Journal of Hydroinformatics

Journal of Hydroinformatics Vol 25 No 5, 1799 doi: 10.2166/hydro.2023.030

Machine learning-enabled calibration of river routing model parameters

Ying Zhao[a,†], Mayank Chadha[b,†], Nicholas Olsen[c], Elissa Yeates[c], Josh Turner[d], Guga Gugaratshan[d], Guofeng Qian[b], 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] Coastal and Hydraulics Laboratory, US Army Corps of Engineers, Vicksburg, MS, USA
[d] Hottinger Bruel & Kjaer Solutions LLC, Southfield, MI 48076, USA
*Corresponding authors. E-mail: zhennhu@umich.edu; mdtodd@ucsd.edu
[†]These authors contributed equally.

ABSTRACT

Streamflow prediction of rivers is crucial for making decisions in watershed and inland waterways management. The US Army Corps of Engineers (USACE) uses a river routing model called RAPID to predict water discharges for thousands of rivers in the network for watershed and inland waterways management. However, the calibration of hydrological streamflow parameters in RAPID is time-consuming and requires streamflow measurement data which may not be available for some ungauged locations. In this study, we aim to address the calibration aspect of the RAPID model by exploring machine learning (ML)-based methods to facilitate efficient calibration of hydrological model parameters without the need for streamflow measurements. Various ML models are constructed and compared to learn a relationship between hydrological model parameters and various river parameters, such as length, slope, catchment size, percentage of vegetation, and elevation contours. The studied ML models include Gaussian process regression, Gaussian mixture copula, Random Forest, and XGBoost. This study has shown that ML models that are carefully constructed by considering causal and sensitive input features offer a potential approach that not only obtains calibrated hydrological model parameters with reasonable accuracy but also bypasses the current calibration challenges.

Key words: hydrological model, machine learning, model calibration, RAPID, rivers streamflow prediction

HIGHLIGHTS

- Calibration of hydrology model using machine learning.
- Learning unknown relationship between model parameters and river features.
- Rapid calibration of hydrological model parameters.
- Comparative study of different machine learning techniques.

1. INTRODUCTION

Watershed and inland waterways management generate a broad range of crucial and often expensive engineering decision-making problems such as dredging (Yeates *et al.* 2020), analysis of a dam's remaining useful life (England 2018), flood prediction (Maidment 2017), understanding climate change (Lucas-Picher *et al.* 2003), and sediment and contamination analysis (Palermo *et al.* 2008). The key constituent of these decision-making analyses is the streamflow discharge prediction of rivers that form a river network. The flow of water in a river network is modeled through *river routing* that simulates the changes in the shape of a hydrograph as water moves through the rivers. The US Army Corps of Engineers (USACE) uses a computational river routing model, RAPID (David *et al.* 2011a) (http://rapid-hub.org/), which offers a numerical solution to river routing modeled by a physics-based hydrological model called the *Muskingum method* to simultaneously predict water discharges in all the rivers in the network using parallelization (David *et al.* 2011a; Rouholahnejad *et al.* 2012).

The application of the Muskingum model in any form involves both a calibration step and a prediction step. The calibration step includes estimating calibrated streamflow parameters k and x (which will be discussed later) for each river by using historical inflow–outflow discharge measurements at certain gauge stations spread across a given geographic region (USACE uses discharge measurements from gauges spread across the United States for calibration). Since the Muskingum method

was first introduced by McCarthy in 1938 (McCarthy 1938; Cunge 1969), numerous research articles have targeted the calibration problem to improve the prediction accuracy of streamflow forecasts by incorporating nonlinearities in river routing problems, making the calibration more computationally efficient via curve-fitting, and employing parallel computing, Bayesian methods, optimization techniques, and other statistical methods (see, for example, Gill 1978; Yoon & Padmanabhan 1993; Mohan 1997; Das 2004; Chu & Chang 2009; Rouholahnejad *et al.* 2012; Xu *et al.* 2012). In terms of effective computational implementation of river routing, a breakthrough was achieved by David *et al.* (David *et al.* 2011a, 2011b, 2013; Tavakoly *et al.* 2017) that developed the RAPID model which is a matrix form of the Muskingum model through which the streamflow could be predicted for a large set of rivers that comprise a large network. The most attractive feature of RAPID is its ability to use efficient parallel computing. However, despite RAPID being a state-of-the-art computational tool that is equipped with parallel computing, calibration of these streamflow parameters is a computationally extensive and time-consuming process (especially when done at a continental or global scale) and requires historic streamflow measurement data which may not be available for certain ungauged locations. This may potentially hinder an accurate prediction of streamflow forecasting in an ungauged region where there is no measurement data available. To address this challenge, our goal is to use machine learning (ML) to establish a causal connection between hydrological parameters and a wide range of hydrological and topological features extracted through satellite imagery. This data-driven approach will enable us to predict these parameters even in ungauged locations without relying on the currently used river routing model-based calibration process.

In recent years, there has been a significant increase in the hydrologic community's interest in machine learning. This surge can be attributed to the rapid expansion of hydrologic data repositories and the successful implementation of ML in diverse academic and commercial applications. This newfound enthusiasm is largely facilitated by the increasing accessibility to supportive hardware and software. Studies have demonstrated the accuracy of data-driven models in predicting extreme events and capturing valuable information from hydrological datasets, outperforming physics-based models in several aspects. Over a decade ago, Todini (2007) highlighted the divergence of opinions between physical process-oriented and system engineering-oriented modelers regarding data-driven models in hydrology. While physical process-oriented modelers expressed skepticism due to the heavy reliance on training sets, system engineering-oriented modelers argued that data-driven models outperformed complex physics-based models in forecasting. With advancements in ML and computational power, hydrology has seen active adoption of data-driven models. Frame *et al.* (2022) demonstrated the accuracy of deep learning-based data-driven models in predicting extreme rainfall–runoff events compared to physics-based models. Similarly, Kratzert *et al.* (2019) and Nearing *et al.* (2021) noted the ability of data-driven models to capture significant information from hydrological datasets and outperform physics-based models, including future predictions in ungauged basins and knowledge transfer between basins using modified LSTM networks. Kan *et al.* (2020) used an ANN with the K-nearest neighbor-based clustering method to propose a hybrid ML hydrological model capable of forecasting floods. Guse *et al.* (2017) noted the importance of model parameters in hydrological models and focused on establishing and investigating the connection between the model parameters and different performance criteria that evaluate the robustness of the hydrological model. Fernandez-Palomino *et al.* (2021) highlighted that model calibration purely using discharge data is problematic since it does not guarantee the correct representation of hydrological processes. Instead, Fernandez-Palomino *et al.* (2021) opted for a multi-objective calibration that includes additional information captured by vegetation data and hydrological signatures. Zhang *et al.* (2018) exploited the regression tree ensemble approach and compared it with three other widely used approaches (multiple linear regression, multiple log-transformed linear regression, and hydrological modeling) to assess the prediction accuracy of 13 runoff characteristics or signatures in Australia. Frame *et al.* (2021) utilized up to 26 parameters, ranging from watershed attributes to meteorological data, to build streamflow prediction models and perform model diagnostics. A recent study by Liu *et al.* (2023) compares the flood simulation capabilities of a hydrological model and ML-based model. Readers are recommended to refer to Xu & Liang (2021) and the references therein for a comprehensive and non-technical review of the application of ML-based methods in hydrology.

The objective of this paper is to establish a relationship between the hydrological parameters and the attributes reported by global satellite monitoring systems, thereby enhancing physics-derived modeling on a large scale. Although ML models can directly be used to develop a hybrid river routing model that improves the discharge predictions by incorporating additional hydrological and topological information in tandem with the standard physics-based model to predict discharge, it is still worthwhile to obtain data-informed hydrological parameters that serve as input to the river routing model. Three primary reasons motivate our current investigation:

1. There are specific applications where physics-based models remain desirable. Despite its simplicity, the Muskingum model continues to be employed in many of the largest-scale hydrological applications (see Tavakoly *et al.* 2017; Grogan *et al.* 2022). It is these users who can benefit from improved methodologies for parameter selection and calibration. This investigative approach aims to explore the implications of using ML to learn hydrological model parameters that feed into river routing models.
2. Some of these hydrological parameters are macro variables that have physical meaning. For example, the k parameter that is the primary focus of our current investigation is linked to the flow travel time within the river reach. The significance of these parameters extends beyond its utilization within the routing model. It proves valuable in other hydrological domains, such as sediment forecasting, reservoir management, river flow estimation and flood control, dam break analysis, and groundwater studies among others (for example, see Marcus 1989; Meyer *et al.* 2018). Consequently, the implications of this research reach far beyond supporting the routing model alone. Moreover, due to the intricate dynamics of rivers and the dependence of their flow on both watershed and its own geometric characteristics, the process is likely influenced by multiple interrelated parameters. This complexity makes ML an appealing approach for obtaining macro hydrological quantities, including the k parameter, by leveraging geography-dependent features. The traditional k parameter is obtained by forcing the RAPID predictions to closely match the measured discharge. As pointed out in Fernandez-Palomino *et al.* (2021), judging the hydrological model reliability purely based on discharge-based calibration is problematic, as it does not guarantee the correct representation of internal hydrological processes. In this paper, our goal is to leverage ML to obtain the k parameter even in ungauged locations by utilizing geographical features that can be obtained from satellite data.
3. River dynamics and their dependency on the proximal surroundings and environmental conditions in the given region are very complex. It is practically impossible to obtain a closed-form physics-based model that accurately models the flow. Defining the prediction bounds is also challenging since quantifying the uncertainties in the causal parameters is difficult. In that regard, our aim was to leverage the power of advanced ML methods to calibrate a hydrological quantity, k, which possibly has dependencies on numerous variables or embedded features. Theoretically, the k parameter obtained using the ML model is more reliable and meaningful, as it is obtained as a functional dependency of factors that influence k. This contrasts with the traditional approach, where the k parameter is obtained by forcing the RAPID predictions to be as close as possible to the measured discharge.

In summary, by obtaining a more informed and improved value of these parameters, we aim to enhance the predictive capabilities of the hydrological model (even if the improvement is marginal) and improve our understanding of the flow dynamics within the river system.

To do so, we start by reasonably assuming that the Muskingum parameters bear a functional relationship with topography and the hydrodynamic characteristics of the system. This paper focuses on calibrating the streamflow parameter k since the Muskingum model is more sensitive to k; the same methodology may be also applied to calibrate x. The objective is to carefully consider various features that could potentially have a causal relationship with the calibrated k (or x), investigate the sensitivity of these features, select the most sensitive features, and finally learn the unknown functional relationship between these features and the calibrated Muskingum parameters. We consider topographical and hydrodynamic features extracted from the rivers and their watershed. These include properties like river length, river slope, vegetation data, catchment area, and elevation data and its gradient. The features related to the vegetation data and the elevation data are extracted using the MODIS Vegetation Continuous Field (VCF) (https://lpdaac.usgs.gov/products/mod44bv006/) and Digital Elevation Model (DEM) satellite images (http://hydro.iis.u-tokyo.ac.jp/ yamadai/MERIT_Hydro/), respectively. We extract two sets of features from the VCF and DEM images. The first set of features comprises the statistical moments of the vegetation and elevation data that capture their spatial distribution, whereas the second set of features is extracted using a Convolution Neural Network (CNN)-based autoencoder that takes these images as input and extracts the useful features. Rivers and watersheds can be grouped into various classes/clusters based on their shared properties (like length and slope of rivers, and mean vegetation percentage of the watershed). We use a Gaussian Mixture Model (GMM)-based clustering technique to group the rivers into different clusters. For each cluster consisting of numerous rivers, these features are used as input, and the calibrated k (obtained from the calibration procedure of RAPID) is used as an output to train the ML models. The studied ML models include Gaussian process regression (GPR), Gaussian mixture copula (GMC), Random Forest, and XGBoost.

The rest of the paper is arranged as follows. Section 2 lays a brief background of the Muskingum model and the calibration procedure of RAPID. Section 3 describes the solution flow of the ML-based calibration of k and investigates various ML models. Section 4 delineates and compares the numerical results of using different ML models and also considers the impact of different feature extraction techniques. Finally, Section 5 concludes the paper and lists ongoing research directions.

2. BACKGROUND OF RIVER ROUTING MODEL

2.1. The linear Muskingum model and RAPID

The simplest form of the linear Muskingum equations consists of two hydrological parameters corresponding to each river in the river network, k and x. The parameter k is the storage time constant for the river reach that has a value reasonably close to the flow travel time within the river reach, and the parameter $x \in [0, 0.5]$ is a dimensionless weighting factor that quantifies the relative influence of the inflow and the outflow on the volume of the channel or river reach. Consider a channel with $S(t)$ denoting the absolute channel storage at time t; as per the linear Muskingum model, we have (see Tung 1985)

$$\mathbb{S}(t) = k(xQ_{\text{in}}(t) + (1-x)Q_{\text{out}}(t)) \tag{1}$$

where $Q_{\text{in}}(t)$ and $Q_{\text{out}}(t)$ are the rates of inflow and outflow at time t, respectively. Therefore, by definition, the flow continuity equation can be written as

$$\frac{\mathrm{d}\mathbb{S}(t)}{\mathrm{d}t} = Q_{\text{in}}(t) - Q_{\text{out}}(t) \tag{2}$$

Considering the time instances t and $t + \Delta t$, the Equations (1) and (2) may be solved numerically to obtain the following (see Tung 1985)

$$Q_{\text{out}}(t+\Delta t) = C_1 Q_{\text{in}}(t+\Delta t) + C_2 Q_{\text{in}}(t) + C_3 Q_{\text{out}}(t) \tag{3}$$

where C_1, C_2, and C_3 are functions of the Muskingum parameters and the time interval, and they satisfy $C_1 + C_2 + C_3 = 1$ such that

$$C_1 = \frac{0.5\Delta t - kx}{k(1-x) + 0.5\Delta t}; \quad C_2 = \frac{0.5\Delta t + kx}{k(1-x) + 0.5\Delta t}; \quad C_3 = \frac{k(1-x) - 0.5\Delta t}{k(1-x) + 0.5\Delta t} \tag{4}$$

This straightforward model for flow in a single channel may be easily extended to a river network by making a simple observation that the inflow in a downstream river reach is fed by the outflow of one or many upstream rivers reaches that converge to the downstream river reach of interest. David *et al.* (2011a) build on this single channel Muskingum model to develop a river routing model for the entire river network, yielding the following matrix form of the Muskingum model

$$(\boldsymbol{I} - \boldsymbol{C_1} \cdot \boldsymbol{N}) \cdot \boldsymbol{Q}(t+\Delta t) = \boldsymbol{C_1} \cdot \boldsymbol{Q^e}(t) + \boldsymbol{C_2} \cdot (\boldsymbol{N} \cdot \boldsymbol{Q}(t) + \boldsymbol{Q^e}(t)) + \boldsymbol{C_3} \cdot \boldsymbol{Q}(t) \tag{5}$$

Here, $\boldsymbol{I}$ is the identity matrix, $\boldsymbol{N}$ is the river network matrix, $\boldsymbol{C}_1$, $\boldsymbol{C}_2$, and $\boldsymbol{C}_3$ are the diagonal matrices with elements $\boldsymbol{C}_{1j}$, $\boldsymbol{C}_{2j}$, and $\boldsymbol{C}_{3j}$, respectively, that are functions of the time interval Δt and the streamflow parameters k_j and x_j with $1 \le j \le m$ for a river network with m river reaches as defined by Equation (4); $\boldsymbol{Q}$ is a vector of outflow from each reach; and $\boldsymbol{Q^e}$ is a vector of lateral inflow for each reach (collected in the watershed of the river). To better understand the matrix form of the Muskingum model defined by (5), the readers are referred to a pedagogical example of a river network consisting of five-reach, two-node, and two-gauge locations provided in David *et al.* (2011a).

2.2. Calibration of the streamflow parameters in RAPID

To briefly summarize the calibration process used in RAPID, we are going to borrow results and discussions presented by David *et al.* (2011a) and Tavakoly *et al.* (2017). The calibration procedure of RAPID involves obtaining the multiplication

factor λ_{k_j} and λ_{x_j} such that

$$k_j = \lambda_{k_j} k_{\text{ini}_j}, \quad x_j = 0.1 \cdot \lambda_{x_j} \tag{6}$$

Here, k_{ini_j} is an initial value of the k-parameter for the jth river reach that depends on wave celerity and river topology. Tavakoly *et al.* (2017) proposes three experiments/options for k_{ini}

$$k^1_{\text{ini}_j} = \frac{L_j}{C_0} \tag{7a}$$

$$k^2_{\text{ini}_j} = \eta_j \frac{L_j}{\sqrt{S_j}} \tag{7b}$$

$$k^3_{\text{ini}_j} = \eta_j \frac{L_j}{\sqrt{S_j}}; \quad \frac{L_i}{\sqrt{S_j}} \in P[0.05, 0.95] \tag{7c}$$

Here, L_j and S_j stand for the length and slope of the jth river in the network, respectively, with $1 \leq j \leq m$ for a river network with m river reaches. Their values can be obtained from the NHD*Plus* dataset. C_0 is the reference water wave celerity, and η_j is the inverse of average velocity based on the first experiment for the jth river in the network, such that (see Tavakoly *et al.* (2017)

$$\eta_j = \frac{\bar{k}^1_{\text{ini}}}{L_j/\sqrt{S_j}}, \text{ where } , \bar{k}^1_{\text{ini}} = \frac{1}{m}\sum_{j=1}^{m} k^1_{\text{ini}_j} = \frac{\bar{L}}{C_0} \tag{8}$$

where $\bar{L}$ is the mean length of the rivers in the network.

As mentioned in Tavakoly *et al.* (2017), for the first experiment (Equation (7a)), the wave celerity is assumed to be independent of topography and is constant for all river reaches. Although a simple assumption, this choice of initial k-parameter makes it devoid of reality since it implies that the travel time of a flow wave is independent of the topography. The second and the third experiments (Equation (7b) and (7c)) tackle the topographical dependence of the initial k-parameter by defining it in terms of the length and slope, which are topographical features. For the third experiment, to avoid the influence of extreme values on celerity, the values of $L_j/\sqrt{S_j}$ are restricted between the 5 and 95% thresholds based on the cumulative probability function. Tavakoly *et al.* (2017) concluded that the predictions of RAPID improved when topological features were explicitly considered by using $k^2_{\text{ini}_j}$, and $k^3_{\text{ini}_j}$ as a choice for initial k-parameter. This is our primary motivation to assume that the Muskingum parameters bear a functional relationship with topography and the hydrodynamic characteristics of the system.

RAPID obtains the set of calibration factors $\lambda^*_{k_j}$ and $\lambda^*_{x_j}$ by minimizing a cost function $\phi(\boldsymbol{k}, \boldsymbol{x})$ that quantifies the deviation between RAPID's prediction of discharge and the observed discharge at multiple gauges located throughout the river basin. Let $\boldsymbol{k} = [k_j] = [\lambda_{k_j} k_{\text{ini}_j}]$ and $\boldsymbol{x} = [x_j] = [0.1 \cdot \lambda_{x_j}]$ denote the vectors of the k and x parameters of all the rivers in the network. For n gauge locations, and for optimization time-window ranging from the first day t_0 and the last day t_f, the cost function is defined as

$$\phi(\boldsymbol{k}, \boldsymbol{x}) = \sum_{t=t_0}^{t_f} \sum_{i=1}^{n} \left[\frac{\bar{Q}_i - Q_i^{\text{obs}}(t)}{\bar{Q}_i^{\text{obs}}} \right]^2 \tag{9}$$

Here, $\bar{Q}_i$ is the daily average of the RAPID's computed flow, $Q_i^{\text{obs}}(t)$ is the daily observed discharge, and $\bar{Q}_i^{\text{obs}}$ is the daily average of the observed flow. Let $\boldsymbol{k}^* = [k_j^*] = [\lambda^*_{k_j} k_{\text{ini}_j}]$ and $\boldsymbol{x}^* = [x_j^*] = [0.1 \cdot \lambda^*_{x_j}]$ denote the vectors of the calibrated k and x parameters of all the rivers in the network. The calibrated parameters are obtained as

$$(\boldsymbol{k}^*, \boldsymbol{x}^*) = \underset{(\boldsymbol{k},\boldsymbol{x})}{arg\,min}\ \phi(\boldsymbol{k}, \boldsymbol{x}) \tag{10}$$

Figure 1 illustrates the calibration and the prediction procedure of RAPID currently used.

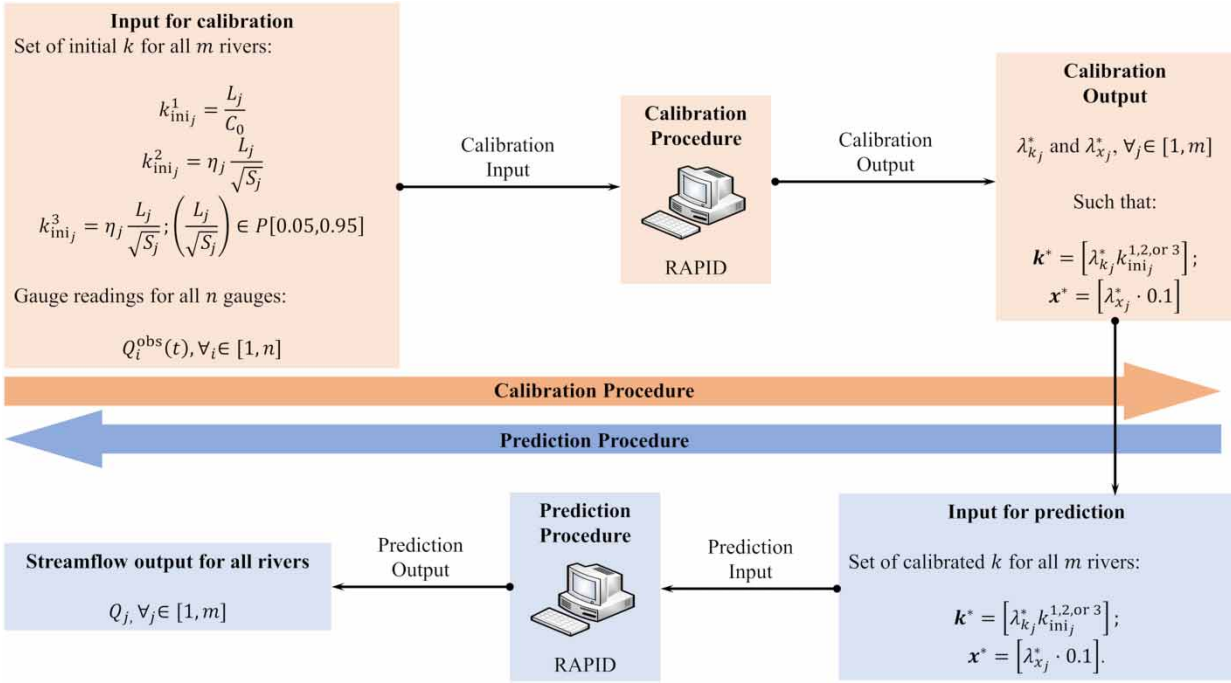


Figure 1 | Schematic diagram of currently used calibration and prediction procedure of RAPID.

The calibration process described in this section (which is currently being used in the RAPID model run by USACE), although state-of-the-art, is time-consuming and depends on the availability of observation data from gauges across the United States. Therefore, we can only expect good predictions in the geographical regions where gauge measurements are available. This limits the streamflow prediction capability in geographic regions where good measurement data are unavailable or inaccessible. As pointed out in Fernandez-Palomino *et al.* (2021), judging the hydrological model reliability purely based on discharge-based calibration is problematic, as it does not guarantee the correct representation of internal hydrological processes. In the next section, we propose a data-driven ML-enabled calibration of the streamflow parameters that tackles the aforementioned challenges in the traditional calibration approach.

3. THE PROPOSED ML-ENABLED CALIBRATION OF THE STREAMFLOW PARAMETERS

3.1. Overview

To establish a data-driven calibration model, we aim to obtain statistical features that have a causal relationship with the calibrated streamflow parameters. We hypothesize that these features are dependent on the topography and the hydrodynamic characteristics of the region of interest. We recall that this paper focuses on calibrating the streamflow parameter k since the Muskingum model is more sensitive to k, and the same methodology can be applied to calibrate x. We extract the features from the raw datasets of various topographic quantities (discussed in Section 3.2) by using two feature extraction techniques: (1) statistical moment analysis (see Section 3.3.1), and (2) CNN-based autoencoder (see Section 3.3.2). We then use the variance-based global sensitivity analysis (GSA) to select sensitive features. In theory, all the rivers and their respective watersheds could be classified into countable numbers of groups/clusters based on their shared topographical and hydrodynamic properties. The sensitive features selected using variance-based global sensitivity analysis can be used to classify a river (and its respective watershed) based on its topographical and hydrodynamic properties. We use a GMM-based clustering technique to obtain these clusters (see Section 3.5). The optimal number of clusters are determined based on Bayesian information criterion (BIC) (see Chen & Gopalakrishnan 1998). For each cluster, we then train ML models (discussed in Section 3.3) using the selected features (selected using the variance-based global sensitivity analysis) as input data and calibrated streamflow parameter k^* as the output. We focus our attention on these four ML techniques: (1) GPR (see Section 3.6.1), (2) GMC (see Section 3.6.2), (3) Random Forest (see Section 3.6.3), and (4) XGBoost (see Section 3.6.4). The following sections discuss these steps comprehensively.

3.2. Capturing and extracting the raw data

The United States of America is divided into seven catchment zones. Each watershed is identified with a *river ID* and each river ID has a *shape file* (in .shp format) depicting the geographic shape of the watershed. We consider the following *raw input data sources*: length of river, slope of river (data obtained from NHD*Plus*), catchment area, vegetation data extracted using the MODIS VCF (https://lpdaac.usgs.gov/products/mod44bv006/), and elevation data obtained using the DEM satellite images (http://hydro.iis.u-tokyo.ac.jp/ yamadai/MERIT_Hydro/). The elevation, and the percentage of vegetation data are available for various land segments in DEM (with extension *.dem*), and Hierarchical Data Format (with the extension *.hdf*), respectively. These images for various land segments are glued together using the open-source Geospatial Data Abstraction Library (GDAL) (https://gdal.org/) to obtain the *global* vegetation, and elevation map in *.tiff* format for the United States consisting of seven catchment zones. Once the *global* vegetation and elevation maps are obtained, the *local* vegetation and elevation clips corresponding to each river ID (identifying a watershed) of interest are obtained by clipping the *global* map using GDAL. Once the elevation clips are obtained, we also obtain the clips of the magnitude of the elevation's gradient since it is reasonable to assume that the elevation's gradient has a direct impact on the river's streamflow. Figure 2 shows a schematic flowchart demonstrating the vegetation data extraction; a similar approach is undertaken to obtain the local elevation clips.

As an expected *output* for the ML model, we obtain the calibrated k^* values using RAPID's calibration procedure as described in Figure 1 for all the river reaches with gauges. At this point, each river ID of interest has a corresponding river length, river slope, and local vegetation clip in *.tiff* format, local elevation clip in *.tiff* format as the input data, and the calibrated k^* as the output data to build the ML model. The raw data in tabulated form looks as shown in the figure below.

Remark 1: The proposed framework is designed to be flexible and can accommodate additional information or features as inputs to obtain hydrological parameters. In this paper, we intentionally focused on showcasing the framework's capabilities by using limited information related to watershed characteristics, specifically river slope, river length, watershed area, vegetation patterns and elevation data. This choice was made for practical reasons, aiming to enable the development of a ML algorithm that can be expanded later to include more features. Moreover, this approach streamlined data collection and pre-processing, saving time that could be dedicated to building the ML algorithm and maintaining a more focused investigation.

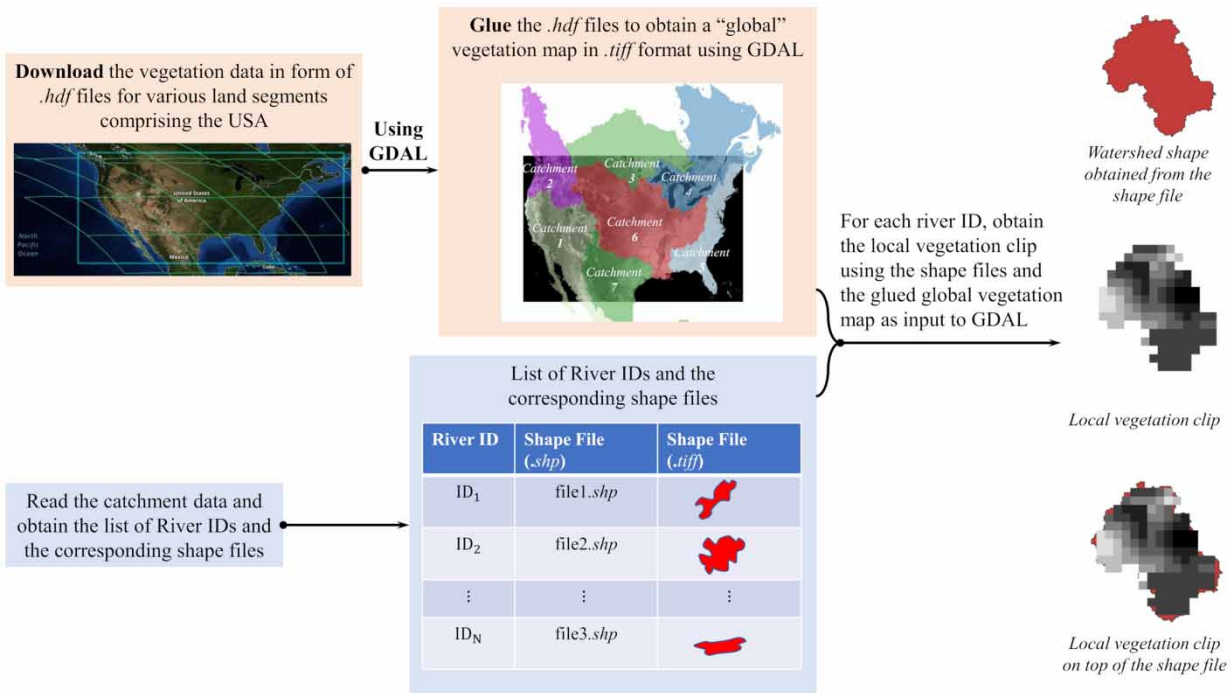


Figure 2 | Data extraction for percentage of vegetation.

3.3. Feature extraction

Utilizing the raw data (in the form of Figure 3), we obtain the features corresponding to the local vegetation and elevation clips by exploiting two approaches described below.

3.3.1. Features extracted via statistical moment analysis

In the first approach, for each river ID, we obtain the statistical moments of the vegetation data, elevation data, and the magnitude of the elevation gradient extracted using the distribution of these quantities in their respective *local* clips. We extracted the first four statistical moments and decided to use the first two (i.e., the mean and the standard deviation) after GSA (discussed in detail in the upcoming Sections 3.4 and 4.1). The catchment size is obtained as the number of pixels in the local vegetation clip. This feature extraction technique gives us 10 features: river length, river slope, catchment size (defined as the number of non-zero pixels in the vegetation clip), catchment geographic area, and the remaining six features corresponding to the mean and standard deviation of the vegetation data, elevation data, and magnitude of the elevation gradient.

3.3.2. Features extracted via a convolution autoencoder

A *convolution autoencoder* integrates an autoencoder and a CNN. The autoencoder comprises the encoder, the decoder, and a latent space known as a bottleneck. The encoder takes an image and extracts the latent features, thereby transforming a high-dimensional image into a lower-dimensional feature space (called the bottleneck). The decoder, on the other hand, uses the features extracted by the encoder to recover the image. In a convolution autoencoder, the encoder and the decoder use a CNN structure to achieve their respective targets of data transformation. The convolution autoencoder, therefore, first compresses the input data from a high-dimensional form to a lower-dimensional latent space using a convolution encoder, and the decoder uses the convolution transpose to convert the lower-dimensional latent features into a reconstruction of the original higher-dimensional input (Chen *et al.* 2017). The difference between the attempted reconstructed data and the original input data is called the *reconstruction error*. Therefore, the *convolution autoencoder* is trained using a large amount of input data (numerous vegetation clips in our case) so as to minimize the reconstruction error. The network learns to exploit natural structure in the input data to find an efficient lower-dimensional representation of the input data. Among various neural networks, the convolution autoencoder is the most suitable architecture to extract features from a high-dimensional image because of two primary benefits:

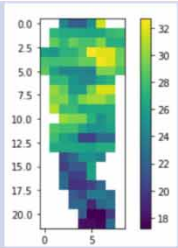
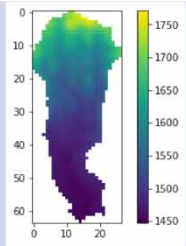
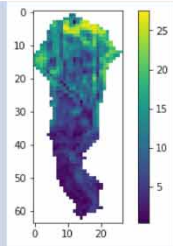
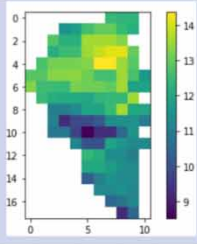
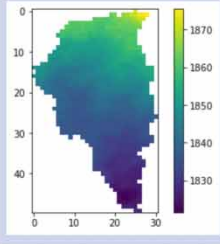
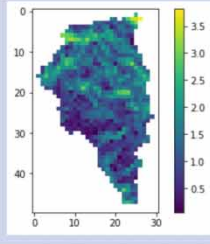
River ID	Shape File	River Length	River Slope	Local vegetation clip	Local elevation clip	Local elevation gradient's magnitude clip	Calibrated k^*
ID_1		L_1	S_1				k_1^*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
ID_N		L_N	S_N				k_N^*

Figure 3 | Raw data in tabulated form.

1. The CNN-based architecture can better retain the connected information between the pixels of an image (Guo *et al.* 2017).
2. Slicing and stacking the data in other neural network leads to a large loss of information. The convolution autoencoder, rather than stack the data, preserves the spatial information of the input image and extracts information gently through the convolution layer.

We use the convolution autoencoder to extract latent features for the local vegetation clips. Since training the convolutional autoencoder is a computationally extensive and time-consuming process, we limit the use of the convolutional autoencoder to extract features for the vegetation clips and use statistical features for the elevation clips. This feature extraction technique gives us 14 features: river length, river slope, catchment size (defined as the number of non-zero pixels in the vegetation clip), catchment geographic area, and six CNN-based features of the vegetation data; the mean and standard deviation of elevation data, and the mean and standard deviation of elevation gradient's magnitude.

To train the model, we use all the raw local vegetation clips (column 5 of Figure 3). We start by first reshaping all the input images into 80×50 size that are then uploaded into the *DataLoader* class of the *PyTorch* package which is then used to train the convolution autoencoder model.

The encoder and decoder: The convolution encoder's structure used for this research consisted of 2D convolution layers followed by a pooling layer for each of them, a flattening layer, and a leaky rectified linear unit (ReLU). The first convolution layer deploys a kernel size of 9 (a 3×3 scanner) and has one input channel (the reshaped image) and 16 output channels. In order to assist the kernel with processing the image, padding is added to the frame of the image to allow for more space for the kernel to cover the image. For the first convolution layer, we use the padding of 4 pixels with zero value and the stride of 1 (stride is the number of pixels the kernel shifts over the input matrix). The first convolution layer is followed by the application of the ReLU that rectifies any negative value in the output channels of the first convolution layer. Following this, two-dimensional max-pooling with a size of 2×2 is imposed to decrease the dimensions of the first convolution layer's output, yielding the output of size $16 \times 40 \times 25$. The output of the first convolution layer acts as the input to the second convolution layer. Therefore, the second convolution layer has 16 input channels and yields six output channels. For the second layer, we use the same kernel size, padding size, stride, and max-pooling size as the first layer. The output of the second convolution layer after max-pooling is the data of size $6 \times 20 \times 12$. These 1,440 numbers are flattened and the latent features are extracted using a Fully Connected Network (FCN), yielding six features. Along similar lines, the decoder uses the 6×1 latent space to reconstruct the original input image using two transpose convolution layers. The convolution autoencoder is trained to minimize the reconstruction loss quantified by the mean-square error between the reconstructed original input dataset. Figure 4 illustrates the convolution autoencoder architecture adopted in this paper.

3.4. Feature selection using the variance-based GSA

GSA quantifies the contributions of variability in the inputs to the variability of an output quantity of interest QoI (generically denoted by Y, which for this paper is the calibrated k^*). Various approaches have been developed for GSA in the past decades, such as the Fourier amplitude sensitivity test (McRae *et al.* 1982), correlation ratio, Kullback-Leibler divergence (Greegar & Manohar 2015), and Sobol's indices (Sudret 2008). Among these various methods, variance decomposition-based GSA is one of the most widely used methods (Sudret 2008). In this method, the variance of a QoI (i.e., $\mathrm{Var}(Y)$) can be decomposed into contributions of individual variables, denoted by V_i, and the interactions between different variables, denoted by $(V_{ij}, \cdots, V_{12\cdots N_d})$, such that

$$\mathrm{Var}(Y) = \sum_{i=1}^{N_d} V_i + \sum_{1 \le i < j}^{N_d} V_{ij} + \cdots + V_{12\cdots N_d} \tag{11}$$

where N_d is the number of uncertain input variables.

Based on the variance decomposition, two types of Sobol's sensitivity indices, namely the main effect index and total effect index, are widely used to analyze the contributions of uncertainty sources (see Sudret 2008). The main effect index, which is also called the first-order index, is given by

$$S_i = \frac{V_i}{\mathrm{Var}(Y)} = \frac{\mathrm{Var}_{X_i}(E_{X_{\sim i}}(Y|X_i))}{\mathrm{Var}(Y)}, \forall i = 1, \cdots, N_d, \tag{12}$$

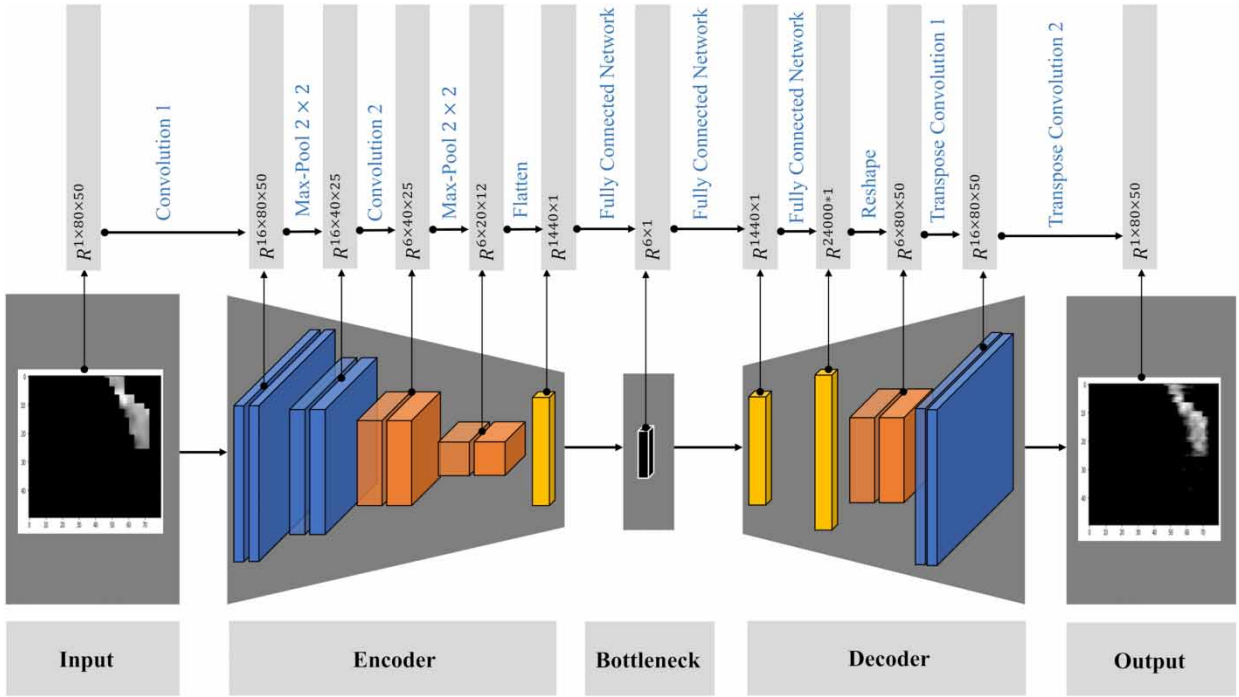


Figure 4 | Convolutional autoencoder architecture to extract vegetation features.

where S_i is the first-order index of the ith input variable, $X_{\sim i}$ represents all the input variables excluding X_i, and $E(\cdot)$ is an expectation operator.

GSA usually requires a double-loop Monte Carlo simulation (MCS) as indicated in the above equation. The double-loop MCS needs to evaluate the simulation model thousands of times. This is inapplicable to the studied problem since there is no model available to represent the relationship between the driving factors and the calibrated streamflow parameter k^*. To overcome this challenge, a data-driven probability model-based GSA method is employed. A data-driven GSA method only requires a data matrix to compute the first-order Sobol's indices (see Hu & Mahadevan 2019). It consists of three main steps, which are briefly explained as follows:

- Step 1: Extract the data of input variable X_i and that of QoI from the original data matrix.
- Step 2: Build a probability model using a GMM.
- Step 3: Compute the Sobol's indices using the constructed GMM.

It has been shown that this method can achieve a similar accuracy level for GSA compared to the double-loop MCS method. We direct interested readers to Hu & Mahadevan (2019) for a detailed description of the probability model-based GSA method.

3.5. Watershed classification via clustering by a GMM

All the sample data are first clustered using the GMM. The sample data are clustered based on the important features identified through the GSA. Let $\boldsymbol{X}$ denote a random vector corresponding to the features with a realization vector $\boldsymbol{x}$ consisting of the selected features as its elements. A GMM model (see Reynolds 2009) is trained as

$$f_{\boldsymbol{X}}(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) = \sum_{i=1}^{Q} \omega_i \phi(\boldsymbol{x}_i;\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i) \tag{13}$$

Here, Q is the number of Gaussian components in the GMM model, $\phi(\cdot)$ is the probability density function (PDF) of a multi-variate Gaussian distribution, ω_j is the weight of the jth Gaussian component, and $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ are respectively the mean vector and covariance of the jth Gaussian component, such that $\boldsymbol{\mu} = [\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_Q]$, and $\boldsymbol{\Sigma} = [\boldsymbol{\Sigma}_1, \cdots, \boldsymbol{\Sigma}_Q]$.

Once the GMM model is trained, for a new prediction sample $\boldsymbol{x}_e$, we first predict the probability $P_{\boldsymbol{X}_i}(\boldsymbol{x}_e)$ of the sample input $\boldsymbol{x}_e$ falling into the ith cluster, where $\boldsymbol{X}_i$ denotes the random vector of an input sample that belongs to the ith cluster. Once the probability $P_{\boldsymbol{X}_i}(\boldsymbol{x}_e)$ is obtained for all the clusters, we identify the cluster i^* in which the input sample $\boldsymbol{x}_e$ most likely belongs, such that

$$i^* = \arg\max_i P_{\boldsymbol{X}_i}(\boldsymbol{x}_e) \tag{14}$$

3.6. ML models for prediction of calibrated streamflow parameter

We have conducted an investigation into the performance of four ML models with varying complexity and characteristics in order to obtain a calibrated k parameter. GPR and GMC generate probabilistic predictions, whereas XGBoost and Random Forest produce single-point predictions. Figure 5 provides an overview of the proposed ML-based calibration framework.

In the following sections, we provide a comprehensive description of all four methods, along with thorough testing and the presentation of their results. This will enable future readers to make informed decisions when applying similar approaches in their own work.

3.6.1. Gaussian process regression

The goal is to find the mapping between the input vector $\boldsymbol{x}$ (consisting of various features) and the output y (the calibrated k in this paper). As described in Williams & Rasmussen (2006), a GPR probabilistically models the output y as a realization $g(\boldsymbol{x})$ of the Gaussian Process (GP), such that

$$\begin{aligned} y &= g(\boldsymbol{x}) + \varepsilon; \\ g(\boldsymbol{x}) &= m(\boldsymbol{x}) + h(\boldsymbol{x}) \end{aligned} \tag{15}$$

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

$$\begin{aligned} g(\boldsymbol{x}) &\sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')); \\ h(\boldsymbol{x}) &\sim \mathcal{GP}(0, k(\boldsymbol{x}, \boldsymbol{x}')); \\ \varepsilon &\sim \mathcal{N}(0, \sigma_\varepsilon^2) \end{aligned} \tag{16}$$

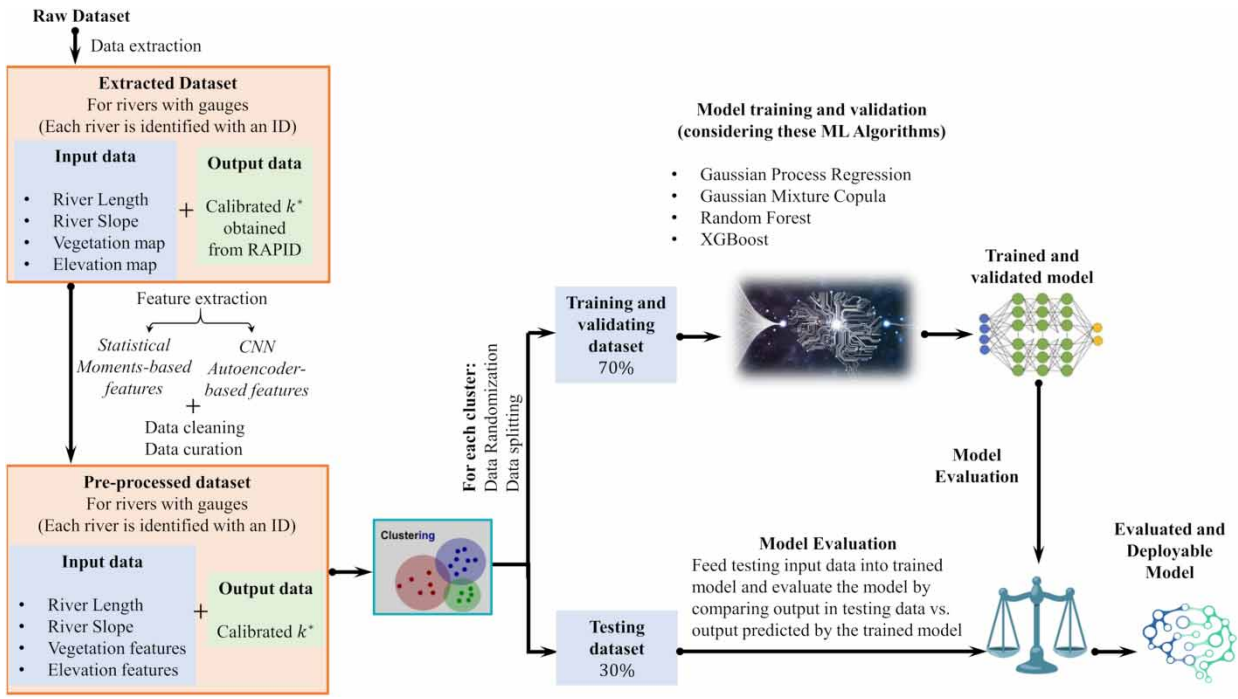


Figure 5 | ML-based calibration framework.

where

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma_h^2 R(\boldsymbol{x} - \boldsymbol{x}', \boldsymbol{\theta}) \tag{17}$$

Here, 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 (see Hu & Mahadevan 2016)

$$R(\boldsymbol{x} - \boldsymbol{x}', \boldsymbol{\theta}) = \exp\left(-\sum_{k=1}^{n_d} \theta_k |x_k - x_k'| \right) \tag{18}$$

where n_d is the dimension of the input variables and x_k is the kth element of $\boldsymbol{x}$.

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

$$g(\boldsymbol{x})|(\boldsymbol{X}, Y) \sim \mathcal{GP}(\sim m(\boldsymbol{x}),\ \sim k(\boldsymbol{x}, \boldsymbol{x}')). \tag{19}$$

Here,

$$\begin{aligned} &\sim m(\boldsymbol{x}) = m(\boldsymbol{x}) + \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{X}) \times (\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) - \sigma_\varepsilon^2 \mathbf{I}_n)^{-1} \times (\boldsymbol{Y} - \boldsymbol{m}); \\ &\sim k(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}, \boldsymbol{x}') - \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{X}) \times (\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) - \sigma_\varepsilon^2 \boldsymbol{I}_n)^{-1} \times \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{x}') \end{aligned} \tag{20}$$

where

$$\begin{aligned} &\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) = \begin{pmatrix} k(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(1)}) & \cdots & k(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(n)}) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(1)}) & \cdots & k(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(n)}) \end{pmatrix} \in \mathbb{R}^{n \times n} \\ &\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{X}) = (k(\boldsymbol{x}, \boldsymbol{x}^{(1)}), \cdots, k(\boldsymbol{x}, \boldsymbol{x}^{(n)})) \in \mathbb{R}^{1 \times n} \\ &\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{x}) = \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{X})^T \in \mathbb{R}^{n \times 1} \\ &\boldsymbol{m} = (m(\boldsymbol{x}^{(1)}), \cdots, m(\boldsymbol{x}^{(n)}))^T \in \mathbb{R}^{n \times 1}. \end{aligned} \tag{21}$$

The hyperparameters $\boldsymbol{\theta}$, σ_h, and σ_ε are estimated using the training data through Bayesian parameter estimation or maximum likelihood estimation (see Hu & Mahadevan 2016; Ramancha *et al.* 2022).

From Equations (15) and (19), the predictive distribution of the output y for the input realization $\boldsymbol{x}$ conditioned on the training data $(\boldsymbol{X}, \boldsymbol{Y})$ is given by

$$y|\boldsymbol{x}, (\boldsymbol{X}, Y) \sim \mathcal{GP}(\sim m(\boldsymbol{x}),\ \sim k(\boldsymbol{x}, \boldsymbol{x}) + \sigma_\varepsilon^2) \tag{22}$$

3.6.2. Gaussian mixture copula

Let X_i, $i = 1 \cdots N_d$ denote the random variable representing the ith component of input random vector $\boldsymbol{X}$, and let Y denote the random variable corresponding to the output. A GMC constructs a probabilistic model to represent the joint PDF of the inputs and outputs, such that (see Hu & Mahadevan 2019):

$$f_{\boldsymbol{X}Y}(\boldsymbol{x}, y) = \mathcal{C}_{\text{GMM}}(u_{x_1}, u_{x_1}, \cdots, u_{x_{N_d}}, u_y; \boldsymbol{\theta}) f_{X_i}(x_i) \cdots f_{X_N}(x_N) f_Y(y) \tag{23}$$

where, $f_{X_i}(x_i)$ is the marginal PDF of X_i, $f_Y(y)$ is the marginal PDF of the output Y, u_{x_i} is the marginal cumulative density function (CDF) of X_i, and $c_{\text{GMM}}(\cdot; \boldsymbol{\theta})$ is a new copula function approximated by a GMM with parameter vector $\boldsymbol{\theta}$ that is to be estimated from the data. The copula function $c_{\text{GMM}}(\cdot; \boldsymbol{\theta})$ is approximated using GMM (see Hu & Mahadevan 2019) as:

$$c_{\text{GMM}}(\cdot; \boldsymbol{\theta}) \approx \frac{1}{\phi(\Phi^{-1}(u_{x_1}))} \cdots \frac{1}{\phi(\Phi^{-1}(u_{x_{N_d}}))} \frac{1}{\phi(\Phi^{-1}(u_y))} \sum_{j=1}^{Q} \lambda_j \phi(\boldsymbol{z}, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \tag{24}$$

Here, $\boldsymbol{z} = [\phi(\Phi^{-1}(u_{x_1})), \phi(\Phi^{-1}(u_{x_2})), \cdots, \phi(\Phi^{-1}(u_{x_{N_d}})), \phi(\Phi^{-1}(u_y))]$, Q is the number of Gaussian components in the GMM model, $\phi(\cdot)$ is the PDF of a standard normal random variable, $\Phi^{-1}(\cdot)$ is the inverse CDF of a standard normal random variable, λ_j is the weight of the jth Gaussian component, and $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ are respectively the mean vector and covariance of the jth Gaussian component.

After the modeling of the joint PDF using GMC, for a new given realization of the input data $\boldsymbol{x}$, the conditional PDF of y is obtained as:

$$f_{Y|\boldsymbol{X}}(y|\boldsymbol{x}) = \frac{f_{\boldsymbol{X}Y}(\boldsymbol{x}, y)}{f_{\boldsymbol{X}}(\boldsymbol{x})} \approx \frac{c_{\text{GMM}}(u_{x_1}, u_{x_1}, \cdots, u_{x_{N_d}}, u_y; \boldsymbol{\theta}) f_Y(\boldsymbol{y})}{c_{\text{GMM}}(u_{x_1}, u_{x_1}, \cdots, u_{x_{N_d}}, u_y; \boldsymbol{\theta}_{\boldsymbol{x}})} \tag{25}$$

The maximum likelihood estimate of y can then be obtained by $y^* = arg\,max_y(f_{Y|\boldsymbol{X}}(y|\boldsymbol{x}))$. The corresponding mean estimate is estimated by $\bar{y} = \int y f_{Y|\boldsymbol{X}}(y|\boldsymbol{x})\, dy$. More details about the GMC model can be found in Hu & Mahadevan (2019). The advantage of the GMC model is that it provides a probabilistic prediction instead of a point estimate. This allows us to quantify the uncertainty in the prediction. Moreover, the probabilistic prediction is not limited to a Gaussian distribution, which is another advantage over the GPR model.

3.6.3. Random forest

Random forest is an ensemble supervised learning algorithm that is constructed from a set of decision trees. It was proposed by Breiman in 2001 (see Breiman 2001; Qi 2012). The goal of this ensemble learning method is to improve the prediction accuracy of decision trees by combining multiple trees using the bagging ensemble algorithm. Compared with decision trees, the random forest has an advantage in dealing with overfitting problems and can efficiently handle a large number of input variables.

As the basis of a random forest, a decision tree consists of three components, namely the root node, decision node, and leaf node. The root node segregates the training dataset into different branches. The decision node then decides on attributes used to predict the output. Each decision node ends up with leaf nodes that represent class labels. Decision trees are sensitive to the training dataset; even small changes in the dataset can lead to significant differences among tree predictions (see Biau & Scornet 2016). The difference between decision trees and the random forest is that decision trees allow each tree to randomly choose data from the training dataset while random forest randomly selects features from several subsets of the input features. The random forest chooses the best split among all available input features. As a result, different trees in a random forest have different input features and thus less correlation and increased diversity. The final prediction of a random forest is obtained by averaging the results of individual decision trees of the forest. Since a large number of trees can be generated, overfitting issues can be minimized.

The random forest can deal with both regression and classification problems. In this paper, it is used for regression. For the random regression forest model used in this paper, the number of trees is 170 which is optimized through cross-validation. This means that the final prediction result is an average of 170 decision trees.

3.6.4. XGBoost

XGBoost is an extreme gradient boost tree algorithm proposed by Chen & Guestrin (2016). It is an improved ensemble learning method based on a gradient boosting decision tree. The objective of XGBoost is to combine several weak models to create a collectively robust model. This algorithm is similar to the random forest described above and comprises a parallel set of decision trees. XGBoost can also deal with regression, classification, and ranking problems. In this paper, XGBoost is used to solve a regression problem. The difference between random forest and XGBoost is that the gradient boosting decision tree uses the tree iteratively, which combines residuals of the previous tree with prediction errors of new trees to perform the final prediction. The term 'gradient boosting' means the gradient descent algorithm is employed to minimize the objective function. The gradient boost first generates an initial model and gets the initial residuals for each input in the training dataset. After that, the gradient boost ensembles the previous models' residuals into the next new model. XGBoost repeats these two steps in a sequential manner to get the final prediction. XGBoost has an advantage over the gradient boosting trees because it uses L1 & L2 regulations to improve the computational speed and prediction accuracy during the training.

3.7. ML-based streamflow parameter calibration and discharge prediction framework

We now have all the components required to build a streamflow parameter calibration framework as illustrated in flowchart 5. The proposed calibration framework can be coupled with the RAPID-based discharge prediction. Figure 6 illustrates the calibration using the ML-based framework followed by the prediction based on the RAPID model.

4. RESULTS AND DISCUSSIONS

In this section, we discuss the predictions of the calibrated k^* obtained using four different ML models, i.e., GPR, GMC, Random Forest, and XGBoost, considering the features obtained using the statistical moment analysis and the convolution autoencoder. For the purpose of building the ML models, we consider the most carefully observed and well-documented catchment 6 region of the United States, which has the Mississippi and Ohio rivers passing through it. Figure 7 shows all

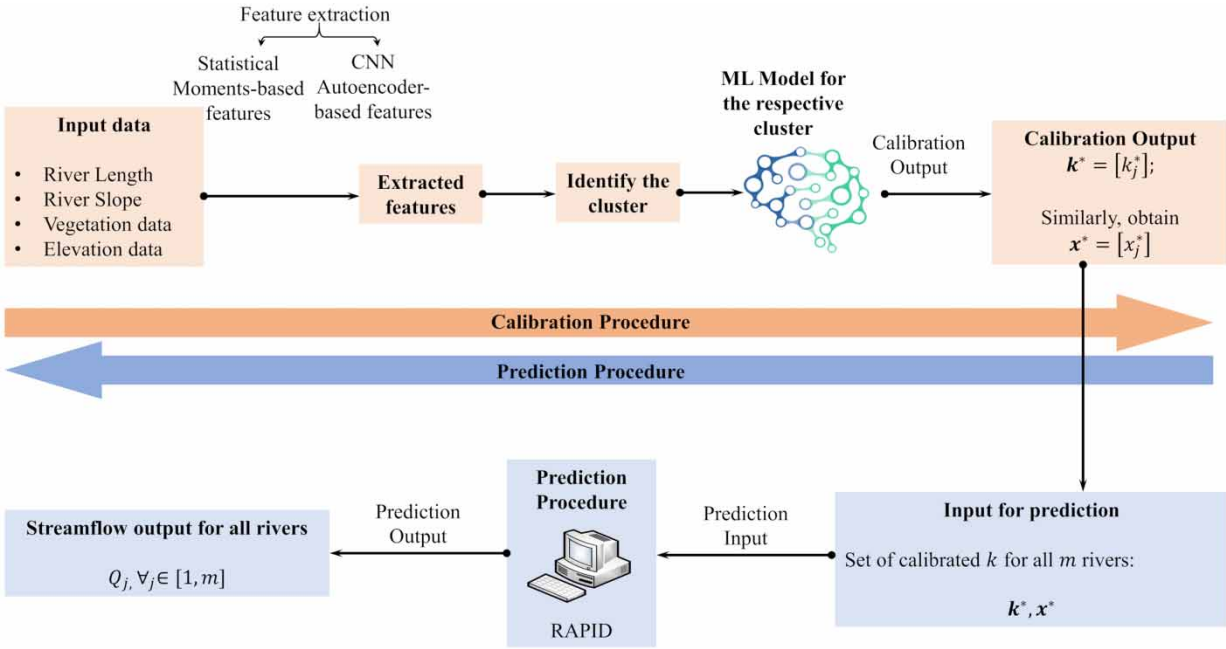


Figure 6 | Schematic diagram of the proposed ML-based calibration framework followed by the RAPID-based discharge prediction.

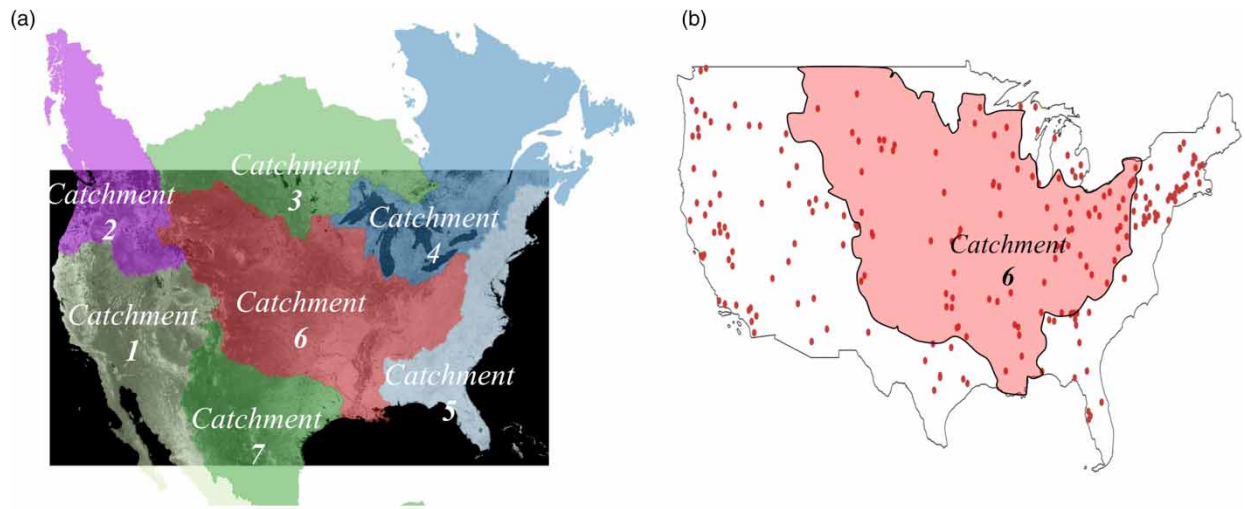


Figure 7 | Different catchments and discharge gauge locations (highlighted for the catchment 6) (a) Different catchments in the USA (b) Discharge gauge locations.

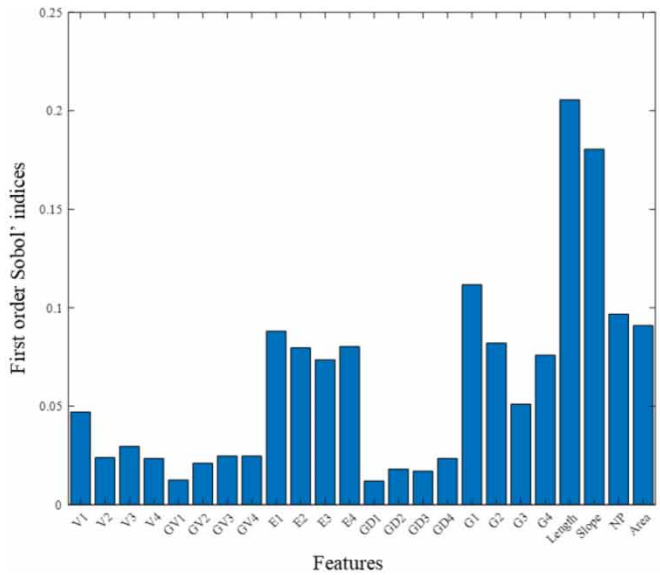
seven catchment regions and illustrates the discharge gauge locations. The data from the discharge gauges in catchment 6 are used to obtain calibrated k^*. As per USACE engineering practice, k^* usually is below 10^4, and a significantly large value of k^* is erroneous. Therefore, to make sure the quality of data used to train the ML models is not meaningless, we reasonably consider the data points with k^* greater than but not equal to 0, and less than 2.5×10^4, that is $k^* \in (0, 2.5 \times 10^4]$. With this constraint, we have a total of 4,569 data points. The 4,569 data points are grouped into six clusters and the results are discussed in the following Section 4.2. The optimal number of clusters are determined based on BIC, which is also a commonly used method to select the number of clusters in Gaussian mixture modeling (see Chen & Gopalakrishnan 1998).

4.1. GSA results

In this section, we present the GSA results focusing on catchment 6 and considering the statistical moment analysis-based features. Figure 8 illustrates the first-order sensitivity index S_i for the ith feature and is calculated using Equation (12) obtained by considering all 4,569 data points belonging to the catchment 6. The numerical value of the first-order sensitivity index can be interpreted as the relative sensitivity of the respective feature. That is, a higher value of the index S_i is indicative of a larger causal contribution and higher sensitivity of the input feature to the output quantity (Sudret 2008). We use this fact to select the most sensitive features.

Based on the first-order Sobol's indices illustrated in Figure 8, obtained through the GSA, we make the following observations:

1. Length and slope are the most sensitive features since they have the top two highest values of the first-order Sobol's index. This observation is consistent with previous engineering knowledge. For example, as shown in Equation (7), the initial calibrated $k_{\text{ini}}^{1,2,3}$ are functions of the river's length and/or the slope.
2. Statistical moments of elevation gradient direction (denoted by GDx in Figure 8) have much lower contributions than the other factors. Elevation gradient direction is therefore not included in the analysis.
3. Statistical moments of the percentage of vegetation (denoted by Vx in Figure 8) are more dominant than the contributions of the percentage of vegetation gradient's magnitude (denoted by GVx in Figure 8). Therefore, we only consider the percentage of vegetation and ignore its gradient.
4. For a given feature, contributions of the higher-order statistical moments are lower than those of the lower-order statistical moments. To maintain the number of variables at a manageable level, higher-order statistical moments are not included in the analysis.



Features	Description
Vx	The x-th statistical moment of the percentage of vegetation
GVx	The x-th statistical moment of the gradient of percentage of vegetation
Ex	The x-th statistical moment of the elevation
GDx	The x-th statistical moment of the elevation gradient direction
Gx	The x-th statistical moment of the elevation gradient magnitude
Length	The river's length
Slope	The river's slope
NP	The number of pixels in the percentage of vegetation clip
Area	The watershed area

Figure 8 | First-order Sobol's indices for different features considering all the data points belonging to the catchment 6.

5. Based on GSA, we consider the following features for building the ML model: river length, river slope, the area of the watershed, the number of pixels in the vegetation clip, the first two moments (mean and the standard deviation) of the percentage of vegetation, elevation contour, and the magnitude of the elevation gradient.

4.2. Clustering results

We cluster the 4,569 watersheds of interest in catchment 6 into six clusters by using the GMM-based clustering technique discussed in Section 3.5. We consider the statistical relationship between the important features identified from GSA. For the sake of illustration, only the three most important features (the river's length, the river's slope, and the mean of vegetation) are plotted in the section. Figure 9 illustrates the clusters obtained using these three input features. For each cluster, we use a 70:30 training/testing split, as listed in Table 1.

Table 2 reports the statistics (mean μ, standard deviation σ, and coefficient of variation ρ) of the input parameters used for clustering (i.e., the river's slope, the river's length, and the mean of the vegetation percentage) for all the six clusters.

Based on the statistics reported in Table 2, we make the following observations:

1. The coefficient of variation for the distribution of length does not change appreciably for different clusters. That is, each cluster has similar variability in the lengths of the river.
2. The coefficient of variation for slope and the vegetation mean is relatively low for clusters 1 and 3.
3. The coefficient of variation for slope and the vegetation mean is relatively high for clusters 2, 4, 5, and 6.

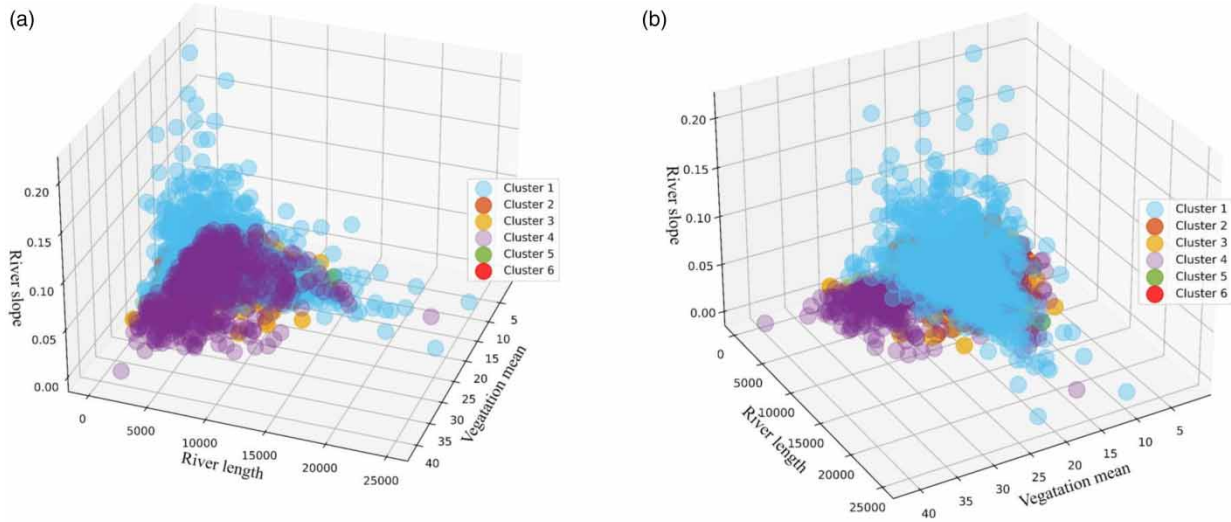


Figure 9 | Visualizing clusters by plotting the three input variables in different orientations. (a) Orientation 1 (b) Orientation 2.

Table 1 | Number of data points in each cluster

Cluster	Number of data points	% of the dataset	Number of training data points	Number of testing data points
1	577	12.63%	402	175
2	524	11.47%	366	158
3	681	14.91%	476	205
4	542	11.86%	379	163
5	1,676	36.68%	1,163	513
6	569	12.45%	402	167

Table 2 | Statistics (mean μ, standard deviation σ, and coefficient of variation ρ) for various clusters

	Slope			Length			Mean of vegetation %		
Clusters	μ	σ	ρ (%)	μ	σ	ρ (%)	μ	σ	ρ (%)
1	0.0059	0.0035	58.9	2,789.7	2,029.8	72.8	10.9955	0.5689	5.2
2	0.0040	0.0032	80.1	1,535.1	949.5	61.8	14.4486	2.8188	19.5
3	0.0025	0.0017	67.5	4,246.3	3,126.7	73.6	15.6748	1.2285	7.8
4	0.0096	0.0085	88.1	2,953.9	2,048.7	69.4	16.1485	3.9934	24.7
5	0.0143	0.0232	161.8	3,662.2	3,006.1	82.1	14.6886	5.5317	37.7
6	0.0030	0.0031	100.8	3,363.4	2,539.4	75.5	14.9984	4.9923	33.3

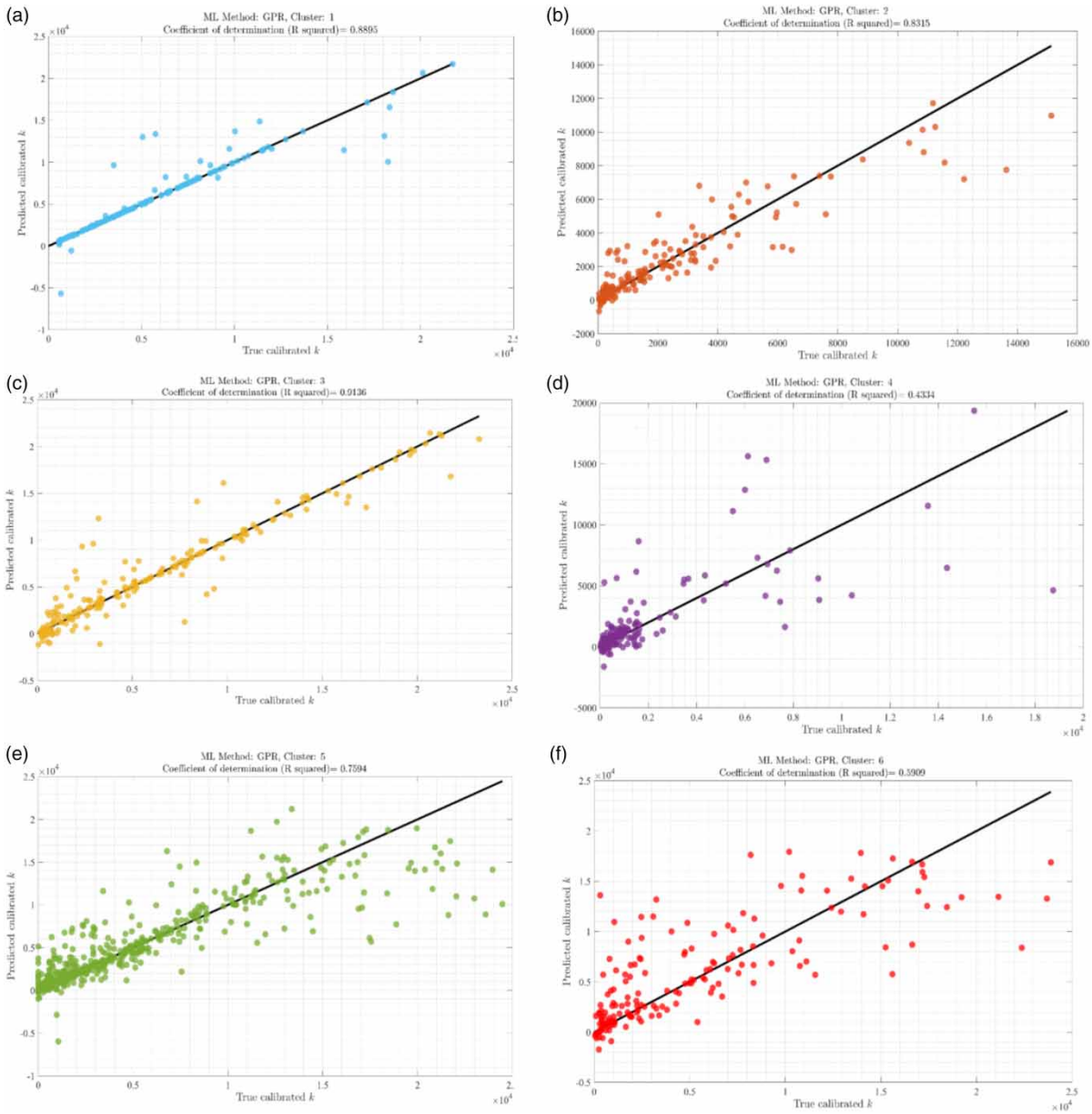


Figure 10 | Predicted vs. true calibrated k^* obtained using GPR considering the statistical features. (a) Cluster 1 (b) Cluster 2 (c) Cluster 3 (d) Cluster 4 (e) Cluster 5 (f) Cluster 6.

4.3. Comparison of different ML models

In this section, we present the prediction results obtained through four ML techniques that use two types of feature extraction techniques: statistical features and features obtained using a CNN-based autoencoder. For illustration purposes, we illustrate plots for predicted vs. true calibrated k^* plotted for the test data for various clusters obtained using statistical features and by deploying four different ML methods: GPR (see Figure 10), GMC (see Figure 11), Random Forest (see Figure 12), and XGBoost (see Figure 13). Tables 3 and 4 report the coefficient of determination R^2 for both the methods of feature extraction.

Based on the presented results, we make the following observations:

1. The predictions for clusters 1 and 3 are better than for clusters 2, 4, 5, and 6 consistently across all ML techniques. This is because, as noted at the end of Section 4.2, clusters 1 and 3 have a relatively low coefficient of variation for slope and vegetation mean whereas clusters 2, 4, 5, and 6 have a relatively high coefficient of variation for slope and vegetation mean. Large variability in the input parameter for the clusters results in a lower coefficient of determination of the prediction.

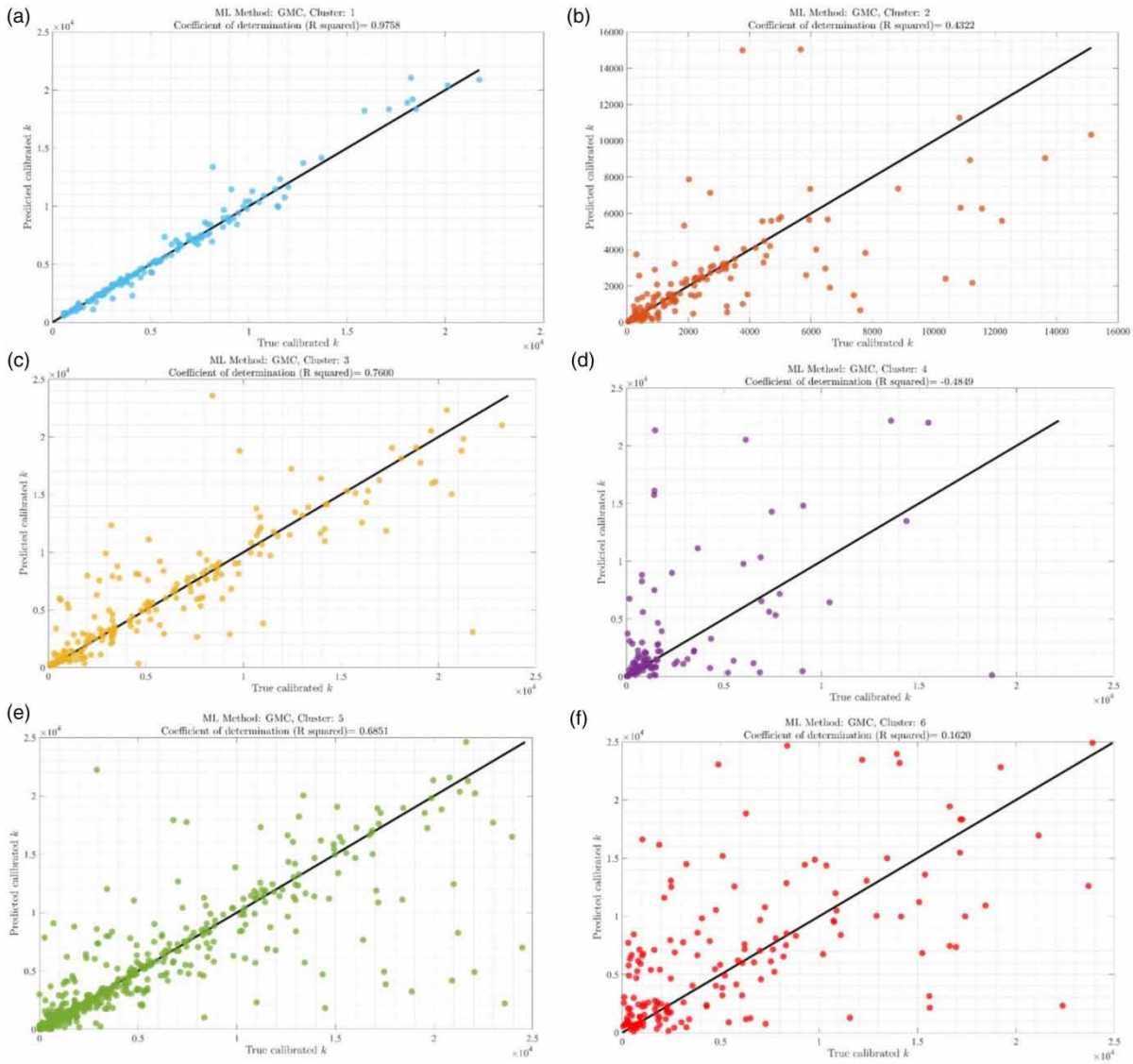


Figure 11 | Predicted vs. true calibrated k^* obtained using GMC considering the statistical features. (a) Cluster 1 (b) Cluster 2 (c) Cluster 3 (d) Cluster 4 (e) Cluster 5 (f) Cluster 6.

2. The performance of most of the ML techniques is consistent. Judging from the R^2 values, XGBoost offers the best performance for most of the clusters followed by Random Forest and GPR, and finally, GMC is relatively the worst performer. As can be observed in Figure 11 and reflected in Tables 3 and 4, the R^2 for clusters 3 and 6 corresponding to the GMC is notably worse because of countable but large outliers.
3. Among the four ML techniques, the GPR and GMC provide probabilistic predictions, whereas the Random Forest (see Figure 12) and the XGBoost (see Figure 13) yield single-point predictions. The Figures 10 and 11 corresponding to the GPR and GMC, respectively, plot the mean value of the predicted calibrated k^*.
4. The ML predictions obtained by considering statistical features are better than the predictions obtained by considering the convolution autoencoder-based features. This is because the statistical moments of the vegetation and the elevation clips are more suitable to capture the distribution of these quantities, and the convolution autoencoder-based features inherently may have noise/bias as a consequence of numerous transformations performed on the higher-dimensional raw data (vegetation clips) to obtain low-dimensional features, as well as model structural bias.

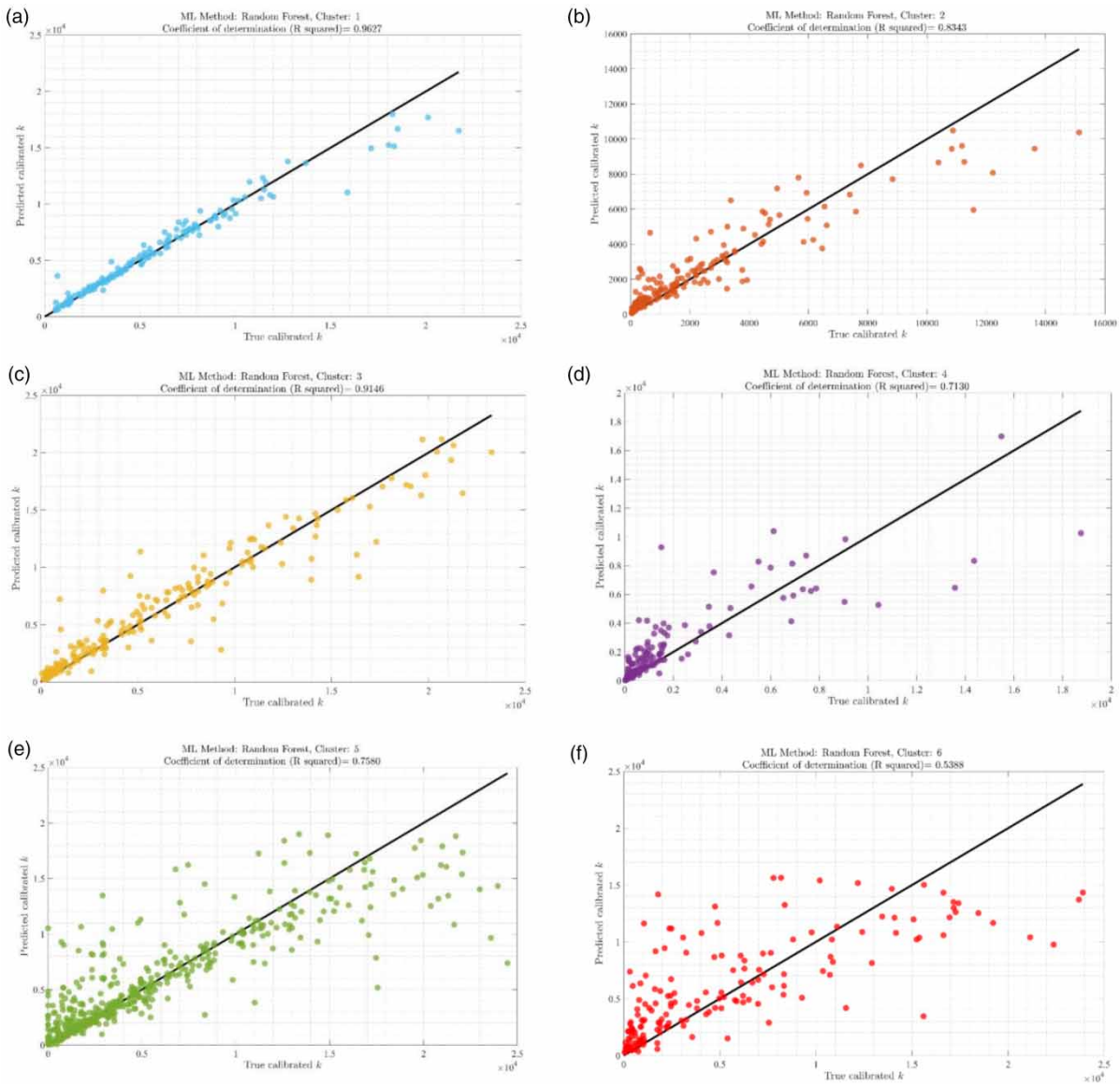


Figure 12 | Predicted vs. true calibrated k^* obtained using Random Forest considering the statistical features. (a) Cluster 1 (b) Cluster 2 (c) Cluster 3 (d) Cluster 4 (e) Cluster 5 (f) Cluster 6.

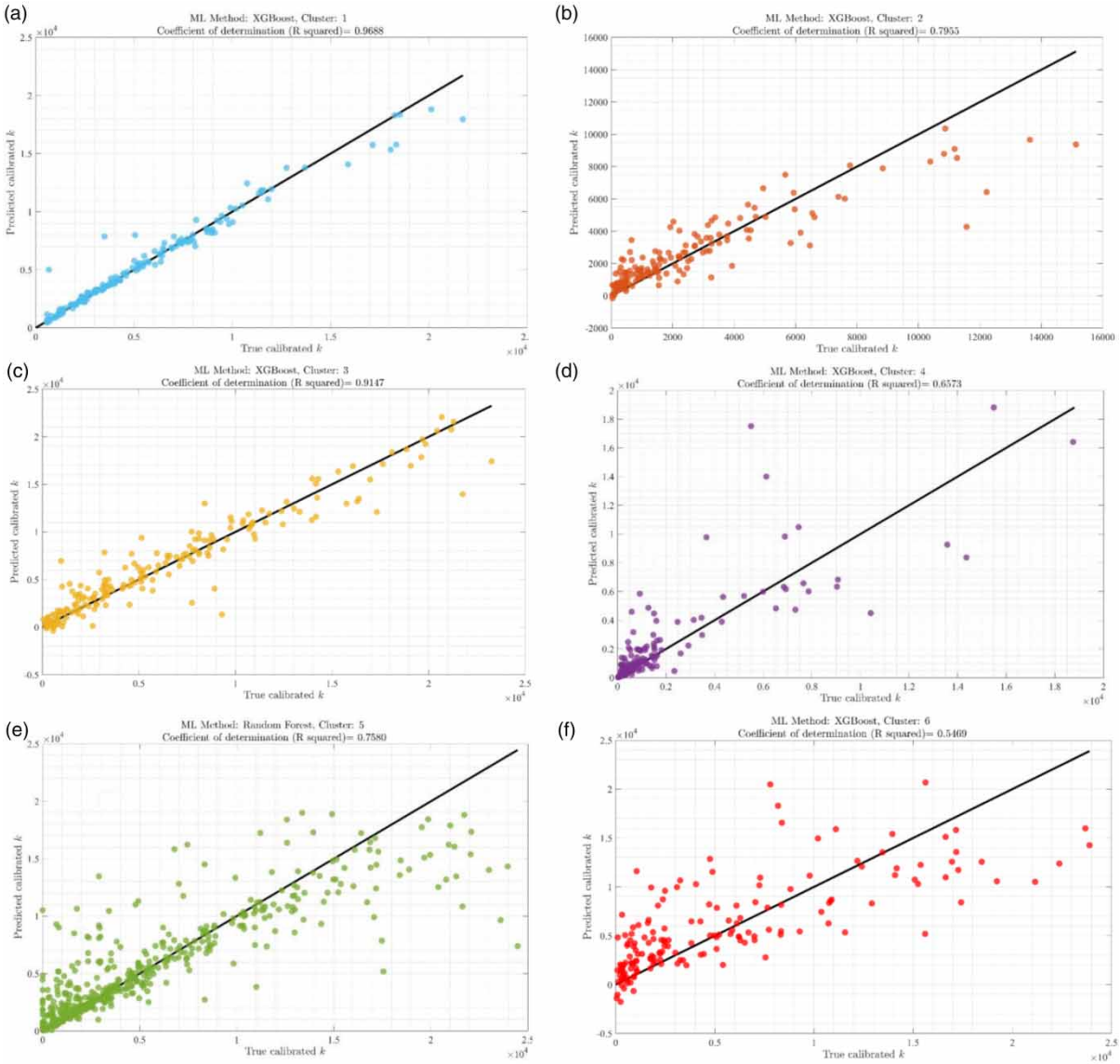


Figure 13 | Predicted vs. true calibrated k^* obtained using XGBoost considering the statistical features. (a) Cluster 1 (b) Cluster 2 (c) Cluster 3 (d) Cluster 4 (e) Cluster 5 (f) Cluster 6.

Table 3 | Coefficient of determination R^2 for various ML predictions on test data considering input features obtained through statistical moment analysis

Clusters	GPR	GMC	Random Forest	XGBoost
1	0.8895	0.9758	0.9627	0.9688
2	0.8315	0.4322	0.8343	0.7855
3	0.9136	0.7600	0.9146	0.9147
4	0.4334	−0.4849	0.7130	0.6574
5	0.7594	0.6851	0.7580	0.7580
6	0.5909	0.1620	0.5388	0.5469

Table 4 | Coefficient of determination R^2 for various ML predictions on test data considering input features obtained through convolution autoencoder

Clusters	GPR	GMC	Random Forest	XGBoost
1	0.8665	0.9856	0.9573	0.9721
2	0.7845	0.3898	0.7918	0.7577
3	0.9186	0.7116	0.9016	0.9120
4	0.3101	−0.3909	0.6621	0.9704
5	0.6548	0.6000	0.6878	0.7342
6	0.4833	−0.1898	0.4642	0.5267

5. SUMMARY AND CONCLUSIONS

This paper focuses on comparing ML techniques to enable computationally-expeditious calibration of hydrological model parameters without requiring streamflow discharge measurements. There are two potential benefits of this study: (1) a trained, validated, evaluated, and deployable ML model can be used to obtain the calibrated hydrological parameters in the regions where there are no discharge measurement gauges; (2) the entire framework is built on an assumption that the hydrological parameters bear a functional relationship with topography and the hydrodynamic characteristics of the river system. Thus, the ML models can take into account numerous features that are sensitive to and correlated with the calibrated hydrological parameters.

The four compared ML architectures included GPR, GMC, Random Forest, and XGBoost. The GPR and GMC are capable of performing probabilistic predictions, and therefore can quantify the uncertainty in the predicted calibrated parameters. Random Forest and XGBoost yield a single-point prediction. The primary input information in the raw form includes the length of the river, the slope of the river, the percentage of vegetation, the elevation profile, and the catchment area. In order to build ML models to predict the calibrated streamflow parameters, we extract numeric features from the vegetation and elevation profiles, which are in *.tiff* image format in the raw form. To do so, we deploy two feature extraction techniques, statistical moment analysis and a convolution autoencoder.

This study has shown encouraging prediction results for this exceedingly complex problem with a better-than-expected coefficient of determination for most clusters. Judging on the coefficient of determination, all four ML techniques have performed reasonably well, with XGBoost being the best, followed by GPR and Random Forest, and GMC being the lowest in the performance rank. Our primary conclusion is that ML models that are carefully constructed by considering causal and sensitive input features (that are related to the rivers and the geographic conditions of the watersheds) do offer a potential approach that can not only obtain calibrated hydrological model parameters with reasonable accuracy but also side-step the current calibration challenges.

ACKNOWLEDGEMENTS

Funding for this work was provided by the United States Army Corps of Engineers through the 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

Biau, G. & Scornet, E. 2016 A random forest guided tour. *Test* **25**, 197–227.
Breiman, L. 2001 Random forests. *Machine Learning* **45** (1), 5–32.

Chen, S. S. & Gopalakrishnan, P. S. 1998 Clustering via the bayesian information criterion with applications in speech recognition. In: *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*. IEEE, Vol. 2, pp. 645–648.

Chen, T. & Guestrin, C. 2016 Xgboost: A scalable tree boosting system. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, 13-17 August 2016. ACM, New York, pp. 785–794.

Chen, M., Shi, X., Zhang, Y., Wu, D. & Guizani, M. 2017 Deep feature learning for medical image analysis with convolutional autoencoder neural network. *IEEE Transactions on Big Data* **7** (4), 750–758.

Chu, H.-J. & Chang, L.-C. 2009 Applying particle swarm optimization to parameter estimation of the nonlinear Muskingum model. *Journal of Hydrologic Engineering* **14** (9), 1024–1027.

Cunge, J. 1969 On the subject of a flood propagation computation method (Musklngum method). *Journal of Hydraulic Research* **7** (2), 205–230.

Das, A. 2004 Parameter estimation for Muskingum models. *Journal of Irrigation and Drainage Engineering* **130** (2), 140–147.

David, C. H., Maidment, D. R., Niu, G.-Y., Yang, Z.-L., Habets, F. & Eijkhout, V. 2011a River network routing on the nhdplus dataset. *Journal of Hydrometeorology* **12** (5), 913–934.

David, C. H., Habets, F., Maidment, D. R. & Yang, Z.-L. 2011b Rapid applied to the Sim-France model. *Hydrological Processes* **25** (22), 3412–3425.

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.

England Jr., J. 2018 Combining streamflow and rainfall-runoff hydrologic hazard estimates for dam safety: data, physics and uncertainty. In: *AGU Fall Meeting Abstracts*. Vol. 2018, Washington, DC, 10-14 December 2018. American Geophysical Union, Washington, DC, pp. NH33C–11.

Fernandez-Palomino, C. A., Hattermann, F. F., Krysanova, V., Vega-Jácome, F. & Bronstert, A. 2021 Towards a more consistent eco-hydrological modelling through multi-objective calibration: a case study in the Andean Vilcanota river basin, Peru. *Hydrological Sciences Journal* **66** (1), 59–74.

Frame, J. M., Kratzert, F., Raney, A., Rahman, M., Salas, F. R. & Nearing, G. S. 2021 Post-processing the national water model with long short-term memory networks for streamflow predictions and model diagnostics. *JAWRA Journal of the American Water Resources Association* **57** (6), 885–905.

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.

Gill, M. A. 1978 Flood routing by the Muskingum method. *Journal of Hydrology* **36** (3–4), 353–363.

Greegar, G. & Manohar, C. 2015 Global response sensitivity analysis using probability distance measures and generalization of Sobol's analysis. *Probabilistic Engineering Mechanics* **41**, 21–33.

Grogan, D. S., Zuidema, S., Prusevich, A., Wollheim, W. M., Glidden, S. & Lammers, R. B. 2022 Water balance model (wbm) v. 1.0. 0: a scalable gridded global hydrologic model with water-tracking functionality. *Geoscientific Model Development* **15** (19), 7287–7323.

Guo, X., Liu, X., Zhu, E. & Yin, J. 2017 Deep clustering with convolutional autoencoders. In *Neural Information Processing: 24th International Conference, ICONIP 2017*, November 14–18, 2017, Guangzhou, China. Proceedings, Part II 24, Springer, pp. 373–382.

Guse, B., Pfannerstill, M., Gafurov, A., Kiesel, J., Lehr, C. & Fohrer, N. 2017 Identifying the connective strength between model parameters and performance criteria. *Hydrology and Earth System Sciences* **21** (11), 5663–5679.

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. & Mahadevan, S. 2019 Probability models for data-driven global sensitivity analysis. *Reliability Engineering & System Safety* **187**, 40–57.

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.

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, L., Liu, X., Bai, P., Liang, K. & Liu, C. 2023 Comparison of flood simulation capabilities of a hydrologic model and a machine learning model. *International Journal of Climatology* **43** (1), 123–133.

Lucas-Picher, P., Arora, V. K., Caya, D. & Laprise, R. 2003 Implementation of a large-scale variable velocity river flow routing algorithm in the Canadian regional climate model (crcm). *Atmosphere-ocean* **41** (2), 139–153.

Maidment, D. R. 2017 Conceptual framework for the national flood interoperability experiment. *JAWRA Journal of the American Water Resources Association* **53** (2), 245–257.

Marcus, W. A. 1989 Lag-time routing of suspended sediment concentrations during unsteady flow. *Geological Society of America Bulletin* **101** (5), 644–651.

McCarthy, G. T. 1938 The unit hydrograph and flood routing. In: *Proceedings of Conference of North Atlantic Division*. US Army Corps of Engineers, Vol. 1938, pp. 608–609.

McRae, G. J., Tilden, J. W. & Seinfeld, J. H. 1982 Global sensitivity analysis – a computational implementation of the Fourier amplitude sensitivity test (fast). *Computers & Chemical Engineering* **6** (1), 15–25.

Meyer, A., Fleischmann, A. S., Collischonn, W., Paiva, R. & Jardim, P. 2018 Empirical assessment of flood wave celerity–discharge relationships at local and reach scales. *Hydrological Sciences Journal* **63** (15–16), 2035–2047.

Mohan, S. 1997 Parameter estimation of nonlinear Muskingum models using genetic algorithm. *Journal of Hydraulic Engineering* **123** (2), 137–142.

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.

Palermo, M. R., Schroeder, P. R., Estes, T. J. & Francingues, N. R. 2008 *Technical Guidelines for Environmental Dredging of Contaminated Sediments*. Tech. Rep., Cold Regions Research and Engineering Laboratory (US).

Qi, Y. 2012 Random forest for bioinformatics. In: *Ensemble Machine Learning* (Zhang, C. & Ma, Y. eds.) Springer, New York, pp. 307–323.

Ramancha, M. K., Vega, M. A., Conte, J. P., Todd, M. D. & Hu, Z. 2022 Bayesian model updating with finite element vs. surrogate mpdels: application to a miter gate structural system. *Engineering Structures* **272**, 114901.

Reynolds, D. A. 2009 Gaussian mixture models. In: *Encyclopedia of Biometrics*, Vol. 741 (Li, S. Z. & Jain, A., eds.). Springer, New York, pp. 659–663.

Rouholahnejad, E., Abbaspour, K. C., Vejdani, M., Srinivasan, R., Schulin, R. & Lehmann, A. 2012 A parallelization framework for calibration of hydrological models. *Environmental Modelling & Software* **31**, 28–36.

Sudret, B. 2008 Global sensitivity analysis using polynomial chaos expansions. *Reliability Engineering & System Safety* **93** (7), 964–979.

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.

Todini, E. 2007 Hydrological catchment modelling: past, present and future. *Hydrology and Earth System Sciences* **11** (1), 468–482.

Tung, Y.-K. 1985 River flood routing by nonlinear muskingum method. *Journal of Hydraulic Engineering* **111** (12), 1447–1460.

Williams, C. K. & Rasmussen, C. E. 2006 *Gaussian Processes for Machine Learning*, Vol. 2. MIT Press, Cambridge, MA.

Xu, T. & Liang, F. 2021 Machine learning for hydrologic sciences: an introductory overview. *Wiley Interdisciplinary Reviews: Water* **8** (5), e1533.

Xu, D.-M., Qiu, L. & Chen, S.-Y. 2012 Estimation of nonlinear muskingum model parameter using differential evolution. *Journal of Hydrologic Engineering* **17** (2), 348–353.

Yeates, E. M., Tavakoly, A. A., Mitchell, K. N., Dreaper, G. W. & Afshari, S. 2020 *Utilizing Stream Flows to Forecast Dredging Requirements*. Tech. Rep., Engineer Research and Development Center (US) Vicksburg United States.

Yoon, J. & Padmanabhan, G. 1993 Parameter estimation of linear and nonlinear muskingum models. *Journal of Water Resources Planning and Management* **119** (5), 600–610.

Zhang, Y., Chiew, F. H., Li, M. & Post, D. 2018 Predicting runoff signatures using regression and hydrological modeling approaches. *Water Resources Research* **54** (10), 7859–7878.

First received 13 February 2023; accepted in revised form 14 August 2023. Available online 1 September 2023