Apply Machine Learning Method And Neuron Network In Forecasting Water Level

Tran Quoc Khanh, Tran Bao Kha, and Truong Hoang Vu

FPT University, Da Nang, Viet Nam

Abstract. Forecasting of water levels in rivers and lakes is crucial for flood warning and water resource management. Traditional process-based numerical models, while precise, are computationally intensive and require extensive hydro-geomorphological data. As an alternative, data-driven methods leveraging statistical and machine learning techniques offer efficient and scalable solutions for real-time water level forecasting. This paper investigates the efficacy of various data-driven approaches, including ARIMA, Random Forest, Support Vector Regression (SVR), K-Nearest Neighbors (KNN), Long Short Term Memory (LSTM) networks and Convolutional Neural Networks (CNN), in predicting water levels. We detail the methodologies and implementation of these models, emphasizing their respective strengths and limitations. Through comprehensive experiments on hydrological datasets, we compare the performance of these models against traditional process-based approaches. Our findings reveal that data-driven models, particularly those utilizing machine learning, not only provide competitive accuracy but also significantly reduce computational complexity. This work aims to guide practitioners in selecting appropriate forecasting models and highlights the potential of integrating advanced machine learning techniques for enhanced water level prediction.

Keywords: Random Forest Regression; Artificial Neural Network; Time series Forecasting

1 Introduction

Preventing natural disaster and managing water resource is crucial for human's life, to do that, we have to have methods which give us precise result of water levels forecasting. Water level data, typically collected from hydrological stations, often exhibit a time series structure. Consequently, researchers frequently employ time series hydrological prediction models to forecast future levels. By utilizing past data to predict future water levels, hidden information can be uncovered, which is vital for mitigating the effects of floods, preventing disasters, and managing water resources effectively.

There are generally two approaches for building water level forecasting models: process-based numerical prediction and data-driven methods. Process-based models, such as HEC-RAS [Mai and De Smedt, 2017], MIKE-11 [Patro et al.,

2009, Kumar et al., 2019], and MIKE-21 [Zhu et al., 2013, Tran Anh et al., 2018], usually provide precise results and fully describe the nature of physical phenomena. However, they are computationally expensive and require various types of hydro-geomorphological data, including topography, geology, conduction roughness, and cross-sections [Fatichi et al., 2016]. Developing these forecasting systems also demands significant expertise to interpret the results and large volumes of data, which can be challenging to obtain for water level and flow predictions [Zhong et al., 2017, Fatichi et al., 2016, Di et al., 2014].

An alternative approach for building forecasting models is the use of data-driven methods (Riad et al., 2004; Wu et al., 2009; Ghumman et al., 2011; Peng et al., 2017; Gjika et al., 2019) [Riad et al., 2004, Wu et al., 2009, Ghumman et al., 2011, Peng et al., 2017, Dhamo Gjika et al., 2019]. Data-driven forecasting models aim to uncover relationships between features or hidden information within the data, primarily using information from available data. These models are often faster and easier to build, making them useful for real-time river and lake flow forecasting with accurate water level predictions. Data-driven models are mainly constructed using statistical and machine learning techniques.

Hydrological data are typically influenced by numerous external factors, resulting in time series that exhibit both linear and nonlinear characteristics. The Autoregressive Integrated Moving Average (ARIMA) model is a widely recognized and effective linear statistical tool for time series forecasting. On the other hand, machine learning (ML) techniques have been extensively used to develop forecasting models for nonlinear time series. Combining these two methods for improved time series forecasting is logical and has been supported by numerous studies demonstrating the efficacy of such hybrid approaches.

The success of hybrid models in various applications has motivated us to apply this approach for water level forecasting, specifically focusing on the Red river. We propose a data-driven hybrid model consisting of two components to address both linear and nonlinear aspects of water level time series, utilizing ARIMA and non-parametric statistical learning methods. Our hypothesis is that using dual-model approaches to capture both linear and nonlinear patterns in water level time series will reveal hidden patterns more effectively than single-model approaches. We plan to test and empirically validate this hypothesis by applying our model to water level forecasting for the Red river using three large datasets from different hydrological stations. [Phan and Nguyen, 2020].

2 Methodology

2.1 Autoregressive Integrated Moving Average model, ARIMA

ARIMA [Box and Jenkins, 1976], stands as one of the most widely used statistical linear models for forecasting univariate time series. The core concept of ARIMA involves decomposing a time series into present and past values, incorporating random errors. ARIMA integrates three primary components: autoregression (AR), moving average (MA), and differencing (d) to achieve stationarity. The

ARIMA model, denoted as ARIMA(p, d, q), is formulated as:

$$\Delta^d y(t) = c + \sum_{j=1}^{p} \alpha_j \cdot y(t-j) + \varepsilon(t) + \sum_{j=1}^{q} \beta_j \cdot \varepsilon(t-j) \tag{1}$$

where $\Delta = (1 - B)^d$, B is the backward shift operator (such that $By(t) = y(t-1)$), $y(t)$ denotes the observation at time t, c is a constant, $\alpha_1, \ldots, \alpha_p$ are autoregressive parameters, $\varepsilon(t)$ represents white noise at time t, and $\beta_1, \ldots, \beta_q$ are the moving average coefficients.

Fitting an ARIMA model typically involves four main steps:

1. **Identification of ARIMA(p, d, q) Structure**: Determining appropriate values of p, d, and q using methods such as autocorrelation function (ACF) and partial autocorrelation function (PACF) analysis, or criteria like AIC and BIC.
2. **Parameter Estimation**: Estimating coefficients α_j and β_j that minimize overall error.
3. **Diagnostic Checking**: Assessing model error assumptions through diagnostic statistics and residual plots.
4. **Forecasting**: Predicting future values using the fitted model based on available data.

The strength of ARIMA lies in its ability to transform non-stationary time series into stationary ones through differencing d times. Stationarity is critical for practical and reliable predictions. Hence, before fitting an ARIMA model, data transformation is often necessary if the data exhibits trends or heteroscedasticity. A time series is stationary when its mean, variance, and covariance (across different lags) remain constant over time.

In conclusion, ARIMA is favored over other statistical models for capturing linear components in time series data, making it a powerful tool for forecasting and analysis.

2.2 Random Forest

Random Forests (RF) represent a powerful ensemble learning technique introduced by Breiman in 2001. They are designed to enhance the predictive accuracy and stability of classification and regression models by leveraging the collective wisdom of diverse decision trees.

Core Principles of Random Forests

– **Ensemble of Diverse Trees**: RF builds an ensemble of decision trees, each trained on a bootstrap sample (randomly sampled with replacement) from the training data. Unlike traditional decision trees, RF introduces additional randomness by considering only a subset of features (random subspace sampling) at each split, promoting tree diversity and reducing correlation between trees.

- **Full-Grown Trees Without Pruning**: Each tree in the Random Forest is typically grown to its maximum depth without pruning. This approach aims to keep bias low while capturing complex relationships within the data.
- **Prediction Aggregation**: The final prediction of a Random Forest for a new input $x \in R^n$ is obtained by averaging predictions from all individual trees:

$$\hat{y} = \frac{1}{T} \sum_{i=1}^{T} \hat{f}_i(x) \tag{2}$$

  where T is the total number of trees in the forest, and $\hat{f}_i(x)$ denotes the prediction of the i-th tree.
- **Tuning Parameters**: Random Forests require tuning parameters to optimize performance. Notably:
  - m_{try}: The number of features considered for splitting at each node. Typically, m_{try} is set to $\sqrt{n}$ for classification and $\frac{n}{3}$ for regression tasks, where n is the total number of features.
  - n_{tree}: The number of trees T in the forest. Increasing T generally improves performance but comes at the cost of increased computational complexity.

Advantages of Random Forests

- **Robustness and Generalization**: By aggregating predictions from multiple trees and introducing randomness in the training process, Random Forests exhibit robust performance and reduce the risk of overfitting.
- **Versatility and Scalability**: RF can handle large datasets with high-dimensional feature spaces and can be parallelized for efficient computation, making them suitable for both small and large-scale applications.
- **Feature Importance**: RF provides insights into feature importance, helping to identify key variables that contribute most to predictive accuracy.

In conclusion, Random Forests offer a powerful and flexible framework for predictive modeling across various domains. By harnessing the strength of ensemble learning and introducing controlled randomness, RF significantly enhances the reliability and accuracy of machine learning models.

2.3 Support Vector Regression

Support Vector Regression (SVR) is the regression counterpart of Support Vector Machines (SVM), a machine learning algorithm grounded in statistical learning theory [Vapnik, 1995]. SVMs aim to find an optimal hyperplane in a high-dimensional feature space to separate linearly separable patterns. Multiple hyperplanes can satisfy this condition, but the objective is to identify the one that maximizes the margin around the separating hyperplane. This is achieved with support vectors, which are data points closest to the decision surface and crucial for determining the optimal placement of the hyperplane.

The linear regression function for estimation is represented as:

$$f(x) = w^T x + b \tag{3}$$

where w is the weight vector, b is the bias term, and x is the input vector. SVMs can handle classification or regression tasks that are not linearly separable by mapping the original data into a higher-dimensional space using Kernel functions. This transformation, known as the feature space, enhances the separability of classes.

2.4 K-Nearest Neighbors

K-nearest neighbors (KNN) [Altman, 1992], is a widely used non-parametric statistical learning method applicable to both classification and regression tasks. This algorithm predicts values for new data points based on a similarity measure, such as a distance function. Essentially, a new data sample is assigned a value based on its proximity to the points in the training set. Specifically, for each test sample, the distance (e.g., Euclidean, Manhattan, or Minkowski) between that sample and all samples in the training set is computed. The K closest training data samples are then selected. The predicted value is the average of the target output values of these K nearest neighbors.

2.5 Long Short Term Memory, LSTMs

Long Short Term Memory (LSTM) networks [Hochreiter and Schmidhuber, 1997], represent a specialized type of recurrent neural networks (RNNs) capable of learning long-distance dependencies. They have found extensive applications in sequence learning tasks and have become a fundamental building block in many prominent deep learning systems. LSTM networks excel at retaining information over extended time intervals and transferring it to subsequent cells. They possess the intrinsic ability to discern and remember significant information without explicit intervention.

The LSTM's ability to manage long-term dependencies and selective information retention makes it highly effective for modeling sequential data in various domains.

The LSTM network comprises interconnected LSTM memory cells, and the architecture of each cell is depicted in Figure 3.

The concept behind LSTM involves considering not only the hidden state h but also the cell internal state c and three gates: the forget gate f_t, input gate i_t, and output gate o_t. At each time step t, these gates receive input values x_t and h_{t-1}, derived from the output of the previous memory cell (at time $t-1$). Each gate serves a distinct filtering purpose:

- **Forget gate**: Removes unnecessary information from the internal cell state.
- **Input gate**: Determines which necessary information should be added to the internal cell state.
- **Output gate**: Decides which information from the internal cell state should be used as the output.

2.6 Convolutional Neural Network

The convolutional neural network (CNN) architecture used in this study processes time series data through multiple layers to predict future values. The architecture begins with an input layer that receives a fixed-length sub-sequence of the time series. This input is then passed through convolutional and pooling layers, followed by a Rectified Linear Unit (RELU) layer and an output layer:

- **Convolutional Layers**: A convolutional layer applies a series of filters to the input data, performing a weighted sum of the inputs within a specified window. This operation smooths the time series and extracts relevant features. Mathematically, for an input vector $\boldsymbol{x} \in R^p$ and a filter $\boldsymbol{w} \in R^k$, the convolution at position i is computed as:

$$\boldsymbol{w}^T \boldsymbol{x}[i : i + k] \tag{4}$$

  Multiple filters can be applied in parallel to capture different features from the input data.
- **Booling**: Pooling layers further reduce the dimensionality of the data by grouping adjacent values and summarizing them using a statistic, such as the maximum, average, or minimum. This process reduces noise and emphasizes the most significant aspects of the data. Common pooling strategies include max pooling, average pooling, and min pooling.
- **RELU Layer**: The RELU layer introduces non-linearity to the network by applying the Rectified Linear Unit function to the output of the pooling layer. This non-linear transformation helps the network learn complex patterns in the data.
- **Output Layer**: The final output layer combines the features extracted by the convolutional and pooling layers, applies a weighted sum, and uses an activation function to produce the desired output. Depending on the task, this could be class probabilities, continuous values, count data, or ordinal data. This layer ensures that the model's predictions are in a suitable format for evaluation and interpretation.

3 Experimental results

Firstly, we describe the datasets and the pre-processing steps involved in our experiments. Next, we define the performance evaluation metrics. Finally, we discuss the experimental results, highlighting key findings and observations.

3.1 Data representation and pre-processing

In this study, we utilize water level data from three hydrology stations on the Red river, which is the primary river in the Northern Delta of Vietnam, located in Hanoi. This data is used to evaluate the proposed hybrid approach and to compare its performance against established machine learning methods.

– **Hanoi Datasets:** The water level data for this study were collected from the and Hanoi hydrology stations in Vietnam, spanning from January 1, 2008, to December 31, 2017. The sampling frequency varied depending on the weather conditions. On days without rainfall or flooding, data were sampled infrequently—approximately 4 to 8 times per day, with some days having no data at all. Conversely, during rainy periods, sampling occurred more frequently, ranging from 10 to 23 times a day.

  To prepare this data for time series forecasting algorithms, it was necessary to resample the data at consistent time intervals. We normalized the sampling frequency to 8 times per day at specific hours: 1h, 4h, 10h, 13h, 16h, 19h, and 22h. The preprocessing steps involved two scenarios:

  - **Scenario 1: Fewer than 8 Samples per Day or No Data Available**
    For days with fewer than 8 samples or no data, the missing periods were treated as gaps. Imputation methods such as linear interpolation and moving averages were used to fill in these gaps. In cases of extended missing data (e.g., a full day or more), linear interpolation would result in a straight line, which fails to capture the dynamic nature of the data. To address this, we employed the DTWBI method [Phan et al., 2017], which completes consecutive missing data while maintaining the data's variability.
  - **Scenario 2: More than 8 Samples per Day**
    For days with more than 8 samples, the data were resampled to match the specified times of 1h, 4h, 7h, 10h, 13h, 16h, 19h, and 22h.

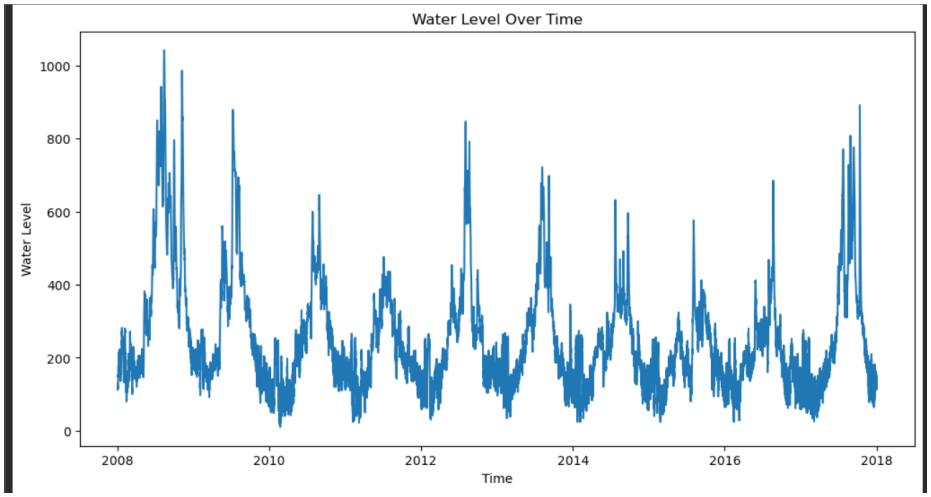

Fig. 1. Water level at the Hanoi hydrology station

No	Dataset name	Period	# Samples	Trend (Y/N)	Seasonal (Y/N)	Frequency
1	Hanoi	2008-2017	29224	N	Y	3 hours

Table 1. Characteristics of the water level time series

3.2 Performance evaluation indicators

After the prediction phase, six evaluation metrics were utilized to assess various models. These metrics include MAE, RMSE, FSD, and NSE. Each of these metrics was chosen for its unique properties, which are crucial for comprehensively understanding the performance of forecasting models from multiple perspectives. The definitions of these metrics are as follows:

1. MAE: The Mean Absolute Error (MAE) between the predicted values y and the actual values x is computed as:

$$\mathrm{MAE}(y, x) = \frac{1}{T}\sum_{i=1}^{T}|y_i - x_i|$$

   A lower MAE value means better performance method for the prediction task.
2. RMSE: The Root Mean Square Error is defined as the average squared difference between the forecast values y and the respective true values x. This indicator is very useful for measuring overall precision or accuracy. In general, the most effective method would have the lowest RMSE.

$$\mathrm{RMSE}(y, x) = \sqrt{\frac{1}{T}\sum_{i=1}^{T}(y_i - x_i)^2}$$

3. FSD: The Fraction of Standard Deviation is defined as:

$$\mathrm{FSD}(y, x) = 2 \cdot \frac{|\mathrm{SD}(y) - \mathrm{SD}(x)|}{\mathrm{SD}(y) + \mathrm{SD}(x)}$$

   This fraction points out whether a method is acceptable or not. For a forecasting method, when the FSD value approaches 0, it is impeccable.
4. NSE: The Nash-Sutcliffe efficiency is used to evaluate the predictive ability of hydrological models. The NSE values range from $-\infty$ to 1, with higher values indicating a better fit between observed and forecasted water levels (Nash and Sutcliffe, 1970).

$$\mathrm{NSE} = 1 - \frac{\sum_{i=1}^{T}(x_i - y_i)^2}{\sum_{i=1}^{T}(x_i - \overline{x})^2}$$

3.3 Results and Discussions

To assess and compare all the tested methods, the entire dataset was split into two segments: one for training and the other for testing. For the Hanoi time series, the training datasets comprised data from 2008 to 2015, representing 80% of the total data. The remaining 20% of the observed samples, collected from 2016 to 2017, were utilized to evaluate the forecasting models.

The problem of water level forecasting was tested using KNN, SVR, RF, LSTM, and other models. In this study, we developed multi-step ahead prediction models based on one-step ahead forecasting. This approach involves using p previous values, $y(t-1), y(t-2), \ldots, y(t-p)$, to predict the value $\hat{y}(t)$ at time t. For subsequent predictions, $\hat{y}(t)$ is used as one of the input values to forecast the next value $\hat{y}(t+1)$, and this process continues iteratively.

Determining the parameters of these models is crucial as they significantly influence forecasting accuracy. In this study, when constructing machine learning methods, we employed 5-fold cross-validation to identify the optimal parameters for each model. Specifically, for the ARIMA model, we utilized the `pmdarima.auto_arima()` function from the `pmdarima` python package to determine the parameters p, d, and q. Additionally, we accounted for seasonality in all datasets when training the ARIMA models. Table 2 presents the details of our parameter configurations.

We found parameters from ARIMA, Optimal parameters: p=5, d=1, q=5.
Parameters for all experiments:
Library: keras Metrics: Accuracy Loss function: MSE Epochs=200
Validation split=0.05 Batch size=3 Optimizer function: Adam

Table 2 show the results of all models we use in this experiment. For this time series, the forcasting models were set up for 12h,48h, 72h and 5-days ahead time intervals. The best results for each forcasting horizon are highlighted in bold. We can see that in most cases, RF give far better results than other models (LSTM, CNN, KNN, SVM). LSTM performs pretty well when compare to other machine learning models except RF and better than CNN.

From table 2, it shows that SVM has the lowest MAE, RMSE and FSD. LSTM has the worst NSE but overall, LSTM achieved good MAE, RMSE, FSD. This indicates that LSTM produces better results than other methods. Surprisely, a deep learning model like CNN does not produces good results but training time is so long.

Method	Size	MAE	RMSE	FSD	NSE
RF	12h	**0.015**	**0.019**	0.33	**-0.52**
LSTM		0.48	0.54	1.88	-6077.01
KNN		2.4	3.52	0.92	0.99
CNN		1.06	1.18	-9.25	0.02
SVM		8.64	13.3	**-0.17**	0.98
RF	24h	**0.024**	**0.028**	0.42	-0.82
LSTM		0.62	0.65	1.90	-16054.93
KNN		10.18	14.96	**0.38**	0.99
CNN		1.81	2.22	-4.88	-1.81
SVM		10.27	15.05	0.38	0.99
RF	48h	**0.024**	**0.028**	**0.3**	-0.97
LSTM		0.68	0.70	1.90	-32230.68
KNN		7.78	11.63	0.68	0.99
CNN		3.40	5.64	-0.52	**-0.22**
SVM		9.45	13.35	-0.58	0.99
RF	72h	**0.025**	**0.029**	**0.21**	-1.38
LSTM		0.68	0.69	1.65	-2787.71
KNN		7.43	10.51	0.71	0.99
CNN		4.96	7.63	-2.16	**-1.17**
SVM		8.45	12.17	0.62	0.99
RF	5days	**0.023**	**0.027**	**0.12**	-1.42
LSTM		0.66	0.67	1.21	-672.66
KNN		6.59	9.22	0.73	0.99
CNN		3.98	6.20	-2.27	**-1.12**
SVM		7.48	10.64	0.64	0.99

Table 2. Performance metrics for different methods on the Hanoi dataset

It can also be seen from table 2 that in general, SVM is not suitable for forecasting water level on Red river because the further we predicted, the worse result we get. A interesting result we can retrieve from the result table is with LSTM models, MAE, RMSE and FSD is stable when we predict further in the future.

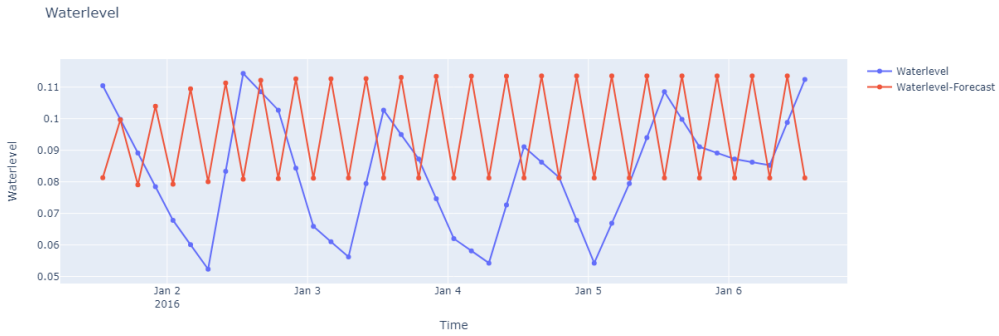

Fig. 2. Forecast first 5 days of the test set

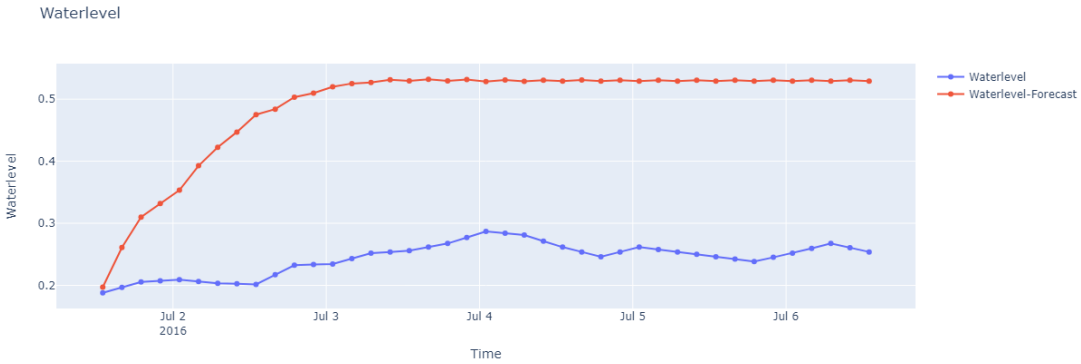

Fig. 3. Forecast randomly 5 other days in the test set

For RF model, although it produces good results for most cases but in general, the further we predict in the future, the larger error we make. So, RF is the best method to apply when the predict time is near, but when the predict time is far in the future, LSTM is more stable.

4 Conclusion

Accurate time series forecasting is a crucial and also challenging task, especially when we apply it into life-changing field like flood warning system and water level forecasting. A lot of widely popular and effective forecasting models like deep learning and machine learning method can be apply to solve the problem. The question is which one most effective when it comes to Red river case. Therefore, in this paper, we apply most popular models and compare with each other to see which one is the most effective in this case. These models have been tested in a real big datasets on the water level time series of the Red river. The results showed that the RF model produced most accurate and reliable results than other models. LSTM is also a good model to apply in this case cause the errors when we predict far in the future is very stable. In the future, we plan to apply RF and LSTM models for forecasting water level on other horological stations of the Red river as well as for other rivers

References

[Altman, 1992] Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175.

[Box and Jenkins, 1976] Box, G. and Jenkins, G. (1976). *Time Series Analysis: Forecasting and Control*. Holden-Day series in time series analysis and digital processing. Holden-Day.

[Dhamo Gjika et al., 2019] Dhamo Gjika, E., Ferrja, A., and Kamberi, A. (2019). A study on the efficiency of hybrid models in forecasting precipitations and water inflow albania case study. *Advances in Science, Technology and Engineering Systems Journal*, 4.

[Di et al., 2014] Di, C., Yang, X., and Wang, X. (2014). A Four-Stage Hybrid Model for Hydrological Time Series Forecasting. *PLoS ONE*, 9(8):e104663.

[Fatichi et al., 2016] Fatichi, S., Vivoni, E. R., Ogden, F. L., Ivanov, V. Y., Mirus, B., Gochis, D., Downer, C. W., Camporese, M., Davison, J. H., Ebel, B., Jones, N., Kim, J., Mascaro, G., Niswonger, R., Restrepo, P., Rigon, R., Shen, C., Sulis, M., and Tarboton, D. (2016). An overview of current applications, challenges, and future trends in distributed process-based models in hydrology. *Journal of Hydrology*, 537:45–60.

[Ghumman et al., 2011] Ghumman, A. R., Ghazaw, Y., Sohail, A., and Watanabe, K. (2011). Runoff forecasting by artificial neural network and conventional model. *Alexandria Engineering Journal*, 50:345–350.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

[Kumar et al., 2019] Kumar, P., Lohani, A. K., and Nema, A. (2019). Rainfall runoff modeling using mike 11 nam model. *Current World Environment*, 14:27–36.

[Mai and De Smedt, 2017] Mai, D. T. and De Smedt, F. (2017). A combined hydrological and hydraulic model for flood prediction in vietnam applied to the huong river basin as a test case study. *Water*, 9(11).

[Patro et al., 2009] Patro, S., Chatterjee, C., Mohanty, S., Singh, R., and Raghuwanshi, N. (2009). Flood inundation modeling using mike flood and remote sensing data. *Journal of the Indian Society of Remote Sensing*, 37:107–118.

[Peng et al., 2017] Peng, T., Zhou, J., Zhang, C., and Fu, W. (2017). Streamflow forecasting using empirical wavelet transform and artificial neural networks. *Water*, 9:406.

[Phan and Nguyen, 2020] Phan, T.-T.-H. and Nguyen, X. H. (2020). Combining statistical machine learning models with arima for water level forecasting: The case of the red river. *Advances in Water Resources*, 142:103656.

[Phan et al., 2017] Phan, T.-T.-H., Poisson Caillault, E., Lefebvre, A., and Bigand, A. (2017). Dynamic time warping-based imputation for univariate time series data. *Pattern Recognition Letters*, 139.

[Riad et al., 2004] Riad, S., Mania, J., Bouchaou, L., and Najjar, Y. (2004). Rainfall-runoff model usingan artificial neural network approach. *Mathematical and Computer Modelling*, 40(7):839–846.

[Tran Anh et al., 2018] Tran Anh, D., Hoang, L. P., Bui, M. D., and Rutschmann, P. (2018). Simulating future flows and salinity intrusion using combined one- and two-dimensional hydrodynamic modelling—the case of hau river, vietnamese mekong delta. *Water*, 10(7).

[Vapnik, 1995] Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer New York.

[Wu et al., 2009] Wu, C., Chau, K., and Li, Y. (2009). Methods to improve neural network performance in daily flows prediction. *Journal of Hydrology*, 372(1):80–93.

[Zhong et al., 2017] Zhong, C., Guo, T., Jiang, Z., Liu, X., and Chu, X. (2017). A hybrid model for water level forecasting: A case study of wuhan station. In *2017 4th International Conference on Transportation Information and Safety (ICTIS)*, pages 247–251.

[Zhu et al., 2013] Zhu, C., Liang, Q., Yan, F., and Hao, W. (2013). Reduction of waste water in erhai lake based on mike21 hydrodynamic and water quality model. *TheScientificWorldJournal*, 2013:958506.