Temporal cluster-based local deep learning or signal processing-temporal convolutional transformer for daily runoff prediction?

Moosavi, Vahid; Mostafaei, Sahar; Berndtsson, Ronny

Published in:
Applied Soft Computing

DOI:
10.1016/j.asoc.2024.111425

2024

Document Version:
Peer reviewed version (aka post-print)

Applied Soft Computing Journal 155 (2024) 111425

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Temporal cluster-based local deep learning or signal processing-temporal convolutional transformer for daily runoff prediction?

Vahid Moosavi [a,*], Sahar Mostafaei [a], Ronny Berndtsson [b,c]

[a] Department of Watershed Management Engineering, Faculty of Natural Resources and Marine Sciences, Tarbiat Modares University, Iran
[b] Division of Water Resources Engineering, Faculty of Engineering, Lund University, P.O. Box 118, Lund 22100, Sweden
[c] Centre for Advanced Middle Eastern Studies, Lund University, P.O. Box 201, Lund 22100, Sweden

HIGHLIGHTS

- A novel Temporal Cluster-Based Local Deep Learning (TCLD) model was proposed.
- Non-stationarity was addressed with signal processing-DL models.
- Signal processing-DL techniques blending approaches were evaluated.
- Discrete Wavelet-Temporal Convolutional Transformer (DWT-TCT) was developed.

ARTICLE INFO

Keywords:
Blending approaches
Deep learning
Discrete wavelet transform
Signal processing
Temporal Convolutional Transformer

ABSTRACT

Water scarcity poses a major obstacle to sustainable development, and precise discharge prediction plays a vital role in enabling effective water resource management. This study investigated improved prediction techniques for nonstationary time series. The study evaluated the effect of signal processing techniques and blending approaches on the performance of deep learning models for daily discharge prediction. It also compared the performance of cluster-based local modeling with hybrid signal processing-deep learning approaches. Two robust deep learning methods, Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN), along with a powerful signal processing approach called discrete wavelet transform, were utilized for prediction of daily discharge. Three blending approaches were assessed: 1) decomposing both inputs and target, 2) decomposing only the target, and 3) decomposing only the inputs and then blending them with deep learning models. Also, a new hybrid deep learning based model namely discrete wavelet transform-Temporal Convolutional Transformer (DWT-TCT) was developed. The results showed that a single-output wavelet transform-deep learning model (3rd blending approach) outperformed multi-output models, demonstrating a relative enhancement of up to 56% for the LSTM model and 51% for the CNN model. Furthermore, temporal cluster-based local modeling displayed promising performance, resulting in an improvement of up to 18% in NRMSE compared to the wavelet transform-deep learning model, while also requiring less computational cost. The successful results of the temporal cluster-based local modeling approach provide a beneficial alternative to hybrid signal processing-deep learning models. In addition the results showed that the proposed DWT-TCT model outperformed all other models with NRMSE ranges from 6.8% to 16.2% in the study areas. The results have implications for hydrology and water resources management, as they can be used to develop more precise and effective models for predicting discharge in view of nonstationarity.

1. Introduction

Water resources management is a critical concern in many regions, with water scarcity posing significant challenges for sustainable development. One of the most important components of water resources management is accurate prediction of discharge [1]. Discharge prediction is required for various purposes, such as flood control, hydropower generation, irrigation planning, and water supply management.

* Corresponding author.
E-mail address: v.moosavi@modares.ac.ir (V. Moosavi).

https://doi.org/10.1016/j.asoc.2024.111425
Received 20 June 2023; Received in revised form 11 February 2024; Accepted 16 February 2024
Available online 27 February 2024

Accurate prediction of daily discharge is particularly important because it assists water resources managers in making informed decisions regarding water allocation and usage. Thus, various methods and models have been developed for discharge prediction, with researchers continually seeking new and more accurate approaches. Traditionally, statistical and empirical models have been used for this purpose relying on a stationary climate. However, in view of the increasing effects of climate change, these models increasingly suffer from poor accuracy due to their inability to capture complex nonlinear relationships between climatic and hydrological variables [2].

Artificial intelligence (AI) models have emerged in recent years as powerful tools for predicting discharge. Several hybrid models have been proposed for runoff prediction including LSTM-ALO, ANFIS-GBO, ELM-PSOGWO, LSSVM-IMVO, SVR-SAMOA, ANN-EMPA, and SVM-FFAPSO [3–7]. Ding, Zhang, Guo and Sun [8] proposed a hybrid EEMD-GRU-TWSVRCSSA model as an empirical mode decomposition with gated recurrent unit and time-weighted support vector regression with cat swarm optimization and Salp swarm algorithm to overcome the extended training time for large-scale datasets, high frequency oscillating, and non-stationary time series data. AI models have the capability to learn from historical data and make accurate predictions by identifying similarities in patterns [9]. These models are capable of taking various variables into account, such as climate, geography, and land use. Deep learning methods, which are an important subset of AI models, have become increasingly popular in recent years. These models excel at recognizing complex patterns and relationships in data without relying on assumptions about the data distribution. Numerous studies have been conducted to assess how well various deep learning models can predict daily discharge using meteorological and hydrological data [10]. Deep learning models have been subject to comparisons with traditional models, and frequently, they surpass empirical models in terms of accuracy [10,11]. For example Liu, Tang, Qin, Liu, Shen, Qu and Zhou [10] proposed a deep neural network (DNN) model for weekly discharge prediction that incorporated both temporal and spatial information. The model was trained using historical data from several gauging stations and was better than traditional statistical models in predicting weekly discharge. Wunsch, Liesch, Cinkus, Ravbar, Chen, Mazzilli, Jourde and Goldscheider [12] employed convolutional neural networks (CNNs) for simulating karst spring discharge and directly learning from spatially distributed climate input data. They demonstrated that the proposed models were highly effective in modeling karst spring discharge. Several other researchers [13–17] proposed deep learning models based on CNNs and LSTM for daily discharge prediction. The models were trained using meteorological and hydrological data from multiple stations and displayed higher accuracy compared to traditional models.

Dealing with nonstationary data poses a significant challenge when it comes to modeling natural phenomena [18]. Nonstationarity refers to the fact that statistical properties of the data change over time or space. In hydrological applications, nonstationarity may arise due to changes in climate patterns or land use. In the context of modeling natural phenomena, such as discharge, the presence of nonstationarity adds complexity to the task, as it entails accounting for evolving dynamic patterns in the underlying processes [19]. Signal processing methods like wavelet transform can be useful for handling nonstationary data. Wavelet transform breaks down signals into various frequency bands, facilitating the identification of distinct components in the data that exhibit changes over time or space [20–23]. By combining deep learning algorithms with signal processing methods like wavelet transform, strengths of both approaches can improve prediction of typically nonstationary data [9,24]. Deep learning algorithms are capable of capturing intricate patterns and correlations in data, whereas signal processing techniques aid in pre-analyzing the data by extracting significant features [25]. By taking into account the nonstationarity of the data, this approach can lead to more accurate predictions of runoff in a basin [26].

However, a remaining issue is related to the deep learning-signal processing blending approach. There are various methods available to combine deep learning techniques with signal processing approaches. One way involves decomposing the input variables (such as precipitation, temperature, evaporation, etc.) as well as the target variable (discharge) data. By treating the components of runoff as targets for the models, a multi-output deep learning model can be developed to predict these components. In the second blending approach, only the target data, i.e., discharge is decomposed and input data remain unchanged. As in the previous approach, a multi-output deep learning model is developed to predict discharge components. In both approaches an inverse wavelet transform is applied for predictions to obtain the final discharge values. In the latter blending method, only the input variables are decomposed, and the discharge values are imported to the model in their original form. This approach utilizes a single-output model and directly estimates the discharge values without predicting the individual components.

Local modeling approach is another strategy to tackle challenges associated with nonstationarities and heterogeneities in natural time series. As previously mentioned, time series are often characterized by heterogeneities, or variations that occur over time. These heterogeneities can pose challenges for accurate modeling and prediction. Temporal cluster-based local modeling of time series involves dividing the time series into clusters or segments based on their similarities in terms of patterns, trends, and other characteristics. Each cluster is then modeled using a local modeling technique that is specific to the data within that cluster. The advantage of cluster-based local modeling is that it allows for more accurate and robust modeling of complex time series by addressing heterogeneities across the data. By identifying and modeling different segments of the data separately, temporal cluster-based local modeling can account for variation in trend, seasonality, and other factors that may strongly influence the data over time [27]. There are several different approaches to perform cluster-based local modeling of time series data, including k-means, hierarchical clustering, density-based clustering, etc. In addition, a new approach called DWT-TCT (Discrete Wavelet Transform-Temporal Convolutional Transformer) was introduced as a method to predict daily runoff. The DWT-TCT combines the discrete wavelet transform with the Temporal Convolutional Transformer, providing a novel modeling approach for accurate runoff prediction.

In view of the above, this study aimed to investigate the effect of signal processing on the performance of deep learning models. It also aimed to assess and compare the performance of different blending approaches between deep learning and signal processing in predicting daily discharge. Additionally, the study evaluated the performance of temporal cluster-based local modeling and compared it with the results obtained from hybrid signal processing-deep learning models. The novelty of this work stems from its exploration of blending approaches that have not been extensively studied in the field of hydrology. Additionally, this study proposes a hybrid model that combines the discrete wavelet transform with the temporal convolutional transformer. Furthermore, the study addresses a critical research gap in water resources management by providing insights into the development of more accurate and efficient models for discharge prediction, especially in view of the nowadays increasing nonstationarity due to climate change. Overall, this research significantly contributes to the advancement of knowledge in the field of nonstationary hydrology and has practical applications for sustainable water resources management. Furthermore, the study compares the performance and computational cost of temporal cluster-based local modeling and hybrid modeling approaches, which are considered powerful methods for overcoming nonstationarities in the data.

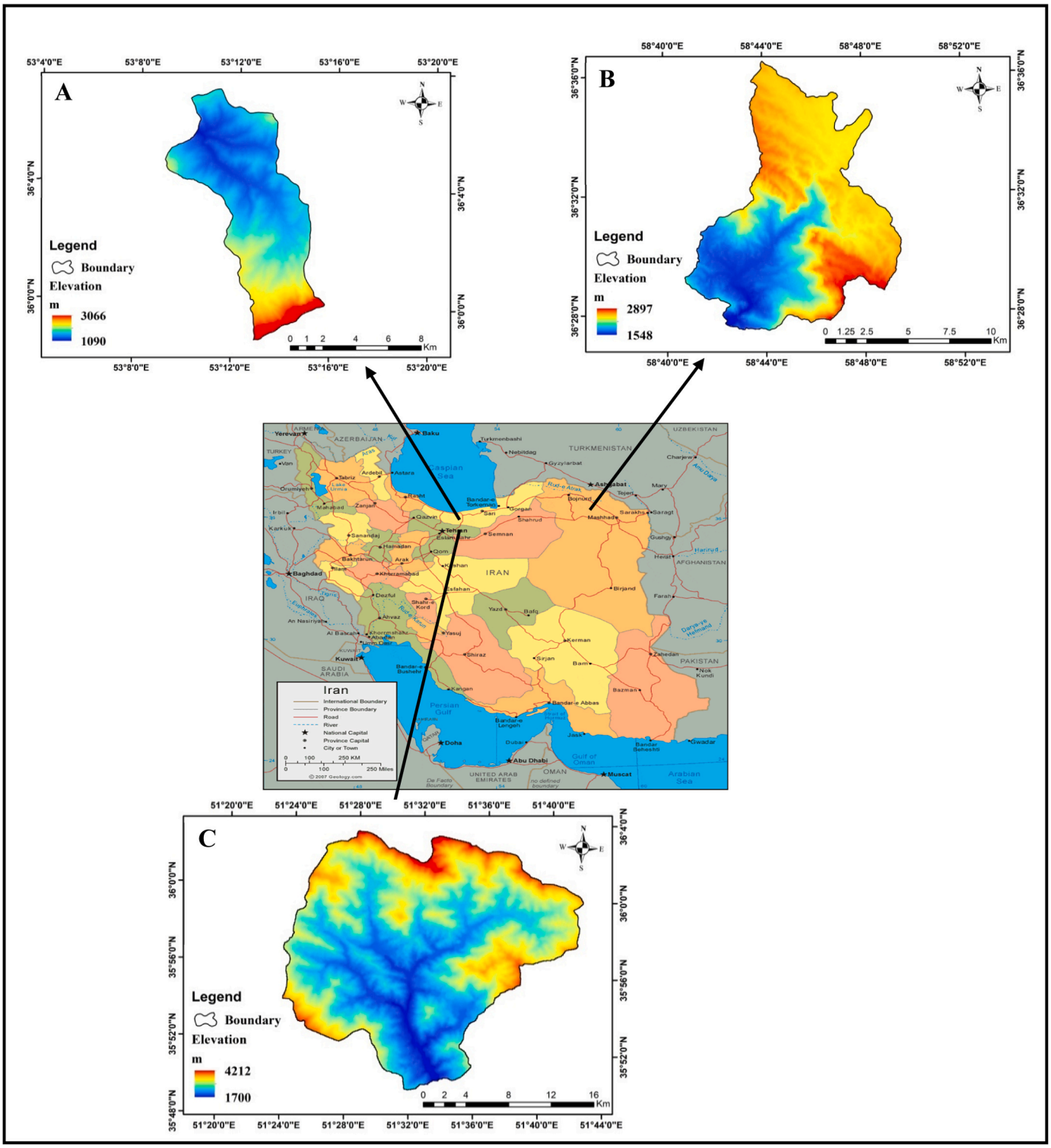

Fig. 1. Experimental study area. A) Bar-Erieh, B) Latian Dam, and C) Kasilian Basins.

2. Materials and methods

2.1. Study area

The study was conducted in three distinct and representative river basins: Kasilian, Latian Dam, and Bar-Erieh, as depicted in Fig. 1. These basins showcase varying physical, biological, and climatological characteristics. The Kasilian basin spans an area of 113 km^2, while the Latian Dam basin covers 436 km^2, and the Bar-Erieh basin encompasses 343 km^2. Bar-Erieh exhibits an average elevation of 2226 m, a precipitation rate of 330 mm per year, and an average slope of 11.9%. In contrast, the Latian Dam Basin features rugged terrain with an average elevation of 2830 m, an annual precipitation of 867 mm, and an average slope of 45.6%. The Kasilian Basin is situated within the Talar River Basin in northern Iran, experiencing an average annual precipitation of 733.3 mm. It has a semi-humid and cold climate, with a minimum altitude of 286 m and a maximum altitude of 3289 m amsl.

The study was conducted for the three representative river basins,

each characterized by distinct climatic, topographic, and physiographic features, aiming to enhance the generalizability of the findings. This approach provides a more comprehensive understanding of the performance of different modeling approaches. If consistent results are obtained across these diverse basins, the findings can be generalized with greater confidence. Consequently, the study's conclusions are more likely to be applicable to other regions exhibiting similar characteristics, thereby increasing the validity and robustness of the research. To construct the input database, observations from the Iran Meteorological Organization and Water Resources Management Organization were utilized, encompassing variables such as rainfall, discharge, temperature, and evaporation. In addition to these variables (R_t, D_t, T_t, E_t), the dataset also included past values of rainfall (R_{t-1}, R_{t-2}) and discharge (D_{t-1}, D_{t-2}), capturing the historical conditions leading up to the present observations. These lagged values provide insights into preceding patterns of hydrological variation. Daily measurements of these variables were collected over a 20-year period, yielding a comprehensive dataset in which each entry included all the variables, facilitating a thorough exploration of their interrelationships. The main objective of this study was to predict the discharge for the following day (D_{t+1}). It is important to note that outliers are frequently encountered in datasets and do not necessarily indicate incorrect values. They can arise from extreme events or uncommon phenomena. However, it is crucial to carefully assess and validate outliers to ensure their legitimacy before incorporating them into the analysis. In this specific study, a comprehensive evaluation of outliers was conducted, confirming their validity as genuine data points. Thus, these data were deemed reliable and were not excluded or disregarded. Missing values in the dataset were addressed through regression imputation, a technique used to estimate the missing values based on the relationships observed within the dataset.

2.2. Deep learning models

The data were first divided into testing and training sets using a 70/30 ratio. Before being imported into the deep learning models, a process called min-max scaling was employed to rescale and transform all variables, ensuring that they were within the range of 0–1. This particular technique was applied to normalize the data and make it suitable for effective analysis and interpretation within the deep learning models. By adjusting the variable ranges, the data was prepared to be efficiently processed by the models, enhancing their performance and accuracy in making predictions. This study employed two deep learning models to predict daily discharge. One of the utilized models was LSTM, which belongs to the category of deep learning models specifically designed for processing sequential data. LSTMs were developed to address the challenge of vanishing gradients commonly encountered when working with traditional recurrent neural networks and lengthy sequences of data [28,29]. LSTM uses memory cells with three gates, input, output, and forget, to control the flow of information. During training, the gate weights are adjusted to selectively store or retrieve important information from its memory cells. This allows capturing dependencies over longer periods in the sequential data, making it useful for predicting future states of long time series. To implement LSTM, layers of LSTM cells were stacked on top of each other. The input for each layer consists of a sequence of vectors representing individual time steps. The final layer's output was then fed into a dense layer for regression [30]. The internal state (h_t) for a traditional RNN is defined as [31]:

$$h_t = \tanh(Wx_t + Uh_{t-1} + b_h) \tag{1}$$

where the input weight matrix is represented by W, the recurrent weight matrix by U, the bias vector by b, the current input vector by x_t, and the last hidden cell state by h_{t-1}. However, LSTM includes a memory block comprising a cell state (c_t) for storing information, along with three gates: forget gate (f_t), input gate (i_t), and the output gate (o_t) [32]. The initial step in computing the forget gate, which regulates the previous cell state, involves using [33]:

$$f_t = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right) \tag{2}$$

After calculating the forget gate using the input weight matrix (W_f), recurrent weight matrix (U_f), and bias vector (b_f) with the logistic sigmoid activation function (σ), the next step involved computing a potential cell state ($\tilde{c}_t$) based on the current input (x_t) and the previous hidden state (h_{t-1}):

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{3}$$

where W_c, U_c, b_c are the input weight matrix, recurrent weight matrix, and bias vector for the potential cell state, respectively. The Eq. (4) was used to calculate the input gate, which determines the information from the potential cell state that should be allowed to update the current cell state (c_t). The input gate acts as a control mechanism for regulating the flow of information from the potential cell state to the current cell state:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{4}$$

Thus, the input gate (i_t) was calculated using the input (x_t), the previous hidden state (h_{t-1}), the weight matrices (W_i and U_i), and bias vector (b_i) associated with the input gate. The σ is a sigmoid function. Then, the current cell state was updated using the results of the previous equations:

$$c_t = f_t \otimes c_{t-1} + i_t \otimes \tilde{c}_t \tag{5}$$

where ⊗ indicates element wise multiplication. The logistic sigmoid function is utilized to generate two vectors, f_t and i_t, which range from 0 to 1. If f_t is close to 1, the previous information stored in the last cell state is retained, while it is discarded if f_t is close to 0. Moreover, the output gate is responsible for determining which information from the current cell state is transmitted to the new hidden state, and it was calculated as:

$$f_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{6}$$

where W_o, U_o, b_o are the input weight matrix, recurrent weight matrix and bias vector for the output gate, respectively. The new hidden state (h_t) was computed by:

$$h_t = o_t * \tanh(c_t) \tag{7}$$

Because the cell state in LSTM networks undergoes a simple linear operation at each time step, it is less likely for the information to get stuck or change drastically, which prevents the problem of vanishing or exploding gradients [29]. To obtain the final predicted values, a traditional dense layer is connected to the last LSTM layer at the last time step:

$$y = W_d h_n + b_d \tag{8}$$

where W_d, and b_d are the hidden-to-output weight matrix and the bias vector of the dense layer, respectively. The following algorithm was performed for prediction of daily discharge with LSTM:

1. A set of memory cells and input/output gates were initialized with the LSTM model.
2. The input sequence was processed, one time step at a time, through the LSTM layers.
3. At each time step, the input was passed through the input gate, which decided which information to store in the memory cell.
4. The current state of the memory cell was then updated based on the input and the previous memory state.
5. Next, the output gate decided which information to output from the memory cell.

6. The predicted value for the current time step was generated by passing the output through a dense layer by regression.
7. A loss function, (mean squared error), was used to compare the predicted values to the actual values.
8. The weights of the input gate, output gate, and memory cell were adjusted using backpropagation with the Adam optimizer.
9. During training, dropout regularization was applied to prevent overfitting.
10. Steps 2–9 were repeated for multiple epochs until convergence was achieved.

There are several parameters in the LSTM structure that should be tuned to achieve its optimum performance, i.e., the number of LSTM layers that determines the depth of the network, number of neurons in each LSTM layer that determines the capacity of the network to learn complex patterns, dropout rate which is a regularization technique used to prevent overfitting, and batch size that determines the number of samples used in each iteration of training and sequence length which determines the number of time steps used in each input sequence.

The other model used in this study was CNN. While CNNs are widely used for image classification tasks, they can also be employed for regression problems where the goal is to predict a continuous output variable. A CNN model for regression problems typically has a similar architecture to a standard CNN, but with some modifications. One modification that is often made to the standard CNN architecture is to replace the final activation function with a linear activation function. This allows the network to output a continuous value rather than a probability distribution over classes. Another modification is to change the loss function used during training to a regression-specific loss function, such as mean squared error (MSE), which measures the difference between the predicted and actual values. The rest of the network architecture can remain largely the same as for a classification CNN, including the use of convolutional and pooling layers to extract different features [34]. Let's consider a scenario where we have a time series consisting of T data points, and our goal is to predict the next K values in the series. To facilitate this prediction, we represent the time series as a matrix X with dimensions$(T - L + 1) \times L$, where L corresponds to the length of the input window or sequence. The input matrix X is fed into a convolutional layer with F filters of size $(h \times L)$, where h is the height of the filter. The output of the convolutional layer is a feature map of size $(\mathrm{T} - \mathrm{L} + 1) \times \mathrm{F}$, which captures local patterns in the time series. The convolution operation is expressed as:

$$\mathrm{h_i} = \mathrm{activation}\left(\mathrm{b_i} + \sum_{\mathrm{j}} \left(\mathrm{W_{ij}} * \mathrm{x_j}\right)\right) \tag{9}$$

where h_i is the $\mathrm{i_{th}}$ feature map, b_i is the bias term, W_{ij} is the weight matrix for the i_{th} filter and j_{th} input channel, x_j is the input channel, and activation is the activation function. The feature map is then passed through a max pooling layer with pool size $(p \times 1)$, where p is the pooling size. The output of the max pooling layer is a downsampled feature map of size $(T - L + 1)/p \times F$, which reduces the dimensionality of the feature map and makes the model more robust to small variations in the input. The max pooling operation is expressed as:

$$\mathrm{y_i} = \mathrm{max_j}\left(h_i * p + j\right) \tag{10}$$

where y_i is the $\mathrm{i_{th}}$ output of the pooling layer, h_i is the $\mathrm{i_{th}}$ feature map, p is the pooling size, and j ranges over the indices of the elements in the pooling window. The output of the pooling layer is flattened into a one-dimensional vector using:

$$\mathrm{output} = \mathrm{flatten(input)} \tag{11}$$

where input is the input feature map. The flattened feature map is then passed through one or more dense layers to perform classification or regression. The output of the dense layer is:

$$\mathrm{y} = \mathrm{activation}\left(\mathrm{b} + \sum_{i} (W_i * x_i)\right) \tag{12}$$

where y is the output of the dense layer, b is the bias term, W_i is the weight for the i_{th} input, x_i is the i_{th} input, and activation is the activation function. The following algorithm was used for predicting daily discharge with CNN:

1. The input data was preprocessed by applying normalization.
2. A CNN model was built with multiple layers of convolutional and pooling layers, followed by fully connected layers.
3. The CNN model was trained using backpropagation and gradient descent optimization algorithms. During training, the weights of the model were adjusted to minimize the loss function, which in this case was typically a MSE loss function.
4. The performance of the trained CNN model on a validation set was evaluated to tune hyperparameters such as the learning rate, number of filters, and kernel size.
5. Once the model was well-tuned, predictions were made on the test set using the trained CNN model.
6. The performance of the CNN algorithm was evaluated using some metrics.

Parameters such as the number of convolutional layers that determines the depth of the network, the number of filters in each convolutional layer that determines the capacity of the network to learn complex patterns, kernel size that determines the size of the sliding window used for convolution, stride which determines the step size of the sliding window used for convolution and pooling which is a down sampling operation used to reduce the spatial dimensions of the feature maps should be calibrated to attain its best performance. Therefore, the optimization algorithms detailed in Section 2.7 were used to fine-tune the structural parameters of the suggested models.

2.3. Hybrid signal processing- deep learning models (assessing blending approaches)

The training and testing datasets were decomposed using the discrete wavelet transform (DWT). The DWT is a signal processing technique that breaks down a signal into different frequency components, enabling a more detailed analysis of its characteristics. This transformation involves passing the signal through a series of filters that extract high- and low-pass frequency components at various scales. DWT serves as a powerful tool in extracting features for deep learning models by decomposing signals into different frequency components, which can subsequently be used as input features for the model. Consequently, a tree-like structure of coefficients is obtained, with each level representing a different level of frequency detail. These coefficients can be further analyzed or used for reconstructing the original signal. The DWT offers valuable insights by capturing both high-frequency details and low-frequency trends in complex signals. To conduct the DWT, a wavelet function and a level of decomposition were selected. The chosen wavelet function was then employed to filter the signal using both low-pass and high-pass filters. This process was repeated for the low-pass filtered signal until the desired level of decomposition was reached. The resulting coefficients represent the approximation at the final level and detailed coefficients at each level. Eqs. 13 and 14 show the mother wavelet and continuous wavelet transform of a signal, respectively.

$$\psi_{j,k}(x) = 2^{\frac{j}{2}} \psi_{j,k}\left(2^j x - k\right) \tag{13}$$

where $\psi_{j,k}(x)$ is the produced form a mother wavelet $\psi(x)$ that is expanded by j and translated by k.

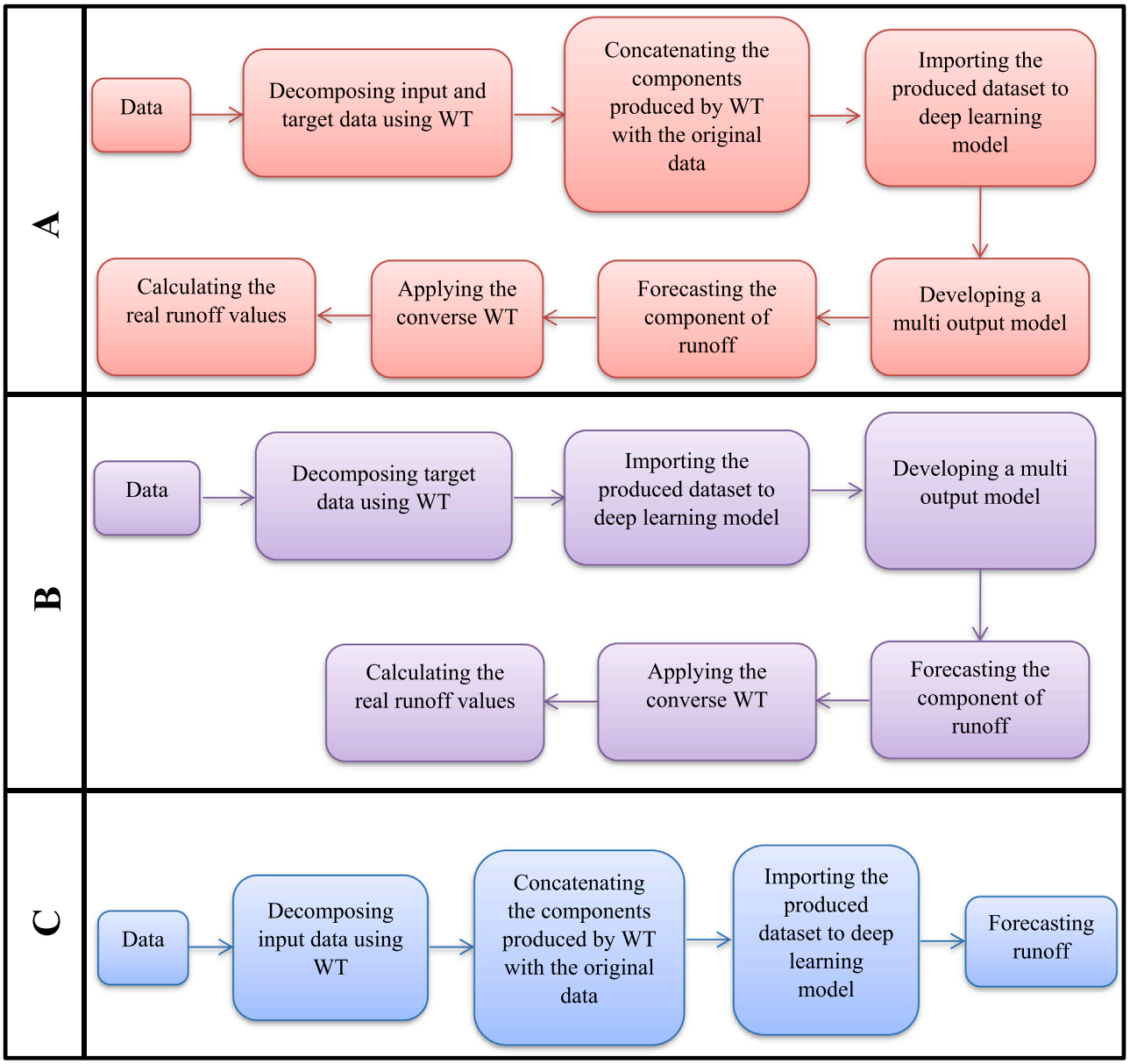

Fig. 2. Three proposed blending approaches.

$$c_{j,k} = \int_{-\infty}^{\infty} f(x)\psi_{j,k}^{*}(x)dx \tag{14}$$

where $c_{j,k}$ is the approximation coeficient of the signal. The mother wavelet is formulated from the scaling function $\varphi(x)$ as:

$$\varphi(x) = \sqrt{2}\sum h_0(n)\varphi\left(2x-n\right) \tag{15}$$

$$\psi(x) = \sqrt{2}\sum h_1(n)\varphi\left(2x-n\right) \tag{16}$$

$$h_1(n) = (-1)^n h_0(1-n) \tag{17}$$

where $\varphi(x)$ represents the scaling function of the wavelet transform, h_0 is a set of filter coefficients for the low-pass filter and $h_1(n)$ is a set of filter coefficients for the high-pass filter. However the discrete form of the wavelet transform to calculate approximation and details is as follows. Let *x[n]* be a discrete-time signal of length *N*, and *h[n]* and *g[n]* be the analysis low-pass and high-pass filters, respectively, of length *L*. The DWT decomposes the signal into approximation coefficients *cA* and detail coefficients *cD* at different scales or levels. The approximation coefficients *cA* represent the low-frequency content of the signal, while the detail coefficients *cD* represent the high-frequency content of the signal. The DWT can be computed recursively using:

$$cA(j+1)[k] = (cA(j) * h[2k] + cD(j) * g[2k]) \Big/ \sqrt{2} \tag{18}$$

$$cD(j+1)[k] = (cA(j) * g[2k+1] + cD(j) * h[2k+1]) \Big/ \sqrt{2} \tag{19}$$

where *j* is the current level or scale, *k* is the sample index, and * denotes the convolution operation.

Different mother wavelets were used, and the decomposition was performed at various levels. Both the decomposed and original input data were considered as inputs for the deep learning models. In this study, three blending approaches that combine signal processing and deep learning models were tested (Fig. 2). The first approach (part A) involved decomposing both input data (such as rainfall (R_t), discharge (D_t), temperature (T_t), evaporation (E_t), rainfall with time lags (R_{t-1}, R_{t-2}), discharge with time lags (D_{t-1}, D_{t-2}), etc.), and target data (one day

ahead discharge (D_{t+1})) using DWT. As the target data were decomposed into sub-series and used as targets, the model was trained to predict the individual components of the target rather than the target itself. Consequently, a multi-output model was created, with each model estimating one component of the target. These estimated components were then combined to calculate the actual target values. In the second approach (part B), only the target data was decomposed using DWT. Once again, a multi-output model was developed, with each model estimating a component of the target. In the third approach (part C), only the input data underwent DWT decomposition. Since the target data remained unchanged, a single output model was sufficient for directly predicting the target values.

Aiming to tackle the challenge of handling a large number of inputs, especially when utilizing signal processing techniques, the neighborhood component analysis method was employed as a reliable approach for feature selection. This method prioritizes the selection of crucial inputs that significantly contribute to enhancing the accuracy of predictions. This algorithm is non-parametric and focuses on optimizing an objective function that calculates the average regression loss over the training data while leaving out one input at a time. By doing so, it determines the weights that help minimize this objective function, leading to improved predictive performance.

2.4. Temporal cluster-based local modeling

In addition to the hybrid models, a temporal cluster-based local modeling approach was proposed in this study. Temporal cluster-based local modeling is a technique used in time series analysis to model the behavior of a system over time. It involves dividing the time series into clusters, where each cluster includes data points that are similar in some way (e.g., they have similar patterns or trends). Within each cluster, a local model was developed to capture the behavior of the system. The idea behind this approach is that a single global model may not be able to capture all the complex dynamics and variations in a long time series. Instead, by dividing the time series into clusters and developing local models for each cluster, we can better capture the behavior of the system at different time periods and under different conditions. This may better describe properties of non-stationary time series.

To implement temporal cluster-based local modeling, the time series was divided into clusters based on similarity criteria. A hierarchical clustering was performed to cluster the data into two clusters. Hierarchical clustering involves constructing a dendrogram that represents the hierarchy of clusters in the data [35]. A dendrogram is constructed by iteratively merging the two closest clusters until all points belong to a single cluster. In time series analysis, the first step is to compute pairwise distance measures between each pair of time series in the dataset. One commonly used distance measure for time series is Dynamic Time Warping (DTW), which finds the optimal alignment between two time series based on their shapes. Once the distance matrix is computed, hierarchical clustering can be applied using any of the available linkage methods, such as average linkage, complete linkage, or ward linkage. Ward linkage is known to produce well-separated clusters compared to other linkage methods. After computing the dendrogram, it can be cut at a particular level to obtain a specific number of clusters. The deep learning local models were then developed for each cluster, which captures the behavior of the system within the context of that cluster [36].

2.5. Discrete wavelet transform-temporal convolutional transformer

Finally a combined discrete wavelet transform-Temporal Convolutional Transformer (DWT-TCT) model was developed in this study for runoff prediction. TCT is a variant of the Transformer architecture specifically designed for processing sequential or time series data. TCT combines the strengths of both temporal convolutions and self-attention mechanisms. The TCT model consists of multiple layers, and each layer has two main components: temporal convolutions and self-attention. Temporal convolutional layers capture local patterns and dependencies within the input sequence [37]. They apply a set of learnable filters across different temporal positions, allowing the model to extract meaningful features. The filters were designed to extract meaningful features by convolving them with the input sequence. By stacking multiple convolutional layers, the TCT model was able to capture increasingly complex temporal patterns. The outputs of the convolutional layers were then passed to the self-attention component. Self-attention component enables the model to capture global dependencies and long-range relationships within the sequence. It attends to different positions in the input sequence and aggregates information from relevant positions to produce context-aware representations. This was achieved through a mechanism known as scaled dot-product attention. In this process, the input sequence was transformed into query, key, and value vectors. The attention mechanism then computed the attention weights by measuring the similarity between the query and key vectors. These weights were used to weight the value vectors, which were then aggregated to produce the context-aware representations. The attention mechanism attended to all positions in the input sequence, allowing the model to capture long-range dependencies effectively.

By combining temporal convolutions and self-attention, the TCT model can effectively capture both local and global temporal dependencies, making it suitable for tasks such as time series forecasting, sequence classification, and anomaly detection in sequential data [38]. There are several key parameters in this method that need to be tuned during the calibration process. Firstly, the number of TCT layers determines the model's depth. Increasing the number of layers allows the model to capture more complex temporal patterns, but it also raises the risk of overfitting. Secondly, the filter sizes in the temporal convolutional layers determine the receptive field of the model trying out different filter sizes helps in capturing patterns of various lengths in the input sequence. Another crucial parameter is the dilation rates, which control the spacing between filter weights in the temporal convolutional layers. Adjusting the dilation rates enables the model to capture patterns at different scales, encompassing both short and long dependencies. In the self-attention component, the number of attention heads determines the model's ability to attend to different positions. Increasing the number of attention heads allows the model to capture more detailed dependencies, but it also increases computational complexity. The learning rate is also an important parameter as it affects the speed of model convergence during training. Tuning the learning rate is crucial for finding the optimal balance between convergence speed and stability. Lastly, the batch size determines the number of samples processed in each training iteration. Modifying the batch size can impact the model's convergence speed and its ability to generalize to new data.

2.6. Uncertainty analysis and inputs weight assessment

The assessment also involved evaluating the uncertainty related to the model type selection and the input variable selection. To evaluate the uncertainty associated with the selection of the model type, the study calculated the predicted values using the same combination of input variables for the proposed models [39]. The R factor index was then used to measure this uncertainty using the following equation.

$$R = \frac{S_P}{S_O} \tag{20}$$

where S_O is the standard deviation of the observed data and S_P is computed as follows:

$$S_P = \sum_{i=1}^{n} (U_{Pi} - UL_{Pi})/n \tag{21}$$

where n is the number of observed data and U_{pi} and L_{pi} are the i^{th} values of the upper quartile (97.5%) and lower quartile (2.5%) of the 95 PPU

band, respectively. To determine the level of uncertainty related to the input variable selection, the predicted data were calculated for several combinations of input variables using a single model for each observed data point. Subsequently, the uncertainty associated with the input variable selection was measured using the R factor method mentioned earlier for quantifying uncertainty related to the model type selection.

The cosine amplitude method was also used for sensitivity analysis and calculating the weights of inputs. It is a technique commonly used in sensitivity analysis to evaluate the impact of input parameter variations on the output of a mathematical model or simulation. It aids in evaluating the responsiveness of the model's outputs to variations in particular inputs. In the cosine amplitude method, input sensitivity is determined by varying its value within a defined range while keeping all other parameters constant. The variation is typically modeled using a cosine wave, which allows for a smooth and continuous transition between different variable values [40]. By systematically varying the input variable and observing the corresponding changes in the model output, analysts can gain insights into which variables have the most significant influence on the output and which ones are less critical.

2.7. Optimization algorithms

We utilized four optimization algorithms, namely genetic algorithm (GA), particle swarm optimization (PSO), grey wolf optimization (GWO), and ant colony optimization (ACO), to optimize the structural parameters of the proposed models. Here are detailed explanations of the optimization algorithms employed in this research. GA is a metaheuristic optimization method inspired by natural selection and genetics. It mimics the evolutionary process by selecting, crossing over, and mutating candidate solutions to enhance model performance. Its purpose is to find the best parameter configuration for models by exploring various combinations systematically. To optimize the structural parameters of the proposed methods, the first step involved defining the model structure within the GA. The parameters were encoded into chromosomes or individuals in the GA representation. A Fitness Function was created to evaluate performance and minimize errors. The process started with a population of individuals with random parameters, undergoing selection, crossover, and mutation to produce new generations. Selection prioritized individuals based on fitness, while crossover and mutation introduced genetic diversity. This iterative process aimed to minimize errors and enhance model performance by refining its structure and optimizing parameters. The primary goal was to reduce errors, validating and improving the model's performance through successive iterations.

The PSO algorithm also employed the following steps. In this algorithm a population of particles is randomly initialized within the search space. Each particle represents a potential solution and is characterized by its position and velocity vectors. The fitness of each particle is evaluated by applying the objective function of the problem being optimized. Each particle adjusts its position and velocity by considering its own best position found so far (p_{best}) and the best position discovered by any particle in the population (g_{best}). By iteratively adjusting particle positions and velocities, the algorithm explores the search space to locate promising regions and exploits them to converge towards optimal solutions [41,42]. The balance between exploration and exploitation is achieved through the inertia weight and social and cognitive acceleration factors. The algorithm iteratively repeats these steps until the termination condition is met.

GWO algorithm, is also inspired by the social hierarchy and hunting behavior of grey wolves [43]. The process of GWO algorithm is as follows. A population of candidate solutions, referred to as wolves, is randomly initialized. Each wolf represents a potential solution and is characterized by its position vector in the search space. The fitness of each wolf is evaluated by applying the objective function of the problem being optimized. This function quantifies the quality or performance of each solution. The three best wolves in the population are assigned the roles of alpha, beta, and delta wolves, respectively. The alpha wolf represents the best solution found so far, while the beta and delta wolves correspond to the second and third-best solutions. Then the wolves update their positions to explore and exploit the search space. Based on the hunting behavior of wolves, three key operators are employed. In the prey position update step, the positions of the non-alpha, non-beta, and non-delta wolves are adjusted to simulate the movement of a prey. This is achieved by updating the wolves' positions towards the positions of the alpha, beta, or delta wolves. In the Alpha wolf update step, the position of the alpha wolf undergoes further modifications to explore potentially promising areas within the search space. Finally boundary handling is performed to ensure that the updated positions remain within the predefined bounds of the problem. The algorithm iteratively repeats these steps until the certain termination condition is met [44].

Another optimization algorithm utilized in this research is ACO, which takes inspiration from the foraging habits of ants [45]. In ACO, the search process is guided by the concept of pheromone trails. Ants deposit pheromones along their paths, and the concentration of pheromone on a particular path is proportional to the quality of the solution associated with that path. Other ants then use the pheromone trails to make decisions about the routes they will explore. To apply ACO for optimizing the structural parameters of deep learning models, we represented the search space as a graph, where each node represents a specific configuration of architectural parameters, and the edges represent the connections between different configurations. The algorithm starts by initializing the pheromone trails on the graph. Initially, the pheromone concentrations are set to low values to encourage exploration of different paths [46]. Each ant then iteratively constructs a solution by probabilistically choosing the next architectural parameter configuration based on the pheromone trail information and potentially domain-specific heuristics. After the ants have constructed their solutions, the pheromone trails are updated. The pheromone evaporation process reduces the pheromone levels on all paths, simulating the decay of pheromones over time. This evaporation helps to prevent the algorithm from converging prematurely to suboptimal solutions. The ants also deposit pheromones on the paths they have traversed, with the amount of pheromone proportional to the quality of the corresponding solution. This pheromone reinforcement process strengthens the paths associated with better solutions, making them more attractive for future ants. The ACO algorithm continues to iterate through multiple iterations, allowing the ants to explore and exploit the search space. The search process is typically terminated after a predetermined number of iterations or when a stopping criterion is met [47].

2.8. Performance evaluation

Assessing the performance of models is vital in order to comprehend how well they perform and to make comparisons between different modeling approaches. Various metrics were used to evaluate the performance of proposed models, including coefficient of determination (R^2), root mean square error (RMSE), normalized root mean square error (NRMSE), and Nash-Sutcliffe (NSE). R^2 measures the proportion of the variation in the target variable that is explained by the model. It ranges from 0 to 1, where a value of 1 indicates a perfect correlation between the observations and the model output. RMSE measures the average difference between the predicted values and the actual values. It is expressed in the same units as the target variable and provides an idea about the magnitude of the prediction error. NRMSE measures the relative performance of the model by normalizing the RMSE with respect to the range of the target variable. NSE measures the relative magnitude of the residual variance compared to the observed data variance. It ranges from negative infinity to 1, where a value of 1 indicates a perfect match between the model and the observations. The equations for these metrics are given as:

Table 1
Descriptive statistics provided for the discharge data in the study regions.

Basin	descriptive statistics			
	Mean	Min	Max	Standard deviation
Bar-Erieh	0.42	0	18.1	0.84
Latian Dam	6.88	0	54.8	7.55
Kasilisn	0.35	0	14	0.59

Table 2
Optimal Values of Parameters for the LSTM and CNN Models.

Model	Parameter	Optimal value
LSTM	Number of LSTM layers	2
	Number of neurons in each LSTM layer	68
	Dropout rate	0.2
	Batch size	41
	Sequence length	40
CNN	Number of convolutional layers	2
	Number of filters in each convolutional layer	28
	Kernel size	3
	Stride	2
	Pooling	4

$$r^2 = \left(\frac{\sum_{i=1}^{n}(o_i - \overline{o})(e_i - \overline{e})}{\sqrt{\sum_{i=1}^{n}(o_i - \overline{o})^2}\sqrt{\sum_{i=1}^{n}(e_i - \overline{e})^2}} \right)^2 \tag{22}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(o_i - e_i)^2}{n}} \tag{23}$$

$$NRMSE = \frac{RMSE}{range\ \ of\ observed\ data} \tag{24}$$

$$NSE = 1 - \frac{\sum_{1}^{n}(o_i - e_i)^2}{\sum_{1}^{n}(o_i - \overline{o_i})^2} \tag{25}$$

where o denotes observed runoff and e the estimated runoff.

3. Results and discussion

Table 1 displays the statistical properties provided for the discharge data in the study regions. Table 2 presents the optimal values for the structural parameters of the LSTM and CNN models. The sensitivity analysis revealed that the CNN model exhibited the highest sensitivity to the number of convolutional layers and stride parameters. This means that variation in these parameters had a significant impact on the performance and behavior of the CNN model. On the other hand, the LSTM model demonstrated greater sensitivity to the batch size and dropout rate.

Fig. 3 illustrates the results of the discrete wavelet-based decomposition (DWT) applied to the discharge data, as an example of data decomposition. In DWT, the time series was decomposed into a collection of discrete basis functions or wavelets at various scales. This decomposition is represented by a tree-like structure, where each level of the tree corresponds to a distinct scale of decomposition. One of the benefits of wavelet-based decomposition is its ability to analyze signals at various scales. This is especially beneficial when dealing with intricate signals that may have varying frequencies and hold valuable information. By breaking down the signal into various scales, it becomes possible to pinpoint the frequencies that are most pertinent to the phenomenon being studied, thereby offering valuable insights into the underlying processes that influence the signal. However, it is important to acknowledge that the choice of wavelet function and level of decomposition can influence the results of wavelet-based decomposition. Therefore, it is crucial to carefully select the appropriate wavelet function and level of decomposition based on the characteristics of the signal. This ensures accurate and meaningful analysis of the signal. In Fig. 3, the top panel in each section demonstrates the original signal, and the subsequent panels show the wavelet coefficients at each level of the decomposition. The low-frequency components of the signal are captured in the first level of decomposition, while the high-frequency components are captured in the subsequent levels. By examining the patterns in the wavelet coefficients at each level, we gain insight into the underlying processes driving the signal and can identify any significant features or changes in the signal over time. In wavelet-based signal processing, a signal can be decomposed into two components: an approximation component and a detail component. The approximation component captures the low-frequency information in the signal, while the detail component captures the high-frequency information. The approximation component represents a coarse-grained version of the original signal that retains only the most important features of the signal. The detail component represents the fine-grained or high-frequency information in the signal. It is obtained by high-pass filtering the signal.

Figs. 4 and 5 show the results of DWT-LSTM and DWT-CNN models developed based on the first blending approach. As mentioned before, both input and target were decomposed using DWT. In this approach multi-output deep learning models were developed to estimate the components of the discharge. The estimated components of the discharge were then summed up to provide the actual discharge estimations. The figures show a moderate performance for both models using this blending approach. Also, the R^2 show that both models were precise, but the NRMSEs (22, 16, and 25 for Bar-Erieh, Latian Dam and Kasilian Basins, respectively, for DWT-LSTM model) showed a moderate performance for all three watersheds. This indicates that this approach was not suitable for blending DWT and deep learning. The multi-output structure of this approach may contribute to its challenges. Even if one of the outputs is estimated with high error, it can have a noticeable impact on the overall results. This is because the blending approach involves combining predictions from multiple models, allowing errors from one model to propagate and affect the final outcome. If one of the models is weak or prone to errors, it can significantly degrade the overall performance of the blending. Multi-output modeling adds complexity to the model, making it more difficult to train, interpret, and optimize. This is particularly true when dealing with a large number of outputs or complex interdependencies between them. Additionally, the computational cost of training and evaluating the model can increase, requiring more resources and time. Another limitation is the potential difficulty in generalizing the output of multi-output models to new data inputs. Poor performance on new data can restrict the practical usefulness of the approach. These challenges highlight the practical barriers associated with utilizing multi-output models, especially when working with large datasets or complex models. Careful consideration is necessary when deciding whether to employ this approach, taking into account the specific requirements and trade-offs involved.

Figs. 6 and 7 display the results of the DWT-LSTM and DWT-CNN models based on the second blending approach. In this approach, only the target was decomposed using DWT, similar to the first blending approach, and multi-output deep learning models were developed to estimate the components of the target, i.e., discharge. The estimated components of the discharge were then summed up to calculate the actual discharge estimations. However, the results indicate that this approach produced weaker results compared to the first approach, in which both input and target were decomposed using DWT, and multi-output deep learning models were built to estimate the components of the target. The reason for this could be that when only the target signal is decomposed using DWT, there is a possibility that some crucial information present in the input signal may disappear or not be effectively

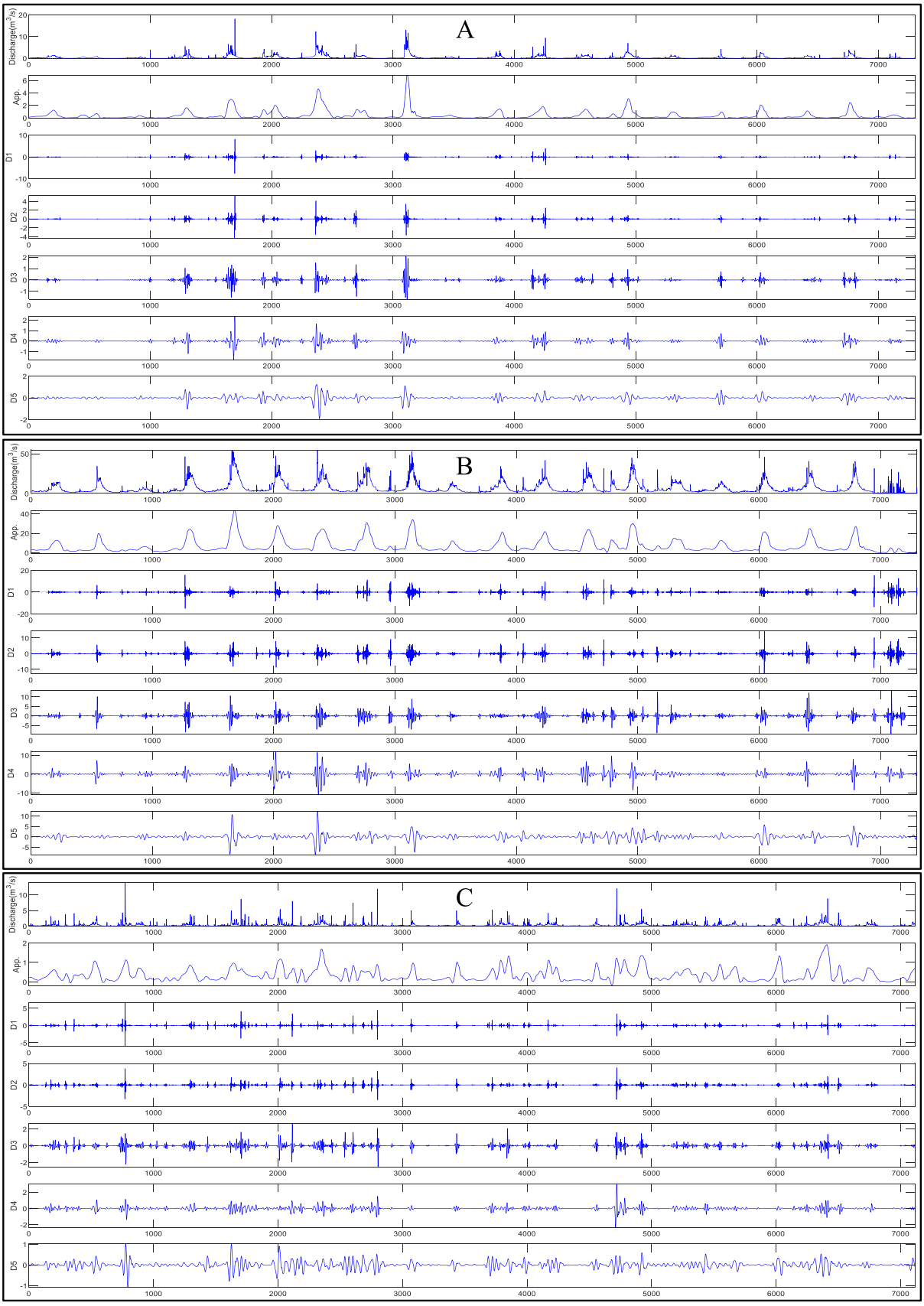

Fig. 3. Wavelet based decomposition of discharge time series at 5 levels using db4. A) Bar-Erieh, B) Latian Dam, and C) Kasilian Basins.

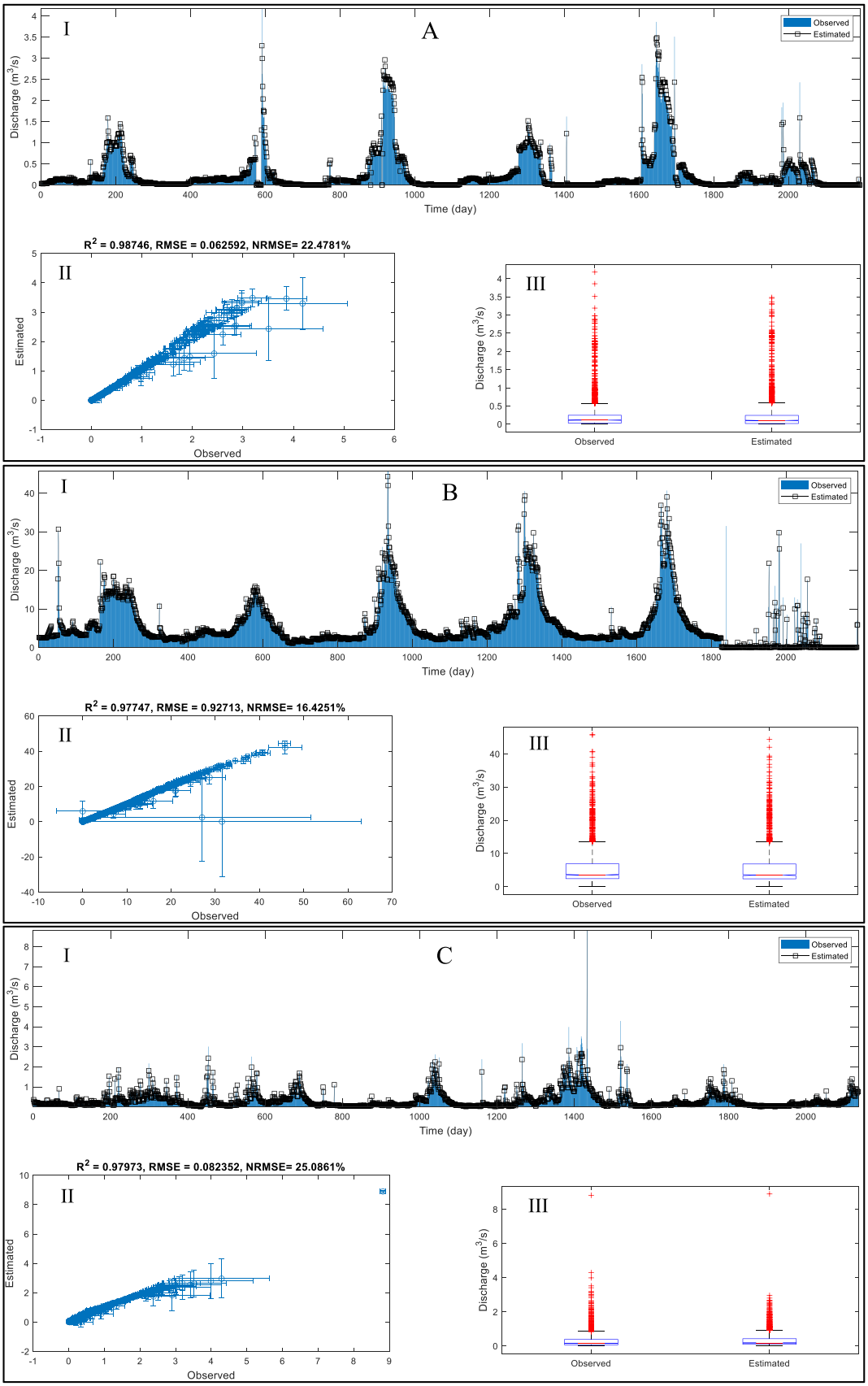

Fig. 4. DWT-LSTM model with first blending approach. A) Bar-Erieh, B) Latian Dam, and C) Kasilian Basins.

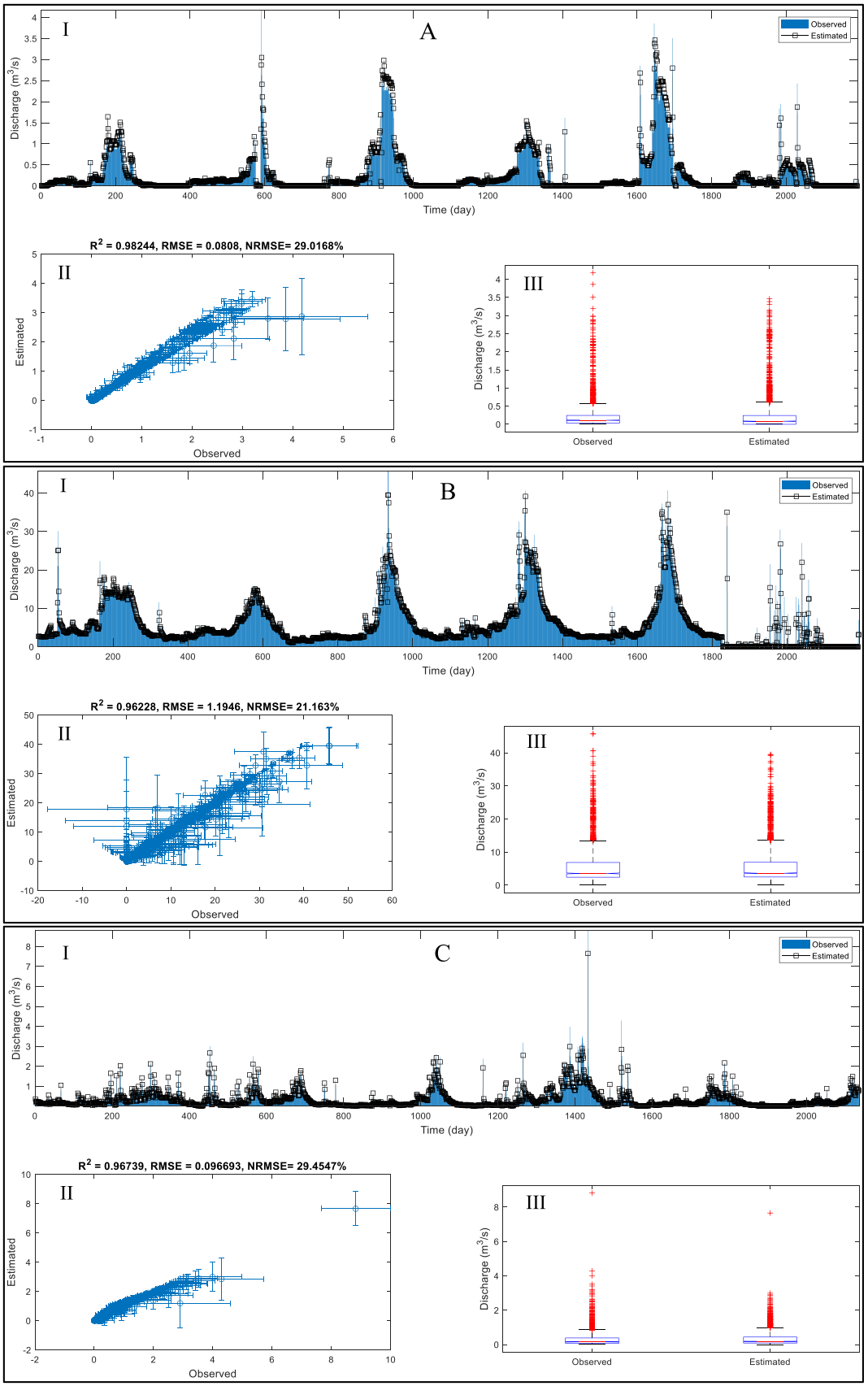

Fig. 5. DWT-CNN model with first blending approach. A) Bar-Erieh, B) Latian Dam, and C) Kasilian Basins.

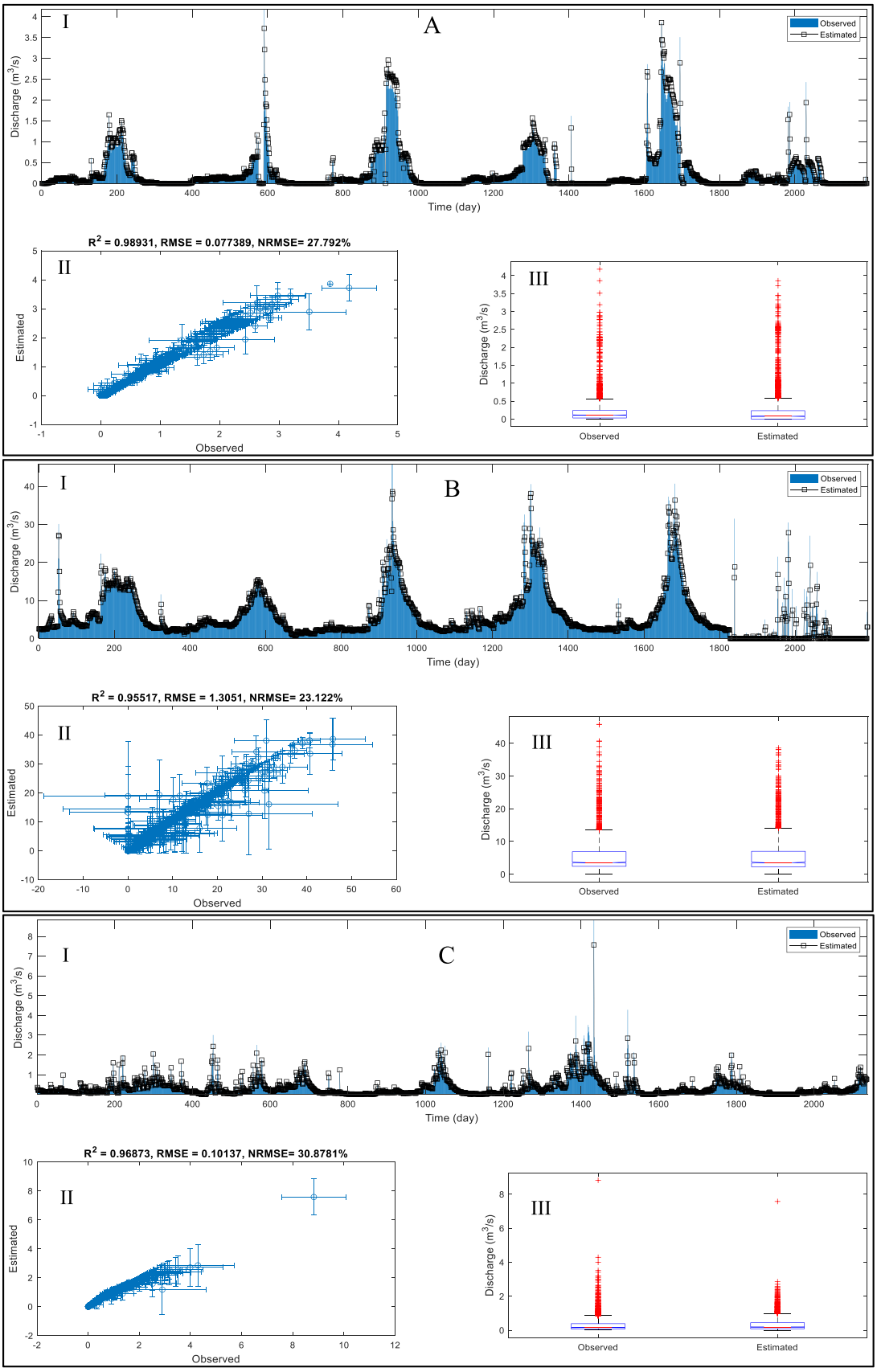

Fig. 6. DWT-LSTM model with second blending approach. A) Bar-Erieh, B) Latian Dam, and C) Kasilian Basins.

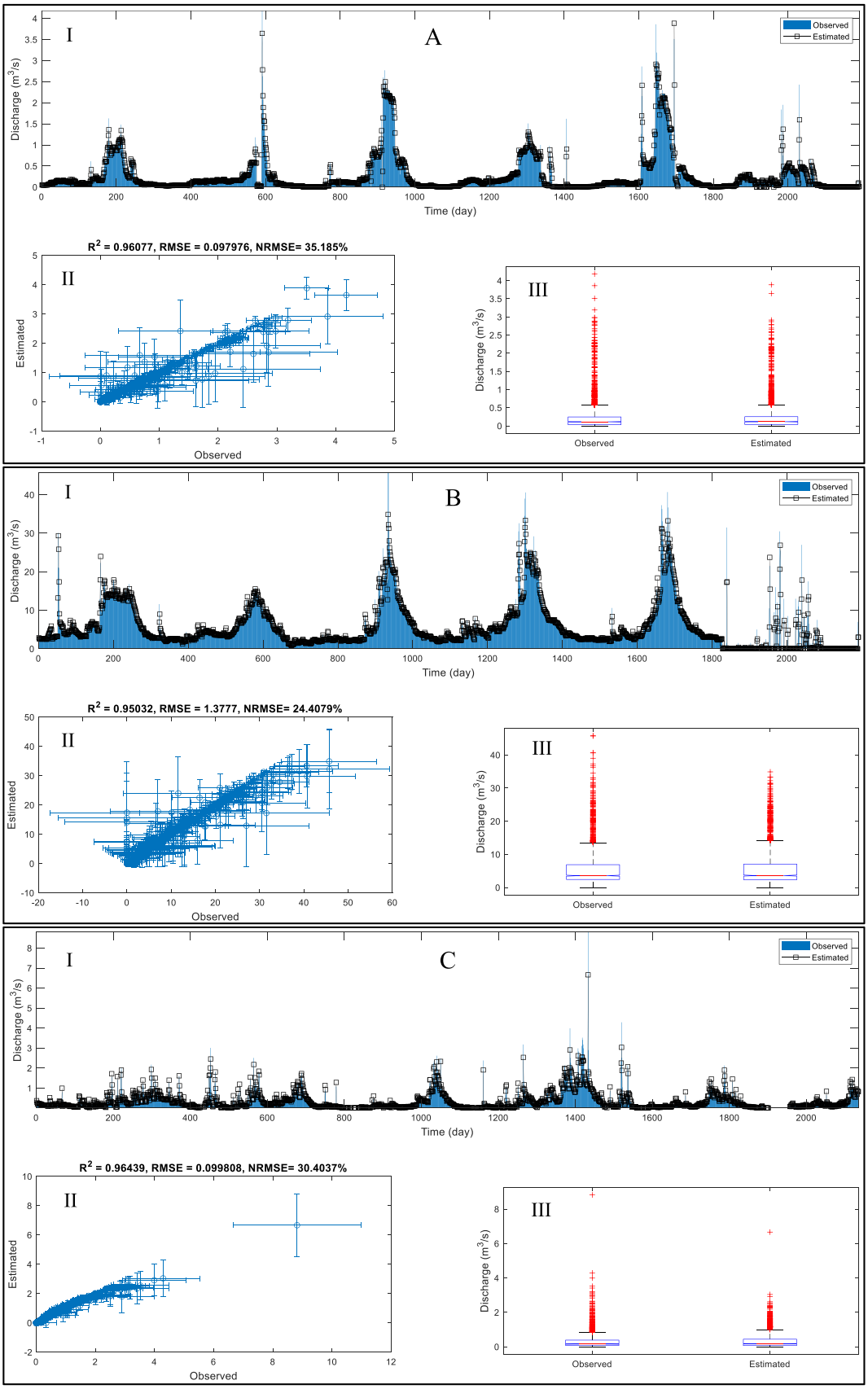

Fig. 7. DWT-CNN model with second blending approach. A) Bar-Erieh, B) Latian Dam, and C) Kasilian Basins.

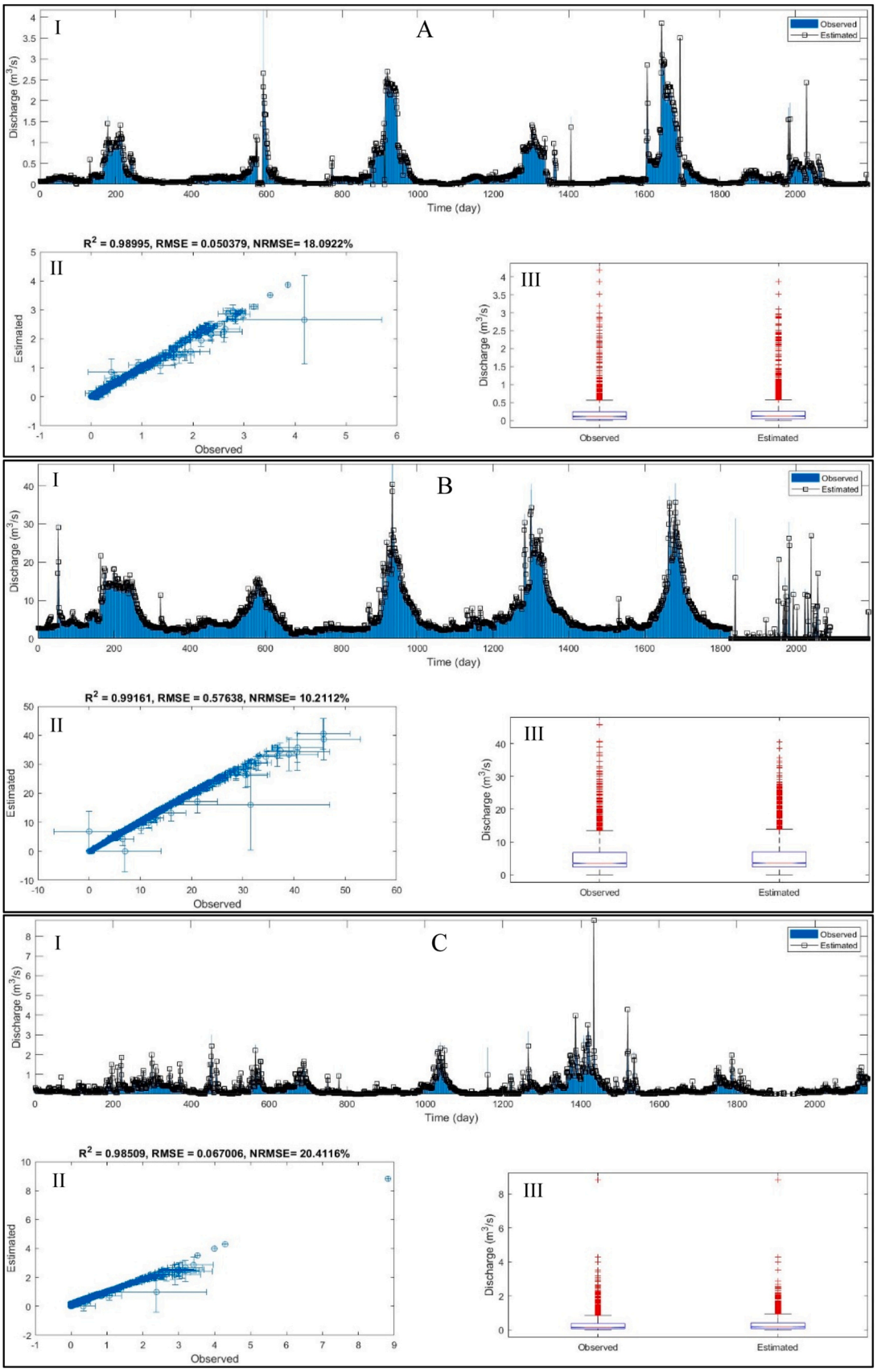

Fig. 8. DWT-LSTM model with third blending approach. A) Bar-Erieh, B) Latian Dam, and C) Kasilian Basins.

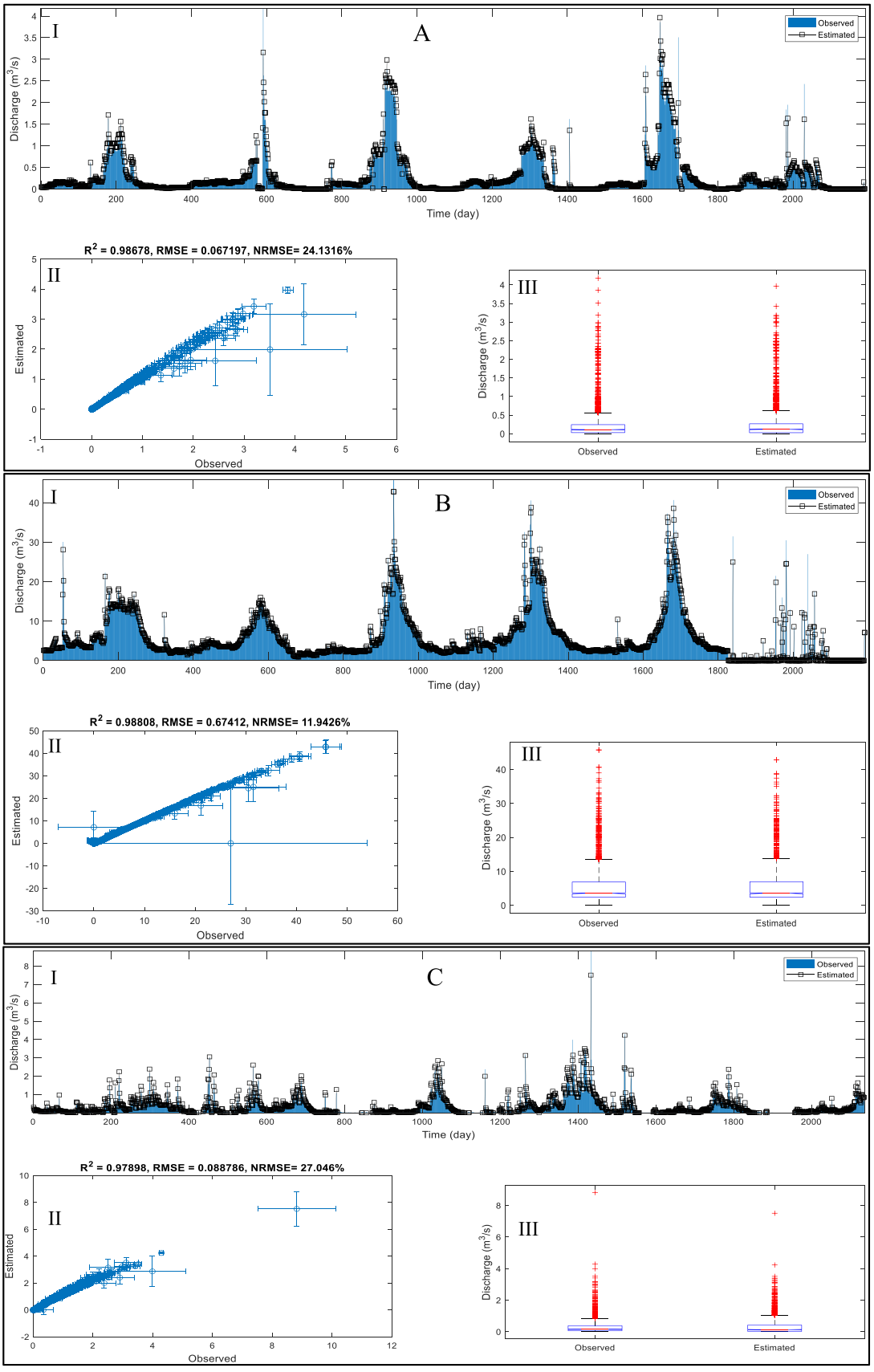

Fig. 9. DWT-CNN model with third blending approach. A) Bar-Erieh, B) Latian Dam, and C) Kasilian Basins.

Table 3
Relative improvement/decline for the third blending approach compared to the second blending approach for DWT-LSTM.

Basin	R^2 Improvement/ decline (%)	RMSE Improvement/ decline (%)	NRMSE Improvement/ decline (%)	Nash- Sutcliffe Improvement/ decline (%)
Bar-Erieh	0.30	35.06	34.90	10.64
Latian Dam	3.63	55.86	55.83	8.51
Kasilian	1.72	33.66	33.89	7.95

Table 4
Relative improvement/decline for the third blending approach compared to the second blending approach for DWT-CNN.

Basin	R^2 improvement/ decline (%)	RMSE improvement/ decline (%)	NRMSE improvement/ decline (%)	Nash- Sutcliffe improvement/ decline (%)
Bar-Erieh	2.64	30.92	31.42	8.79
Latian Dam	3.85	51.05	51.07	10.53
Kasilian	1.43	11.11	11.04	2.41

utilized, resulting in weaker model performance. Discharge estimation requires a good understanding of the complex relationship between the input data and the target signal. By decomposing both inputs and the target using DWT, the developed models can capture and utilize the interdependencies between these signals more efficiently than when only the target is decomposed. However, just like the first blending approach, this approach still has the deficiencies of multi-output modeling of discharge components.

Figs. 8 and 9 show the results of the DWT-LSTM and DWT-CNN models based on the third blending approach. In this approach, only the inputs were decomposed using DWT and the discharge values were estimated directly. The results indicate that this blending approach yields improved outcomes compared to the previous two approaches, where targets were decomposed and multi-output modeling was employed. One possible reason for this improvement is that multi-output modeling can introduce complexity, requiring careful consideration of the inter-relationships between different output variables. This complexity can sometimes lead to errors and reduced prediction accuracy. In contrast, by solely utilizing DWT to decompose the input data, the models may be better able to capture essential information without sacrificing detail or complexity. Estimating discharge values directly also eliminates the need to model the inter-relationships between different output variables, simplifying the model and potentially enhancing accuracy. Furthermore, the results demonstrate that LSTM outperforms CNN in discharge modeling. LSTMs are specifically designed to process sequential data, making them well-suited for modeling discharge as it represents a time-series dataset with temporal dependencies. In comparison, CNNs are more suitable for tasks involving image processing, where spatial correlations are more significant. LSTMs incorporate memory units that allow them to retain information from previous inputs and utilize this to make better predictions. This ability was especially valuable when modeling discharge since there might be lasting connections between input variables that impact the output. This is a common characteristic of hydrological time series. In contrast, CNNs can only learn local patterns within a fixed window size and lack this capability. Overall, LSTMs offer greater flexibility than CNNs, as they can model both short- and long-term dependencies, making them well-suited for discharge modeling, where the relationship between input variables and discharge can vary across different time scales.

To compare the performance of different blending approaches, the percentage improvement/decline of the metrics for the third blending approach (the best one) compared to the second blending approach (the worst one) was calculated.

Tables 3 and 4 showcase the relative improvement or decline in the performance of the third blending approach compared to the second blending approach for both the DWT-LSTM and DWT-CNN models. In terms of the DWT-LSTM model, improvements in R^2 were observed, indicating that the second blending approach was outperformed by the third blending approach in capturing the variance of the target variable. Improvement ranged from 0.30% to 3.6%, signifying increased accuracy in modeling the discharge for different basins. Decrease in RMSE and NRMSE were observed for the third blending approach compared to the second one. These results demonstrate enhanced accuracy and reduced error in discharge prediction. Improvements ranged from 33.7% to 55.9% for RMSE and 33.9–55.8% for NRMSE, highlighting the superiority of the third approach in terms of accuracy. Furthermore, improvements ranging from 8.0% to 10.6% were noted for the Nash-Sutcliffe efficiency coefficient, indicating that the observed discharge trend and variability were better captured by the third blending approach compared to the second blending approach. Similarly, for the DWT-CNN model, improvements in R^2 ranging from 1.4% to 3.9% were observed, indicating that the second blending approach was surpassed by the third blending approach in explaining the variance of the target variable. Lower RMSE and NRMSE were also observed for the third blending approach, suggesting improved accuracy in discharge prediction. Improvements ranged from 11.1% to 51.1% for RMSE and 11.0–51.1% for NRMSE, highlighting the enhanced performance of the third approach compared to the second approach. The third blending approach showed significant improvements in capturing the observed discharge trend and variability, with enhancements ranging from 2.4% to 10.5% for the Nash-Sutcliffe efficiency coefficient. This means that the third blending approach was better at accurately modeling discharge compared to the second blending approach. Both the DWT-LSTM and DWT-CNN models benefited from the third blending approach, achieving better results in terms of R^2, RMSE, NRMSE, and Nash-Sutcliffe efficiency coefficient.

As mentioned before, dynamic time warping (DTW) was utilized as a method for clustering time series data, with the goal of identifying patterns that can help to separate data into distinct clusters. The results of using DTW to cluster the data into two groups are presented in Fig. 10, with cluster 1 represented by red circles and cluster 2 represented by blue diamonds. This method involves employing different models for different subsets of the data. For instance, one model can be used to describe the behavior of the time series during periods of high flow, while another model can be used for periods of low flow. This technique is particularly useful when the behavior of the time series varies significantly depending on external factors such as weather patterns or seasonal influence. DTW provides a convenient means of separating the data into high and low flow clusters, allowing for the creation of separate models for each.

Figs. 11 and 12 demonstrate the performance of LSTM and CNN models using the temporal cluster-based local modeling approach to predict discharge. The results are comparable to the best results achieved by the DWT-deep learning models. This highlights the effectiveness of the temporal local modeling approach for accurate discharge prediction. Furthermore, creating separate models for different clusters facilitate better estimations, particularly for high flows. This is a critical factor as accurate prediction of high flow events is essential for flood management and inundation prevention. The ability of the temporal local modeling approach to capture the inherent characteristics of discharge data is noteworthy. By considering the temporal dynamics of the data, the models can effectively capture the complex relationships among the inputs and predict discharge with high accuracy. Clustering the data into separate groups enables the development of models that are tailored to each cluster's unique characteristics. This results in models that are better at predicting discharge, particularly during

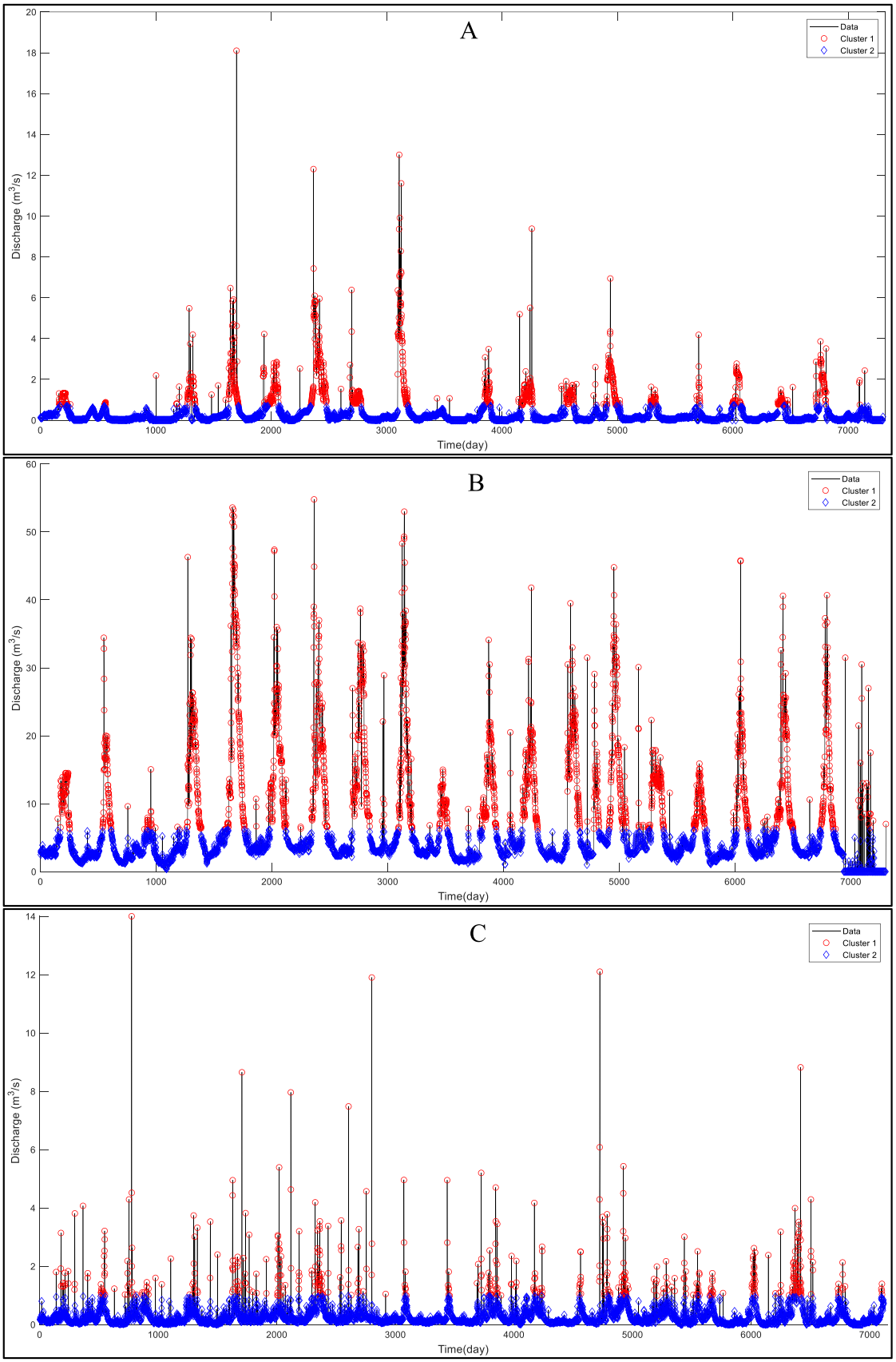

Fig. 10. Time series clustering using DTW A) Bar-Erieh, B) Latian Dam, and C) Kasilian Basins.

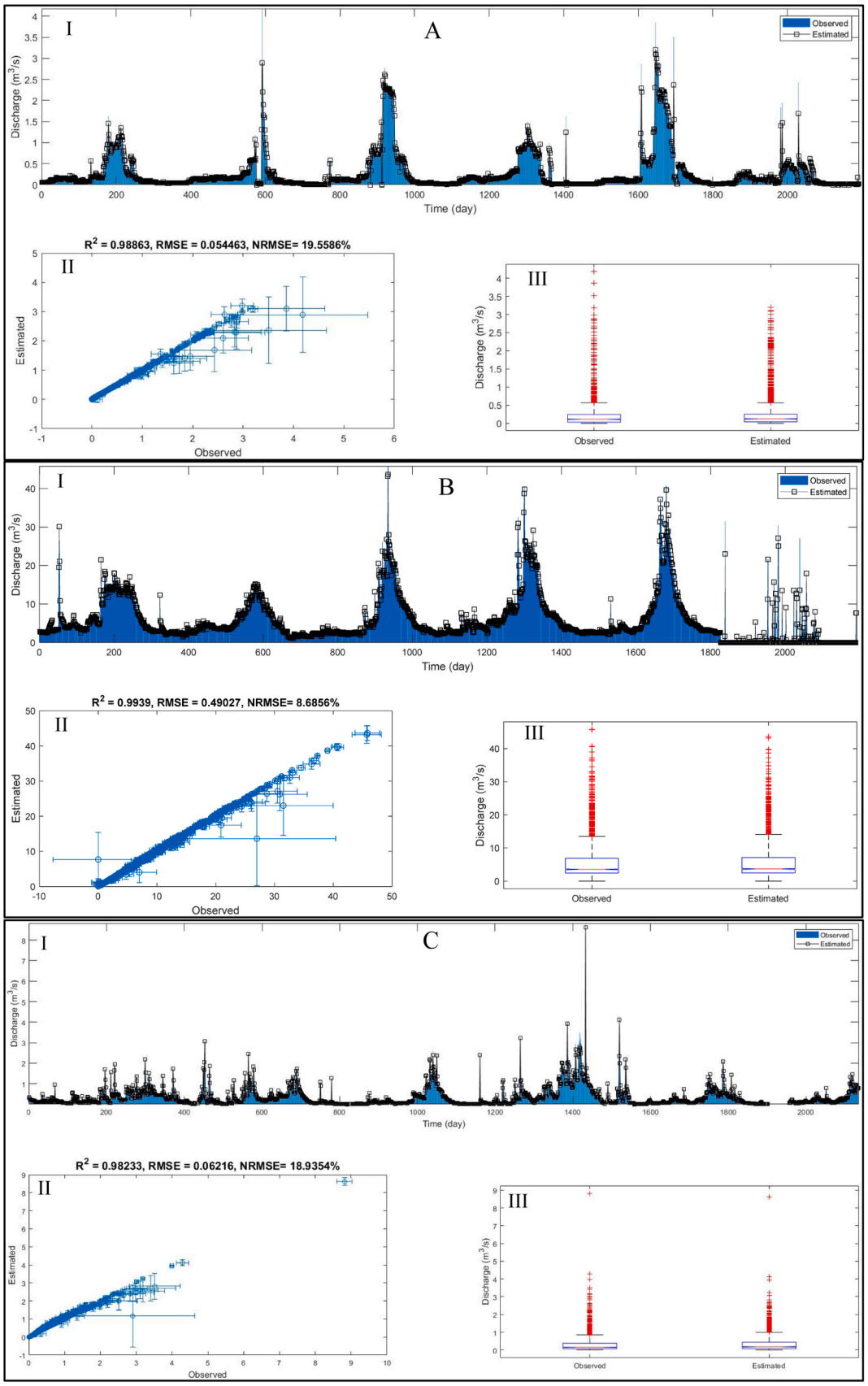

Fig. 11. Temporal Cluster-Based Local LSTM model. A) Bar-Erieh, B) Latian Dam, and C) Kasilian Basins.

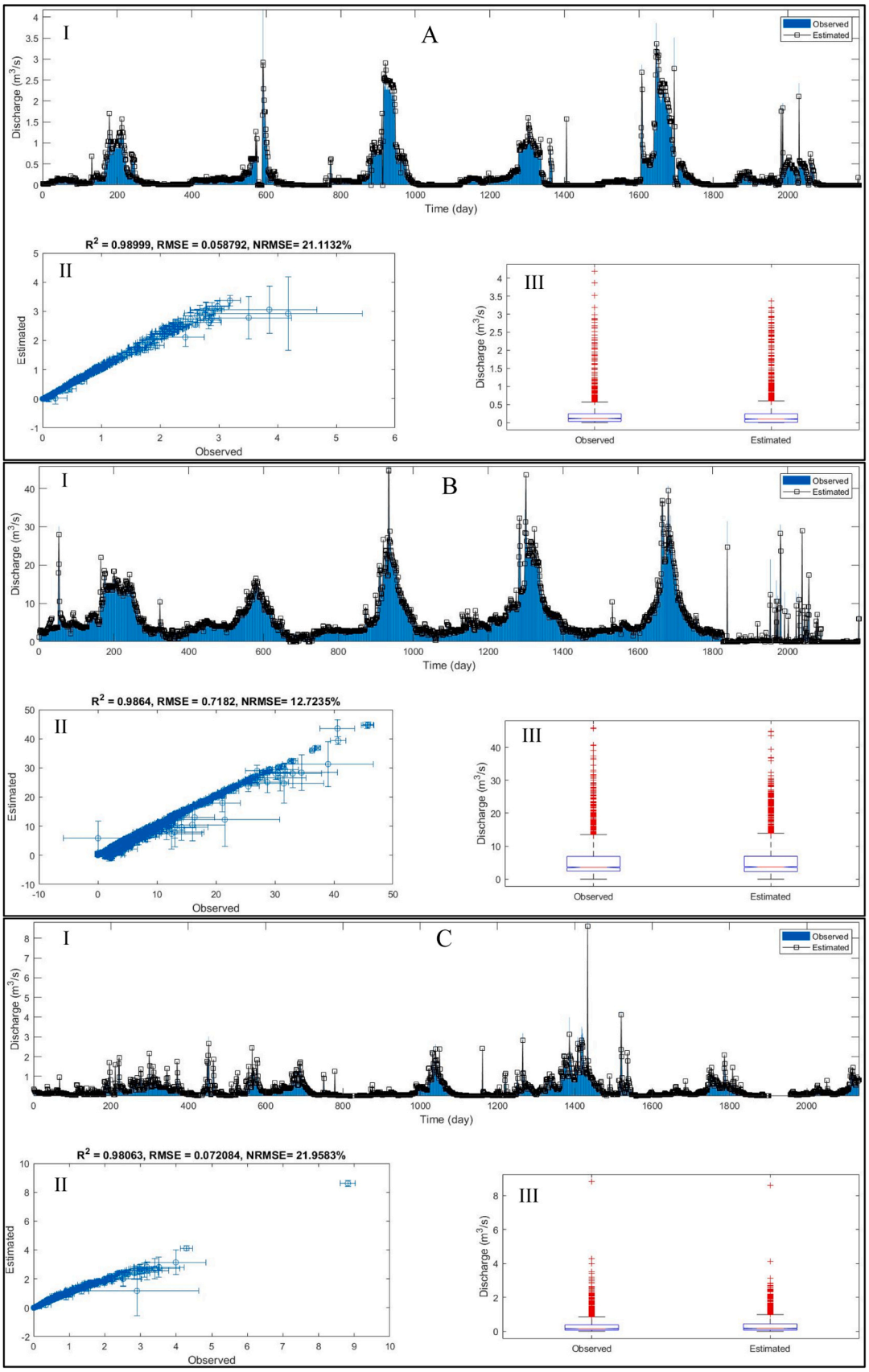

Fig. 12. Temporal Cluster-Based Local CNN model. A) Bar-Erieh, B) Latian Dam, and C) Kasilian Basins.

Table 5
Relative improvement/decline for the Temporal Cluster-Based Local compared to the DWT-LSTM as the best hybrid model.

Basin	R^2 improvement/ decline (%)	RMSE improvement/ decline (%)	NRMSE improvement/ decline (%)	Nash- Sutcliffe improvement/ decline (%)
Bar-Erieh	-0.101	-7.54	-7.50	-1.07
Latian Dam	0.202	17.55	17.60	1.05
Kasilian	-0.306	7.40	7.82	2.22

Table 6
Accuracy assessment metrics for Train and Test phases for temporal cluster-based local model.

Basin	Train/Test	R^2	RMSE	NRMSE	Nash- Sutcliffe
Bar-Erieh	Train	0.98	0.050	18.21	0.94
	Test	0.98	0.054	19.55	0.93
Latian Dam	Train	0.99	0.413	7.32	0.96
	Test	0.99	0.49	8.68	0.95
Kasilian	Train	0.99	0.055	16.80	0.93
	Test	0.98	0.062	18.93	0.90

extreme events, where accurate forecasting is critical. The clustering approach is beneficial for identifying patterns and trends in the data, which helps in understanding the factors influencing discharge at a particular time. Instead of creating a single global model in conjunction with signal processing approaches, developing separate models for each cluster can be computationally efficient, especially when dealing with large datasets. This approach helps save computational resources and reduces the time required for model development, validation, and testing.

To compare the performance of DWT-deep learning and Temporal Cluster-Based Local modeling approaches, the percentage improvement or decline of metrics for the Temporal Cluster-Based Local model, compared to the DWT-LSTM model (blended by the third approach), as the best hybrid model, was computed. Table 5 shows the results for the three basins.

For the Bar-Erieh basin, a slight deterioration in R^2, RMSE, NRMSE, and Nash-Sutcliffe efficiency was observed when the Temporal Cluster-Based Local model was used compared to the DWT-LSTM model. This suggests that the DWT-LSTM model was slightly better in capturing the variance and accurately predicting discharge for this particular basin. In the case of Latian Dam basin, improvements in R^2, RMSE, NRMSE, and Nash-Sutcliffe efficiency were observed with the Temporal Cluster-Based Local model. This indicates that the Temporal Cluster-Based Local model outperformed the DWT-LSTM model in terms of explaining the variance and achieving better accuracy in discharge prediction for this basin. Similarly, for the Kasilian basin, a decline in R^2 was observed, but improvements in RMSE, NRMSE, and Nash-Sutcliffe efficiency were seen with the Temporal Cluster-Based Local model compared to the DWT-LSTM model. This implies that while the Temporal Cluster-Based Local model may have slightly lower performance in capturing the variance, it achieved better accuracy and improved the overall trend and variability of the observed discharge for this basin. In summary, the results in Table 5 demonstrate that the performance of the Temporal Cluster-Based Local model compared to the DWT-LSTM model varied across different basins.

The results presented in Table 6 demonstrate that the model did not suffer from overfitting across all three basins. To ensure the model's reliability, various evaluation metrics were used, showing strong performance and consistency between the training and test datasets. This suggests that the model is capable of effectively generalizing to new, unseen data in all three basins. In order to address potential overfitting, Elastic Net regularization was implemented. This technique combines

Table 7
The results of DWT-TCT model for the three study areas.

Basin	R^2	RMSE	NRMSE (%)	Nash- Sutcliffe
Bar-Erieh	0.97	0.044	16.23	0.94
Latian Dam	0.98	0.386	6.85	0.96
Kasilian	0.98	0.046	14.21	0.94

L1 (Lasso) and L2 (Ridge) regularization methods to strike a balance between feature selection and feature shrinkage. Two penalty terms were added to the loss function during training. The L1 penalty term encouraged sparsity in the model by penalizing features with negligible or zero weights, aiding in feature selection and reducing the number of parameters. The L2 penalty term promoted smaller weights by penalizing the square of the weight magnitude, preventing individual weights from becoming excessively large and facilitating feature shrinkage. Elastic Net regularization achieves a balance between sparsity and shrinkage by combining L1 and L2 regularization. It effectively selects features and prevents overfitting. The trade-off between L1 and L2 penalties was controlled by a hyper-parameter, with higher values favoring sparsity and lower values favoring shrinkage. The model's convergence and trapping behavior were also assessed by monitoring R^2, RMSE, NRMSE, and Nash- Sutcliffe values at each iteration. Initially, the R^2 and Nash- Sutcliffe values were low but gradually increased over time, indicating an improvement in the model's performance. Conversely, the RMSE and NRMSE values were relatively high at the beginning but steadily decreased with each iteration, implying that the model's predictions became more accurate. Upon analyzing the R^2, RMSE, NRMSE, and Nash- Sutcliffe values at each iteration, there were no indications of early convergence or trapping behavior. Instead, the model exhibited a gradual improvement in performance, with higher R^2 values, lower RMSE and NRMSE values, and an increased Nash- Sutcliffe value. This indicates that the model continuously refined its predictions and approached a more precise representation of the relationships between the variables. Table 7 illustrates the outcomes of the DWT-TCT model across three distinct study areas. The findings clearly demonstrate the superior performance of this model in comparison to previously proposed models. Several reasons contribute to the DWT-TCT model's superior performance. The TCT model leverages the power of temporal convolutions, which allow it to capture and model temporal dependencies in the data. By considering the sequential nature of the time series, the TCT model can effectively capture dependencies and intricate temporal patterns, leading to enhanced predictive performance. It also incorporates the transformer mechanism, which utilizes self-attention mechanisms to capture global dependencies within the time series data. This mechanism enables the model to effectively weigh the importance of different time steps, allowing it to focus on relevant temporal information and ignore irrelevant noise, resulting in improved prediction accuracy. The TCT model employs multi-head attention, which enables it to capture diverse patterns and dependencies at various resolutions. This capability allows the model to effectively extract and utilize both local and global temporal features, leading to a more comprehensive understanding of the data and improved predictive performance.

Fig. 13 presents a detailed evaluation of the performance of the optimization algorithms on the top-performing models, namely the DWT-LSTM/Third blending approach, Temporal Cluster-Based Local LSTM, and DWT-TCT, across the three study areas. The ranking of optimization algorithms based on their effectiveness in enhancing model performance was determined to be as follows: GWO ranked highest, followed by GA (with very similar performance), ACO, and then PSO. According to the figure, both GA and GWO algorithms exhibited very similar performance, indicating their suitability for tuning deep learning model parameters effectively. They are known for their ability to strike a good balance between exploration and exploitation. They exhibit strong exploration capabilities, allowing them to efficiently search the solution

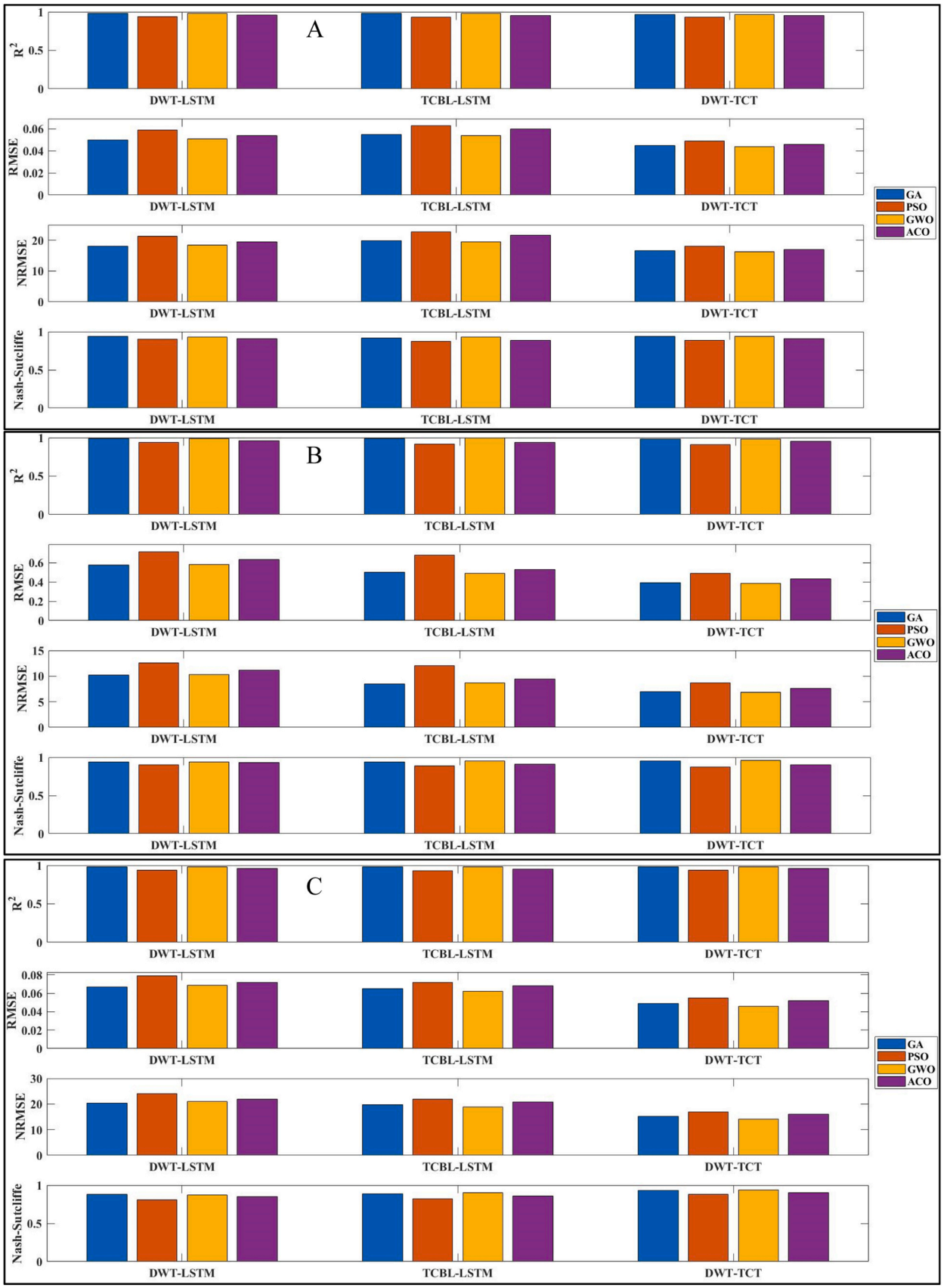

Fig. 13. A comprehensive comparison of the performance of the proposed optimization algorithms on the best models, A) Bar-Erieh, B) Latian Dam, and C) Kasilian Basins.

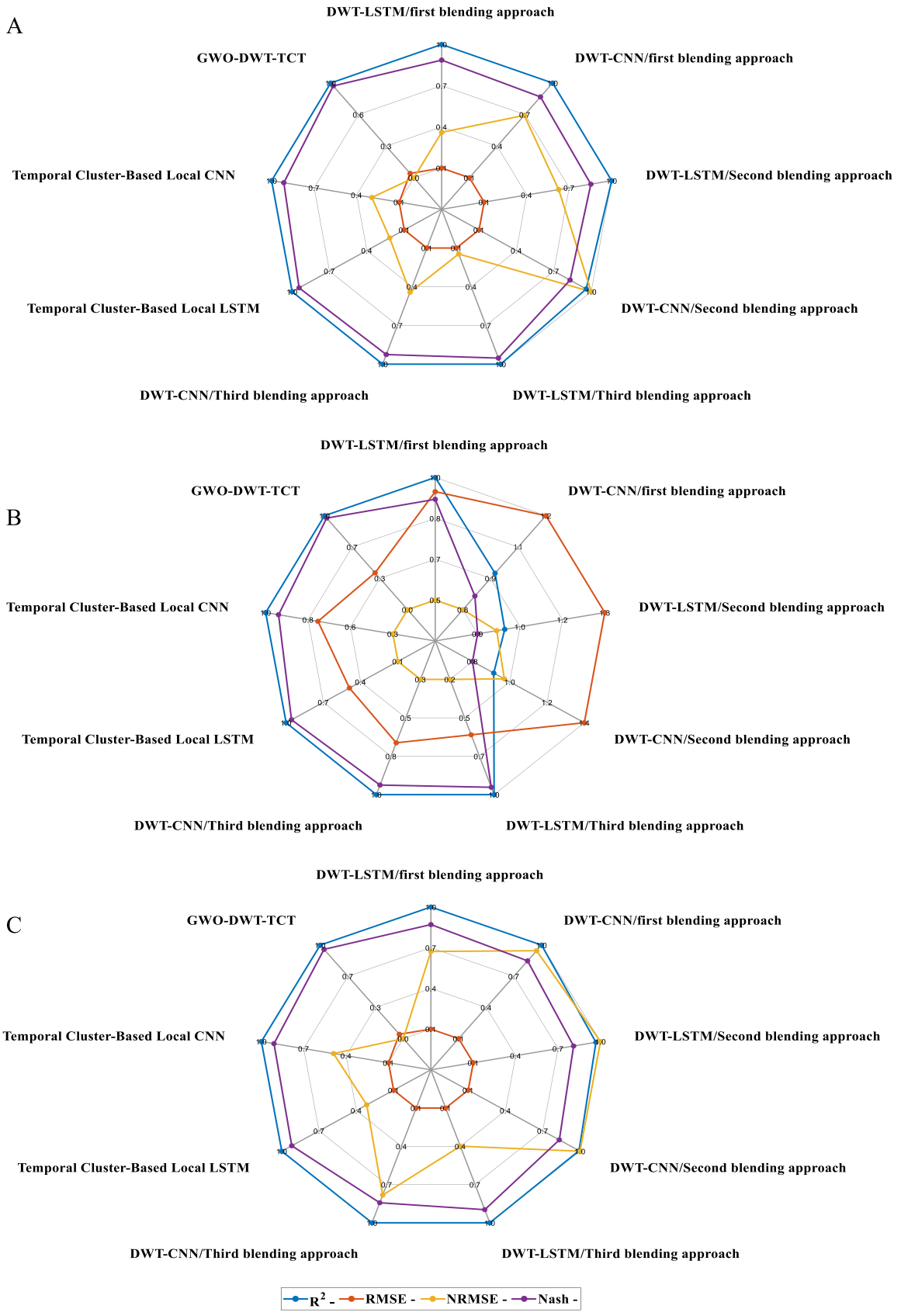

Fig. 14. A comprehensive comparison of the performance of the proposed models, A) Bar-Erieh, B) Latian Dam, and C) Kasilian Basins.

space and discover promising regions. Additionally, they effectively exploit the discovered regions to refine the solutions. This balance in GA and GWO may have allowed them to effectively search for optimal structural parameters in the models, leading to better performance compared to PSO and ACO. Each optimization algorithm has its own strengths and weaknesses, and their suitability may vary depending on the characteristics of the problem being solved. It is possible that the GWO and GA algorithms are well-suited to the specific optimization requirements of the models being evaluated in this study, allowing them to outperform the other algorithms. GWO is also known for its ability to maintain a diverse set of solutions during the optimization process. This diversity helps prevent the algorithm from getting stuck in local optima and provides a higher chance of finding better solutions. This characteristic might have contributed to its ability to optimize the models' performance effectively. Also GA and GWO algorithms are known for their ability to handle local optima. GA employs genetic operators such as crossover and mutation, which provide mechanisms for escaping local optima and exploring new regions of the search space. GWO, inspired by the hunting behavior of grey wolves, utilizes different update equations to balance exploration and exploitation, enabling it to avoid getting trapped in suboptimal solutions. These characteristics may allow GA and GWO to find better solutions in optimizing the structural parameters compared to the PSO and ACO algorithms.

Fig. 14 presents a detailed analysis as a spider plot, showcasing a comprehensive performance comparison of the proposed models. To ensure consistency across metrics, the NRMSE range was standardized between different models, scaling it to a range from 0–1. The spider plot reveals that the R^2 values among the models were not significantly different. However, the remaining metrics, such as RMSE, NRMSE, and Nash- Sutcliffe, demonstrated varying values, indicating diverse levels of accuracy across the models. In this context, similar R^2 values indicate that the models are generally consistent in their ability to explain the variance in the data. However, when it comes to the other evaluation metrics, such as RMSE, NRMSE, and Nash- Sutcliffe, distinct disparities emerge. These metrics provide insights into different aspects of model accuracy and performance. The divergent values obtained for RMSE, NRMSE, and Nash- Sutcliffe across the models indicate discrepancies in their respective accuracies. This information becomes crucial in selecting the most suitable model based on the specific objectives and requirements of the study at hand.

The results showed that the proposed DWT-TCT model outperformed both local temporal cluster-based and DWT-deep learning models in daily runoff prediction. It may be due to some reasons. The DWT-TCT model combines the wavelet transform and the Temporal Convolutional Transformer (TCT) architecture. By incorporating the wavelet transform, the model can capture both the frequency information and temporal dependencies present in the daily runoff data. This ability to capture both frequency and temporal patterns can lead to improved prediction accuracy compared to models that focus solely on one aspect. The TCT component of the model also enables the capture of both global and local temporal dependencies. The transformer layers in the TCT model utilize self-attention mechanisms to capture long-range dependencies across the entire sequence. At the same time, the convolutional layers capture local dependencies and short-term patterns. This combination enables the model to effectively capture and utilize both types of temporal relationships, leading to improved prediction accuracy. The ability to capture dependencies across the entire sequence helps the TCT model better understand the relationships between past and future runoff values, potentially leading to improved prediction accuracy. The convolutional layers can identify relevant and discriminative features at different temporal scales, enabling the model to capture important characteristics of the daily runoff time series. By effectively extracting informative features, the TCT model may have a better representation of the underlying patterns in the data, leading to improved prediction performance. The TCT model can perform multi-scale analysis of the input data. The temporal convolutional layers with different kernel sizes allow the model to capture patterns at various temporal resolutions. This multi-scale analysis helps the TCT model capture and leverage information at different levels of granularity, potentially improving its ability to capture complex temporal patterns and variations in the daily runoff data. The transformer layers in the TCT model utilize self-attention mechanisms, allowing the model to assign different weights to different time steps based on their relevance for prediction. This attention mechanism helps the model focus on the most informative parts of the input sequence, giving more importance to relevant time steps and potentially reducing the impact of noisy or irrelevant data. By attending to the most relevant information, the TCT model can make more accurate predictions.

Table 8
R Factor values calculated for model type selection and input variable selection.

	Model Type Selection	Input Variable Selection
R Factor	0.21	0.15

Table 8 shows the R factors calculated for both model type selection and input variable selection. The results showed an R factor of 0.21 for model type selection and 0.15 for input variable selection. For model type selection, an R factor of 0.21 suggests that there is a moderate level of uncertainty associated with choosing the appropriate model type. This means that there are some variations or discrepancies between the predicted values generated by different models considered in the study. However, the relatively low R factor indicates that the models produced reasonably consistent results with a certain degree of agreement. For input variable selection, an R factor of 0.15 indicates a lower level of uncertainty regarding the choice of input variables. This implies that the predicted data generated by various combinations of input variables using a single model showed relatively little variation or inconsistency. The low R factor values indicate a relatively low level of uncertainty in the predictions, demonstrating the effectiveness and reliability of the chosen models and input variables. The results suggest that the selection of model types might have a more significant impact on reducing uncertainty in the analysis compared to selecting the input variables. It could imply that there might be a wider variation in predicted values among different model types using the same set of input variables, leading to the higher uncertainty.

Fig. 15 elucidates the findings derived from the implementation of the cosine amplitude method, which facilitated the determination of weights assigned to various input variables. Akin to assigning importance levels, these weights delineate the relative significance of each input factor in describing the output of the process at hand. To provide contextual clarification, the abbreviations P, T, ET, P, D, WS, and H denote precipitation, temperature, evapotranspiration, discharge, wind speed, and humidity, respectively. The inclusion of the variable index, denoted as t, links the respective measurements to specific points in time. For instance, P(t-1) signifies the precipitation value recorded for the day immediately preceding the reference time. Through the analysis, it was revealed that precipitation and discharge played the most pivotal role in accurately forecasting runoff.

4. Conclusions

This study explored different data decomposition and modeling methods for improving nonstationary discharge prediction, with significant implications for hydrology and water resources management, particularly in the context of climate change and withering stationarity. The performance of three different signal processing and deep learning blending approaches were tested for daily discharge modeling. The results showed that a single output DWT-deep learning model provided acceptable results, while multi-output models had moderate to weak performance. On the other hand, the temporal local modeling approach, which involved clustering the data into separate groups and developing tailored models for each cluster, proved to be a successful approach for

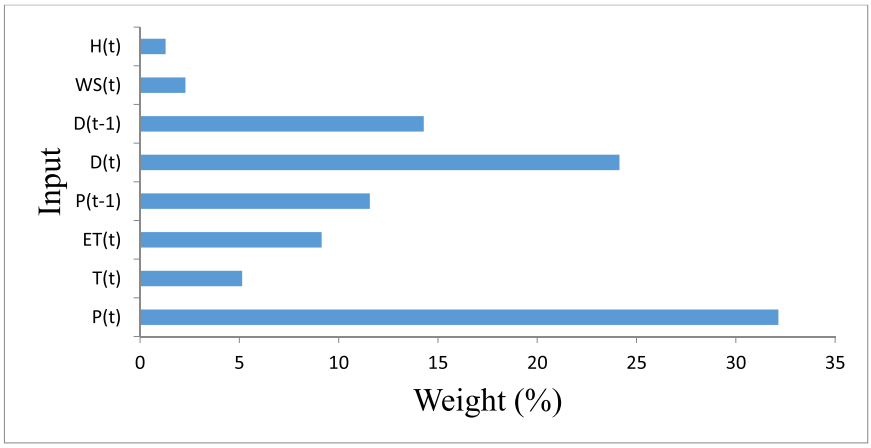

Fig. 15. The calculated sensitivity indices of input variables on predicted runoff.

accurate discharge prediction. This approach can help identify patterns and trends within the data, making it easier to understand the underlying factors driving discharge at a specific location. Additionally, it can reduce computational resources and time required for model development, validation, and testing rather than hybrid signal processing- deep learning models. The promising results of the temporal local modeling approach provide a beneficial alternative to hybrid signal processing-deep learning models. This approach has significant potential for improving the accuracy of discharge predictions and advancing our understanding of non-stationary hydrological processes. Moreover, the fusion of discrete wavelet transform and Temporal Convolutional Transformer yielded superior performance compared to other modeling approaches, making it a formidable technique for predicting runoff at a daily time step. It is important to note that the selection of an appropriate modeling approach should be based on careful consideration of the nature of the data and the specific objectives of the study. While hybrid signal processing-deep learning models may offer advantages in certain cases, they require extensive tuning of parameters and can be computationally intensive. It is important to acknowledge the limitations of this study, including the fact that the time scale of the study was daily and should be tested for different time scales, such as monthly or seasonal. Additionally, while the temporal local modeling approach showed promising results, it may not be suitable for all types of data. The study did not consider the effects of external factors, such as land use changes and human activities, on discharge prediction. Therefore, future studies could consider incorporating these external factors into the modeling process to improve the accuracy of discharge prediction. Furthermore, the study only evaluated the performance of three deep learning methods, and future studies could explore the performance of other deep learning methods and compare them with the proposed models.

CRediT authorship contribution statement

Sahar Mostafaei: Visualization, Software, Resources, Data curation. **Ronny Berndtsson:** Writing – review & editing, Validation, Formal analysis. **Vahid Moosavi:** Writing – original draft, Validation, Supervision, Software, Methodology.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] J. Seibert, S. Uhlenbrook, C. Leibundgut, S. Halldin, Multiscale calibration and validation of a conceptual rainfall-runoff model, Physics and Chemistry of the Earth, Part B: Hydrology, Oceans Atmos. 25 (2000) 59–64.
[2] X. Min, B. Hao, Y. Sheng, Y. Huang, J. Qin, Transfer performance of gated recurrent unit model for runoff prediction based on the comprehensive spatiotemporal similarity of catchments, J. Environ. Manag. 330 (2023) 117182.
[3] X. Yuan, C. Chen, X. Lei, Y. Yuan, R. Muhammad, Adnan, Monthly runoff forecasting based on LSTM–ALO model, Stoch. Environ. Res. Risk Assess. 32 (2018) 2199–2212.
[4] S.E. Priestly, K. Raimond, Y. Cohen, J. Brema, D.J. Hemanth, Evaluation of a novel hybrid lion swarm optimization – AdaBoostRegressor model for forecasting monthly precipitation, Sustain. Comput.: Inform. Syst. 39 (2023) 100884.
[5] K. Przybył, K. Koszela, Applications MLP and other methods in artificial intelligence of fruit and vegetable in convective and spray drying, Appl. Sci. 13 (2023) 2965.
[6] R.M. Adnan, R.R. Mostafa, O. Kisi, Z.M. Yaseen, S. Shahid, M. Zounemat-Kermani, Improving streamflow prediction using a new hybrid ELM model combined with hybrid particle swarm optimization and grey wolf optimization, Knowl. -Based Syst. 230 (2021) 107379.
[7] R.M. Adnan, R.R. Mostafa, O. Kisi, Z.M. Yaseen, S. Shahid, M. Zounemat-Kermani, Improving streamflow prediction using a new hybrid ELM model combined with hybrid particle swarm optimization and grey wolf optimization, Knowl. -Based Syst. 230 (2021) 19.
[8] S. Ding, Z. Zhang, L. Guo, Y. Sun, An optimized twin support vector regression algorithm enhanced by ensemble empirical mode decomposition and gated recurrent unit, Inf. Sci. 598 (2022) 101–125.
[9] V. Moosavi, A. Karami, R. Aliramaee, High-resolution soil moisture mapping using PSO-based optimized cerebellar model articulation controller (CMAC), Sci. Total Environ. 857 (2023) 159493.
[10] G. Liu, Z. Tang, H. Qin, S. Liu, Q. Shen, Y. Qu, J. Zhou, Short-term runoff prediction using deep learning multi-dimensional ensemble method, J. Hydrol. 609 (2022) 127762.
[11] H. Han, R.R. Morrison, Improved runoff forecasting performance through error predictions using a deep-learning approach, J. Hydrol. 608 (2022) 127653.
[12] A. Wunsch, T. Liesch, G. Cinkus, N. Ravbar, Z. Chen, N. Mazzilli, H. Jourde, N. Goldscheider, Karst spring discharge modeling based on deep learning using spatially distributed input data, Hydrol. Earth Syst. Sci. 26 (2022) 2405–2430.
[13] Y. Xie, W. Sun, M. Ren, S. Chen, Z. Huang, X. Pan, Stacking ensemble learning models for daily runoff prediction using 1D and 2D CNNs, Expert Syst. Appl. 217 (2023) 119469.
[14] J. Zhang, H. Yan, A long short-term components neural network model with data augmentation for daily runoff forecasting, J. Hydrol. 617 (2023) 128853.
[15] B. Li, R. Li, T. Sun, A. Gong, F. Tian, M.Y.A. Khan, G. Ni, Improving LSTM hydrological modeling with spatiotemporal deep learning and multi-task learning: A case study of three mountainous areas on the Tibetan Plateau, J. Hydrol. 620 (2023) 129401.
[16] H. Deng, W. Chen, G. Huang, Deep insight into daily runoff forecasting based on a CNN-LSTM model, Nat. Hazards 113 (2022) 1675–1696.
[17] D.K. Ghose, V. Mahakur, A. Sahoo. Monthly Runoff Prediction by Hybrid CNN-LSTM Model: A Case Study, Springer International Publishing, Cham, 2022, pp. 381–392. Advances in Computing and Data SciencesM.SinghV.TyagiP.K. GuptaJ.FlusserT.Ören.
[18] T.Z. Addis, K.S. Kasiviswanathan, Chapter 3 - Nonstationarity analyses of design rainfall using Bayesian approaches, in: K.S. Kasiviswanathan, B. Soundharajan, S. Patidar, J. He, C.S.P. Ojha (Eds.), Developments in Environmental Science, Elsevier, 2023, pp. 31–56.
[19] Y. Gao, J. Xia, X. Chen, L. Zou, J. Huang, J. Yu, Analysis of the nonstationarity characteristics and future trends of flood extremes in the Dongting Lake Basin, J. Hydrol.: Reg. Stud. 44 (2022) 101217.

[20] H. Wang, T. Wu, A. Hilbert-wavelet-based, nonstationarity index for multi-level quantification of extreme winds, J. Wind Eng. Ind. Aerodyn. 215 (2021) 104682.
[21] A. Prokoph, F. Barthelmes, Detection of nonstationarities in geological time series: Wavelet transform of chaotic and cyclic sequences, Comput. Geosci. 22 (1996) 1097–1108.
[22] M. Alizamir, J. Shiri, A.F. Fard, S. Kim, A.D. Gorgij, S. Heddam, V.P. Singh, Improving the accuracy of daily solar radiation prediction by climatic data using an efficient hybrid deep learning model: long short-term memory (LSTM) network coupled with wavelet transform, Eng. Appl. Artif. Intell. 123 (2023) 106199.
[23] V. Moosavi, M. Vafakhah, B. Shirmohammadi, N. Behnia, A. Wavelet-ANFIS, Hybrid model for groundwater level forecasting for different prediction periods, Water Resour. Manag. 27 (2013) 1301–1321.
[24] B. Shirmohammadi, H. Moradi, V. Moosavi, M.T. Semiromi, A. Zeinali, Forecasting of meteorological drought using Wavelet-ANFIS hybrid model for different time steps (case study: southeastern part of east Azerbaijan province, Iran), Nat. Hazards 69 (2013) 389–402.
[25] S. Zhu, J. Wei, H. Zhang, Y. Xu, H. Qin, Spatiotemporal deep learning rainfall-runoff forecasting combined with remote sensing precipitation products in large scale basins, J. Hydrol. 616 (2023) 128727.
[26] V. Moosavi, A. Talebi, M.R. Hadian, Development of a hybrid wavelet packet-group method of data handling (WPGMDH) model for runoff forecasting, Water Resour. Manag. 31 (2017) 43–59.
[27] S. Mishra, C. Saravanan, V.K. Dwivedi, J.P. Shukla, Rainfall-runoff modeling using clustering and regression analysis for the River Brahmaputra Basin, J. Geol. Soc. India 92 (2018) 305–312.
[28] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Trans. Neural Netw. 5 (1994) 157–166.
[29] S. Hochreiter, The vanishing gradient problem during learning recurrent neural nets and problem solutions, Int. J. Uncertain., Fuzziness Knowl. -Based Syst. 06 (1998) 107–116.
[30] X. Yuan, C. Chen, M. Jiang, Y. Yuan, Prediction interval of wind power using parameter optimized Beta distribution based LSTM model, Appl. Soft Comput. 82 (2019) 105550.
[31] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.
[32] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (1997) 1735–1780.
[33] F. Kratzert, D. Klotz, C. Brenner, K. Schulz, M. Herrnegger, Rainfall–runoff modelling using Long Short-Term Memory (LSTM) networks, Hydrol. Earth Syst. Sci. 22 (2018) 6005–6022.
[34] S.-Y. Huang, W.-J. An, D.-S. Zhang, N.-R. Zhou, Image classification and adversarial robustness analysis based on hybrid quantum–classical convolutional neural network, Opt. Commun. 533 (2023) 129287.
[35] M. Kloska, G. Grmanova, V. Rozinajova, Expert enhanced dynamic time warping based anomaly detection, Expert Syst. Appl. 225 (2023) 120030.
[36] M. Herrmann, G.I. Webb, Amercing: an intuitive and effective constraint for dynamic time warping, Pattern Recognit. 137 (2023) 109333.
[37] R. Casal, L.E. Di Persia, G. Schlotthauer, Temporal convolutional networks and transformers for classifying the sleep stage in awake or asleep using pulse oximetry signals, J. Comput. Sci. 59 (2022) 101544.
[38] H. Peng, B. Jiang, Z. Mao, S. Liu, Local Enhancing Transformer With Temporal Convolutional Attention Mechanism for Bearings Remaining Useful Life Prediction, IEEE Trans. Instrum. Meas. 72 (2023) 1–12.
[39] A. Sharafati, S.B. Haji Seyed Asadollah, D. Motta, Z.M. Yaseen, Application of newly developed ensemble machine learning models for daily suspended sediment load prediction and related uncertainty analysis, Hydrol. Sci. J. 65 (2020) 2022–2042.
[40] R. Asheghi, S.A. Hosseini, M. Saneie, A.A. Shahri, Updating the neural network sediment load models using different sensitivity analysis methods: a regional application, J. Hydroinformatics 22 (2020) 562–577.
[41] J. Guo, C. Chen, H. Wen, G. Cai, Y. Liu, Prediction model of goaf coal temperature based on PSO-GRU deep neural network, Case Stud. Therm. Eng. 53 (2024) 103813.
[42] M. Zanganeh, A. Chaji, A new aspect of the ApEn application to improve the PSO-ANFIS model to forecast Caspian sea levels, Reg. Stud. Mar. Sci. (2023) 103347.
[43] J. Liang, Y. Du, Y. Xu, B. Xie, W. Li, Z. Lu, R. Li, H. Bal, Using Adaptive Chaotic Grey Wolf Optimization for the daily streamflow prediction, Expert Syst. Appl. 237 (2024) 121113.
[44] Y. Wang, S. Ran, G.-G. Wang, Role-oriented binary grey wolf optimizer using foraging-following and Lévy flight for feature selection, Appl. Math. Model. 126 (2024) 310–326.
[45] A. Jayaprakash, C. KeziSelvaVijila, Feature selection using ant colony optimization (ACO) and road sign detection and recognition (RSDR) system, Cogn. Syst. Res. 58 (2019) 123–133.
[46] H.R. Kanan, K. Faez, An improved feature selection method based on ant colony optimization (ACO) evaluated on face recognition system, Appl. Math. Comput. 205 (2008) 716–725.
[47] M.D. Toksari, A hybrid algorithm of ant colony optimization (ACO) and iterated local search (ils) for estimating electricity domestic consumption: case of Turkey, Int. J. Electr. Power Energy Syst. 78 (2016) 776–782.