) A Comparative Study on Prediction of Monthly Streamflow Using Hybrid ANFIS-PSO Approaches. KSCE J Civ Eng 25:4032–4043. doi: 10.1007/s12205-021-2223-y.

Sanikhani H, Kisi O (2012) River Flow Estimation and Forecasting by Using Two Different Adaptive Neuro-Fuzzy Approaches. Water Resour Manag 26:1715–1729. doi: 10.1007/s11269-012-9982-7.

Song S, Lam JCK, Han Y, Li VOK (2020) ResNet-LSTM for Real-Time PM 2.5 and PM_{10} Estimation Using Sequential Smartphone Images. IEEE Access 8:220069–220082.

Sun Y, Niu J, Sivakumar B (2019) A comparative study of models for short-term streamflow forecasting with emphasis on wavelet-based approach. Stoch Environ Res Risk Assess 33:1875–1891. doi: 10.1007/s00477-019-01734-7.

Takagi T, Sugeno M (1985) Fuzzy identification of systems and its applications to modeling and control. IEEE Trans Syst Man Cybern 116–132.

Tao H, Habib M, Aljarah I, et al (2021) An intelligent evolutionary extreme gradient boosting algorithm development for modeling scour depths under submerged weir. Inf Sci (Ny) 570:172–184. doi: 10.1016/j.ins.2021.04.063.

Yaseen ZM, Ebtehaj I, Bonakdari H, et al. (2017) Novel approach for streamflow forecasting using a hybrid ANFIS-FFA model. J Hydrol 554:263–276. doi: 10.1016/j.jhydrol.2017.09.007.

Yaseen ZM, Faris H, Al-Ansari N (2020) Hybridized Extreme Learning Machine Model with Salp Swarm Algorithm: A Novel Predictive Model for Hydrological Application. Complexity 2020: doi: 10.1155/2020/8206245.

Yonaba H, Anctil F, Fortin V (2010) Comparing Sigmoid Transfer Functions for Neural Network Multistep Ahead Streamflow Forecasting. J Hydrol Eng 15:275–283. doi: 10.1061/(ASCE)he.1943-5584.0000188.

Zealand CM, Burn DH, Simonovic SP (1999) Short term streamflow forecasting using artificial neural networks. J Hydrol 214:32–48.

Zhu S, Luo X, Yuan X, Xu Z (2020) An improved long short-term memory network for streamflow forecasting in the upper Yangtze River. Stoch Environ Res Risk Assess 34:1313–1329. doi: 10.1007/s00477-020-01766-4.