16th International Conference on Advanced Computational Intelligence (ICACI 2024)
Zhangjiajie, China, May 16-19, 2024

Bayesian Optimized CNN-RNN Hybrid Model for Predicting Streamflow in Potomac River Basin

Nian Zhang
Department of Electrical and Computer Engineering
University of the District of Columbia
Washington, D.C., USA
nzhang@udc.edu

Stephanie Rouamba
Department of Electrical and Computer Engineering
University of the District of Columbia
Washington, D.C., USA
sougrenonma.rouamba@udc.edu

Gavin Robinson
Department of Electrical and Computer Engineering
University of the District of Columbia
Washington, D.C., USA
gavin.robinson@udc.edu

Tolessa Deksissa
Water Resources Research Institute
University of the District of Columbia
Washington, D.C., USA
tdeksissa@udc.edu

2024 16th International Conference on Advanced Computational Intelligence (ICACI) | 979-8-3503-0850-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICACI60820.2024.10537003

***Abstract*—Accurately estimating streamflow quantity within a watershed is essential for anticipating potential severe drought conditions in the future. This paper introduces a hybrid model that combines Convolutional Neural Networks and Recurrent Neural Networks (CNN-RNN) to predict streamflow discharge using historical data from the Potomac River Basin. The experimental outcomes showcase the model's proficiency in successfully predicting streamflow discharge values for the upcoming month. Additionally, a new sequence, derived from a one-step-ahead approach, is constructed to forecast streamflow discharge values beyond the testing data horizon. Notably, the CNN-RNN model demonstrates superior performance with a lower Symmetric Mean Absolute Percentage Error (sMAPE) at 0.048592 compared to the RNN method at 0.067662. Moreover, the prediction accuracy of the CNN-RNN model surpasses that of the RNN model. The correlation between the test sequence and the CNN-RNN prediction is notably high at 0.97753, while the RNN correlation lags slightly at 0.96694. Furthermore, the results affirm the CNN-RNN model's potential capability to forecast unobserved values beyond the time series horizon when observed data is unavailable.**

***Index Terms*—Time series prediction, water quantity prediction, machine learning, deep learning, convolutional neural networks (CNNs), recurrent neural networks (RNNs), long short-term memory Networks (LSTM).**

I. INTRODUCTION

The streamflows within a river basin can hover around or dip below normal levels, influenced by below-average precipitation and significantly reduced soil moisture content. If the trend of below-average rainfall persists, further deterioration is anticipated. Regular monthly monitoring of the river basin is crucial to anticipate the potential development of severe drought conditions in the future [1].

As the challenges posed by drought and streamflow quantity intensify, establishing an effective early warning system becomes imperative. This system should utilize historical data to predict the likelihood of flow dropping below drought trigger levels. Precisely gauging streamflow poses a notable challenge for professionals in water resources and water management districts. However, it is indispensable for the effective management of water supply, flood control, and drought mitigation. The characteristics of streamflow are chiefly shaped by climatic conditions and features within the watershed.

Precise estimation of streamflow quantity from a watershed is crucial for informing urban watershed modeling, water quantity management, legislative development, and water supply strategies. Additionally, drought prediction stands out as one of the most intricate hydrological challenges due to the random and unpredictable nature of drought variables and the complexity of underlying physical processes. The intricacy is further heightened by a constrained understanding of influencing factors and their effects on streamflow, coupled with a deficiency in reliable prediction and design methodologies. Consequently, precise drought prediction, encompassing predictions for streamflow quantity, becomes imperative to improve water resource management plans and operational performance assessments.

In recent developments, several deep learning algorithms have found success in addressing water quantity prediction and drought prediction challenges. Recurrent Neural Networks (RNNs) offer several advantages for time series prediction tasks. They can handle sequential data of varying lengths, capturing long-term dependencies and temporal patterns effectively. RNNs accommodate irregularly spaced time intervals and adapt to different forecasting tasks with input and output sequences of varying lengths.

However, RNNs have limitations like the vanishing or exploding gradient problem, which affects their ability to capture long-term dependencies because RNNs may be unrolled very far back in this Memory constraints may also limit their performance with very long sequences. The

vanishing gradient problem is a challenge that affects the training of deep neural networks, including Recurrent Neural Networks (RNNs). It occurs when gradients, which indicate the direction and magnitude of updates to network weights during training, become very small as they propagate backward through layers. The vanishing gradient problem is particularly problematic in sequences where information needs to be remembered or propagated over a long span of time, affecting the network's ability to capture important patterns. This phenomenon hinders the ability of RNNs to learn long-range dependencies and can lead to slow or ineffective training.

However, this challenge is elegantly addressed by LSTM which is a specific kind of RNN, as it incorporates specialized memory cells and gating mechanisms that preserve and control the flow of gradients over extended sequences [2]. This enables the network to capture long-term dependencies more effectively and significantly enhances its ability to learn from sequential data. LSTM has three gates (input, forget, and output) and excels at capturing long-term dependencies. In addition, Gated Recurrent Unit (GRU), a simplified version of LSTM with two gates (reset and update), maintains efficiency and performance similar to LSTM, making it widely used in time series tasks.

Nevertheless, it shows that in some specific applications, the combination of convolutional neural networks (CNNs) and recurrent neural networks (RNNs), often referred to as CNN-RNN, can offer certain advantages over Long Short-Term Memory Recurrent Neural Networks (LSTM)-RNN [3]-[5]. There are some known advantages of CNN-RNN over LSTM-RNN: (1) Spatial Hierarchies in Data: CNNs are well-suited for capturing spatial hierarchies and patterns in data. If the time series data has inherent spatial structures or patterns, a CNN can efficiently capture these features, providing an advantage over LSTM, which primarily focuses on temporal dependencies; (2) Feature Extraction: CNNs excel at feature extraction from input data. They can automatically learn relevant spatial features, reducing the need for manual feature engineering. When working with time series data, especially in applications such as video analysis or sensor data, a CNN's ability to automatically extract spatial features can be beneficial; (3) Parallel Processing: CNNs can process input data in parallel, which is particularly advantageous for large-scale data. In scenarios where parallel processing is crucial, such as processing multiple time series simultaneously, CNN-RNN architectures may be more efficient compared to the sequential processing of LSTM-RNNs.

The combination of CNNs and RNNs for water quantity prediction is a relatively less explored area in the field of hydroinformatics and environmental modeling [6]. No detailed work has been found on CNN-RNN. Since both CNNs and RNNs have their unique strengths, and their combination can be powerful for handling spatial and temporal dependencies in water quantity data [7]. Therefore, it's imperative to investigate the performance of CNN-RNN on streamflow prediction and compare its performance with other models, such as LSTM-RNN. We will utilize diverse data sources, including historical water quantity measurements to enhance the model's ability to generalize across different environmental conditions.

The remainder of the paper is organized as follows: Section II details the methodology, encompassing the LSTM-RNN and CNN-RNN. Section III delves into time series prediction using the LSTM network. Section IV provides the experimental results. Section V concludes the paper.

II. PROPOSED METHODS

A. LSTM-RNN Model for Time Series Prediction

The depicted diagram Fig. 1 showcases the data flow within an LSTM layer, with input denoted as $\mathbf{x}$ and output as $\mathbf{y}$ across T time steps. Here, $\mathbf{h}_t$ signifies the output, also referred to as the *hidden state*, while $\mathbf{c}_t$ represents the cell state at each time step, t.

When the layer produces the entire sequence, it yields $\mathbf{y}_1, \ldots, \mathbf{y}_T$, equivalent to $\mathbf{h}_1, \ldots, \mathbf{h}_T$. However, if the layer outputs solely the last time step, it generates $\mathbf{y}_T$, matching $\mathbf{h}_T$. The output's channel count aligns with the number of hidden units within the LSTM layer.

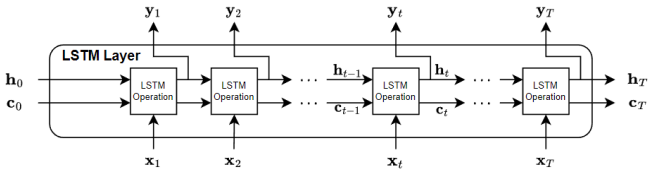


Fig. 1. Unfolded single layer of LSTM network.

The initial LSTM operation utilizes the RNN's initial state alongside the sequence's first time step to calculate the initial output and update the cell state [8]. As for time step t, the operation employs the current RNN state (c_{t-1}, h_{t-1}) and the subsequent time step of the sequence to compute both the output and the updated cell state, c_t.

The layer's state encompasses both the *hidden state*, also referred to as the *output state*, and the *cell state*. At time step t, the hidden state holds the LSTM layer's output for that specific step, while the cell state retains knowledge acquired from preceding steps. With each time step, the layer either adds or removes information from the cell state, a process regulated by gates.

In contrast to the conventional RNN, the LSTM distinguishes itself as a recurrent neural network equipped with built-in gates. At each time step, the LSTM layer has the capability to selectively incorporate or discard information from the cell state, with these actions governed by gates. The gated circuit within the LSTM is intentionally crafted to regulate the flow of data at time step t, as depicted in Fig. 2. The introduction of self-loops in the LSTM creates pathways that allow gradients to endure for extended periods, enhancing the model's capability to effectively learn long-term dependencies.

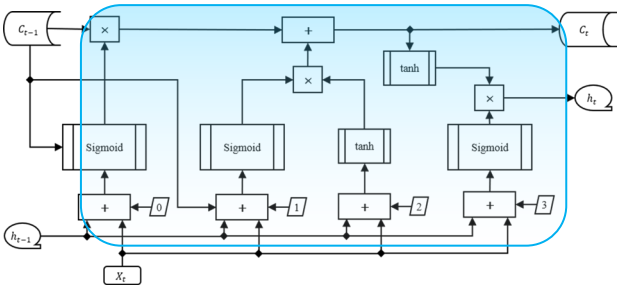


Fig. 2. Block diagram illustrating LSTM operations applied to a time series sequence.

The trainable parameters of an LSTM layer encompass the input weights W (Input Weights), the recurrent weights R (Recurrent Weights), and the bias b (Bias). Matrices W, R, and b are constructed by amalgamating the input weights, the recurrent weights, and the bias of each component, respectively. The layer merges these matrices based on the following equations:

$$W = \begin{pmatrix} W_i \\ W_f \\ W_g \\ W_o \end{pmatrix} \quad R = \begin{pmatrix} R_i \\ R_f \\ R_g \\ R_o \end{pmatrix} \quad b = \begin{pmatrix} b_i \\ b_f \\ b_g \\ b_o \end{pmatrix}$$

where i, f, g, and o denote the input gate, forget gate, cell candidate, and output gate, respectively.

The equations describing the operations are listed below.

$$f_t = \sigma_g(W_f x_t + R_f h_{t-1} + b_f) \quad (1)$$

$$i_t = \sigma_g(W_i x_t + R_i h_{t-1} + b_i) \quad (2)$$

$$o_t = \sigma_g(W_o x_t + R_o h_{t-1} + b_o) \quad (3)$$

$$g_t = \sigma_c(W_g x_t + R_g h_{t-1} + b_g) \quad (4)$$

σ_g denotes the sigmoid function

σ_c denotes the hyperbolic tangent function

The cell state at time step t is given by

$$c_t = f_t \circ c_{t-1} + i_t \circ g_t \quad (5)$$

Where $\circ$ denotes the element-wise multiplication of vectors.

The hidden state at time step *t* is given by

$$h_t = o_t \circ \sigma_c(c_t) \quad (6)$$

B. CNN-RNN Model for Time Series Prediction

A hybrid CNN-RNN model will be developed to address the water quantity prediction. This model complements the advantages of both CNNs and RNNs. On one hand, CNNs are well-suited for extracting spatial features from input data. In the context of water quantity prediction, satellite imagery or sensor data from various geographical locations can be fed into the CNN layers. These layers can capture spatial patterns, such as land cover, terrain characteristics, and other relevant features. RNNs, on the other hand, are effective in capturing temporal dependencies in sequential data. Time-series data related to water quantity, such as streamflow measurements, precipitation, and temperature, can be input into the RNN layers. The RNN component helps in modeling the dynamic behavior and dependencies over time.

In the new hybrid CNN-RNN architecture, the CNN and RNN layers can be integrated in a hybrid architecture. The output from the CNN layers capturing spatial features can be fed into the RNN layers to model temporal dependencies. This combination allows the model to learn both spatial and temporal patterns simultaneously. The hybrid CNN-RNN architecture is shown in Fig. 3.

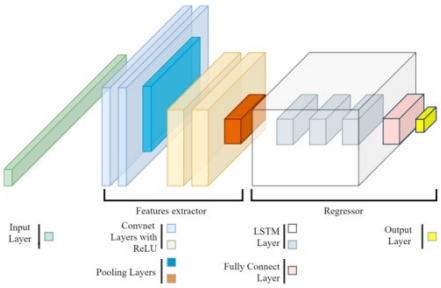


Fig. 3. The hybrid CNN-RNN architecture.

The CNN excels in extracting features, making it a strong neural network for this purpose, while the RNN has demonstrated its proficiency in predicting values within sequence-to-sequence series. In each time step, the CNN extracts key features from the sequence, whereas the RNN focuses on learning to predict the next value in the sequence. The input size of the sequence is lagged by n-months, consequently, the RNN anticipates an input size of n-months cases to generate the prediction for the subsequent month, specifically one step ahead.

In the performance evaluation, one of the most common evaluation metrics for water quantity prediction problem is the symmetric mean absolute percentage error (sMAPE). sMAPE, commonly employed to evaluate forecasting model performance, measures accuracy, with lower values indicating higher accuracy. This measure of accuracy relies on percentage (or relative) errors and is defined as follows:

$$\text{sMAPE} = \frac{100}{n}\sum_{t=1}^{n}\frac{|F_t - A_t|}{(|F_t| + |A_t|)/2} \quad (7)$$

where A_t represents the actual value and F_t represents the forecast value. The absolute difference between A_t and F_t is divided by half the sum of the absolute values of the actual value A_t and the forecast value F_t. The result is then summed for each fitted point *t* and further divided by the total number of fitted points *n*. The result is a percentage, and the goal is to minimize this percentage.

III. Experimental Results

In this section, we will perform a one-step ahead forecast, where we predict the next time point based on historical data. In addition, we will compare the forecast performance of CNN-RNN and RNN.

A. Study Area and Streamflow Data

Since we focus on Potomac River Basin, i.e. the Four Mile Run stream station at Alexandria, VA is selected to retrieve real-time streamflow water data from the U.S. Geological Survey (USGS)'s national water information system. Four Mile Run is a tributary of the Potomac River and is located in the Potomac River basin. It flows through the Washington, D.C. metropolitan area and into the Potomac River in northern Virginia. The Potomac River basin encompasses a large area, and many smaller rivers and streams, including Four Mile Run, contribute to the overall drainage system that feeds into the Potomac River.

The dataset comprises streamflow discharge values (measured in cubic feet per second) recorded from July 31, 2010, to November 20, 2010. The time series data are organized with timestamps and corresponding values., as illustrated in Fig. 4.

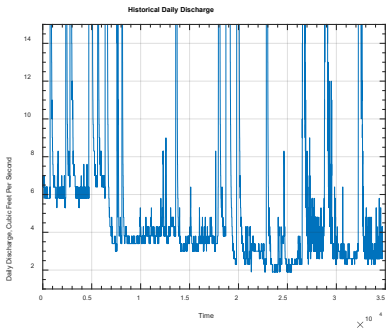


Fig. 4. Discharge information, measured in cubic feet per second, gathered at the Four Mile Run site in Alexandria, VA, spanning from July 31, 2010, to November 20, 2010.

Real-time time series data is typically logged at intervals of 15 to 60 minutes. Consequently, each graph displays 34,721 data points collected over the 120-day period. The input data is represented by a 34,721x1 matrix, capturing dynamic data across 34,721 time steps, each consisting of a single element.

Then the time series data will be split into training and testing sets. The training set will be used to train the model, and the testing set will be used to evaluate its performance. Thus, the data is randomly split into 34,721 time steps, with 90% allocated for training, contributing 31,249 target time steps. These are presented to the network during training, and adjustments are made based on the network's error. The remaining 10% of the data, consisting of 3,472 target time steps, is utilized for testing.

B. Feature Engineering

To improve the convergence process, the data is standardized. The most important task in feature engineering is to create lag features by shifting the target variable (the variable we want to predict) by a certain number of time steps to create lagged values. This represents the "look-back" effect. We have monthly data, so we create a lag feature with a lag of 15 month, which means using the value of the previous 15 months as a feature. In addition, hyperparameter tuning and optimization are also important to achieve the best results. The input parameters are specified as follows: Learning rate is set to 0.006. Moreover, we use the adaptive moment estimation, a.k.a. Adam optimization algorithm to train both CNN-RNN and RNN. It is an extension of the stochastic gradient descent (SGD) optimization method and is known for being computationally efficient. One of the main features of Adam is its adaptive learning rate. Furthermore, the Bayesian regularization backpropagation algorithm is used for training feedforward neural networks. This algorithm combines the advantages of both Bayesian regularization and Levenberg-Marquardt backpropagation. Bayesian regularization is used to prevent overfitting by adding a penalty term to the error function, while the Levenberg-Marquardt algorithm is employed for backpropagation to update updates the weight and bias values.

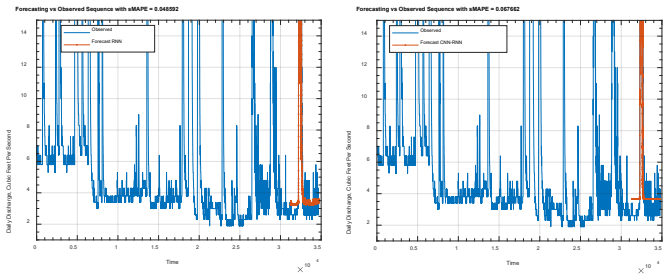
C. Model Training of CNN-RNN and RNN

For the proposed CNN-RNN time series forecasting model, the maximum number of epochs for training is set to 600. The learning rate is reduced by a factor of 0.25 every 96 epochs. A mini-batch size with 64 observations is used at each iteration. On the other hand, RNN time series forcasting model uses the same setting. We train the two models on the training set using the lagged features as input and the target variable as output.

D. Forecasting the Testing Data

We forecast the streamflow discharge value for the next month (next step). The network is expecting a sequence of Lag values roll back window to predict the streamflow discharge value for the next month. The trained network still "remember" the training time sequence and it expects a new sequence to predict one step ahead. The training time series and the forecasted values for CNN-RNN and RNN are plotted in Fig. 5 (a) and (b), respectively.

As we can observe from Fig. 5, the sMAPE for CNN-RNN in (a) is lower, which is 0.048592, while the sMAPE for RNN in (b) is higher, which is 0.067662. A lower sMAPE indicates that the forecasted values are closer to the actual values, reflecting better accuracy in the forecasting model.



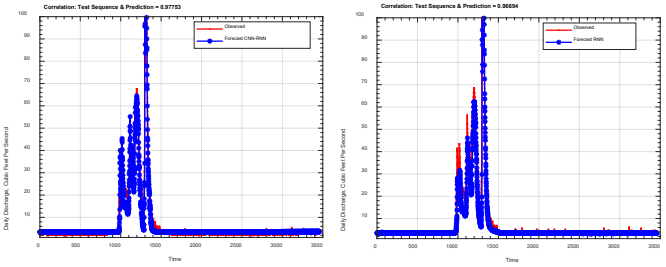
(a) (b)

Fig. 5. The forecasted values are show in red, while the observed values (targets) are shown in blue. (a) CNN-RNN (b) RNN.

E. Correlation between forecasted values and the testing data

In addition, we compare the forecasted values with the testing data. The result is shown in Fig. 6. As we can see from Fig. 6, the correlation of the test sequence and prediction for CNN-RNN is higher, which is 0.97753, while the correlation

of the test sequence and prediction for RNN is lower, which is 0.96694.

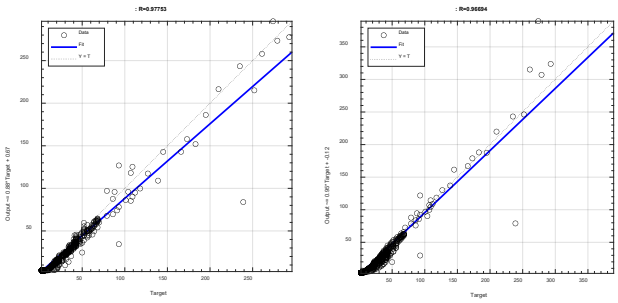


(a) (b)

Fig. 6. Comparison of the forecasted values with the testing data. (a) CNN-RNN (b) RNN.

In order to better visualize the correlation, the correlation between the forecasted values and the testing data for CNN-RNN and RNN can be plotted in Fig. 7 (a) and (b). Fig. 7 shows that CNN-RNN has higher correlation, which is 0.97753, and RNN has lower correlation, which is 0.96694. In time series prediction, the correlation between the forecasted values and the testing data is an important measure of the accuracy of the predictive model. A higher correlation between the test sequence and the predictions is considered better.

The correlation coefficient quantifies the strength and direction of a linear relationship between two variables. In the context of forecasting or prediction, a high positive correlation suggests that the predicted values move in the same direction as the actual values. For a perfect prediction model, all of the correlations should be 1.



(a) (b)

Fig. 7. The correlation between the forecasted values and the testing data for CNN-RNN and RNN. (a) CNN-RNN (b) RNN.

F. Forecasting Unobserved Values Beyond the Horizon

We also conduct some preliminary experiments on the ability of the CNN-RNN to predict sequences extending beyond the testing data. This experiment elucidates the approach for approximately forecasting "future" streamflow discharge values in the absence of observable data. The trained network retains its memory, signifying its ability to recall the sequence from the training data period. Specifically, the network retains information from the last lag time step. Consequently, the network anticipates a new sequence, enabling it to predict one step ahead.

Initially, we establish a novel network and transfer its knowledge and memory to a fresh variable. Subsequently, the network is prepared to extend its prediction to the subsequent step. However, generating a new sequence becomes imperative, relying on the previous prediction. In this phase, we aim to forecast a sequence matching the number of steps in the testing data (horizon). The earlier predicted value is placed at the beginning of the sequence, and the list is shifted downward to maintain a consistent number of features. Fig. 8 illustrates the forecasted sequence derived from the testing data (complete testing sequence) alongside the new prediction based on a sequence constructed from the preceding forecasted value.

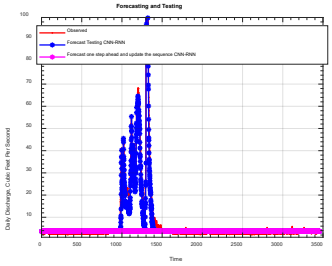


Fig. 8. The forecasted sequence derived from the testing data (complete testing sequence) alongside the new prediction based on a sequence constructed from the preceding forecasted value for CNN-RNN.

From Fig. 8, we observe a decline in prediction accuracy as we extend further away from the last known time step in the sequence. The disparity between the aforementioned testing prediction and this final prediction is contingent on the sequence information. The testing prediction encompasses a complete sequence of observable values, whereas this ultimate prediction comprises only a limited set of observable values, with the remainder being constructed or populated through bootstrapping from the preceding predicted value.

IV. Conclusions

This paper proposes a hybrid convolutional neural networks (CNNs) and recurrent neural networks (RNNs) model, referred to as CNN-RNN model to forecast the streamflow discharge values using the historical streamflow discharge values for potomac river basin. The experimental results demonstrated that the designed network successfully predicts the streamflow discharge values for the next month. In addition, a new sequence based on one-step ahead was built to forecast streamflow discharge values beyond the horizon of the testing data. Moreover, The CNN-RNN model demonstrates a lower symmetric mean absolute percentage error (sMAPE) and a higher correlation between the forecasted values and the testing data than the RNN model, indicating higher prediction accuracy.

V. Future Work

We will implement transfer learning utilizing ResNet50, Xception, and a feedforward neural network to forecast streamflow discharge beyond the horizon of the testing data. Furthermore, we will fine-tune the parameters of the model

based on its performance on the testing set in order to further enhances the accuracy of the forecasts.

ACKNOWLEDGMENT

This work is supported by the National Science Foundation (NSF) grants #2401880 and #2011927, and USGS DCWRRI grant #2023DC104B.

REFERENCES

[1] Water Supply Outlook and Status, Interstate Commission on the Potomac River Basin, https://www.potomacriver.org.

[2] N. Zhang, X. Dai, M. A. Ehsan, and T. Deksissa, "Development of a drought prediction system based on long short-term memory networks (LSTM)," *Advances in Neural Networks - 17th International Symposium on Neural Networks (ISNN 2020)*, Cairo, Egypt, December 4–6, 2020, doi: 10.1007/978-3-030-64221-1_13.

[3] G. Singh et al., "CNN-RNN based hybrid machine learning model to predict the currency exchange rate: USD to INR, " *2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering* (ICACITE), Greater Noida, India, 2022, pp. 1668-1672, doi: 10.1109/ICACITE53722.2022.9823844.

[4] Z. Tu, and Z. Wu, "Predicting Beijing air quality using bayesian optimized CNN-RNN hybrid model," *2022 Asia Conference on Algorithms, Computing and Machine Learning* (CACML), Hangzhou, China, 2022, pp. 581-587, doi: 10.1109/CACML55074.2022.00104.

[5] X. Zhou, Y. Li and W. Liang, "CNN-RNN based intelligent recommendation for online medical pre-diagnosis support," *in IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 18, no. 3, pp. 912-921, 1 May-June 2021, doi: 10.1109/TCBB.2020.2994780.

[6] J. F. R. Rochac, N. Zhang, T. Deksissa, J. Xu, and L. Thompson, "A hybrid ConvLSTM deep neural network for noise reduction and data augmentation for prediction of non-linear ynamics of streamflow," *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*, Orlando, FL, USA, 2022, pp. 1120-1127, doi: 10.1109/ICDMW58026.2022.00146.

[7] J. F. R. Rochac, N. Zhang, T. Deksissa, and W. H. Mahmoud, "Streamflow prediction using a hybrid methodology based on convolutional neural network and long short-term memory," *The 8th IEEE International Conference on Big Data Computing Service and Machine Learning Applications (BigDataService)*, San Francisco Bay Area, CA, August 15-18, 2022, doi: 10.1109/BigDataService55688.2022.00035.

[8] Y. Shi, C. Yang, J. Wang, Z. Zhang, F. Meng, and H. Bai, "A forecasting model of ionospheric foF2 using the LSTM network based on ICEEMDAN decomposition," *in IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1-16, 2023, Art no. 4108416, doi: 10.1109/TGRS.2023.3336934.

[9] S. Hochreiter, and J. Schmidhuber. "Long short-term memory," *Neural computation*. vol. 9, no. 8, pp. 1735-1780, 1997.