


Physics-enhanced machine learning models for streamflow discharge forecasting

Ying Zhao^{a,†}, Mayank Chadha^{b,†}, Dakota Barthlow^c, Elissa Yeates^d, Charles J. Mcknight^d,
Natalie P. Memarsadeghi^d, Guga Gugaratshan^c, Michael D. Todd^b and Zhen Hu ^{a,*}

^a Department of Industrial and Manufacturing Systems Engineering, University of Michigan-Dearborn, Dearborn, MI 48128, USA

^b Department of Structural Engineering, University of California San Diego, La Jolla, CA 92093, USA

^c Hottinger Bruel & Kjaer Solutions LLC, Southfield, MI, 48076, USA

^d Coastal and Hydraulics Laboratory, Engineer Research and Development Center, US Army Corps of Engineers, Vicksburg, MS, 39180, USA

*Corresponding author. E-mail: zhennhu@umich.edu

[†]These authors contributed equally.

 ZH, 0000-0003-1661-515X

ABSTRACT

Accurate river discharge forecasts for short to intermediate time intervals are crucial for decision-making related to flood mitigation, the seamless operation of inland waterways management, and optimal dredging. River routing models that are physics based, such as RAPID ('routing application for parallel computation of discharge') or its variants, are used to forecast river discharge. These physics-based models make numerous assumptions, including linear process modeling, accounting for only adjacent river inflows, and requiring brute force calibration of hydrological input parameters. As a consequence of these assumptions and the missing information that describes the complex dynamics of rivers and their interaction with hydrology and topography, RAPID leads to noisy forecasts that may, at times, substantially deviate from the true gauged values. In this article, we propose hybrid river discharge forecast models that integrate physics-based RAPID simulation model with advanced data-driven machine learning (ML) models. They leverage runoff data of the watershed in the entire basin, consider the physics-based RAPID model, take into account the variability in predictions made by the physics-based model relative to the true gauged discharge values, and are built on state-of-the-art ML models with different complexities. We deploy two different algorithms to build these hybrid models, namely, delta learning and data augmentation. The results of a case study indicate that a hybrid model for discharge predictions outperforms RAPID in terms of overall performance. The prediction accuracy for various rivers in the case study can be improved by a factor of four to seven.

Key words: hybrid routing model, hydrological model, model calibration, physics-enhanced machine learning, rivers, streamflow prediction

HIGHLIGHTS

- Hybrid model that combines physics-based model with machine learning model.
- Two hybrid modeling techniques: delta learning and data augmentation.
- Hybrid model improves streamflow prediction accuracy over time.
- Multiple machine learning models are implemented.

1. INTRODUCTION

The management of stream and river systems entails significant and frequently costly decision-making procedures. These processes include floodplain mapping, citizen water usage, national-level infrastructure designs, and agricultural arrangements. Notable modeling investigations were conducted in the early 1950s, and over the past two decades, hydrologists have made significant efforts to develop large-scale hydrologic models (David *et al.* 2011a). Prior to the utilization of data-driven analysis through computational simulation, hydrologists predominantly relied on creating mathematical models of river systems as the primary method for generating hydrometeorological forecasts (David *et al.* 2016). Currently, significant research is dedicated to creating river models that focus on ground-level stream wave propagation across vast geographical boundaries. Notable examples of such models include the routing model proposed by Lohmann *et al.* (1996), total runoff integrating pathways by Oki & Sud (1998), LISFLOOD-FP by Bates & De Roo (2000), river transport model by Branstetter (2001), RiTHM

This is an Open Access article distributed under the terms of the Creative Commons Attribution Licence (CC BY 4.0), which permits copying, adaptation and redistribution, provided the original work is properly cited (<http://creativecommons.org/licenses/by/4.0/>).

(river-transfer hydrological model) by Ducharne *et al.* (2003), hillslope river routing by Beighley *et al.* (2009), and CaMa-Flood (catchment-based macro-scale floodplain) by Yamazaki *et al.* (2011).

Despite the advancements in these models, a significant challenge persists: a substantial portion of river stretches remains unmonitored. Streamflow measurements for the United States are primarily provided by the National Water Information System (NWIS) of the US Geological Survey (USGS) (see <https://waterdata.usgs.gov/nwis>). Local state or municipal institutions also contribute short-duration streamflow observations, which are generally less reliable than the NWIS dataset (Ghimire *et al.* 2023). As of 2023, the USGS reports that only 10,996 sites are equipped with direct automatic monitoring devices to record current conditions, despite the United States being home to approximately 250,000 rivers and having over 3.5 million miles of waterways. Furthermore, monitoring intervals vary widely and lack consistency. The only monitoring technique available aside from USGS observations entails using a bucket and timer to measure the volume of water collected over a specific period, but this is impractical for most streams except for those of the smallest order. Therefore, the challenge in predicting streams and waterways lies in the limited availability of discharge observations that are obtained from a restricted number of gauged locations. As a result, flow estimations rely on alternative techniques, such as the use of an acoustic Doppler current profiler or the utilization of the routing application for parallel computation of discharge (RAPID) model (McCarthy 1938).

This study focuses on RAPID, which is a river routing model based on the Muskingum algorithm developed by McCarthy (1938). David *et al.* (2011b, 2013, 2015) expanded on McCarthy's work by proposing a matrix-based version of the Muskingum method that enables computation of flow and water volume in numerous sections of a river network, even those without measurement data. RAPID was specifically designed for the National Hydrography Dataset Plus (NHDPlus), a comprehensive dataset that effectively captures the water flow wave across a wide range of catchment boundaries (David *et al.* 2011b; Tavakoly *et al.* 2017). Within the Geographic Information System (GIS), NHDPlus contains monthly and annual streamflow estimates for over 3 million stream segments and provides a systematic representation of how these streams are interconnected within global river networks (David *et al.* 2011b).

Despite its high-performance computing capabilities, calibrating parameters in the RAPID model poses challenges due to the necessity for numerous measurements and the intensive computational and data requirements (Zhao *et al.* 2023). RAPID requires access to streamflow measurement data, which may be unattainable for certain unmonitored catchments. Meanwhile, RAPID's process-based hydrology simulation also demands a substantial amount of computational processing time (Ghimire *et al.* 2023). To address these aforementioned difficulties, various research studies have been conducted. One such study integrates anticipated discharges from the variable infiltration capacity model with the RAPID model on a nationally scalable framework, incorporating waterway measurements from the NHDPlusV2 river network. This approach leverages high-level computational efforts to enhance parameter calibration (Ghimire *et al.* 2023). Another coupled model, AutoRAPID, combines runoff data from RAPID with watershed storm surge values generated by the AutoRoute model. This integration enables swift flood inundation predictions across large geographic areas in response to catastrophic situations (Follum *et al.* 2017). In addition to these coupled models, data-driven machine learning (ML) models have witnessed tremendous success in hydrological predictions and related fields. In particular, various ML models and hybrid models that combine two different types of ML models have been developed for time series prediction related to hydrology. For instance, a long short-term memory (LSTM) method tuned by ant lion optimization (called LSTM-ALO) has been applied to monthly runoff forecasting in the study by Yuan *et al.* (2018), where the ALO algorithm is employed to optimize parameters of the LSTM model. Similarly, a method called LSTM-INFO has been developed to predict water temperature by optimizing the LSTM model parameters using reptile search algorithm and weighted mean of vectors optimizer (Ikram *et al.* 2023). In addition to LSTM-based methods, support vector machine (SVM)-based methods have been explored in the studies by Adnan *et al.* (2023b) and Adnan *et al.* (2022) for evaporation prediction, resulting in methods called relevance vector machine tuned with improved Manta-Ray foraging optimization (RVM-IMRFO) and firefly algorithm and particle swarm optimization (SVM-FFAPSO). The other studied ML models for time series forecasting including the random vector functional link networks (Mostafa *et al.* 2023), hybrid heuristic algorithm (Adnan *et al.* 2021), and extreme learning machine methods tuned with metaheuristic algorithms (Adnan *et al.* 2023a). In the past decade, predicting time series data using ML models has become a very crowded field and numerous approaches have been proposed in the literature. Among all these available approaches, deep learning (DL)-based methods have shown promising potentials, especially for river discharge prediction. Kratzert provides convincing evidence that DL, specifically through the utilization of LSTM, produces superior runoff simulations from precipitation data in ungauged sites compared to conceptual models in monitored basins (Kratzert *et al.* 2019). Similarly, Nearing's research also highlights the notable advantage of LSTM-based DL models in

capturing dynamic watershed behavior and improving hydrological performance across various climatological circumstances (Nearing *et al.* 2021). Remarkably, even when catastrophic scenarios were not explicitly included in the training data, the LSTM network and its variants consistently predicted severe events with remarkable accuracy compared to the conceptual Sacramento model and the process-based US National Water Model (Frame *et al.* 2022). Furthermore, the hybrid hydrological ML model, which efficiently leverages the strengths of artificial neural networks and the k -nearest neighbor method, has demonstrated accurate and robust forecasting capabilities for extreme events (Kan *et al.* 2020).

The challenges associated with calibrating RAPID streamflow parameters, coupled with the calibration process's reliance on gauged measurement data, motivated us to develop a novel data-driven parameter calibration model that utilizes the hydrological and topological features of the river's watershed (Zhao *et al.* 2023). These calibrated parameters can be fed into a physics-based model that produces discharge forecasts. However, physics-based models are often simplified by numerous assumptions, such as RAPID being a linear river routing model, and therefore, they can be improved. While the aforementioned reviewed data-driven ML models has shown promising potential in river discharge prediction, their performance is often affected by both the quantity and quality of the data used for training the models. In this article, we extend our research efforts to build a hybrid streamflow discharge prediction model for *gauged* rivers that leverages the strengths of both the physics-based RAPID simulation model and data-driven ML models. It is worth noting that the 'hybrid model' in this article is different from the variety of hybrid data-driven models previously reviewed, which typically amalgamate various ML algorithms or optimization techniques. The 'hybrid model' in this article refers to the integration of the physics-based analysis model with the data-driven ML model, which is a concept also known as physics-informed/enhanced ML models. The major contributions and novelty of the proposed methods can be summarized as follows:

1. Incorporating physics-based principles: Integrating the intrinsic physics of river flow as encapsulated by the RAPID model into the data-driven ML prediction framework.
2. Leveraging data-informed insights: Harnessing hydrological insights embedded in the runoff data of nearby gauged rivers to address the gaps, discrepancies, and missing information in the physics-based model created as a result of inherent assumptions in physics-based models.
3. Utilizing cutting-edge ML techniques: Building a hybrid model that captures the capabilities of the physics-based model while being data-informed, by employing cutting-edge ML-based forecasting techniques of varying complexities, including Gaussian process (GP)-based nonlinear autoregressive with exogenous inputs (NARX) regressor (GP-NARX), neural network-based NARX regressor (NARX-Net), LSTM, and bidirectional LSTM (BiLSTM).
4. Forecasting for extended time periods: The model should be capable of predicting river discharge for long-time horizons (such as predicting the discharge a year in advance) and capturing the observed seasonality and trends.

In this article, we limit ourselves to forecasting river discharge only for gauged rivers that have available historical true discharge values. The primary idea is to consider a basin that consists of numerous watersheds, each with a respective river crossing the watershed. A small subset of these rivers is gauged, and we aim to forecast the discharge of these rivers. Building a machine learning forecasting model for these gauged rivers will include three primary data sources: (a) the past time series of true discharge values from gauges; (b) the physics-based RAPID discharge predictions, and (c) the runoff data of all the rivers in the basin from a land surface model. These three datasets are useful for two purposes: (a) the physics-based RAPID data, in tandem with runoff data (with reduced dimensionality using singular value decomposition (SVD) or LSTM-based autoencoder) can be used to build a more informed digital surrogate of the RAPID model, and (b) the RAPID data can be compared with the gauged data to learn about the discrepancies in the physics-based model. We use these capabilities to build two hybrid forecasting models: (a) delta Learning and (b) data augmentation. The incorporation of true discharge data and runoff data from the entire basin, along with the adjustment and enhancement of the RAPID model's performance through these data-driven hybrid models, collectively contributes to the development of a reliable framework for forecasting river discharge. This framework outperforms the currently used physics-based RAPID model in terms of prediction accuracy.

The following sections of this article are structured as follows: Section 2 summarizes the datasets used and the techniques employed for feature extraction from the runoff data. Section 3 provides detailed information on the two algorithms, namely, delta learning and data augmentation, used to build the hybrid model. Section 4 outlines various ML methods employed to implement the delta learning and data augmentation algorithms and obtain the hybrid discharge prediction models. Section 5 presents the case study, and Section 6 concludes the article.

2. DATASETS AND EXTRACTION OF FEATURES

2.1. Datasets

Building these forecasting algorithms requires leveraging three data sources. The first data stream consists of the true measured discharge from USGS, obtained using river gauges. However, not all the rivers in a basin are gauged, and the forecasting algorithms presented in this article are limited to rivers with gauges since these algorithms are contingent upon the availability of historic true gauged measurements. The second data stream is RAPID's historic discharge prediction time series for the river of interest. Readers are referred to Section 2 of our previous work (Zhao *et al.* 2023) to understand the workings of the RAPID model, which consists of two steps: (1) calibrating the input hydrological parameters of the RAPID model, and (2) predicting future discharge (Figure 1 of Zhao *et al.* 2023). Unlike the first two data streams associated with the specific rivers whose discharge we aim to forecast, the third data stream used for building the hybrid model is the runoff data for all the rivers/watersheds that are part of the basin of interest. These data are simulated using the Noah with multiparameterization (Noah-MP) land surface model (Niu *et al.* 2011). We observe that unlike true gauge data, runoff data can be projected for the intermediate term. Let $N_{\text{rivers-in-basin}}$ denote the number of rivers in the basin of interest. Incorporating the runoff data of all the rivers present in the basin provides information about the dynamics of the river network in the region surrounding the specific river of interest. This helps us gain a better understanding of hydrological conditions, particularly in areas lacking streamflow monitoring networks.

The true discharge data and RAPID's predictions are vector time series, with each element representing a discharge value on a given day. Let $N_{\text{total-days}}$ denote the total number of days for which the data are available (these days will be split later for the purpose of training and testing the hybrid model). We denote true gauge data as a vector $\mathbf{Y} = \{y_i\} \in \mathbb{R}^{N_{\text{total-days}}}$ and the RAPID's prediction data time series as a vector $\mathbf{R} = \{r_i\} \in \mathbb{R}^{N_{\text{total-days}}}$, with $1 \leq i \leq N_{\text{total-days}}$. Unlike the true gauge discharge and RAPID values, which are associated with the river whose discharge we aim to predict, the runoff dataset consists of time series data for runoff from all the rivers in the basin. Therefore, the runoff data are represented as a higher-dimensional matrix, denoted by $\mathbf{M} = \{\mathbf{m}_i\} \in \mathbb{R}^{N_{\text{rivers-in-basin}} \times N_{\text{total-days}}}$, with $1 \leq i \leq N_{\text{total-days}}$.

The j th element of the i th column \mathbf{m}_i of the matrix \mathbf{M} , denoted as m_{ij} , represents the runoff values for the j th rivers in the basin on the i th day. Therefore, the dimensionality of the runoff data is $N_{\text{rivers-in-basin}} \times N_{\text{total-days}}$. Using these three datasets in their raw format presents two major challenges that require further data processing.

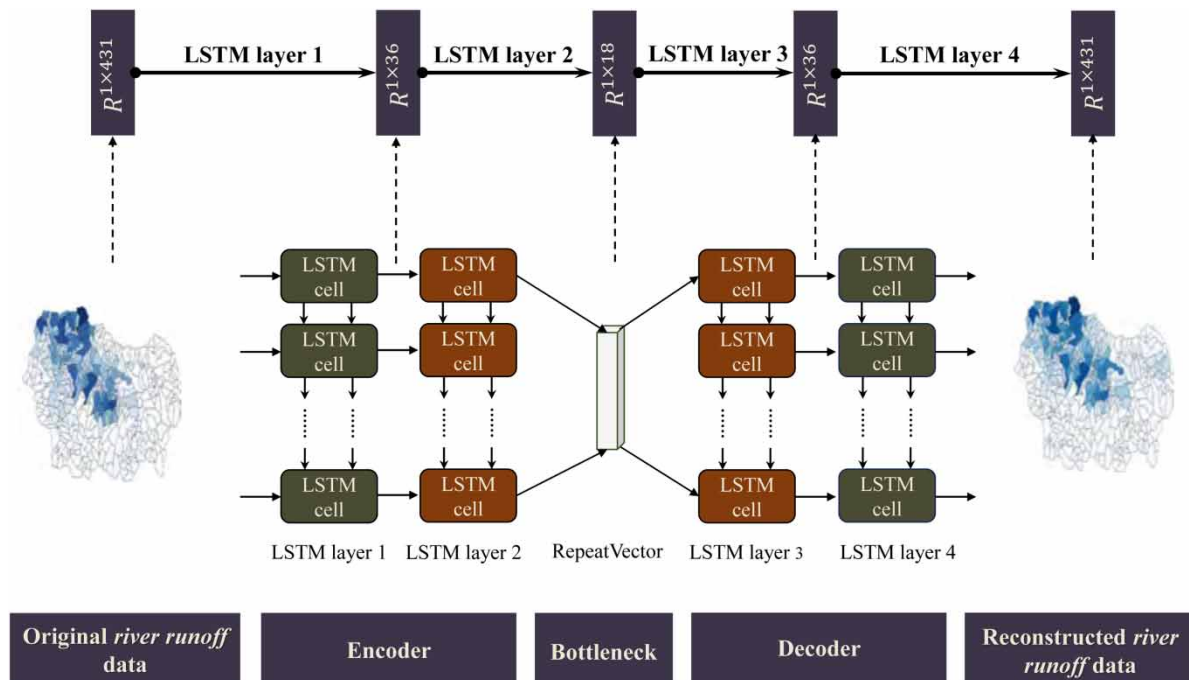


Figure 1 | LSTM autoencoder architecture.

- First, for basins with a large number of watersheds, the dimension of runoff data can be very high, as there is one time series of runoff data for each watershed. For example, the basin studied in the case study has 431 watersheds, which means that the dimension of the runoff data is $431 \times N_{\text{total-days}}$. This high-dimensional data poses significant challenges to the training of the hybrid models. Since these watersheds are in the geographic neighborhood of each other, there is bound to be correlation between the runoff time series of rivers passing through these watersheds. Therefore, we first reduce the high-dimensional data into low-dimensional features using dimension reduction techniques (i.e., reducing the number of rows in the \mathbf{M} matrix). Two types of dimension reduction techniques, namely, SVD and LSTM-based autoencoder, have been studied.
- Second, the RAPID model exhibits considerable variations and noise due to the presence of unmodeled features and underlying assumptions. To enhance the accuracy of hybrid models based on RAPID model predictions, it is essential to employ data smoothing techniques that help eliminate outliers and highlight underlying trends. We use a Hanning filter based on the convolution of a scaled window with the signal to smooth RAPID model predictions (Oppenheim 1999). The window size used in this article is 30.

2.2. Surface runoff features extracted by singular value decomposition

To reduce the dimension of the runoff data for training the hybrid models, we perform SVD on the matrix \mathbf{M} as follows:

$$\mathbf{M} = \mathbf{V}\boldsymbol{\lambda}\mathbf{U}^T, \quad (1)$$

where $\mathbf{V} \in \mathbb{R}^{N_{\text{rivers-in-basin}} \times N_{\text{rivers-in-basin}}}$ and $\mathbf{U} \in \mathbb{R}^{N_{\text{total-days}} \times N_{\text{total-days}}}$ are orthogonal matrices, and $\boldsymbol{\lambda} \in \mathbb{R}^{N_{\text{rivers-in-basin}} \times N_{\text{total-days}}}$ is a rectangular matrix with eigenvalues $\boldsymbol{\chi} = \{\chi_i\} \in \mathbb{R}^{\min\{N_{\text{rivers-in-basin}}, N_{\text{total-days}}\}}$. In our case, since $N_{\text{rivers-in-basin}} \ll N_{\text{total-days}}$, we have $\boldsymbol{\chi} \in \mathbb{R}^{N_{\text{rivers-in-basin}}}$.

Following this decomposition, we define another matrix $\tilde{\mathbf{M}} = [\tilde{\mathbf{m}}_1; \tilde{\mathbf{m}}_2; \dots; \tilde{\mathbf{m}}_{N_{\text{rivers-in-basin}}}] = \mathbf{V}\boldsymbol{\lambda} \in \mathbb{R}^{N_{\text{rivers-in-basin}} \times N_{\text{total-days}}}$ that represent the latent features of the runoff data. Here, $\tilde{\mathbf{m}}_j \in \mathbb{R}^{1 \times N_{\text{total-days}}}$ is the j th row of the $\tilde{\mathbf{M}}$ matrix. Assuming that we reduce the original dimension of the \mathbf{M} matrix to a much lower dimension, denoted by N_{SVD} , we have the runoff data in the reduced space as follows:

$$\tilde{\mathbf{M}} = [\tilde{\mathbf{m}}_1; \tilde{\mathbf{m}}_2; \dots; \tilde{\mathbf{m}}_{N_{\text{SVD}}}] \in \mathbb{R}^{N_{\text{SVD}} \times N_{\text{total-days}}}. \quad (2)$$

Readers are referred to the study by Hu *et al.* (2017) for a more detailed discussion and application of the SVD technique.

2.3. Surface runoff features extracted by a long short-term memory autoencoder

2.3.1. LSTM overview

LSTM is a recurrent neural network (RNN)-based architecture specifically developed to address the challenge of learning sequence data with long-term dependencies. LSTM networks have been employed to tackle this issue by ensuring a consistent backward flow in the error signal. The utilization of LSTM networks not only excels in capturing short-term data dependencies for accurate predictions but also effectively considers the explicit dependencies among multiple outputs over a long-term time horizon.

In this article, we leverage LSTM for both the dimension reduction of the runoff data and hybrid modeling. This section explains the application of LSTM for feature extraction of the runoff data, while the LSTM architecture is later described in Section 4.3, which discusses the hybrid modeling.

2.3.2. Long short-term memory autoencoder network

An autoencoder is a form of unsupervised neural network that uses data to find the best encoding-decoding architecture (Nguyen *et al.* 2021). It can be regarded as an n -layer neural network that aims to reconstruct the inputs based on the provided code. The autoencoder comprises five components: an input layer, an output layer, an encoder, a decoder, and a bottleneck region known as the latent space.

The encoder reduces the input dimensions to generate a representation for the bottleneck, while the decoder expands the bottleneck's dimensions to recreate the input layers. The latent space represents the high-level inputs using compressed datasets at a lower level. In this article, considering the advantage of LSTM in dealing with time series data and the promising performance of LSTM shown in the literature, LSTM is used for constructing the autoencoder. The structures of the encoder and decoder in this article are similar, with two LSTM layers stacked, as shown in Figure 1. The encoder takes high-dimensional datasets as inputs and generates compressed datasets as outputs. The decoder uses the compressed dataset stored in the bottleneck to reconstruct the original input datasets.

2.3.3. Encoder structure

The encoder consists of two LSTM layers, followed by a RepeatVector layer. It serves as an encoder that takes high-dimensional datasets as inputs and generates compressed datasets as outputs.

To implement the encoder, we begin by reshaping all the input datasets at each time step into a size of (1,431). This allows us to standardize the input datasets before passing them into the LSTM layers. The first LSTM layer (LSTM layer 1) sets the input channel to one, indicating two dimensions. The output dimension of LSTM layer 1 is configured as 36, and the kernel initialization is defined as *he_uniform*, which follows a truncated normal distribution. Following the first LSTM layer, we proceed to create the second LSTM layer (LSTM layer 2). In this layer, the input dimension is set to 36, reflecting the output channel from the previous layer, while the output dimension is configured as 18. Like the first LSTM layer, we maintain the kernel initialized as *he_uniform*. The final layer utilized in the encoder structure is the RepeatVector layer, which is repeated a defined number of times using this architecture.

This suggests that this autoencoder model can be viewed as a generative model, particularly when dealing with sparse gradients. We have:

$$\tilde{\mathbf{m}} = k_e(\mathbf{m}), \quad (3)$$

where $\tilde{\mathbf{m}}$ denotes the compressed reduced-order representation learned from the encoder $k_e(\cdot)$ for the input \mathbf{m} , which represents the runoff data at a particular instance of time.

2.3.4. Decoder structure

The structure of the decoder comprises two LSTM layers. Its purpose is to utilize the compressed dataset stored in the bottleneck to reconstruct the original input datasets.

To implement the decoder structure, we begin by utilizing an LSTM layer to create the first decoding layer (LSTM layer 3), which is the same as the second layer in the encoder structure (LSTM layer 2). In this layer, the input size is set to (1,18), while the output size is 36. Next, we employ a second decoding LSTM layer (LSTM layer 4). The input feature size is 36, and the output feature size is 431. The remaining parameters, including the kernel initialization, are the same as those used in the encoder structure. The river runoff sequence for a particular time instant is reconstructed as:

$$\hat{\mathbf{m}} = k_d(\tilde{\mathbf{m}}), \quad (4)$$

where $\hat{\mathbf{m}}$ represents reconstructed inputs learned from decoder $k_d(\cdot)$ using the compressed dataset $\tilde{\mathbf{m}}$.

2.3.5. The reconstruction error and the parameters used for LSTM autoencoder architecture

The reconstruction error P , defined in Equation (5), must be minimized when training the LSTM autoencoder.

$$P = \frac{1}{2} \sum \|\mathbf{m} - \hat{\mathbf{m}}\|^2. \quad (5)$$

The primary objective of the autoencoder is to reduce the dimensionality of surface runoff features. By introducing a bottleneck with a lower dimension than the original 431-dimensional input, the autoencoder is compelled to capture the most essential features from the runoff dataset. This combination of components empowers us to achieve the intended goals of analyzing large-dimensional input datasets and adjusting their resolutions. Table 1 displays the parameters used in the LSTM autoencoder.

Table 1 | LSTM autoencoder architecture model parameters

Model architecture	Description
Number of layers	5
Batch size	32
Number of epochs	500
Optimizer	Adam
Loss function	MSE

2.4. Data processing workflow

Processing the data involves three steps:

- 1. Smoothing the RAPID's prediction data:** The prediction data \mathbf{R} from RAPID is smoothed using a Hanning filter, with no change in the dimensionality of the data (Oppenheim 1999).
- 2. Reducing the dimensions of the runoff data:** By employing SVD or a LSTM autoencoder, we can reduce the number of rows in the \mathbf{M} matrix, consequently altering its dimensions from $N_{\text{rivers-in-basin}} \times N_{\text{total-days}}$ to either $N_{\text{SVD}} \times N_{\text{total-days}}$ when using SVD or $N_{\text{AE}} \times N_{\text{total-days}}$ when using a LSTM autoencoder.
- 3. Downsampling the number of days:** To further reduce the dimensionality of all three datasets, we downsample the number of days if the ML method deems it necessary. For example, if we downsample by a factor of 2, it would require deleting every other data point in RAPID's predictions and the true gauge values (since we use daily average RAPID estimates for building the hybrid model), as well as deleting alternate columns in the runoff matrix. Let the number of remaining days after downsampling be denoted by $N_{\text{sampled-days}}$.

Let $\tilde{\mathbf{R}} = \{\tilde{r}_i\}$ denote the smoothed and downsampled RAPID data, and let $\tilde{\mathbf{Y}} = \{\tilde{y}_i\}$ denote the downsampled gauge data. Both $\tilde{\mathbf{R}}$ and $\tilde{\mathbf{Y}}$ are vectors of size $N_{\text{sampled-days}}$. In addition, let $\tilde{\mathbf{M}} = \{\tilde{m}_i\}$ denote the dimensionally reduced runoff matrix via SVD or autoencoder, with dimensions of $N_{\text{SVD}} \times N_{\text{reduced-days}}$ when using SVD or $N_{\text{AE}} \times N_{\text{reduced-days}}$ when using a LSTM autoencoder. We split these three datasets into training and testing subsets, such that $\tilde{\mathbf{R}} = [\tilde{\mathbf{R}}_{\text{train}}, \tilde{\mathbf{R}}_{\text{test}}]$, $\tilde{\mathbf{Y}} = [\tilde{\mathbf{Y}}_{\text{train}}, \tilde{\mathbf{Y}}_{\text{test}}]$, $\tilde{\mathbf{M}} = [\tilde{\mathbf{M}}_{\text{train}}, \tilde{\mathbf{M}}_{\text{test}}]$, and $N_{\text{reduced-days}} = N_{\text{train-days}} + N_{\text{test-days}}$. Here, $N_{\text{train-days}}$ and $N_{\text{test-days}}$ denote the number of training and testing data points, respectively. In the following write-up, we will detail two algorithms deployed to build a hybrid discharge forecasting model for a gauged river.

3. ALGORITHMS FOR PHYSICS-ENHANCED ML MODEL FOR DISCHARGE FORECASTING

Having presented two data compression techniques previously, in this section, we describe two algorithms used in building the hybrid forecasting models: (a) delta learning and (b) data augmentation (Thelen *et al.* 2022). Both of these hybrid models utilize RAPID as the physics-based model and enhance it by incorporating runoff data from the entire basin and the historical discharge measurements of the specific gauged river whose discharge we aim to forecast.

3.1. Delta learning

The delta learning algorithm is based on learning the error between RAPID's discharge prediction and measured gauge data.

3.1.1. Training

Training the hybrid forecast model using the delta learning algorithm involves building two ML models and includes three steps, as detailed below.

- 1. Step 1:** The first step is to construct a runoff data-informed surrogate for the RAPID model, enabling the prediction of future RAPID discharge based on the historical time series of RAPID data and runoff data. Incorporating the runoff data into the surrogate model of RAPID helps include some of the missing information about river dynamics not captured by the physics-based RAPID model. This surrogate model can be considered an *enhanced physics-based model*. The goal of this surrogate is to predict the discharge at the next time step given a window of RAPID data and runoff data in the immediate past. For instance, consider the *current* time step $N < N_{\text{train}}$ and the window size w_r and w_m , such that $\tilde{\mathbf{R}}_{\text{current}} = [\tilde{r}_1, \tilde{r}_2, \dots, \tilde{r}_{N-w_r+1}, \dots, \tilde{r}_{N-1}, \tilde{r}_N]$ and $\tilde{\mathbf{M}}_{\text{current}} = [\tilde{m}_1, \tilde{m}_2, \dots, \tilde{m}_{N-w_m+1}, \dots, \tilde{m}_{N-1}, \tilde{m}_N]$. Since Noah-MP

provides runoff forecasts for the near future, we also consider the next runoff data \tilde{m}_{N+1} as the input to train RAPID surrogate since it informs the future. The input to the surrogate model \mathcal{R} will be the vector $[\tilde{r}_{N-w_r+1}, \dots, \tilde{r}_{N-1}, \tilde{r}_N, \tilde{m}_{N-w_m+1}^T, \dots, \tilde{m}_{N-1}^T, \tilde{m}_N^T, \tilde{m}_{N+1}^T]$. For the current time step N and a window size of w , the discharge predicted by the RAPID's surrogate for the next time step is given by:

$$\hat{r}_{N+1} = \mathcal{G}_r(\tilde{r}_{N-w_r+1}, \dots, \tilde{r}_{N-1}, \tilde{r}_N, \tilde{m}_{N-w_m+1}^T, \dots, \tilde{m}_{N-1}^T, \tilde{m}_N^T, \tilde{m}_{N+1}^T). \quad (6)$$

2. *Step 2:* The second step is to obtain the error in the anticipated outflow based on the physics-based prediction relative to the gauged measurement. To obtain the physics-based prediction, we utilize the previously trained surrogate \mathcal{G}_r defined in Equation (6). However, instead of using the RAPID data as input, we use the gauge measurements along with the runoff data to evaluate the physics-based surrogate response. This tweak is necessary for a like-for-like comparison of the physics-based prediction relative to the gauged measurement. That is,

$$\hat{r}_{N+1} = \mathcal{G}_r(\tilde{y}_{N-w_r+1}, \dots, \tilde{y}_{N-1}, \tilde{y}_N, \tilde{m}_{N-w_m+1}^T, \dots, \tilde{m}_{N-1}^T, \tilde{m}_N^T, \tilde{m}_{N+1}^T). \quad (7)$$

The error term (or the delta term) is given as follows:

$$\delta_{N+1} = \tilde{y}_{N+1} - \hat{r}_{N+1}. \quad (8)$$

Using this construct, the delta term can be evaluated for each day by considering data points from the preceding days, except for the first $w = \max(w_r, w_m)$ time steps. This exception is necessary because evaluating the predicted discharge using the surrogate \mathcal{G}_r requires data from the last w time steps. This yields the delta vector denoted by $\Delta = [\delta_{w+1}, \delta_{w+2}, \dots, \delta_{N_{\text{train}}}]$.

3. *Step 3:* The goal of this step is to build a predictor for the error in discharge (the delta term surrogate) capable of predicting the error for a future time step, considering the preceding gauge measurements and runoff data for a window size of w_r and w_m , respectively. To maintain compatibility in dimensionality between the input and output, we delete the first w columns of $\tilde{\mathbf{Y}}_{\text{train}}$ and $\tilde{\mathbf{M}}_{\text{train}}$. The predicted error $\hat{\delta}_{N+1}$ for the step $(N+1)$ is then defined as follows:

$$\hat{\delta}_{N+1} = \mathcal{G}_\delta(\tilde{r}_{N-w_r+1}, \dots, \tilde{r}_{N-1}, \tilde{r}_N, \tilde{m}_{N-w_m+1}^T, \dots, \tilde{m}_{N-1}^T, \tilde{m}_N^T, \tilde{m}_{N+1}^T). \quad (9)$$

Figure 2 illustrates the steps involved in training the hybrid model using delta learning algorithm.

3.1.2. Testing and forecasting

The two ML models, \mathcal{G}_r and \mathcal{G}_δ , trained based on the steps defined in the previous subsection, will be deployed to predict the discharge at the next time step using the gauge data and runoff data from preceding time steps as shown in Figure 3. We aim for a time series forecast of future discharge. This involves two steps:

1. *Step 1:* For the current time step N , we obtain the discharge forecast \hat{r}_{N+1} and the delta term $\hat{\delta}_{N+1}$ for time step $(N+1)$ using the following:

$$\hat{r}_{N+1} = \mathcal{G}_r(\tilde{y}_{N-w_r+1}, \dots, \tilde{y}_{N-1}, \tilde{y}_N, \tilde{m}_{N-w_m+1}^T, \dots, \tilde{m}_{N-1}^T, \tilde{m}_N^T, \tilde{m}_{N+1}^T), \quad (10a)$$

$$\hat{\delta}_{N+1} = \mathcal{G}_\delta(\tilde{y}_{N-w_r+1}, \dots, \tilde{y}_{N-1}, \tilde{y}_N, \tilde{m}_{N-w_m+1}^T, \dots, \tilde{m}_{N-1}^T, \tilde{m}_N^T, \tilde{m}_{N+1}^T). \quad (10b)$$

Here, the forecast runoff data for the time step $(N+1)$, denoted by \hat{m}_{N+1} , are obtained from Noah with multiparameterization (Noah-MP) land surface model.

2. *Step 2:* By using the quantities obtained earlier, we obtain the forecasted gauge measurement \hat{y}_{N+1} for future time step $(N+1)$ as follows:

$$\hat{y}_{N+1} = \hat{\delta}_{N+1} + \hat{r}_{N+1}. \quad (11)$$

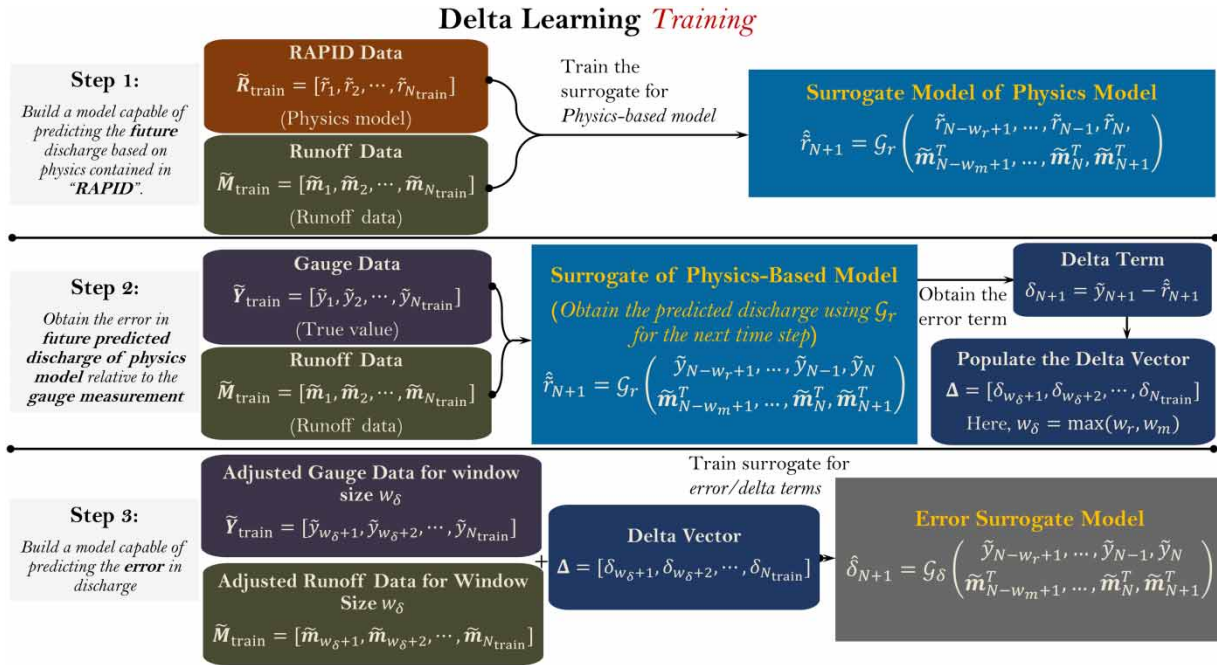


Figure 2 | Schematic flowchart for training the surrogate of the physics-based model and the error (or delta) term in the delta learning algorithm.

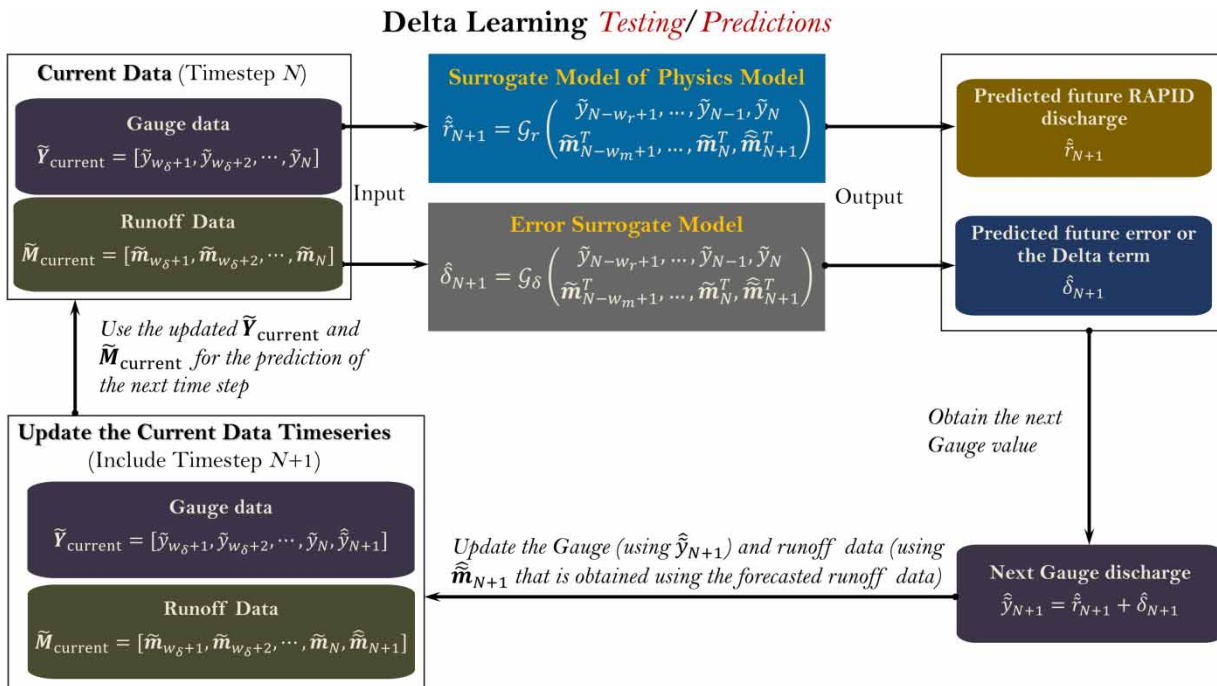


Figure 3 | Schematic flowchart for testing and making discharge forecasts using the delta learning algorithm.

These steps can be repeated to obtain the predicted gauge measurement for the next time step ($N + 2$) by using the predicted gauge discharge \hat{y}_{N+1} and the forecasted runoff data \hat{m}_{N+2} as part of the input, such that

$$\hat{r}_{N+2} = \mathcal{G}_r(\tilde{y}_{N-w_r+2}, \dots, \tilde{y}_N, \hat{y}_{N+1}, \tilde{m}_{N-w_m+2}^T, \dots, \tilde{m}_N^T, \hat{m}_{N+1}^T, \hat{m}_{N+2}^T), \quad (12a)$$

$$\hat{\delta}_{N+2} = \mathcal{G}_\delta(\tilde{Y}_{N-w_r+2}, \dots, \tilde{Y}_N, \hat{Y}_{N+1}, \tilde{\mathbf{m}}_{N-w_m+2}^T, \dots, \tilde{\mathbf{m}}_{N-1}^T, \hat{\mathbf{m}}_{N+1}^T, \hat{\mathbf{m}}_{N+2}^T), \quad (12b)$$

$$\hat{Y}_{N+2} = \hat{\delta}_{N+2} + \hat{r}_{N+2}. \quad (12c)$$

Following the same line of reasoning, the gauge discharge forecast can be obtained for as long as the runoff forecast is available.

3.2. Data augmentation

3.2.1. Training

Similar to delta learning, training the hybrid forecast model using the data augmentation algorithm involves constructing two ML models and consists of three steps, as outlined below:

1. *Step 1:* The first step is identical to delta learning as discussed in Section 3.1.1, and it entails creating a runoff data-informed surrogate for the RAPID model, denoted as \mathcal{G}_r and defined in Equation (6).
2. *Step 2:* At time step N , the model \mathcal{G}_r can be used to obtain the predicted physics-based discharge \hat{r}_{N+1} for the next time step $(N + 1)$ using Equation (7). The predicted physics-based discharge can be evaluated for each day by considering data points from the preceding days, except for the first w time steps. This exception is necessary because evaluating the predicted discharge using the surrogate \mathcal{G}_r requires data from the last w time steps. This yields the vector of predictions made by the model \mathcal{G}_r for the entire training period denoted by $\hat{\mathbf{R}} = [\hat{r}_{w+1}, \hat{r}_{w+2}, \dots, \hat{r}_{N_{\text{train}}}]$.

We emphasize that to obtain the prediction vector $\hat{\mathbf{R}}$ produced by the model \mathcal{G}_r , we use gauge measurements in conjunction with runoff data as inputs to evaluate the physics-based surrogate response, rather than relying on RAPID data that were one of the training input for the model \mathcal{G}_r in the first step (Equation (7)). This approach is taken because the physics-based prediction vector will be used to augment the inputs for training the surrogate model that forecasts gauge discharge measurements, as will be explained in the next step.

3. *Step 3:* This step involves utilizing the gauge data $\tilde{\mathbf{Y}}_{\text{train}}$ and runoff data $\tilde{\mathbf{M}}_{\text{train}}$, along with the prediction vector $\hat{\mathbf{R}}$ obtained in the previous step, to train a ML model capable of directly predicting the gauge measurements. This is unlike the delta Learning approach, where the error term was predicted first before the gauge measurements.

To maintain compatibility in dimensionality between the inputs and output vectors, we delete the first w columns of $\tilde{\mathbf{Y}}_{\text{train}}$ and $\tilde{\mathbf{M}}_{\text{train}}$. The predicted gauge measurement \hat{y}_{N+1} for the step $(N + 1)$ is then defined as follows:

$$\hat{y}_{N+1} = \mathcal{G}_y(\tilde{Y}_{N-w+1}, \dots, \tilde{Y}_{N-1}, \tilde{Y}_N, \tilde{\mathbf{m}}_{N-w+1}^T, \dots, \tilde{\mathbf{m}}_{N-1}^T, \hat{\mathbf{m}}_N^T, \hat{r}_{N+1}). \quad (13)$$

Notice that, in addition to the preceding gauge measurements and runoff data, the output of the physics-based surrogate \mathcal{G}_r , denoted as \hat{r}_{N+1} , is included as part of the input to the surrogate \mathcal{G}_y . That is, the input for the model \mathcal{G}_y was obtained by *augmenting* the input for the model \mathcal{G}_r . This is the reason why this approach is referred to as *data augmentation*. The term \hat{r}_{N+1} contains information about the immediate future (the next time step). Consequently, the model \mathcal{G}_y is trained to incorporate this immediate future information as predicted by the physics-based surrogate \mathcal{G}_r (at the next time step).

3.2.2. Testing and forecasting

Make a time series prediction of discharge into the future using data augmentation utilizes the models \mathcal{G}_r and \mathcal{G}_y . This involves two steps:

1. *Step 1:* For the current time step N , we obtain the discharge forecast \hat{r}_{N+1} using the following:

$$\hat{r}_{N+1} = \mathcal{G}_r(\tilde{Y}_{N-w_r+1}, \dots, \tilde{Y}_{N-1}, \tilde{Y}_N, \tilde{\mathbf{m}}_{N-w_m+1}^T, \dots, \tilde{\mathbf{m}}_{N-1}^T, \hat{\mathbf{m}}_N^T, \hat{\mathbf{m}}_{N+1}^T). \quad (14)$$

2. *Step 2:* By using the value of \hat{r}_{N+1} obtained earlier, we obtain the forecasted gauge measurement \hat{y}_{N+1} for future time step $(N + 1)$ as follows:

$$\hat{y}_{N+1} = \mathcal{G}_y(\tilde{y}_{N-w_r+1}, \dots, \tilde{y}_{N-1}, \tilde{y}_N, \tilde{\mathbf{m}}_{N-w_m+1}^T, \dots, \tilde{\mathbf{m}}_{N-1}^T, \tilde{\mathbf{m}}_N^T, \hat{r}_{N+1}^T, \hat{r}_{N+1}). \quad (15)$$

The forecast runoff data for the time step $(N + 1)$, denoted by $\hat{\mathbf{m}}_{N+1}$, is obtained from Noah-MP runoff forecasting model. These steps can be repeated to obtain the predicted gauge measurement for the next time step $(N + 2)$ by using the predicted gauge discharge \hat{y}_{N+1} and the forecasted runoff data $\hat{\mathbf{m}}_{N+2}$ as part of the input, such that

$$\hat{r}_{N+2} = \mathcal{G}_r(\tilde{y}_{N-w_r+2}, \dots, \tilde{y}_N, \hat{y}_{N+1}, \tilde{\mathbf{m}}_{N-w_m+2}^T, \dots, \tilde{\mathbf{m}}_{N-1}^T, \tilde{\mathbf{m}}_N^T, \hat{\mathbf{m}}_{N+1}^T, \hat{\mathbf{m}}_{N+2}^T), \quad (16a)$$

$$\hat{y}_{N+2} = \mathcal{G}_y(\tilde{y}_{N-w_r+2}, \dots, \tilde{y}_N, \hat{y}_{N+1}, \tilde{\mathbf{m}}_{N-w_m+2}^T, \dots, \tilde{\mathbf{m}}_{N-1}^T, \tilde{\mathbf{m}}_N^T, \hat{\mathbf{m}}_{N+1}^T, \hat{\mathbf{m}}_{N+2}^T, \hat{r}_{N+2}). \quad (16b)$$

Following the same line of reasoning, the gauge discharge forecast can be obtained for as long as the runoff forecast is available. Figures 4 and 5 illustrate the training and testing process for the data augmentation algorithm.

4. IMPLEMENTATION OF PHYSICS-ENHANCED ML ALGORITHMS FOR DISCHARGE PREDICTION

As discussed in the previous section, building a hybrid discharge forecast model using the delta learning algorithm requires the construction of two ML models, \mathcal{G}_r and \mathcal{G}_δ . In addition, using the delta augmentation algorithm necessitates the creation of two ML models, \mathcal{G}_r and \mathcal{G}_y . We have investigated four different ML methods with varying complexity and characteristics to train these models. The following write-up discusses these ML methods. To maintain generality in our descriptions, we use the notation of a generic input vector \mathbf{x} and a scalar output y that is available for the training purpose. We denote a matrix of multivariate training inputs as \mathbf{X} , where each row represents an instance of the input vector \mathbf{x} . Similarly, we represent the output as a column vector \mathbf{Y} , with each element denoting an instance of the output variable y . Finally, let the input vector for a testing point be denoted by $\hat{\mathbf{x}}$ and the predicted output be denoted by \hat{y} .

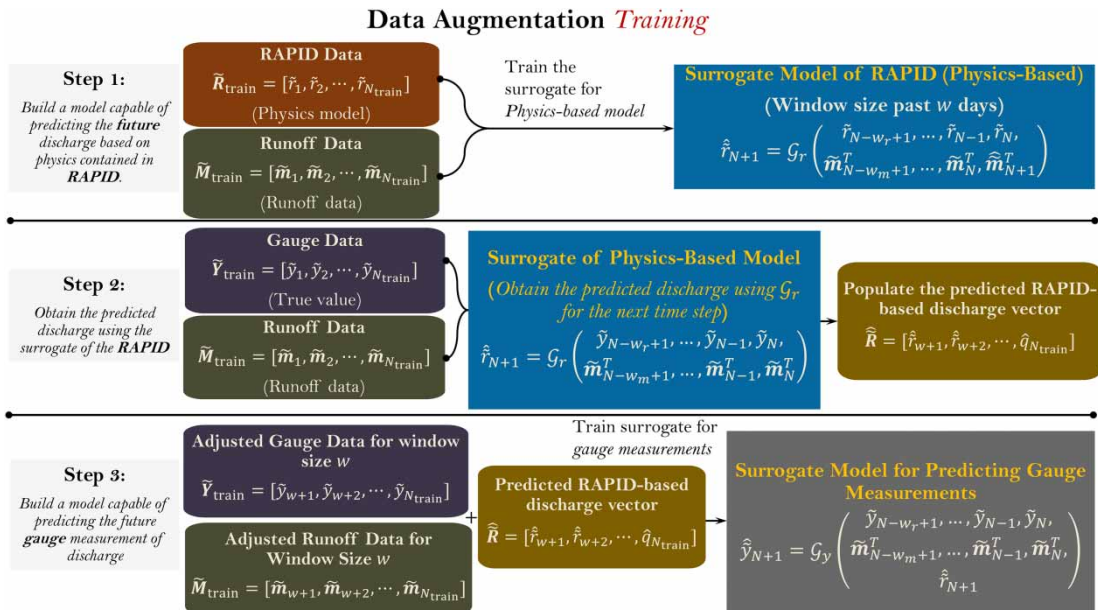


Figure 4 | Schematic flowchart for training the surrogate of the physics-based model and the gauge measurement in the data augmentation algorithm.

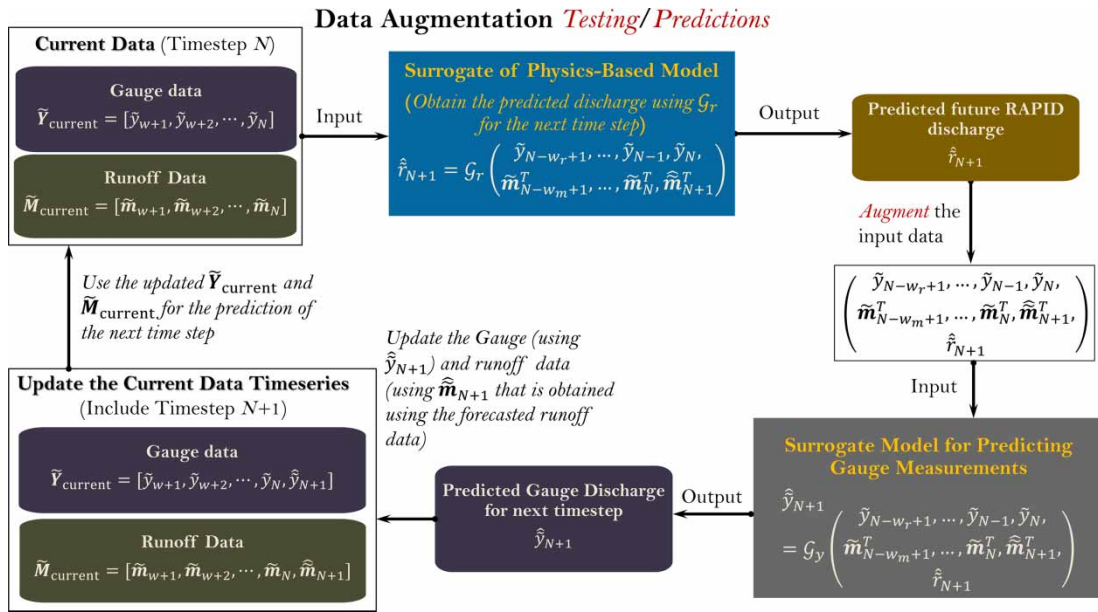


Figure 5 | Schematic flowchart for testing and making discharge forecasts using the data augmentation algorithm.

4.1. Gaussian process regression-based nonlinear autoregressive network with exogenous inputs

4.1.1. Gaussian process regression

We start by detaining the Gaussian process (GP) regression (GPR) for the training datasets (\mathbf{X}, \mathbf{Y}) . The goal is to find the mapping between the input vector \mathbf{x} (consisting of various features) and the output y . As described in Williams & Rasmussen (2006), a GPR probabilistically models the output y as a realization $g(\mathbf{x})$ of the GP, such that

$$y = g(\mathbf{x}) + \varepsilon, \tag{17}$$

$$g(\mathbf{x}) = m(\mathbf{x}) + h(\mathbf{x}),$$

where $m(\mathbf{x})$ is the mean of the GP and $h(\mathbf{x})$ is assumed to be a GP with zero mean and covariance function $k(\mathbf{x}, \mathbf{x}')$, and ε is the noise in the output, such that

$$g(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')), \tag{18}$$

$$h(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}')),$$

$$\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2),$$

where

$$k(\mathbf{x}, \mathbf{x}') = \sigma_h^2 R(\mathbf{x} - \mathbf{x}', \boldsymbol{\theta}), \tag{19}$$

where σ_h^2 is the constant variance of the GP, $R(\cdot, \cdot)$ is the correlation function, and $\boldsymbol{\theta}$ is the unknown parameter vector of the covariance function. There are a variety of correlation functions available. The most commonly used one is the Gaussian correlation function given by (Hu & Mahadevan 2016)

$$R(\mathbf{x} - \mathbf{x}', \boldsymbol{\theta}) = \exp \left(- \sum_{k=1}^{n_d} \theta_k |x_k - x'_k| \right), \tag{20}$$

where n_d is the dimension of the input variables and x_k is the k th element of \mathbf{x} .

The GP conditioned on the training data (\mathbf{X}, \mathbf{Y}) , where \mathbf{X} is also a GP with the mean function $\tilde{m}(\mathbf{x})$ and the covariance function $\tilde{k}(\mathbf{x}, \mathbf{x}')$, is given by

$$g(\mathbf{x})|\mathbf{X}, \mathbf{Y} \sim \mathcal{GP}(\tilde{m}(\mathbf{x}), \tilde{k}(\mathbf{x}, \mathbf{x}')). \quad (21)$$

Here,

$$\begin{aligned} \tilde{m}(\mathbf{x}) &= m(\mathbf{x}) + \mathbf{K}(\mathbf{x}, \mathbf{X}) \times (\mathbf{K}(\mathbf{X}, \mathbf{X}) - \sigma_\varepsilon^2 \mathbf{I}_n)^{-1} \times (\mathbf{Y} - \mathbf{m}), \\ \tilde{k}(\mathbf{x}, \mathbf{x}') &= k(\mathbf{x}, \mathbf{x}') - \mathbf{K}(\mathbf{x}, \mathbf{X}) \times (\mathbf{K}(\mathbf{X}, \mathbf{X}) - \sigma_\varepsilon^2 \mathbf{I}_n)^{-1} \times \mathbf{K}(\mathbf{X}, \mathbf{x}'), \end{aligned} \quad (22)$$

where

$$\begin{aligned} \mathbf{K}(\mathbf{X}, \mathbf{X}) &= \begin{pmatrix} k(\mathbf{x}^{(1)}, \mathbf{x}^{(1)}) & \dots & k(\mathbf{x}^{(1)}, \mathbf{x}^{(n)}) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}^{(n)}, \mathbf{x}^{(1)}) & \dots & k(\mathbf{x}^{(n)}, \mathbf{x}^{(n)}) \end{pmatrix} \in \mathbb{R}^{n \times n}, \\ \mathbf{K}(\mathbf{x}, \mathbf{X}) &= (k(\mathbf{x}, \mathbf{x}^{(1)}), \dots, k(\mathbf{x}, \mathbf{x}^{(n)})) \in \mathbb{R}^{1 \times n}, \\ \mathbf{K}(\mathbf{X}, \mathbf{x}) &= \mathbf{K}(\mathbf{x}, \mathbf{X})^T \in \mathbb{R}^{n \times 1}, \\ \mathbf{m} &= (m(\mathbf{x}^{(1)}), \dots, m(\mathbf{x}^{(n)}))^T \in \mathbb{R}^{n \times 1}. \end{aligned} \quad (23)$$

The hyperparameters θ , σ_h , and σ_ε are estimated using the training data through Bayesian parameter estimation or maximum likelihood estimation (Hu & Mahadevan 2016).

From Equations (17) and (21), the predictive distribution of the output \hat{y} for the input realization $\hat{\mathbf{x}}$ conditioned on the training data (\mathbf{X}, \mathbf{Y}) is given by

$$\hat{y}|\hat{\mathbf{x}}, \mathbf{X}, \mathbf{Y} \sim \mathcal{GP}(\tilde{m}(\hat{\mathbf{x}}), \tilde{k}(\hat{\mathbf{x}}, \hat{\mathbf{x}}) + \sigma_\varepsilon^2). \quad (24)$$

4.1.2. NARX model based on GPR

The GPR can be used to build a NARX predictive model for time series data. This model assumes that the current value of the output at time step $(N + 1)$ is predicted using a nonlinear function F of previous inputs and the output, such that

$$\hat{y}_{N+1} = F(y_{N-w_y+1}, \dots, y_N, \mathbf{x}_{N-w_x+1}^T, \dots, \mathbf{x}_N^T) + \varepsilon_N, \quad (25)$$

where the residual noise ε_N is assumed to be Gaussian. The terms w_x and w_y denote the input and output lags, respectively. The nonlinear function F can be employed to train the ML models incorporated in the hybrid discharge prediction model (i.e., \mathcal{G}_r , \mathcal{G}_δ , and \mathcal{G}_y).

For a given training set (\mathbf{X}, \mathbf{Y}) , this nonlinear function F can be estimated probabilistically using the GPR trained for the training set (\mathbf{Z}, \mathbf{Y}) . The training input matrix \mathbf{Z} is constructed such that its N th row, denoted by \mathbf{z}_N , is given by

$$\mathbf{z}_N = [y_{N-w_y+1}, \dots, y_N, \mathbf{x}_{N-w_x+1}^T, \dots, \mathbf{x}_N^T]. \quad (26)$$

4.2. NARX-Net

The NARX basis discussed in Section 4.1.2 can be implemented using a neural network as the regressor function, as opposed to a polynomial or GP regressor (Xie *et al.* 2009). In this approach, the predictor basis comprises lags of both autoregressive and exogenous terms, which are then processed through a ML model to predict future time steps of the output. Specifically, the function F in Equation (25) is estimated using a neural network model, which may be a fully connected feed-forward network with multiple layers, activation functions, dropout, and numerous parameters in both the neuron weights and biases.

Unlike the immediate explicit solution of model parameters, which is typical in ordinary least squares regression and GPR, due to the high nonlinearity of the network function, iterative training on a subset of the data is now necessary. This training involves optimization schemes commonly used in the ML/AI community, such as ADAM, RMSProp, Adagrad, and others.

The choice and application of this modeling methodology are motivated by its utilization of potentially sophisticated neural network architectures, which can capture complex patterns and data features, as well as its traditional aspect of establishing a relationship between previous and current values in an autoregressive sense. This characteristic is favorable for physics-based problems that are typically governed by differential equations (Liu *et al.* 2022). Furthermore, this method serves as a suitable middle ground in terms of computational complexity when compared to GP-NARX and more intensive ML methods, such as LSTM or gated response unit (GRU), when assessing its performance.

4.3. Long short-term memory

As mentioned in Section 2.3.1, we recall that LSTM is an RNN-based algorithm that is especially useful for learning sequences of data with long-term dependencies. Apart from using LSTM to extract features from high-dimensional runoff data, we also utilize LSTM to build a hybrid discharge forecasting model. In this section, we present the general description of the LSTM algorithm and its application for hybrid modeling.

Figure 6 illustrates an LSTM cell. A standard LSTM cell consists of an input gate, a forget gate, and an output gate. The input gate integrates new information, which includes the previously observed value from the current iteration, the LSTM unit's output, and the current input. The forget gate determines whether to retain or discard this information. The output gate regulates how to compute the values for the next neuron (Yu *et al.* 2019).

As illustrated in Figure 6, multiple LSTM cells are connected together to map an input sequence $\mathbf{x} = [x_{t-1}, x_t, x_{t+1}, \dots]$ to an output sequence $\mathbf{h} = [h_{t-1}, h_t, h_{t+1}, \dots]$. In each LSTM cell, the operations in the three gates are explained as follows.

4.3.1. Input gate

At a specific time step t , the input gate adds new information from the input x_t to the current cell state, scaling it by the amount we want to update the cell state, as follows:

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \\ \tilde{C}_t &= \tanh(W_c \cdot [h_{t-1}, x_t] + b_c). \end{aligned} \quad (27)$$

Here, $\sigma(\cdot)$ and $\tanh(\cdot)$ are, respectively, a sigmoid function and a hyperbolic tangent function with weights W_i and W_c , and biases b_i and b_c .

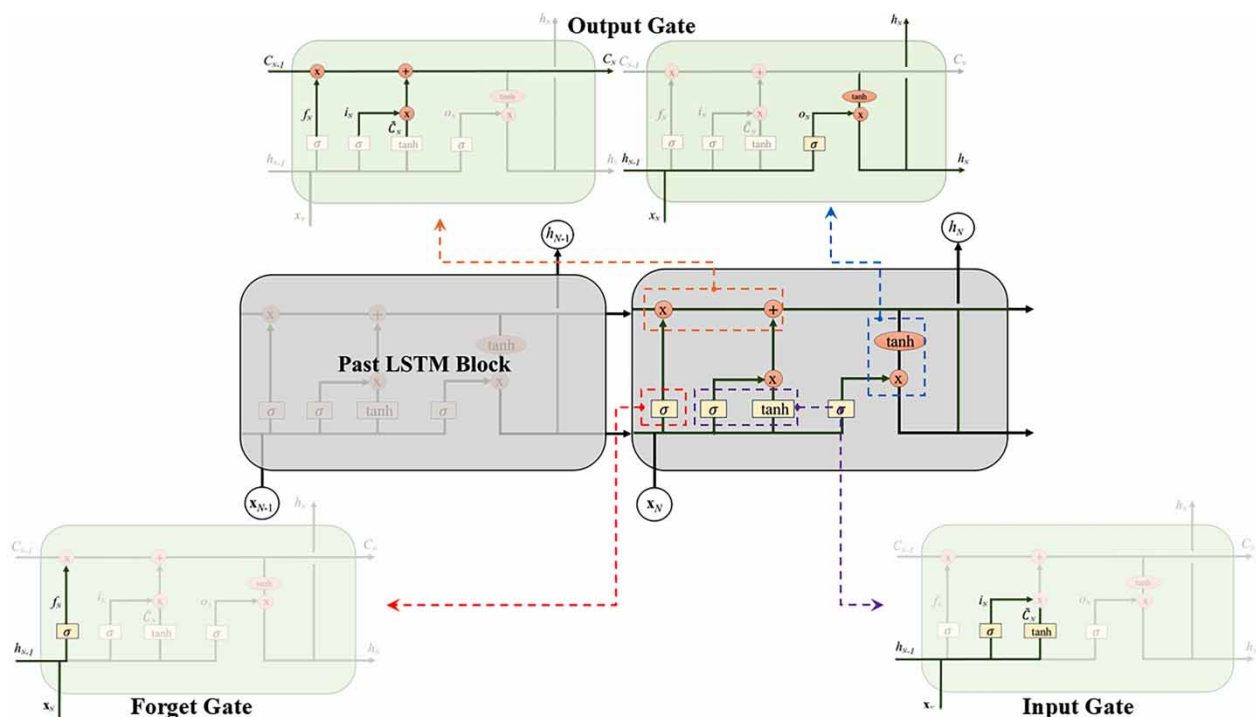


Figure 6 | Schematic diagram of an LSTM cell.

4.3.2. Forget gate

The forget gate, employing the sigmoid function, processes the input data before outputting a vector of values between 0 and 1, which determines whether to retain or discard information. The LSTM unit uses this vector to decide which data from its previous hidden state h_{t-1} should be forgotten. When the value in the sigmoid function's vector is 0, it means that the LSTM forgets the corresponding information from the previous hidden state, and when it is 1, the information is retained and the gate outputs the value f_t such that

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \quad (28)$$

where W_f and b_f denote the weight and bias, respectively, used in the sigmoid function to obtain the value of f_t .

4.3.3. Output gate

The output gate first aggregates cell state C_{t-1} from the previous cell, the data f_t from forget gate, and data i_t and \tilde{C}_t from the input gate to modify the results for the newly allocated cell state C_t as follows:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t, \quad (29)$$

where $*$ stands for Hadamard product operation.

Finally, the outcomes of this LSTM cell can be found through the Hadamard product operation, together with refreshed cell state (at the scale between -1 and 1) and output gate findings, such that

$$\begin{aligned} o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \\ h_t &= o_t * \tanh(C_t), \end{aligned} \quad (30)$$

where W_o and b_o denote the weight and bias used in the sigmoid function to obtain the value of o_t .

The hidden states h_t are ultimately transformed into the output y_t through a fully connected layer. In a conventional LSTM topology, groups of neurons are organized into layers, comprising the input layer, output layer, and multiple hidden layers. The input layer receives input from external sources, while the output layer transmits information to the surroundings. The hidden layers, consisting of several levels, do not have a direct connection to the external physical world but are linked to the input and output layers. In the multilayer LSTM architecture used in this article, feed-forward neural networks are constructed by stacking cells across multiple hidden levels.

4.4. Bidirectional long short-term memory

BiLSTM is a modification of the conventional LSTM. In 2005, Graves and Schmidhuber introduced bidirectional training to an LSTM network for the first time (Graves & Schmidhuber 2005). The fundamental concept behind BiLSTM is to train an additional set of LSTM cells in the reverse direction. Each unit's output, denoted as h_t , results from the combination of both forward and reversed subunits, represented as \overrightarrow{h}_t^p and \overleftarrow{h}_t^p (Ma *et al.* 2020). If we remove the backward LSTM component, this configuration simplifies into a conventional forward LSTM, illustrated in Figure 7. Likewise, removing the forward LSTM yields a conventional LSTM with a reversed time axis (Schuster & Paliwal 1997).

The production of the hidden gate in BiLSTM is achieved by stacking two BiLSTM layers, as shown in Equation (31). The integration of both forward and backward directions within the same network enables the prompt utilization of input data from both past and future time frames to minimize errors efficiently.

$$h_t^p = WL[\overrightarrow{h}_t^p, \overleftarrow{h}_t^p], \text{ where, } p = 1, \dots, n, \quad (31)$$

where h_t^p is the p th BiLSTM hidden state signal at time t , n is the number of cells, \overrightarrow{h}_t^p is the BiLSTM layer's forward direction algorithm, and \overleftarrow{h}_t^p is the BiLSTM layer's backward orientation algorithm. Here, WL is the function that combines the forward and backward computation procedures.

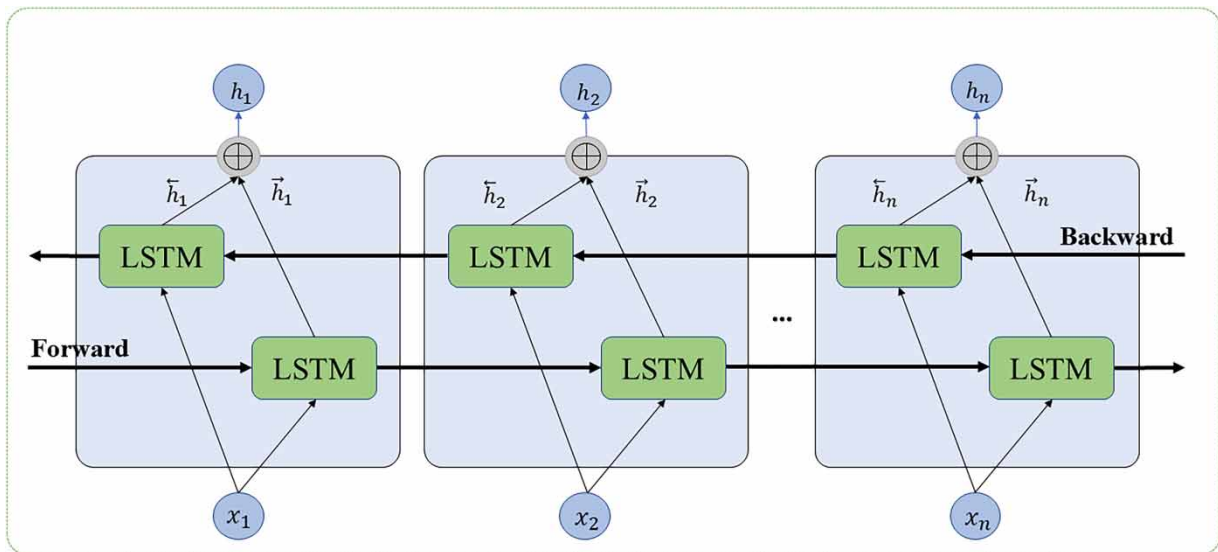


Figure 7 | Architecture of bidirectional LSTM.

5. CASE STUDY

5.1. Basin and rivers of interest

As discussed by *Zhao et al. (2023)*, the United States is primarily divided into seven catchment zones. We focus our attention on a basin located in Colorado, as shown in *Figure 8*. This basin is a part of catchment zone 6. This particular basin is home to 431 watersheds with a river passing through each of them. Each river is assigned a unique river ID, and there is a corresponding shape file (.shp file) representing the watershed in the basin. Among these 431 rivers, six have gauges that measure and record discharge data. Our goal is to forecast the discharge of these six rivers. *Figure 9* illustrates the river network within the basin of interest. The red lines depict rivers crossing the watersheds, while the dark blue watersheds indicate the locations containing gauged rivers. Our objective is to demonstrate the applicability of the proposed forecasting algorithms in predicting the discharge of these rivers. This particular basin is chosen to demonstrate the proposed approaches for three main reasons: (1) the runoff data of this basin has been verified using a benchmark dataset, which allows us to isolate model uncertainty in runoff data from the studied problem; (2) this basin contains a variety of rivers, including rivers with low discharge rates, rivers with high discharge rates, and a human-controlled river with a dam; and (3) this basin has sufficient historical data, allowing for the training and testing of the hybrid modeling methods developed in this article. It should be noted that even though the proposed approaches are demonstrated using this particular basin, their application is not limited to this basin.

To develop the hybrid model, three datasets are required, as detailed in Section 2.1. The first and second datasets pertain to gauge measurements and discharge predictions produced by the RAPID model for the six gauged rivers, while the third

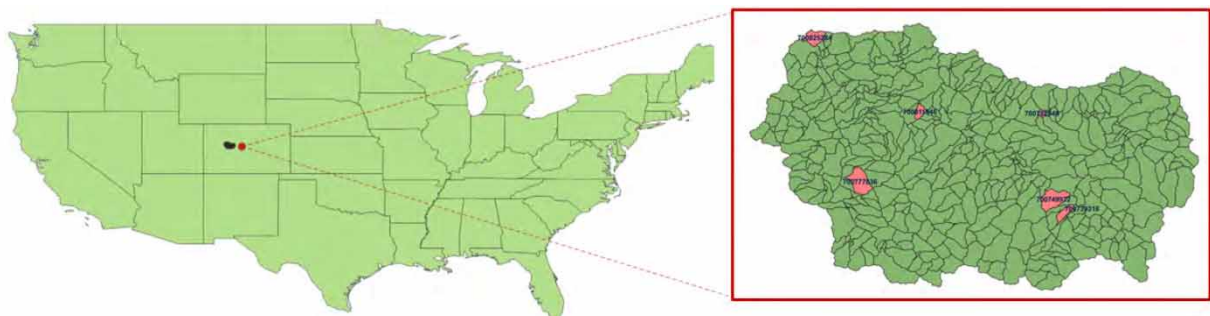


Figure 8 | Basin of interest.

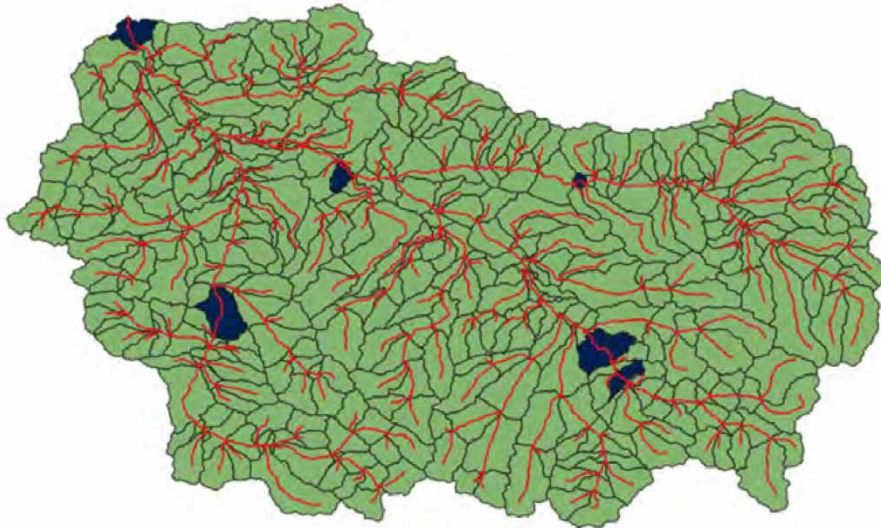


Figure 9 | River network (marked in red) and the watersheds corresponding to the six gauged rivers (colored in blue).

dataset consists of runoff data for all 431 rivers in the basin, and all these datasets span 5,311 days. Specifically, the dimensions of the gauge measurements \mathbf{Y} , the RAPID prediction vector \mathbf{R} , and the runoff data \mathbf{M} are 1×5000 , 1×5000 , and 431×5000 , respectively. The six gauged rivers for which we aim to forecast discharge are identified by the following USGS river IDs: 700739316, 700749972, 700777836, 700782048, 700811946, and 700825284. In this article, we refer to these rivers using numerals 1–6, respectively.

5.2. Data preprocessing

We perform the following:

1. *Dimension reduction of the runoff data:* Since these rivers all belong to the same basin and their dynamics are interrelated, it is expected that the runoff data for all 431 rivers will exhibit some correlation. This correlation arises from the interconnected nature of the river network, where the dynamics of each river are influenced by the dynamics of the entire network and vice versa. Consequently, this correlation allows for the reduction of dimensions in the runoff data to extract essential information in a latent space. Reducing the dimensionality of the runoff dataset \mathbf{M} is crucial for enhancing the overall construction of the hybrid model, as it significantly improves computational efficiency. To achieve this, we employ both the single value decomposition (SVD) technique (Section 2.2) and an LSTM-based autoencoder (Section 2.3) for dimension reduction. By employing both of these techniques, we extract 18 essential features from the original dataset. The dimensionality of features obtained through SVD and an LSTM-based autoencoder is the same, consisting of 18 features.

Figure 10 depicts the reconstruction of the runoff data distributed over the entire basin for a particular day using an LSTM-based autoencoder. The top figure displays the original runoff data, while the bottom figure shows the reconstructed runoff dataset obtained through the LSTM-based autoencoder. Each pixel in this figure represents the runoff volume for a specific catchment, with varying colors indicating different volume ranges as listed on the color-coded legend. The darker hues signify larger volumes, and among the 431 catchments, the largest recorded volume is $54,500 \text{ m}^3/\text{s}$. The similarity between the original and reconstructed datasets indicates that the autoencoder efficiently captures the properties of the runoff dataset using just 18 dimensions.

2. *Smoothing of the RAPID prediction data:* The RAPID prediction is smoothed using a Hanning filter with a window size of 65 days (Oppenheim 1999).
3. *Quality assessment and data standardization:* We conduct a thorough data quality assessment to address issues such as mismatched data types, mixed data values, data outliers, and missing data. To ensure standardized data for analysis, the StandardScaler transform in Python is applied to normalize all three datasets. This normalization method ensures that the data are scaled consistently, allowing for accurate outcomes in subsequent studies while limiting the impact of various scales on hybrid model building.

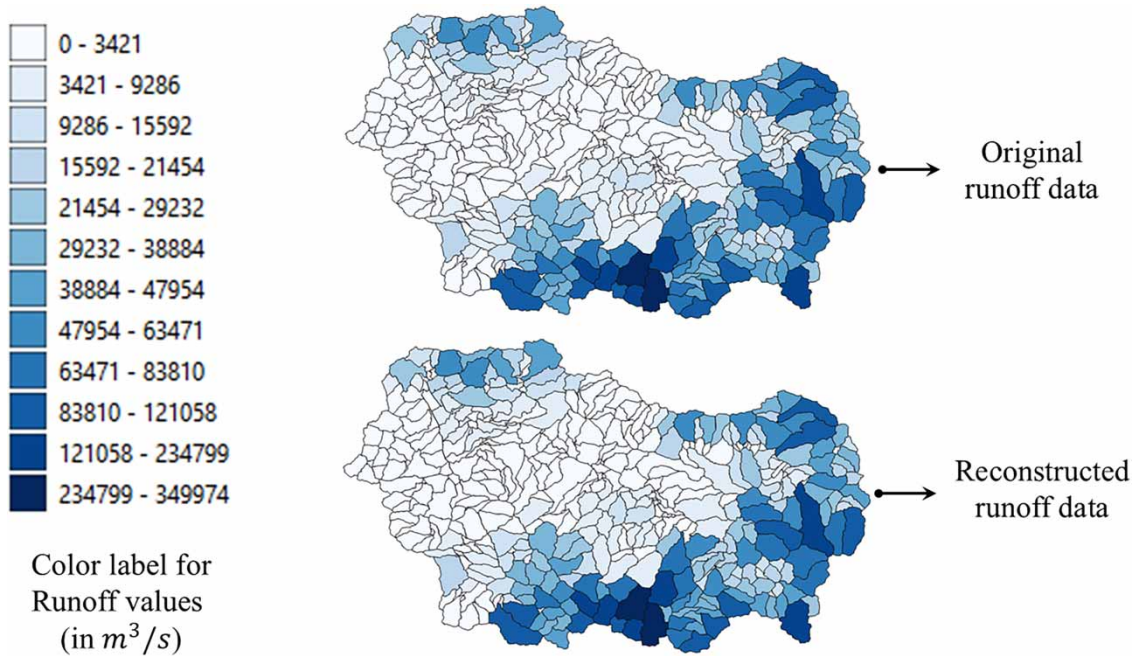


Figure 10 | An instance of an original runoff dataset compared to the reconstructed runoff dataset.

4. *Down sampling the number of days:* We downsampled the data by a factor of 2 for GP-NARX. This was necessary due to GP-NARX's inability to handle large dimensionality, as will be discussed in more detail in Section 5.4. No downsampling was performed for the other methods.

5.3. Performance evaluation metrics

To comprehensively evaluate the performance of the proposed methods, we employed multiple performance metrics, which include the well-acknowledged and widely used Kling–Gupta efficiency (KGE) score and other commonly used accuracy metrics in the ML and hydrology domain. In what follows, we explain the metrics used in this article in detail. The mean-squared error (MSE) and its associated normalization, along with the Nash–Sutcliffe efficiency (NSE), are widely used metrics for the calibration and assessment of hydrological simulations using observable information (Nash & Sutcliffe 1970; Murphy 1988). To compute the MSE, we calculate the squared difference between the model's predictions and the ground truth and then average this value across the entire dataset. Root-mean-square error (RMSE) is obtained by taking positive square root of MSE. The NSE is obtained by dividing the MSE by the overall variability of the data and subtracting the resulting ratio from 1. Let y_i , \hat{y}_i , μ_y , and $\mu_{\hat{y}}$ denote the observed discharge, forecasted discharge, mean of the observed discharge, and mean of the predicted discharge, respectively, and let N_d denote the aggregate quantity of samples tested. The MSE and NSE are defined as follows:

$$\text{MSE} = \frac{1}{N_d} \sum_{i=1}^{N_d} (y_i - \hat{y}_i)^2, \quad (32a)$$

$$\text{RMSE} = \sqrt{\frac{1}{N_d} \sum_{i=1}^{N_d} (y_i - \hat{y}_i)^2}, \quad (32b)$$

$$\text{NSE} = 1 - \frac{\sum_{i=1}^{N_d} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{N_d} (y_i - \mu_y)^2}. \quad (32c)$$

Another commonly used metric to evaluate the similarity between the observed discharge and the predicted discharge is the

KGE (Gupta *et al.* 2009; Kling *et al.* 2012). Let σ_y and $\sigma_{\hat{y}}$ denote the standard deviation of observed and simulated discharge, respectively.

The KGE score is defined as follows:

$$\text{KGE} = 1 - \sqrt{(\rho - 1)^2 + (\beta - 1)^2 + (\gamma - 1)^2},$$

where

$$\begin{aligned} \beta &= \frac{\mu_{\hat{y}}}{\mu_y}, \\ \gamma &= \frac{\sigma_{\hat{y}}\mu_y}{\sigma_y\mu_{\hat{y}}}. \end{aligned} \quad (33)$$

where ρ , β , and γ denote the Pearson correlation coefficient, bias ratio, and variability ratio, respectively.

Since MSE can be obtained by squaring RMSE, and RMSE has the same physical units as the predictions, we report RMSE in addition to KGE and NSE error metrics.

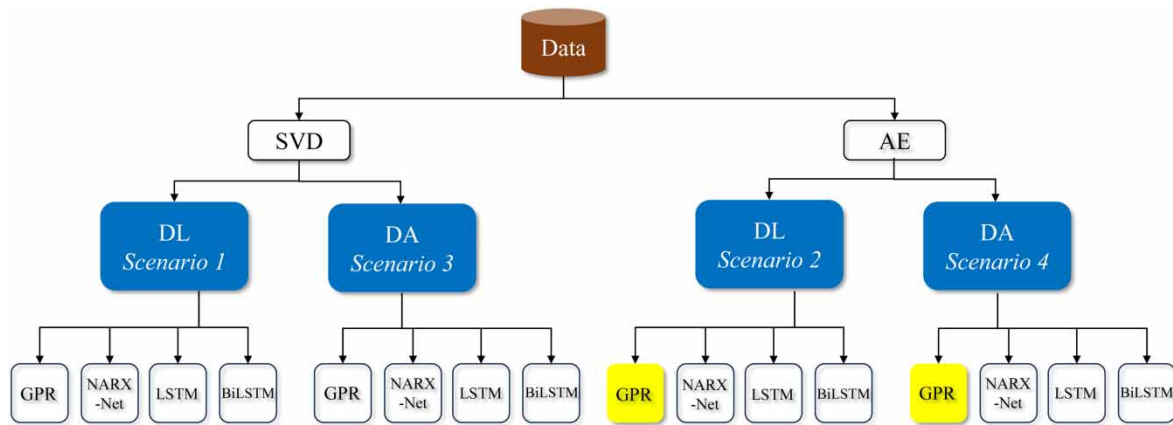
5.4. Numerical results of hybrid forecasting models

5.4.1. Prediction results

We recall that our aim is to forecast the discharge of six gauged rivers in the chosen basin by using two different algorithms: delta learning (Section 3.1) and data augmentation (Section 3.2). These algorithms are built using four different ML algorithms as detailed in Section 4. In addition, two different feature extraction techniques, namely, SVD and LSTM-based autoencoder, are used to reduce the dimensionality of the runoff data, as detailed in Sections 2.2 and 2.3. This gives us four combinations of algorithm and two feature extraction scenarios for the runoff data: (1) *Scenario 1*: delta learning algorithm combined with runoff data features extracted using SVD; (2) *Scenario 2*: delta Learning algorithm combined with runoff data features extracted using LSTM-based autoencoder; (3) *Scenario 3*: data augmentation algorithm combined with runoff data features extracted using SVD; and (4) *Scenario 4*: data augmentation algorithm combined with runoff data features extracted using LSTM-based autoencoder. We call these the *four scenarios of hybrid models*, and these are implemented using different ML methods as illustrated in Figure 11. We present prediction results for all four of these scenarios, implemented using various ML approaches discussed in Section 4.

LSTM, BiLSTM, and NARX-Net are employed to forecast river discharge for six gauged rivers in all four scenarios mentioned earlier. The key parameters of the models are tuned through a grid search method. Four different dropout rates, 0.001, 0.005, 0.01, and 0.1, are used. Four different batch sizes (16, 32, 64, and 128) are also compared to determine the best batch size. The studied number of layers are 2, 3, 4, and 5. The studied learning rates are 0.0001, 0.001, 0.01, and 0.1. The studied number of neurons are 30, 40, 60, 80, and 100. The optimal hyperparameters for the models are as follows: dropout rate, 0.01; batch size, 32; number of hidden layers, 3; learning rate, 0.001; and number of neurons, 80. LSTM and BiLSTM use a past window size of $w_r = w_m = 4$. NARX-Net utilizes a window size of $w_r = 3$ and $w_m = 2$ for data augmentation. For delta learning, it employs a window size of $w_r = 6$ and $w_m = 2$. However, GPR is used only for *Scenario 1* and *Scenario 3*, which use SVD to reduce runoff data dimensionality, with the window size of $w_r = w_m = 4$ days. GPR is sensitive to high-dimensional input, and we sidestep this challenge by using a shorter lag time and by utilizing only the most important runoff features obtained by SVD. Although SVD reduced the dimensionality of runoff data from 431 rivers to just 18 values, most of the information is contained in the first few terms. We exploit this fact and use only the first two terms of the 18 features extracted by SVD to build the GPR model.

To limit the length of this article, we illustrate the prediction results for Rivers 1 and 6 in the four scenarios discussed earlier. Rivers 1 and 6 are chosen as representative examples, with River 1 having low discharge relative to River 6, which has a much higher discharge value. Figures 12, 13, 14, and 15 represent the predictions vs. true discharge measurements for *Scenarios 1* to *4*, respectively. We note that all rivers are free-flowing except for River 4, which has a dam. Figure 16 illustrates the predictions vs. true discharge measurements for River 4, considering both delta learning and data augmentation with runoff data reduced using SVD (*Scenario 1* and *Scenario 3*). The performance metrics of predicted discharge are depicted in Figures 17, 18, and 19. These figures illustrate the evaluation of various hybrid models proposed in this article, along with RAPID predictions, in comparison to the true measured discharge. The evaluation is conducted using metrics such as



DL: Delta learning

DA: Data augmentation

SVD: Singular value decomposition

AE: Autoencoder

Note: GPR is sensitive to high-dimensional input and is not suitable in the case of Autoencoder-based data reduction (indicated by the yellow box in this flowchart), where the reduced latent space needs to be fully utilized. This is in contrast to SVD, where the first few terms are the most important information holders. That is why GPR is used exclusively for *Scenario 1* and *Scenario 3*.

Figure 11 | Flowchart showing different combinations of data reduction techniques and ML methods exploited to build the hybrid prediction models.

RMSE, NSE, and Kling–Gupta efficiency (KGE) for a comprehensive analysis. Lower RMSE values indicate higher accuracy of the predictions, while higher KGE and NSE scores suggest better predictive performance.

5.4.2. Discussion

Table 2 shows the best-performing ML method for all the rivers and various scenarios of algorithms used in hybrid modeling. We make the following observations based on these analysis results:

1. *Improved overall prediction results relative to the original RAPID model:* Based on the error plots for RMSE, GPR-based hybrid algorithms (Scenario 1 and Scenario 3) outperforms RAPID 100% of the time. LSTM- and BiLSTM-based algorithms outperform RAPID 91.667% of the time for delta Learning and 100% of the time for data augmentation. Narx-Net-based hybrid algorithms outperform RAPID 100% of the time for all the four scenarios of hybrid models. This clearly indicates that the predictions of the hybrid algorithms perform better than the RAPID predictions. That is, the ML component in the hybrid model successfully retrieves the information that has not been modeled in the original RAPID model.
2. *Data augmentation vs. delta learning:* It is concluded from the error plots that data augmentation is overall a more robust algorithm than delta Learning. Overall, all the ML methods perform marginally better with data augmentation relative to delta Learning.
3. *Predictions in human-controlled river with a dam:* As discussed earlier, River 4 is an anomaly in that it has a dam that obstructs the flow. Consequently, the true measured discharge is almost static and low (near zero). The RAPID model, unable to incorporate this information, produces a noisy prediction of a large positive discharge. This particular river, however, reveals a very interesting behavior of the delta learning and data augmentation algorithm. Since the delta learning algorithm involves evaluating the error between smoothed RAPID and the true discharge, it is highly sensitive to the deviation of RAPID's predictions relative to the true value. In cases where the true discharge measurements are near zero, the delta terms become almost equal to smoothed RAPID predictions. As a result, the delta learning algorithm's predictions are influenced by RAPID's measurements and follow a similar pattern to RAPID's predictions. On the other hand, the data augmentation algorithm does not have this sensitivity to RAPID predictions and yields a prediction pattern closer to the true discharge. This also justifies the overall better behavior of data augmentation relative to the delta learning algorithm (the first observation).

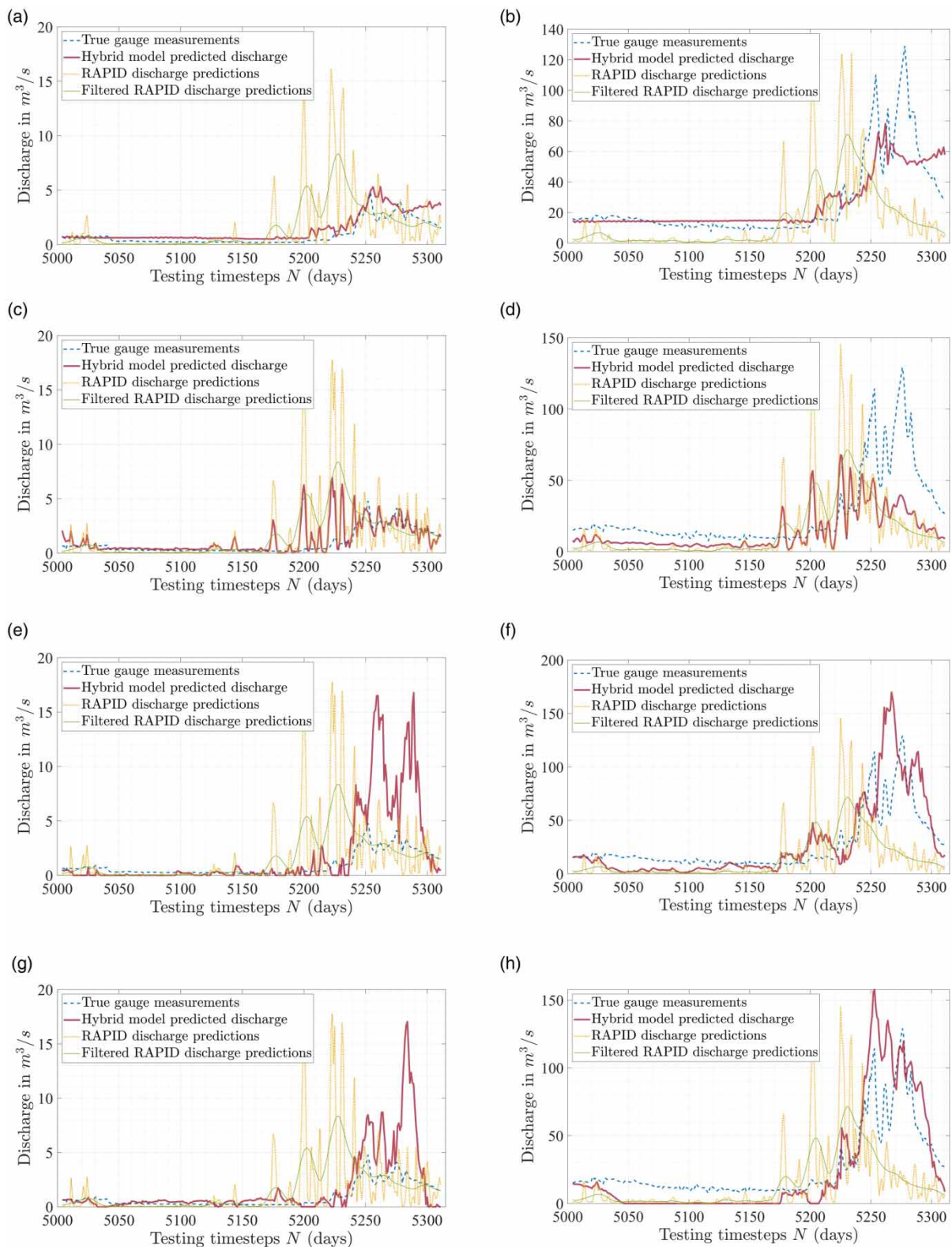


Figure 12 | Predicted vs. true measured discharge of rivers 1 and 6 obtained using delta learning algorithm considering runoff data reduced using SVD technique (Scenario 1). (a) River 1; ML method: GPR, (b) River 6; ML method: GPR, (c) River 1; ML method: NARX-net, (d) River 6; ML method: NARX-net, (e) River 1; ML method: LSTM, (f) River 6; ML method: LSTM, (g) River 1; ML method: BiLSTM, and (h) River 6; ML method: BiLSTM.

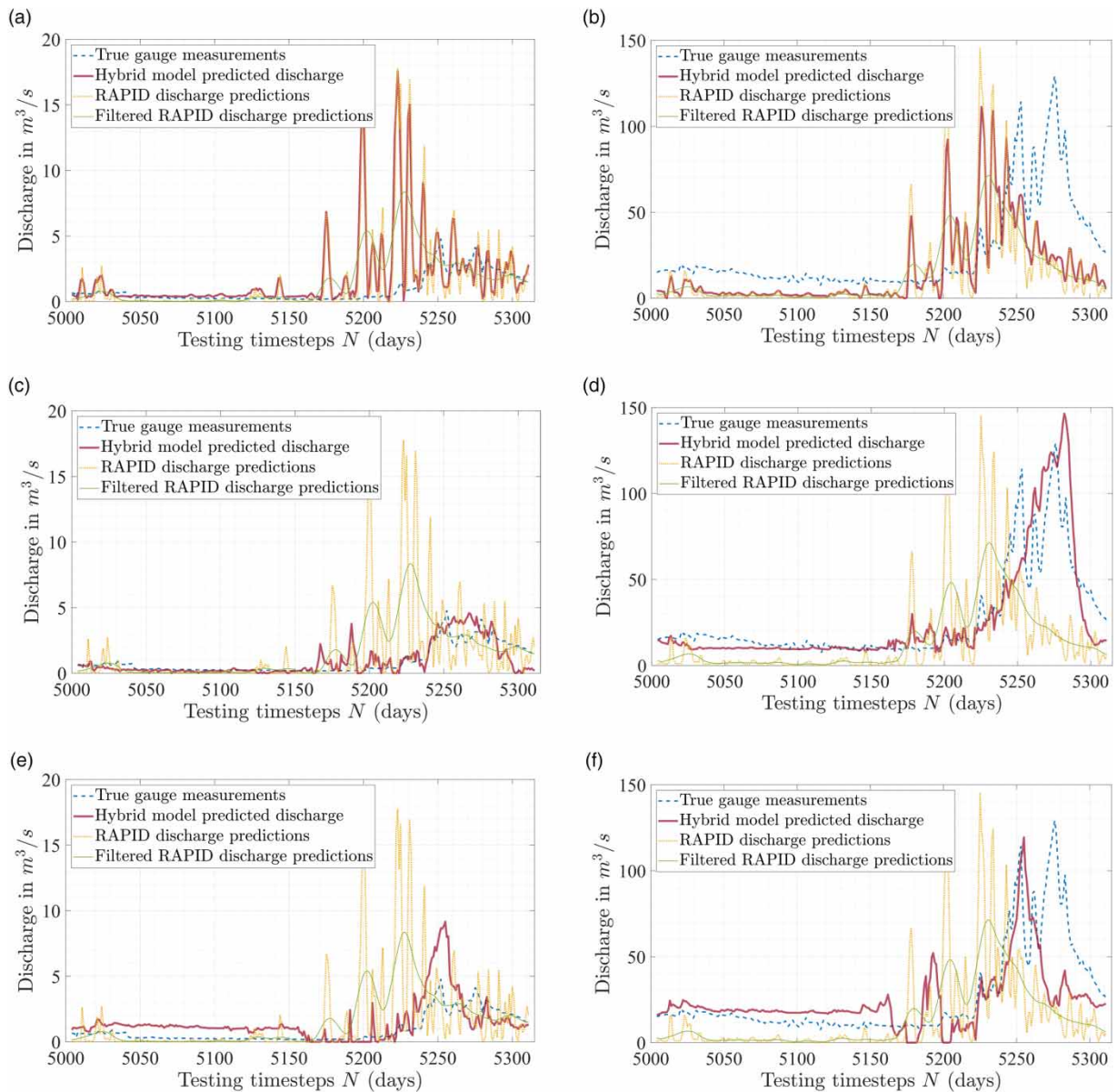


Figure 13 | Predicted vs. true measured discharge of Rivers 1 and 6 obtained using delta learning algorithm considering runoff data reduced using LSTM-based autoencoder (Scenario 2). (a) River 1; ML method: NARX-net, (b) River 6; ML method: GPR, (c) River 1; ML method: LSTM, (d) River 6; ML method: LSTM, (e) River 1; ML method: BiLSTM, (f) River 6; ML method: BiLSTM.

4. *Impact of feature extraction techniques on predictions:* LSTM performs better with the delta learning algorithm that utilizes autoencoder-based runoff features relative to the SVD-based runoff features. Contrarily, the data augmentation algorithm that uses SVD runoff features performs better relative to the autoencoder-based runoff features.
5. *Performance of different ML algorithms in different scenarios:* GPR is the overall best performer for Scenario 1 of the hybrid model; LSTM for Scenario 2; BiLSTM for Scenario 4. In Scenario 3, different methods perform the best for different rivers.
6. *Training cost and complexities of different ML algorithms:* BiLSTM is harder to train than LSTM. Therefore, even though LSTM and BiLSTM perform relatively similarly, LSTM has an edge in terms of training ease. Ignoring River 4, overall, GPR marginally outperforms both LSTM and BiLSTM, even though it is a less complex architecture in terms of required training resources. This makes GPR a powerful method to support the hybrid model. However, GPR suffers from the curse

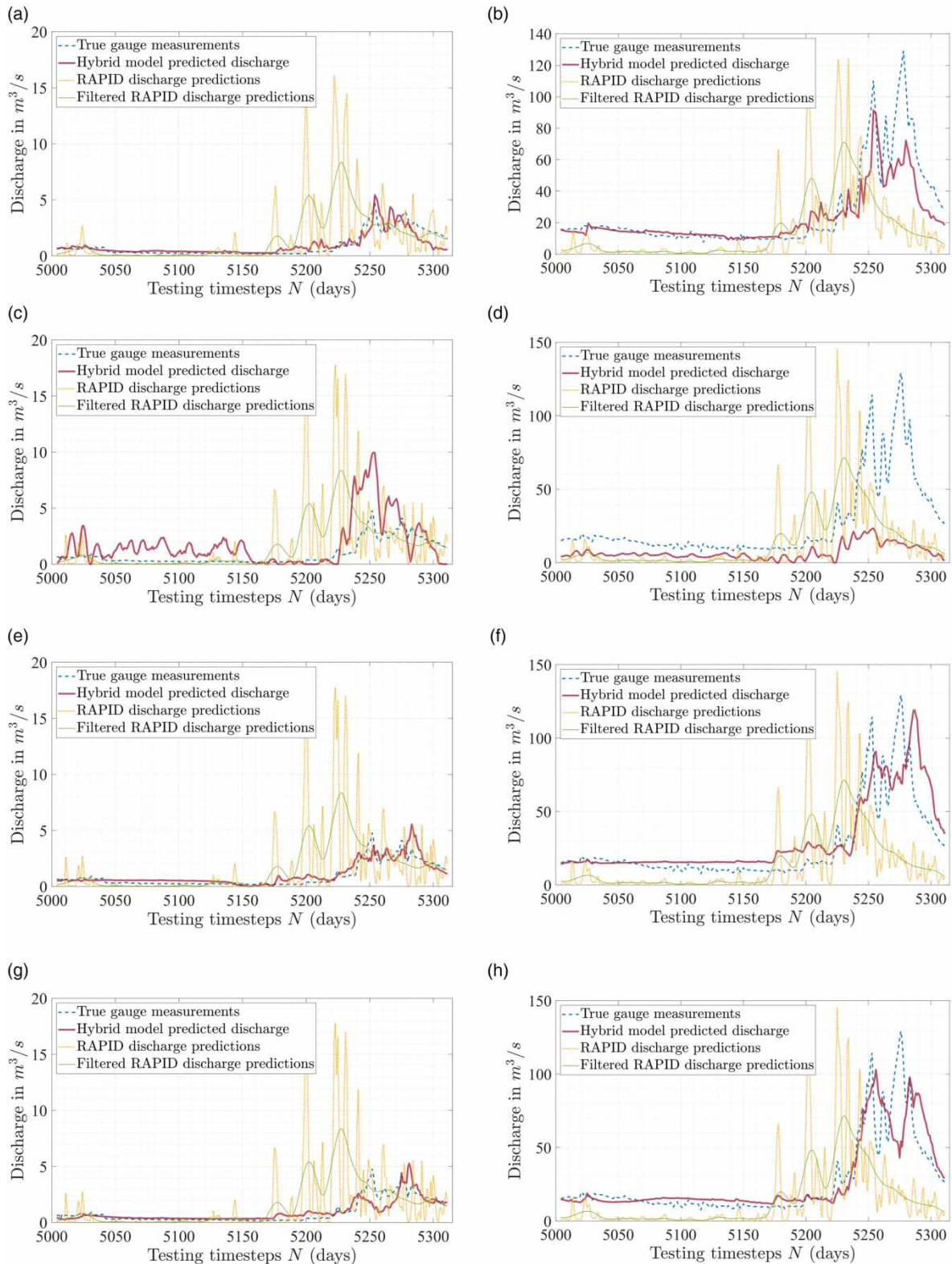


Figure 14 | Predicted vs. true measured discharge of Rivers 1 and 6 obtained using data augmentation algorithm considering runoff data reduced using SVD technique (Scenario 3). (a) River 1; ML method: GPR, (b) River 6; ML method: GPR, (c) River 1; ML method: NARX-net, (d) River 6; ML method: NARX-net, (e) River 1; ML method: LSTM, (f) River 6; ML method: LSTM, (g) River 1; ML method: BiLSTM, and (h) River 6; ML method: BiLSTM.

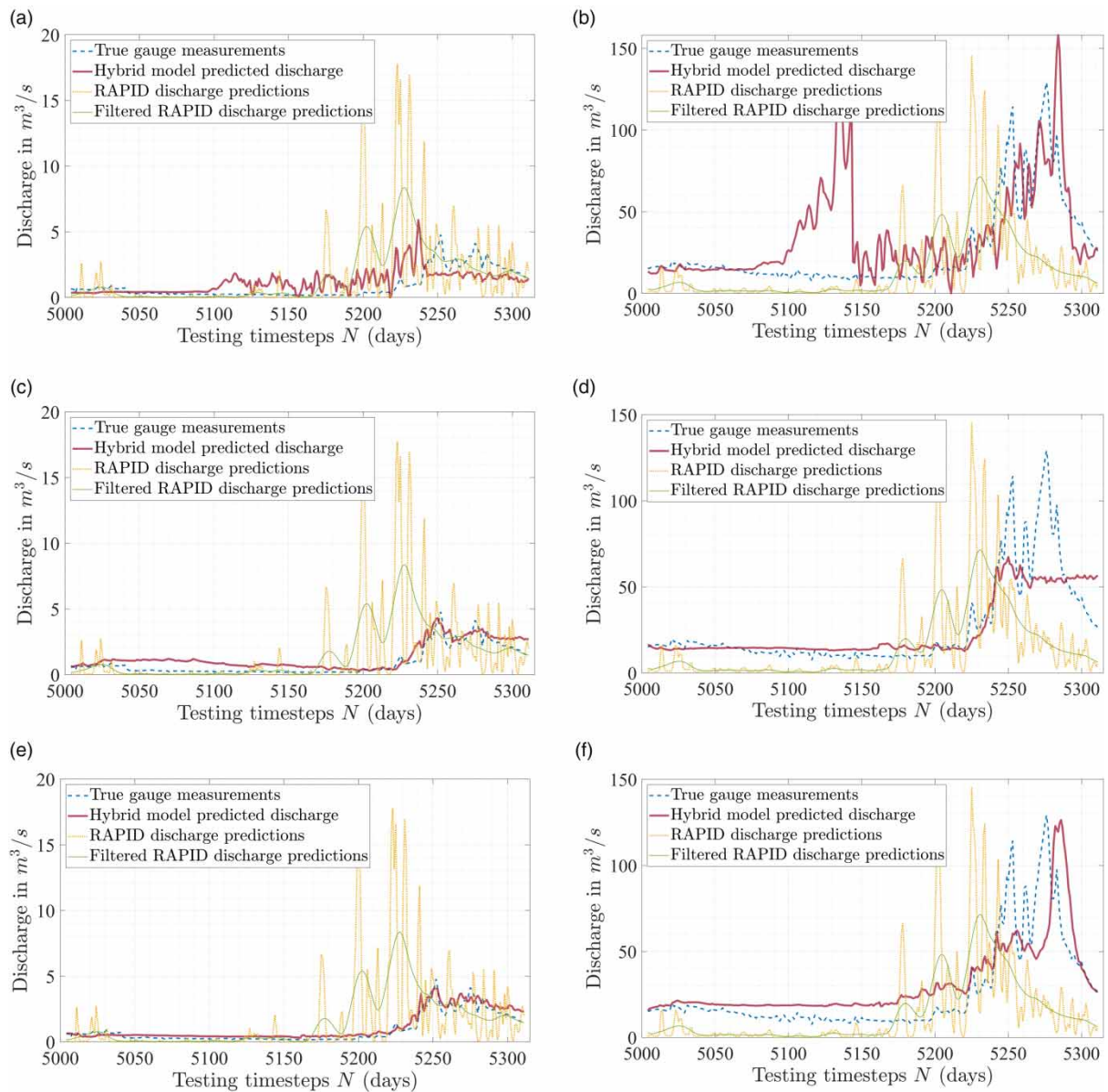


Figure 15 | Predicted vs. true measured discharge of Rivers 1 and 6 obtained using data augmentation algorithm considering runoff data reduced using LSTM-based autoencoder (Scenario 4). (a) River 1; ML method: NARX-net, (b) River 6; ML method: NARX-net, (c) River 1; ML method: LSTM, (d) River 6; ML method: LSTM, (e) River 1; ML method: BiLSTM, and (f) River 6; ML method: BiLSTM.

of dimensionality, as discussed earlier. That is, it cannot handle larger input dimensions and is, hence, less informed than LSTM and BiLSTM.

7. *Performance of NARX-Net*: The NARX-Net method of modeling with delta learning consistently demonstrated an overly strong correlation with the RAPID prediction. This trend was also observed in other modeling techniques, albeit to a lesser extent. Two possible explanations arise: there may be an excessive bias toward RAPID as the sole predictor for the output, implying that the utilization of runoff data and historic gauge information was insufficient to rectify the exclusive reliance on RAPID. Alternatively, and more likely, a very small and inadequate delta term is being generated and added back to the original estimation as a correction. This phenomenon could be attributed to the presence of noisy, erratic, and higher frequency variance in the training data, as is the case in some instances of RAPID and gauge data. Models, when exposed to

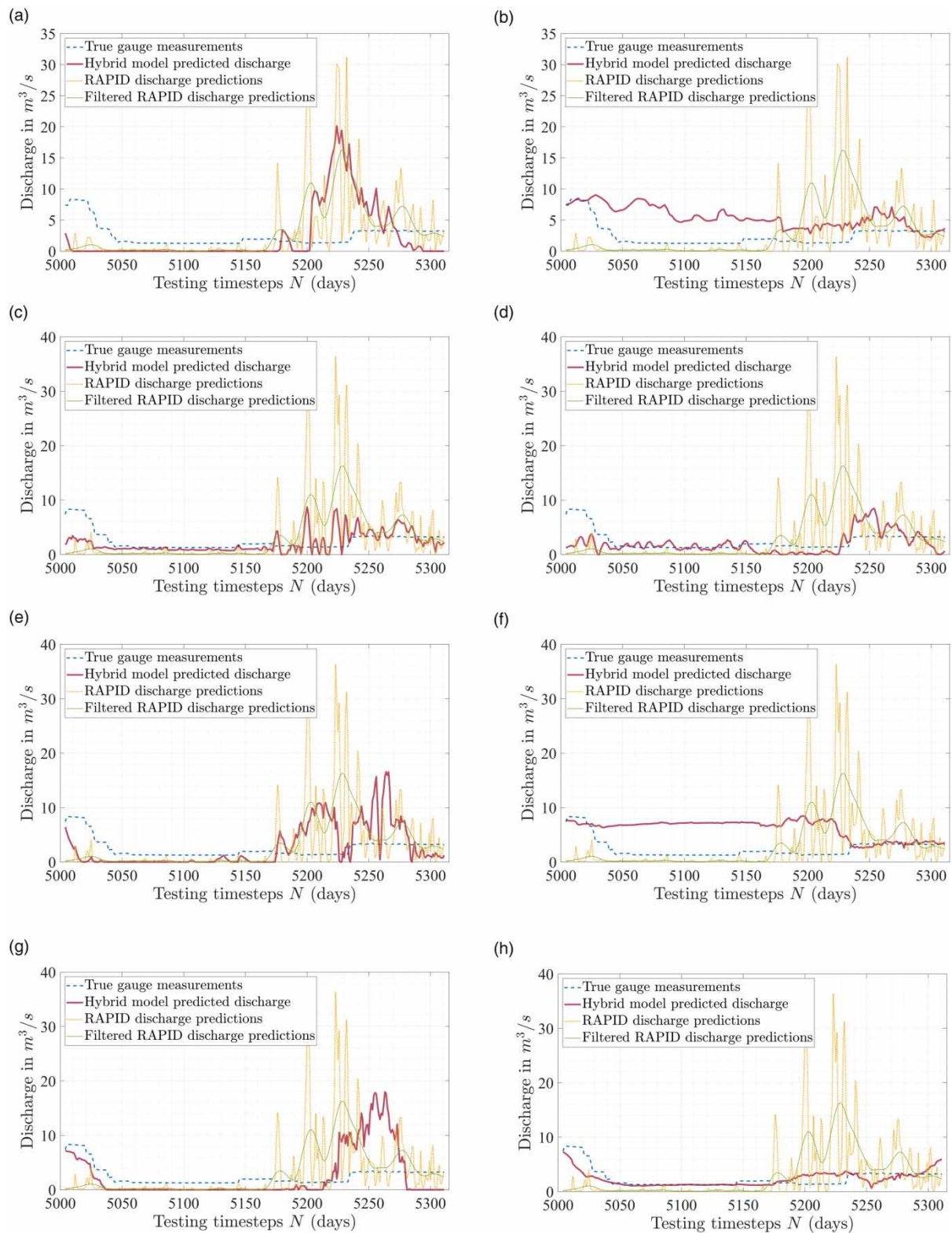


Figure 16 | Predicted vs. true measured discharge of River 4, which has a dam. (a) Delta learning implemented with GPR, (b) data augmentation implemented with GPR, (c) delta learning implemented with NARX-Net, (d) data augmentation implemented with NARX-Net, (e) delta learning implemented with LSTM, (f) data augmentation implemented with LSTM, (g) delta learning implemented with BiLSTM, and (h) data augmentation implemented with BiLSTM.

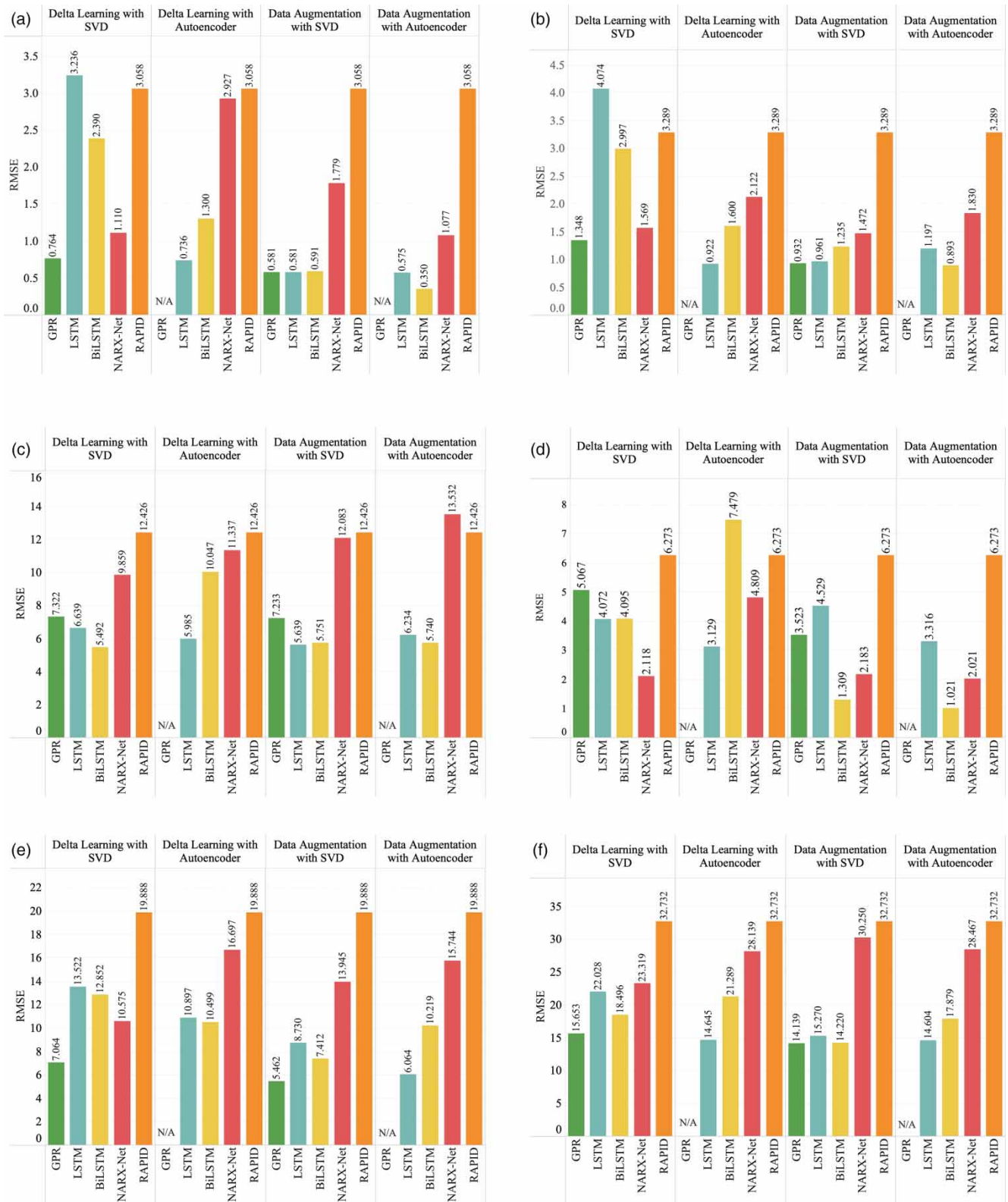


Figure 17 | The RMSE for hybrid models and RAPID predictions relative to the true measured discharge. (a) RMSE for River 1, (b) RMSE for River 2, (c) RMSE for River 3, (d) RMSE for River 4, (e) RMSE for River 5, and (f) RMSE for River 6.

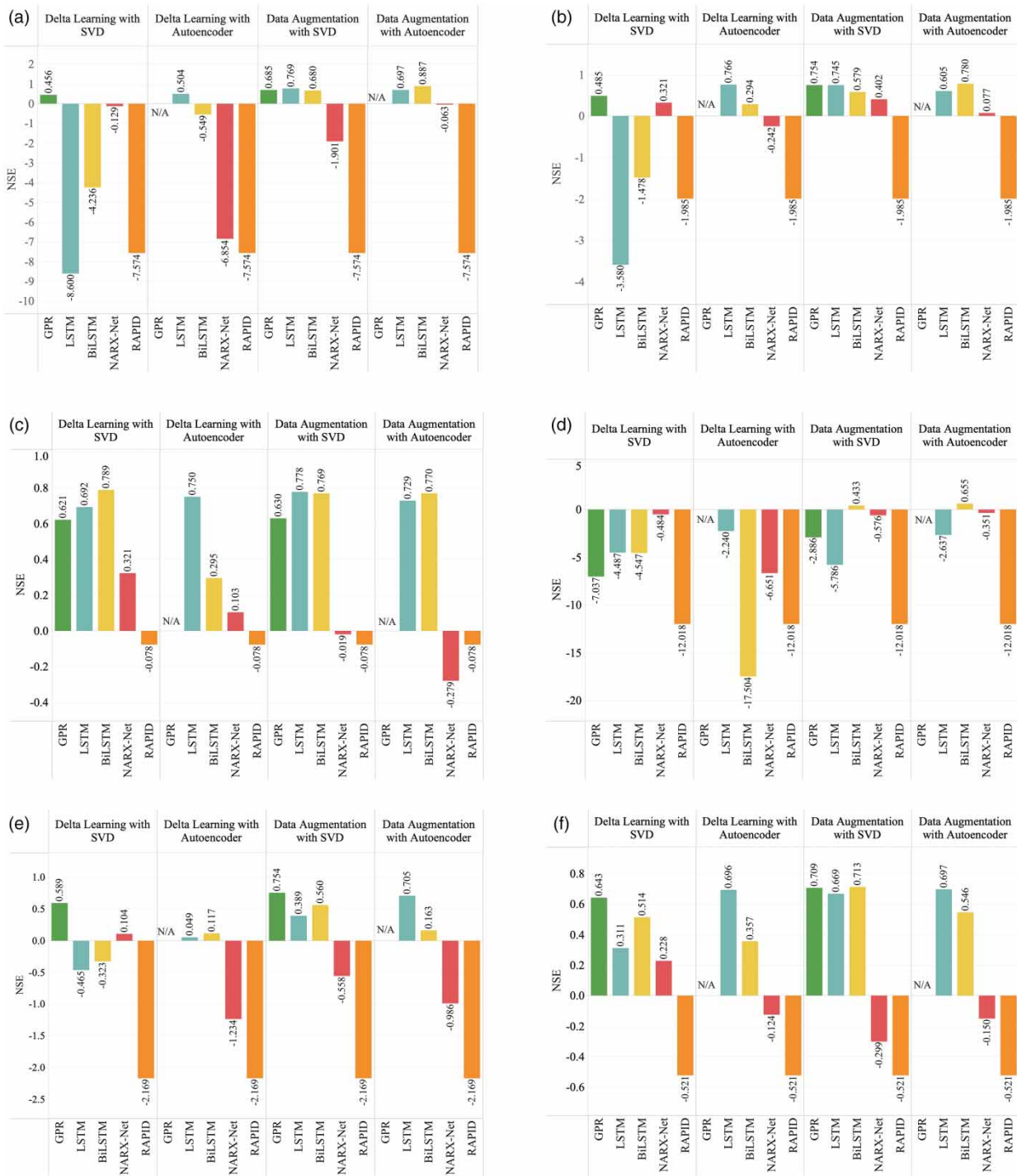


Figure 18 | The NSE for hybrid models and RAPID predictions relative to the true measured discharge. (a) NSE for River 1, (b) NSE for River 2, (c) NSE for River 3, (d) NSE for River 4, (e) NSE for River 5, and (f) NSE for River 6.

such data, may undesirably learn the variance through potential overfitting, resulting in a biased model that either inappropriately predicts noise or abandons the natural features in the data, defaulting to a smooth or flat average. In the case of NARX-Net, it appears that, during training, the model produces a very small delta term that provides minimal correction to the RAPID prediction.

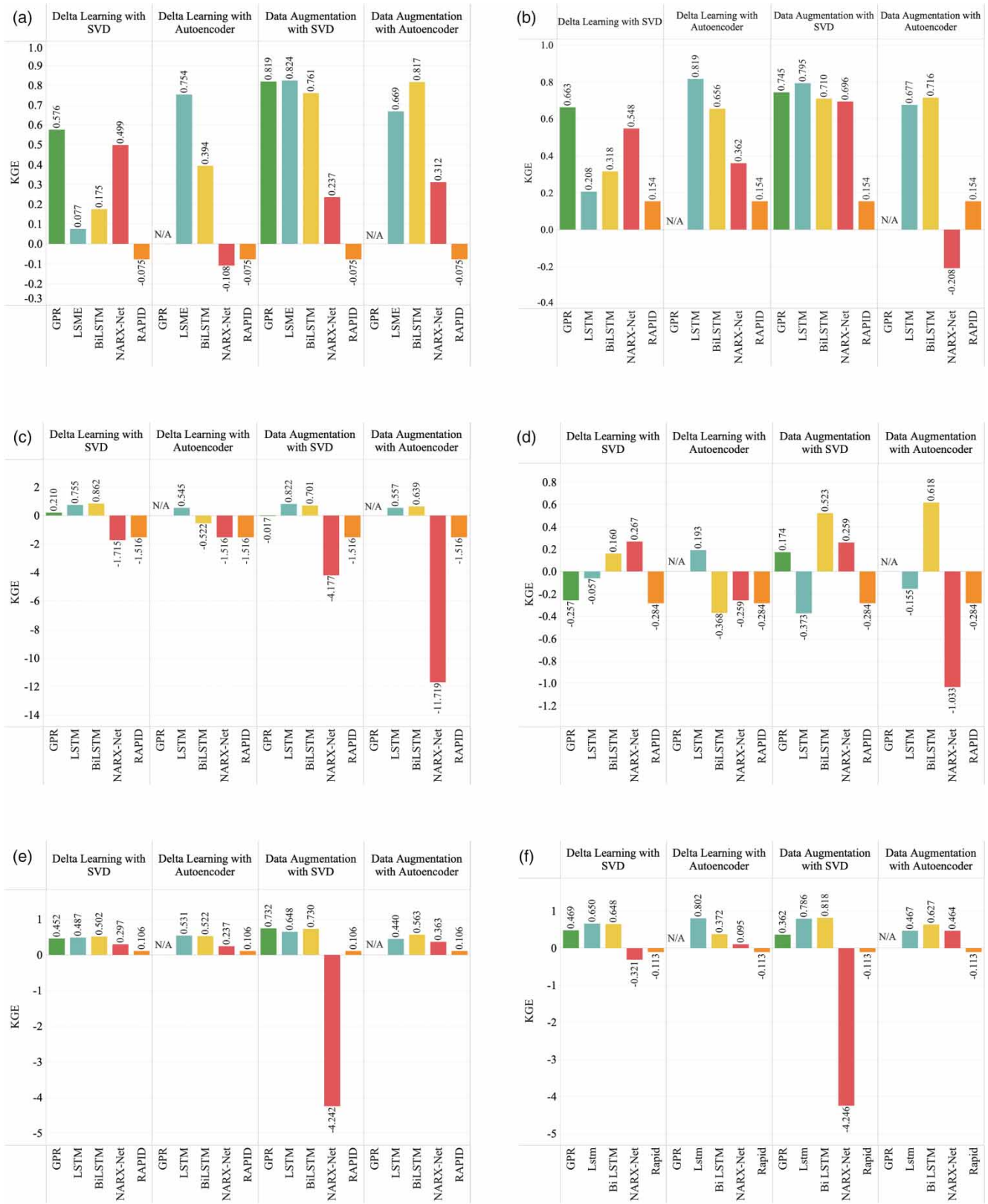


Figure 19 | The KGE for hybrid models and RAPID predictions relative to the true measured discharge. (a) KGE for River 1, (b) KGE for River 2, (c) KGE for River 3, (d) KGE for River 4, (e) KGE for River 5, and (f) KGE for River 6.

Table 2 | Best-performing ML method for various rivers and algorithm scenarios based on the three error metrics

Rivers	Scenario 1 delta learning with SVD runoff features	Scenario 2 delta learning with autoencoder runoff features	Scenario 3 data augmentation with SVD runoff features	Scenario 4 data augmentation with autoencoder runoff features
1	RMSE: GPR	RMSE: LSTM	RMSE: LSTM	RMSE: BiLSTM
	NSE: GPR	NSE: LSTM	NSE: LSTM	NSE: BiLSTM
	KGE: GPR	KGE: LSTM	KGE: LSTM	KGE: BiLSTM
2	RMSE: GPR	RMSE: LSTM	RMSE: GPR	RMSE: BiLSTM
	NSE: GPR	NSE: LSTM	NSE: GPR	NSE: BiLSTM
	KGE: GPR	KGE: LSTM	KGE: LSTM	KGE: BiLSTM
3	RMSE: BiLSTM	RMSE: LSTM	RMSE: BiLSTM	RMSE: BiLSTM
	NSE: BiLSTM	NSE: LSTM	NSE: LSTM	NSE: BiLSTM
	KGE: BiLSTM	KGE: LSTM	KGE: LSTM	KGE: BiLSTM
4	RMSE: NARX-Net	RMSE: LSTM	RMSE: BiLSTM	RMSE: BiLSTM
	NSE: NARX-Net	NSE: LSTM	NSE: BiLSTM	NSE: BiLSTM
	KGE: NARX-Net	KGE: LSTM	KGE: BiLSTM	KGE: BiLSTM
5	RMSE: GPR	RMSE: LSTM	RMSE: GPR	RMSE: LSTM
	NSE: GPR	NSE: BiLSTM	NSE: GPR	NSE: LSTM
	KGE: BiLSTM	RMSE: LSTM	KGE: GPR	KGE: BiLSTM
6	RMSE: GPR	RMSE: LSTM	RMSE: GPR	RMSE: LSTM
	NSE: GPR	NSE: LSTM	NSE: BiLSTM	NSE: LSTM
	KGE: LSTM	KGE: LSTM	KGE: BiLSTM	KGE: BiLSTM

6. SUMMARY AND CONCLUSIONS

This article focuses on building a hybrid data-driven physics-enhanced model to forecast river discharge. The model is based on three data sources: true discharge values measured by gauges, discharge predicted by the physics-based RAPID, and runoff data from all the rivers in the basin. Two cutting-edge algorithms, namely, delta learning and data augmentation, are proposed for hybrid modeling. Building a hybrid model using either of these algorithms requires constructing three different ML models. We exploit four different ML methods, including GP-NARX, NARX-Net, LSTM, and BiLSTM.

The inclusion of runoff data from all rivers within the basin offers a comprehensive view of the dynamics of river flow across the entire surrounding region. The runoff data from neighboring rivers provide insights into the dynamic interactions and inter-connected physics of the broader river network, contributing to more informed discharge forecasts for the specific river of interest. However, considering the runoff data of all the rivers constituting a basin of interest makes the runoff dataset have higher dimensions. We utilize SVD and LSTM-based autoencoder techniques to extract features with reduced dimensionality.

There are four combinations of algorithms and two feature extraction scenarios for the runoff data: (1) *Scenario 1*: Delta Learning algorithm combined with runoff data features extracted using SVD; (2) *Scenario 2*: Delta Learning algorithm combined with runoff data features extracted using an LSTM-based autoencoder; (3) *Scenario 3*: Data augmentation algorithm combined with runoff data features extracted using SVD; and (4) *Scenario 4*: Data augmentation algorithm combined with runoff data features extracted using an LSTM-based autoencoder. Each of these scenarios can be built by exploiting various ML techniques. In the hybrid model, GPR outperforms other methods in Scenario 1, while LSTM excels in Scenario 2, and BiLSTM is the top performer in Scenario 4. In Scenario 3, various methods demonstrate the best performance for different rivers.

This study has demonstrated encouraging results in discharge forecasting using various hybrid models to address this exceedingly complex problem. The performance metrics examined, namely, RMSE, KGE, and NSE, consistently show that hybrid algorithms outperform RAPID. For example, when considering RMSE as the error measure, GPR-based hybrid algorithms (Scenario 1 and Scenario 3) outperform RAPID 100% of the time. LSTM- and BiLSTM-based algorithms outperform RAPID 91.667% of the time for delta learning and 100% of the time for data augmentation. Narx-Net-based hybrid algorithms (Scenario 2 and Scenario 3) also outperform RAPID 91.667% of the time.

One major requirement upon which these models were contingent was the availability of true discharge measurement data. The encouraging results in this study have motivated us to extend this research and build a hybrid model capable of predicting the discharge of ungauged rivers where true discharge measurements are not available. Currently, the developed method is mainly designed for rivers with gauges. The extension of the proposed methods to un-gauged rivers through transfer learning is an interesting direction worth studying in the future. In addition, for rivers with a large volume of discharge data, purely data-driven ML models can be employed. The method developed in this article is more suitable for situations where we have only a limited amount of river discharge data. Another important research direction worth studying is the selection or ensemble of ML models using optimization algorithms. In this article, we compared the performance of different hybrid modeling techniques. These models and techniques can be integrated together to achieve better performance using ensemble learning methods.

ACKNOWLEDGEMENTS

Funding for this work was provided by the <http://dx.doi.org/10.13039/100006752> United States Army Corps of Engineers through the <http://dx.doi.org/10.13039/100006505> U.S. Army Engineer Research and Development Center Research Cooperative Agreement W912HZ-21-C-0042.

DATA AVAILABILITY STATEMENT

All relevant data are included in the paper or its Supplementary Information.

CONFLICT OF INTEREST

The authors declare there is no conflict.

REFERENCES

- Adnan, R. M., Mostafa, R. R., T. Islam, A. R. M., Kisi, O., Kuriqi, A. & Heddham, S. (2021) Estimating reference evapotranspiration using hybrid adaptive fuzzy inferencing coupled with heuristic algorithms, *Computers and Electronics in Agriculture*, **191**, 106541.
- Adnan, R. M., Dai, H.-L., Mostafa, R. R., Parmar, K. S., Heddham, S. & Kisi, O. (2022) Modeling multistep ahead dissolved oxygen concentration using improved support vector machines by a hybrid metaheuristic algorithm, *Sustainability*, **14** (6), 3470.
- Adnan, R. M., Dai, H.-L., Mostafa, R. R., T. Islam, A. R. M., Kisi, O., Heddham, S. & Zounemat-Kermani, M. (2023a) Modelling groundwater level fluctuations by elm merged advanced metaheuristic algorithms using hydroclimatic data, *Geocarto International*, **38** (1), 2158951.
- Adnan, R. M., Mostafa, R. R., Dai, H.-L., Heddham, S., Kuriqi, A. & Kisi, O. (2023b) Pan evaporation estimation by relevance vector machine tuned with new metaheuristic algorithms using limited climatic data, *Engineering Applications of Computational Fluid Mechanics*, **17** (1), 2192258.
- Bates, P. D. & De Roo, A. (2000) A simple raster-based model for flood inundation simulation, *Journal of Hydrology*, **236** (1–2), 54–77.
- Beighley, R. E., Eggert, K., Dunne, T., He, Y., Gummadi, V. & Verdin, K. (2009) Simulating hydrologic and hydraulic processes throughout the Amazon River Basin, *Hydrological Processes: An International Journal*, **23** (8), 1221–1235.
- Branstetter, M. L. (2001) *Development of a Parallel River Transport Algorithm and Applications to Climate Studies*. The University of Texas at Austin, Austin, Texas, USA.
- David, C. H., Habets, F., Maidment, D. R. & Yang, Z.-L. (2011a) RAPID applied to the SIM-France model, *Hydrological Processes*, **25** (22), 3412–3425.
- David, C. H., Maidment, D. R., Niu, G.-Y., Yang, Z.-L., Habets, F. & Eijkhout, V. (2011b) River network routing on the NHDPlus dataset, *Journal of Hydrometeorology*, **12** (5), 913–934.
- David, C. H., Yang, Z.-L. & Famiglietti, J. S. (2013) Quantification of the upstream-to-downstream influence in the Muskingum method and implications for speedup in parallel computations of river flow, *Water Resources Research*, **49** (5), 2783–2800.
- David, C. H., Famiglietti, J. S., Yang, Z.-L. & Eijkhout, V. (2015) Enhanced fixed-size parallel speedup with the Muskingum method using a trans-boundary approach and a large subbasins approximation, *Water Resources Research*, **51** (9), 7547–7571.
- David, C. H., Famiglietti, J. S., Yang, Z.-L., Habets, F. & Maidment, D. R. (2016) A decade of RAPID—reflections on the development of an open source geoscience code, *Earth and Space Science*, **3** (5), 226–244.
- Ducharne, A., Golaz, C., Leblois, E., Laval, K., Polcher, J., Ledoux, E. & de Marsily, G. (2003) Development of a high resolution runoff routing model, calibration and application to assess runoff from the LMD GCM, *Journal of Hydrology*, **280** (1–4), 207–228.

- Follum, M. L., Tavakoly, A. A., Niemann, J. D. & Snow, A. D. (2017) AutoRAPID: A model for prompt streamflow estimation and flood inundation mapping over regional to continental extents, *JAWRA Journal of the American Water Resources Association*, **53** (2), 280–299.
- Frame, J. M., Kratzert, F., Klotz, D., Gauch, M., Shalev, G., Gilon, O., Qualls, L. M., Gupta, H. V. & Nearing, G. S. (2022) Deep learning rainfall–runoff predictions of extreme events, *Hydrology and Earth System Sciences*, **26** (13), 3377–3392.
- Ghimire, G. R., Hansen, C., Gangrade, S., Kao, S.-C., Thornton, P. E. & Singh, D. (2023) Insights from dayflow: A historical streamflow reanalysis dataset for the conterminous United States, *Water Resources Research*, **59** (2), e2022WR032312.
- Graves, A. & Schmidhuber, J. (2005) Framewise phoneme classification with bidirectional LSTM and other neural network architectures, *Neural Networks*, **18** (5–6), 602–610.
- Gupta, H. V., Kling, H., Yilmaz, K. K. & Martinez, G. F. (2009) Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling, *Journal of Hydrology*, **377** (1–2), 80–91.
- Hu, Z. & Mahadevan, S. (2016) Global sensitivity analysis-enhanced surrogate (GSAS) modeling for reliability analysis, *Structural and Multidisciplinary Optimization*, **53** (3), 501–521.
- Hu, Z., Ao, D. & Mahadevan, S. (2017) Calibration experimental design considering field response and model uncertainty, *Computer Methods in Applied Mechanics and Engineering*, **318**, 92–119.
- Ikram, R. M. A., Mostafa, R. R., Chen, Z., Parmar, K. S., Kisi, O. & Zounemat-Kermani, M. (2023) Water temperature prediction using improved deep learning methods through reptile search algorithm and weighted mean of vectors optimizer, *Journal of Marine Science and Engineering*, **11** (2), 259.
- Kan, G., Liang, K., Yu, H., Sun, B., Ding, L., Li, J., He, X. & Shen, C. (2020) Hybrid machine learning hydrological model for flood forecast purpose, *Open Geosciences*, **12** (1), 813–820.
- Kling, H., Fuchs, M. & Paulin, M. (2012) Runoff conditions in the upper danube basin under an ensemble of climate change scenarios, *Journal of Hydrology*, **424**, 264–277.
- Kratzert, F., Klotz, D., Herrnegger, M., Sampson, A. K., Hochreiter, S. & Nearing, G. S. (2019) Toward improved predictions in ungauged basins: Exploiting the power of machine learning, *Water Resources Research*, **55** (12), 11344–11354.
- Liu, Y., Barthlow, D., Mourelatos, Z. P., Zeng, J., Gorsich, D., Singh, A. & Hu, Z. (2022) Mobility prediction of off-road ground vehicles using a dynamic ensemble of NARX models, *Journal of Mechanical Design*, **144** (9), 091709.
- Lohmann, D., Nolte-Holube, R. & Raschke, E. (1996) A large-scale horizontal routing model to be coupled to land surface parametrization schemes, *Tellus A*, **48** (5), 708–721.
- Ma, J., Li, Z., Cheng, J. C., Ding, Y., Lin, C. & Xu, Z. (2020) Air quality prediction at new stations using spatially transferred bi-directional long short-term memory network, *Science of the Total Environment*, **705**, 135771.
- McCarthy, G. T. (1938) The unit hydrograph and flood routing. In: *Proceedings of Conference of North Atlantic Division, US Army Corps of Engineers, 1938*, pp. 608–609.
- Mostafa, R. R., Kisi, O., Adnan, R. M., Sadeghifar, T. & Kuriqi, A. (2023) Modeling potential evapotranspiration by improved machine learning methods using limited climatic data, *Water*, **15** (3), 486.
- Murphy, A. H. (1988) Skill scores based on the mean square error and their relationships to the correlation coefficient, *Monthly Weather Review*, **116** (12), 2417–2424.
- Nash, J. E. & Sutcliffe, J. V. (1970) River flow forecasting through conceptual models Part I—A discussion of principles, *Journal of Hydrology*, **10** (3), 282–290.
- Nearing, G. S., Kratzert, F., Sampson, A. K., Pelissier, C. S., Klotz, D., Frame, J. M., Prieto, C. & Gupta, H. V. (2021) What role does hydrological science play in the age of machine learning? *Water Resources Research*, **57** (3), e2020WR028091.
- Nguyen, H. D., Tran, K. P., Thomassey, S. & Hamad, M. (2021) Forecasting and anomaly detection approaches using LSTM and LSTM autoencoder techniques with the applications in supply chain management, *International Journal of Information Management*, **57**, 102282.
- Niu, G.-Y., Yang, Z.-L., Mitchell, K. E., Chen, F., Ek, M. B., Barlage, M., Kumar, A., Manning, K., Niyogi, D., Rosero, E. & Tewari, M. (2011) The community Noah land surface model with multiparameterization options (Noah-MP): 1. Model description and evaluation with local-scale measurements, *Journal of Geophysical Research: Atmospheres*, **116** (D12), 1–19.
- Oki, T. & Sud, Y. (1998) Design of total runoff integrating pathways (TRIP)—A global river channel network, *Earth Interactions*, **2** (1), 1–37.
- Oppenheim, A. V. (1999) *Discrete-Time Signal Processing*. Pearson Education India. Prentice-Hall Inc., Upper Saddle River, New Jersey, 07458.
- Schuster, M. & Paliwal, K. K. (1997) Bidirectional recurrent neural networks, *IEEE Transactions on Signal Processing*, **45** (11), 2673–2681.
- Tavakoly, A. A., Snow, A. D., David, C. H., Follum, M. L., Maidment, D. R. & Yang, Z.-L. (2017) Continental-scale river flow modeling of the Mississippi River Basin using high-resolution NHDplus dataset, *JAWRA Journal of the American Water Resources Association*, **53** (2), 258–279.
- Thelen, A., Zhang, X., Fink, O., Lu, Y., Ghosh, S., Youn, B. D., Todd, M. D., Mahadevan, S., Hu, C. & Hu, Z. (2022) A comprehensive review of digital twin—Part 1: Modeling and twinning enabling technologies, *Structural and Multidisciplinary Optimization*, **65** (12), 354.
- Williams, C. K. & Rasmussen, C. E. (2006) *Gaussian Processes for Machine Learning*, vol. 2. Cambridge, MA: MIT Press.
- Xie, H., Tang, H. & Liao, Y.-H. (2009) Time series prediction based on NARX neural networks: An advanced approach. In: *2009 International Conference on Machine Learning and Cybernetics*, vol. 3. IEEE, pp. 1275–1279.
- Yamazaki, D., Kanae, S., Kim, H. & Oki, T. (2011) A physically based description of floodplain inundation dynamics in a global river routing model, *Water Resources Research*, **47** (4) W04501, 1–21.

- Yu, Y., Si, X., Hu, C. & Zhang, J. (2019) A review of recurrent neural networks: LSTM cells and network architectures, *Neural Computation*, **31** (7), 1235–1270.
- Yuan, X., Chen, C., Lei, X., Yuan, Y. & Muhammad Adnan, R. (2018) Monthly runoff forecasting based on LSTM-ALO model, *Stochastic Environmental Research and Risk Assessment*, **32**, 2199–2212.
- Zhao, Y., Chadha, M., Olsen, N., Yeates, E., Turner, J., Gugaratshan, G., Qian, G., Todd, M. D. & Hu, Z. (2023) Machine learning-enabled calibration of river routing model parameters, *Journal of Hydroinformatics*, **25**, 1799–1821.

First received 28 February 2024; accepted in revised form 6 September 2024. Available online 20 September 2024