IOSR Journal of Mechanical and Civil Engineering (IOSR-JMCE)
e-ISSN: 2278-1684, p-ISSN: 2320-334X, Volume 22, Issue 3 Ser. 2 (May. – June. 2025), PP 34-43
www.iosrjournals.org

Evaluating ANN Algorithms And LSTM Deep Learning For River Flow Prediction

Chandra Upadhyaya, Dr. Arnab Sarma
Department Of Civil Engineering, Royal School Of Engineering & Technology / The Assam Royal Global University, Guwahati-781034, India

Abstract:
Monthly flow data for the Noa-Dehing River, covering a period of 19 years, are available; however, they are inadequate for optimising the proposed 71 MW hydropower project. To address this limitation, additional flow data were generated using an Artificial Neural Network (ANN) and deep learning model. Various algorithms have been employed to develop ANN models. Among these, the Conjugate Gradient (CG) algorithm yields the most favourable results. In contrast, the commonly used Gradient Descent with Momentum Backpropagation (GDMB) algorithm proved to be less effective, owing to its requirement for a longer duration and a greater number of iterations during the network's training phase. Although the Levenberg-Marquardt (LM) algorithm demonstrated high training efficiency, it failed to accurately predict high flows. Similarly, the Gradient Descent with Momentum and Adaptive Learning algorithms did not effectively predict high flows despite achieving the highest R value among the ANN algorithms. Consequently, a novel approach, the long short-term memory (LSTM) deep learning model, was employed to forecast the flows. The forecast was evaluated using the root-mean-square error (RMSE), mean absolute error (MAE), and correlation coefficient (R). The LSTM model yielded RMSE, MAE, and R-values of 55, 42, and 0.90, respectively. These assessment values were superior to those obtained using the ANN algorithms.

Key Word*: Artificial Neural Network, Deep Learning, LSTM, River Flow Prediction, Hydropower Project, Streamflow Prediction.*

Date of Submission: 11-05-2025 Date of Acceptance: 21-05-2025

I. Introduction

Approximately 70 years ago, there was a surge in interest in artificial neural networks driven by the aspiration to comprehend and replicate the functions of the human brain. In the past three decades, this field has experienced a significant revival owing to advances in algorithms and the availability of powerful computing resources. Extensive research has explored the potential of artificial neural networks (ANNs) as computational tools for deriving, establishing, and evaluating maps from multidimensional input spaces to other spaces. This potential has been made feasible by advancements in estimation methods with self-organising properties and parallel information systems [1] [2] [3]. These developments have led to widespread utilisation of ANNs over the last 30 years. Additionally, there was an introduction of a new algorithm called backpropagation, designed for networks consisting of neuron-like units which sparked a considerable increase in interest in this computational approach. The weights of the connections are adjusted through iterative minimisation of the difference between the output and target vectors in the neural networks [4] [5] [6]. These developments have contributed to the extensive adoption of ANNs across various research fields through the implementation of the backpropagation rule within parallel-distributed information-processing frameworks. Consequently, ANNs have found applications in a wide range of areas, such as finance, cybernetics, neuroscience, physics, biochemistry, mechanical engineering, computer technology, medicine, robotics, and electronics. Additionally, during the late 1980s, they were widely utilised to comprehend dynamic and nonlinear relationships in machine learning [7] [8].

The architecture of an ANNs is influenced by the working of the human brain [9]. Although they do not match the complexity of the brain, there are two main parallels between biological neural networks and ANNs. First, both consist of basic, interconnected computing components. Second, the role of the network is determined by the relationships between neurones [10] [11] [12]. ANNs operate as parallel-distributed processing networks and share basic properties with biological neural systems [13] [14]. Neurones receive a host of signals. Each input is assigned a weighting that affects the result of the input. This is similar to the various synaptic capacities of the neurones in the human brain. A few inputs are much more appropriate than others in the way they produce an output. Weights are functional parameters within the system that determine the strength of an input parameter. The resultant signal of the neuron is generated by the addition block, which approximately corresponds to the biological cell body and algebraically sums all inputs after multiplying by the weights [15] [16]. Since the early

1990s, ANNs have been used in various hydrology-related applications, including streamflow prediction, rainfall-runoff modelling, water quality analysis, water management strategies, precipitation forecasting, and groundwater modelling [17] [18] [19] [20].

Streamflow prediction is essential for various timeframes, such as short-term (hours) flood management and longer terms (days, months) for the efficient functioning of dams in hydropower generation and irrigation. Reliable flow forecasts enable project managers to effectively allocate water resources according to different user needs, including agriculture, energy production, domestic use, and ecological requirements [21] [22] [23] [24] [25] [26]. Streamflow forecasting becomes particularly important when multiple users rely on the same reservoir, particularly for flood control purposes. Additionally, flow forecasting includes information on sediment transport volume [27].

In recent years, artificial neural networks (ANNs) have been increasingly utilised in the modelling of water resource systems, presenting an effective alternative to traditional methods [13]. Rainfall-runoff ANN models have also been widely used. Researchers have examined the effectiveness of this modelling approach for predicting streamflow over a 1-week period. Many researchers have employed backpropagation algorithms to estimate river flow on an hourly basis. Some researchers have constructed ANN models for forecasting monthly average flow rates, whereas others have used various ANN algorithms capable of predicting both shorter- and longer-duration flows [28] [29]. Some researchers have assessed the accuracy of ANNs in groundwater prediction and utilised Levenberg-Marquardt and back-propagation algorithms to train their ANN models [30].

Artificial Neural Network (ANN) models have been employed using a widely utilised back-propagation (BP) algorithm for daily forecasting. The American Society of Civil Engineers (ASCE) Task Committee highlighted the principles and applications of ANNs in hydrology and concluded that ANNs could serve as alternative modelling methods, meriting further investigation. It has been noted that the back-propagation algorithm has been applied in approximately 90% of hydrology cases. [31].

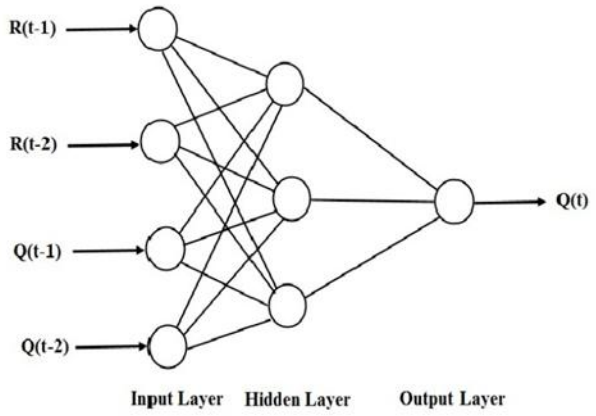


Fig.1. Typical Artificial Neural Network Configuration

Nacar et al. examined three different algorithms for predicting streamflow several days in advance and concluded that the Conjugate Gradient algorithm exhibited superior prediction capabilities compared to the backpropagation and Levenberg-Marquardt algorithms [32]. Mutlu et al. assessed the use of ANN models for daily flow predictions at various measurement sites in the Eucha Watershed using a multilayer perceptron and radial neural network, with both models demonstrating satisfactory performance. The MLP model performed better than the RBFNN model did [33]. Hu et al. employed Long-Short Term Memory deep learning model for forecasting time-series data and compared its prediction accuracy with Support Vector Regression and Multilayer Perceptions, concluding that LSTM showed better performance [34].

In Figure 1, we can see the standard three-layer feed-forward arrangement of the Artificial Neural Network. The network comprises three layers: the initial input layer, hidden layer, and output layer. Each layer contains multiple neurones chosen based on the nature and complexity of the problem to be addressed. Neurones serve as fundamental computational units of a neural network. They incorporated weights, bias, summation points, and output transfer functions. Neurones within the same layer are not interconnected, but are connected to the previous or corresponding layers, as depicted in the diagram. Each connection is assigned specific weights which interact with inputs from the preceding layers through multiplication, resulting in aggregated products that ultimately produce solutions after undergoing an output transfer mechanism. The Sigmoid function is one of the most commonly employed transfer functions because of its S-shaped monotonic behaviour, which maps numbers onto a finite interval (-1, +1) within the range (-∞, +∞). The yn output from neuron n is determined by Eq. 1.

$$y_n = f(\sum w_{nm} x_m + \theta_n) = \frac{1}{1+e^{-(\sum w_{nm} x_m + \theta_n)}} \quad (1)$$

Where θ_n= bias in the n^{th} neuron in a layer; w_{nm}= weight of the connection joining the n^{th} neuron in a layer with the m^{th} neuron in the previous layer; and x_m= value of the m^{th} neuron in the previous layer.

Initially, artificial neural networks were trained using input and target data and then tested with a portion of the data not used for training. It is important to note that, as ANN are utilised for predicting future outcomes, it is crucial to ensure that the network is capable of generalisation. Sufficient data play an essential role in enhancing network performance during training. The number of neurones in the hidden layer is another key aspect of the ANN. Insufficient neurones may lead to incomplete learning of the underlying physical mechanisms through the network, whereas an excessive number could result in overfitting and hinder its capability for accurate forecasting [6].

II. Material And Methods

Training algorithms used in neural networks

Although the back-propagation algorithm is commonly utilised in hydrology, it may not yield optimal results and often exhibits slow performance compared to other algorithms. For comparison, both the original and modified versions were included. The training algorithms employed include backpropagation with momentum, adaptive learning, conjugate gradient, and Levenberg-Marquardt algorithm. Each of these algorithms has its own strengths and weaknesses, leading to their use in the search for the most effective algorithm in this study.

The aim of training is to minimise the mean square error (MSE), defined by Eq. 2:

$$Mean\ Square\ Error\ (E) = \frac{1}{k}\sum_{m=1}^{k}(Target_m - Output_m)^2 \quad (2)$$

The letter k denotes the total count of the output nodes. The next section introduces the training algorithms. These algorithms are designed to minimise the mean square error in each iteration by adjusting the weights and biases until they satisfy the output target. Upon completion of this process, the model can predict future values.

Backpropagation with momentum

Backpropagation with momentum is commonly employed for training neural networks. This technique uses gradient descent to minimise the error function, which is typically represented by the mean squared error [35]. The input vector of the designated dataset, organised as training data, undergoes forward propagation through the network. Subsequently, after processing the input vector, the ANN yielded the results. The mean square error was then computed based on these results and compared with the target values from the dataset. Following this computation, the resulting error is propagated backwards through the network, and adjustments are made to the weights connecting the neurones using Eq. 3.

$$\Delta w_{mn}(p) = -\varepsilon * \frac{\partial E}{\partial w_{mn}} + \propto * \Delta w_{mn}(p-1) \quad (3)$$

Where $\Delta w_{mn}(p)$ and $\Delta w_{mn}(p-1)$ are difference in weights between m and n during the p^{th} and $(p-1)^{th}$ iterations. $\propto$ and ε are momentum and learning rate, respectively. Similarly, for a bias, the relationship can also be formulated. Gradient descent with the inclusion of momentum efficiently captures both the local gradient and overall pattern of error. The momentum acts as a low-pass filter, allowing the network to disregard insignificant error values, which in turn helps prevent getting stuck at shallow minima. The training process was outlined as follows:

i. We assume small weight values (both positive and negative). Training of the network does not begin if all weights are equal.
ii. A training pair is selected from the training dataset.
iii. Apply the input vector to the network.
iv. The output of the network is evaluated.
v. The error is determined by calculating the difference between the output and the target.
vi. The network weights were adjusted to reduce this error.
vii. The process outlined in steps 2-6 until the error is satisfactorily minimised across the entire network.

Gradient descent with momentum and adaptive learning rate backpropagation

In gradient descent with momentum and backpropagation of the adaptive learning rate, the weight and bias values are modified based on the gradient descent momentum and adaptive learning rate [36] [37]. The learning rate remained consistent throughout the training process when using the conventional steepest descent method. The training outcome was linked to finding an appropriate value for the learning rate. A high learning rate is not optimal because it may prevent the network from converging to the desired output. Conversely, a low learning rate is also suboptimal because it requires a significantly higher number of iterations for the algorithm to yield results. Unfortunately, determining the appropriate learning rate in advance is challenging. Moreover, the ideal learning rate fluctuated during the various stages of training. The performance of the gradient descent

algorithm is significantly improved by employing a mechanism that allows the adjustment of the learning rate during training. An adaptive learning rate algorithm accomplishes this, enabling it to effectively address challenging scenarios such as local errors. Some enhancements are required in the training methodology employed for gradient descent and backpropagation to incorporate the adaptive learning rates. Initially, preliminary outputs and errors were computed, followed by the adjustment of weights and biases at each epoch using the current learning rate. Subsequently, the new outputs and errors were re-evaluated. If the current error exceeds approximately 1.05 of the previously calculated error, the derived weights and biases are discarded; however, if this condition is not met, they are deemed acceptable. Adjustments to the learning rate occur when the newly calculated error is roughly 1.06 times less than the previous one, leading to an appropriate learning rate.

Levenberg-Marquardt algorithm

The Levenberg-Marquardt algorithm is as follows [38]:

i. Provide all network input values and calculate the corresponding network errors from the output (e = t − a). We then determine the sum of the squares of all input errors (V(x)).

ii. Evaluate the Jacobian matrix J(x).

iii. To determine the value of Δx, we must solve Eq. 4:

$$\Delta x = [J^T(x)J(x) + sI]^{-1}J^T(x)e(x) \quad (4)$$

When a step increases by V(x), s is multiplied by factor c. When V(x) decreases in steps, s is divided by c. The algorithm becomes the steepest descent when s is large and Gauss-Newton for small s. In the trust region, the Levenberg-Marquardt algorithm can be viewed as an extension of Gauss-Newton.

iv. Using $x + \Delta x$, we calculate the errors, square, and sum them again. If the current result is smaller than that calculated earlier in the 1st step, then decreases by c and moves backward to the first step. If the value of the squares does not decrease, then increases s by c and returns to the third step.

v. If the gradient is lower than a certain specified amount or if the (V(x)) is decreased to less than a target, the algorithm is believed to have converged.

Conjugate gradient algorithm

The conjugate gradient algorithm does not require the computation of 2nd derivatives, but converges in a manner similar to quadratic convergence. In this approach, the descent direction is not aligned with the error gradient but rather perpendicular to the previous step [39]. The Conjugate Gradient algorithm operates as follows:

i. First, the weight vectors x_j are first initialized. Error gradient g_0 is computed. Select the first search direction q_0 to be the negative of the gradient as $q_0 = -g_0$, where $g_j \equiv \Delta F(x)|_{x=x_j}$

ii. In each iteration *j*, select the learning rate α_j to minimize the function along the search direction by taking the step as $x_{j+1} = x_j + \alpha_j q_j$

iii. Choose the direction of the next search as per, $q_{j+1} = -g_{j+1} + \beta_j q_j$,

Where $\beta_j = \frac{g_j^T g_j}{g_{j-1}^T g_{j-1}}$

iv. Step 2 is repeated if the algorithm does not converge.

Long short-term memory (LSTM) network

The issue of exploding and vanishing gradients in artificial neural networks was addressed by introducing the LSTM network. This neural network is based on deep learning principles. The network incorporates cells and gates with memory to manage the retention or removal of long-term data stored in the system [40].

Experiment using LSTM: Evaluation of the LSTM's performance involves data preparation, model training, selection of performance criteria, and presenting final results and comparing them to the results obtained from the ANN.

Preparation of LSTM data: Training data were standardised to improve the fitting and prevent divergence during training. The standardised data had an average of 0 and a standard deviation of 1. The validation data were similarly standardised using the parameters applied to the training set. The employed formula is shown in Eq. 5 and Eq. 6.

$$\text{New Training data} = \frac{\text{Training data} - \mu}{\sigma} \quad (5)$$

$$\text{New validation data} = \frac{\text{Validation data} - \mu}{\sigma} \quad (6)$$

Where, μ = Mean of all the data; σ = Standard deviation of all the data.

Training of LSTM model

The various parameters utilised in the LSTM model are as follows.

Using a trial-and-error approach, 64 hidden nodes were determined to be required for optimal performance. The most commonly used optimisers include stochastic gradient descent with momentum (sgdm), Adaptive Moment Estimation (Adam), and root-mean-square propagation (rmsprop). From the experiment, the Adam optimiser demonstrated superior performance. To calculate the gradient of the loss function and update the weights, a batch method was employed, with an optimum batch size of 72. An epoch constitutes a complete pass through the entire dataset; after experimentation, it was found that utilising 250 epochs resulted in the lowest value for the loss function. A gradient threshold of 1 was used to prevent exploding gradients during the training. In this study, an initial learning rate of 0.006 was established through experimentation to balance the training time constraints while avoiding becoming stuck at suboptimal points. Additionally, a learning rate drop factor of 125 was applied to adjust global learning rates every specified period of epochs; further adjustments were made using a multiplicative factor of 0.2 after a certain number of epochs had passed.

Evaluation of LSTM results

The root means square error (RSME), mean absolute error (MAE), and correlation coefficient (R) were employed to assess the outcomes of the aforementioned experiment. The RMSE is calculated by subtracting the observed values from the predicted values and is given by Eq. 7:

$$\text{RSME} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y_t})^2} \quad (7)$$

Where, y_t is the predicted value, $\hat{y_t}$ is the observed value and n is the number of observations. The RSME result was consistently positive, with a reduced RSME value indicating an enhanced prediction. The MAE can be expressed using Eq. 8:

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|(y_t - \hat{y_t})| \quad (8)$$

A smaller MAE value indicates a more accurate prediction, whereas the RSME value decreases with improved predictions. R represents the correlation coefficient, which can be calculated using Eq. 9:

$$R = \sqrt{1 - \frac{\sum_{t=1}^{n}(y_t - \hat{y_t})^2}{\sum_{t=1}^{n}(y_t - \bar{y})^2}} \quad (9)$$

The average of the estimated value is represented by $\bar{y}$. The range for R falls between 0 and 1, with a higher R signifying a more accurate prediction result.

III. Result

Fig. 2. Location of the study area (Source: Google Map)

Table 1: Hydrological Statistics of Monthly Data

Flow parameters						
Period	Average (m^3/s)	Standard deviation (m^3/s)	Skewness coefficient	Coefficient of variation	Minimum flow (m^3/s)	Maximum flow (m^3/s)
Training	186.64	121.15	0.77	0.65	31.83	566.77
Validation	152.41	120.31	1.30	0.79	33.00	549.50

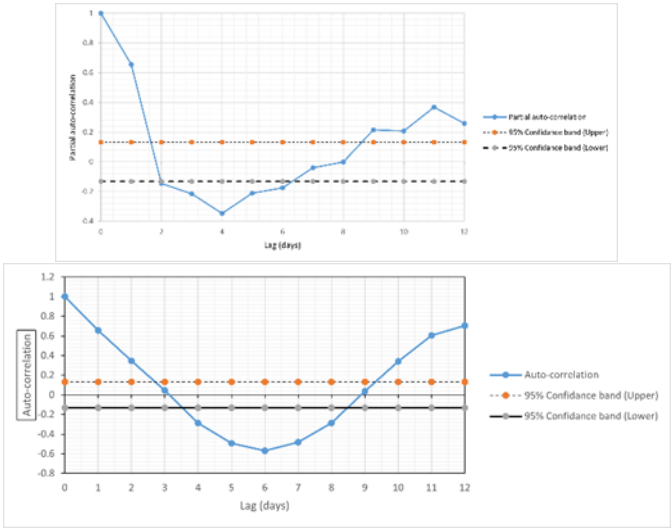


Fig.3. Autocorrelation plots of the time series

Monthly river flow data from 1987 to 2005 were available alongside rainfall data in a 10-day format, which was converted to monthly data (Ref. Fig.2 for location of the study area). A total of 228 months of data were obtained: 160 were used for training and 68 for validation. The hydrological statistics from the observed data during the training and testing periods are presented in Table 1. The analysis revealed that the mean and skewness of the test data differed from those of the training data. However, both sets exhibit similar extreme minimum and maximum flow values, facilitating the forecasting accuracy for low and high flow levels. The time series underwent correlation analysis to assess the impact of previous flows. This analytical method is typically applied to investigate the influence of earlier flows on the current flow levels. An assessment was conducted on auto- and partial autocorrelation values using streamflow data with 95% confidence bands calculated up to lag 12 (Figure 3). The oscillating pattern observed in the partial autocorrelation function indicates correlations at lags 1, 3, 4, 5, 9, 10, 11, and 12, as shown in Figure 3. As a result of this observation, it can be concluded that it is important to consider 12 preceding flow values when constructing input vectors for ANNs. Moreover, the maximum value of partial autocorrelation was only 0.65 which may not be significant. This required the identification of a connection between the monthly rainfall in the basin and streamflow (Ref. Fig. 4). A correlation of 0.84 was determined from a scatter plot of monthly rainfall and streamflow data. These supports utilise monthly rainfall information as part of the input, along with 12 previous flow values.

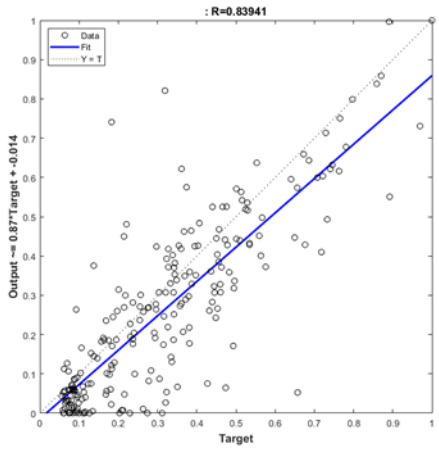


Fig.4. Scatter plot of rainfall and runoff.

Before applying the ANN techniques, the rainfall and flow data were normalised using the following method: the maximum value in the flow vector was identified and used to divide all elements of the vector, resulting in normalised flow values. These normalised values were then utilised for both the training and testing of the ANN networks with various algorithms, as outlined below.

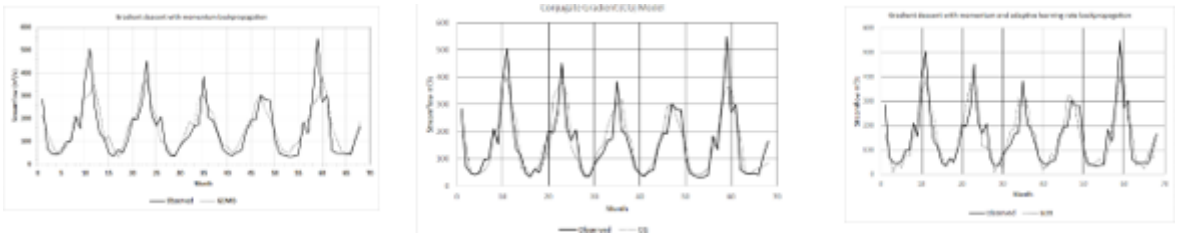
The following configurations of discharge and rainfall data were tried for the assessment: (1) Q_{t-1}, R_{t-1}; (2) Q_{t-1}, Q_{t-2}, R_{t-1}, R_{t-2}; … (12) Q_{t-1}, Q_{t-2}, Q_{t-3} … Q_{t-12}, Qt_{-1}, Q_{t-2}, R_{t-1}, R_{t-2}, R_{t-3}… R_{t-12}, *where Qt is* the flow value at time t. Similarly, R_t represents rainfall. Discharge Q_t is found out from the output neuron by using the

above combination. The network was trained using four distinct ANN algorithms and an LSTM deep learning model. These ANN algorithms include backpropagation with momentum, adaptive learning, conjugate gradient, and Levenberg–Marquardt algorithms.

Table 2. MSE values for different algorithms for 68 months forecasting

Model inputs	Mean Square Error (MSE)			
	LM	CG	GDMB	GDMAL
$Q_{t-1}, Q_{t-2}, \dots, Q_{t-12}$ $R_{t-1}, R_{t-2}, \dots, R_{t-12}$	4566	3673	10906	12405
$Q_{t-1}, Q_{t-2}, \dots, Q_{t-12}$	3606	3574	4197	3784

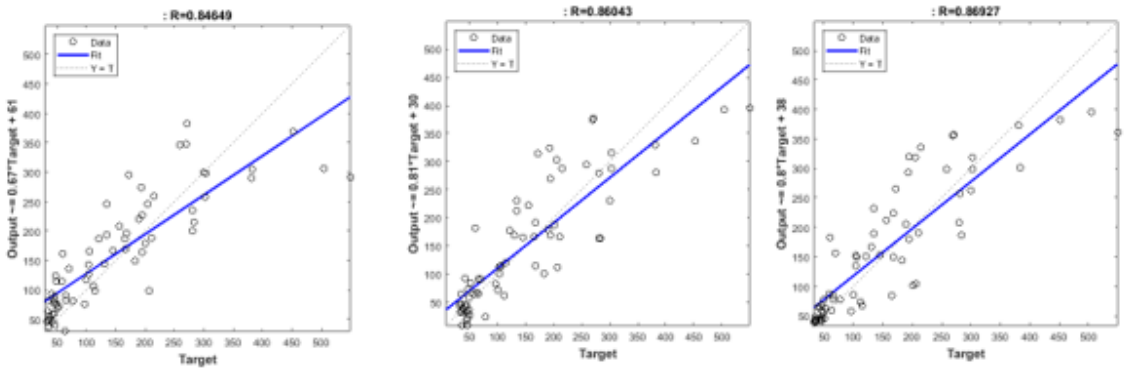
Upon completion of the training, the performance of all the networks was evaluated using the forecasted data generated by various algorithms. The results from the artificial neural network models, initially in normalised form, were then de-normalised, and mean squared errors were calculated for each model's forecasted data. Table 2 displays the MSE results for the predicted flow values, indicating that the Conjugate Gradient yielded the lowest MSE, followed by the Levenberg-Marquardt algorithm. Notably, the GDMB and GDMAL models exhibited the highest MSE with a correspondingly high R-value over a 68-month forecast period. Nine hidden layers were used for the analysis. An additional noteworthy observation is that networks incorporating an input vector consisting of 12 previous flows without rainfall input exhibited lower MSE despite findings from the rainfall-flow correlation analysis, suggesting otherwise. Fig. 5 and 6 display the results of forecasting 68 months ahead using the GDMB, GDMAL, and CG algorithms. Analysis of the fit line equations in the diagrams indicates that the forecasts from the Conjugate Gradient approach closely align with the observed flows compared with those from GDMB and GDMAL. The slope and intercept coefficients for the Conjugate Gradient algorithm were approximately equal to 1 and 0, respectively, in contrast to those for GDMB and GDMAL.



(a) GDMB **(b) CG** **(c) GDMAL**

Fig.5. Graph of 68-months-flow prediction for the testing period

Figure 5 (b) serves as a visual representation to demonstrate the similarity between the observed and forecasted values for a 68-month forecasting period, using the Conjugate Gradient algorithm. This is further depicted in the scatter plot shown in Figure 6 (b), highlighting the close correspondence between the observed values and the neural network predictions produced by the Conjugate Gradient algorithm. The accuracy of predicting both lower and higher flow values is evident, although there are instances where under-prediction arises in the 18th and 50th months, which is potentially attributable to limitations inherent in ANN algorithms. Additionally, upon examining the scatter diagrams, it becomes apparent that forecasts exhibit an approximately linear relationship with observations, affirming the suitability of employing linear correlation coefficients for these comparisons.



(a) GDMB **(b) CG** **(c) GDMAL**

Fig.6. Scatter plot for the validation period (68-months-ahead forecast)

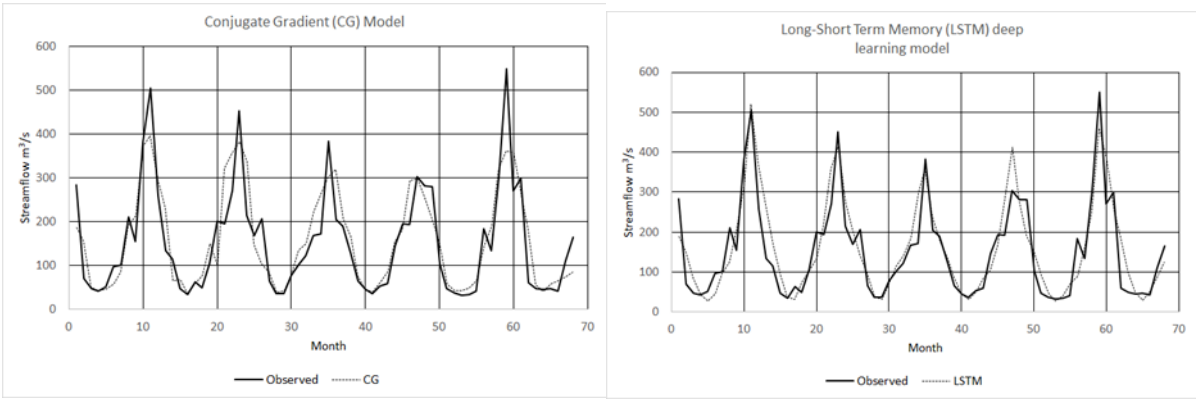
The Adaptive Learning algorithm achieves the highest R-value of 0.86 for forecasting 68 months ahead. Figure 5 (c) illustrates both the observed and forecasted plots generated by the Adaptive Learning algorithm for this time period, whereas Figure 6 (c) presents a scatter plot of the same data. There is a noticeable disparity between the observed values and the corresponding predictions made by the neural network through the Adaptive Learning algorithm, particularly in terms of accurately predicting higher flow values. However, lower flow values are generally predicted reasonably well, with just one exception noted during the analysis of the scatter plot, as shown in Fig. 6(c), which shows that predictions tend to be on the higher side for low-value targets.

Four different Artificial Neural Network algorithms were evaluated for monthly streamflow forecasting using flow data from the Noa-Dehing River and rainfall data from the Noa-Dehing Basin. The results indicate that the Conjugate Gradient (CG) algorithm outperforms the other algorithms in providing accurate monthly flow forecasts. Additionally, the Levenberg–Marquardt algorithm demonstrated satisfactory performance, particularly in predicting low flow values compared to high flow values. Conversely, the commonly used Gradient Descent with Momentum Backpropagation algorithm exhibited slow processing and yielded less promising results among the four algorithms, owing to its first-order gradient. It was observed that second-order gradients contributed to a faster execution, as evidenced by the speed of the Levenberg–Marquardt and Conjugate Gradient algorithms. Furthermore, incorporating a correlation analysis before finalising the input vectors proved effective in determining the input vector composition and hidden layer count. These findings apply specifically to the Noa-Dehing Basin, and further research is essential to validate these conclusions across other Himalayan River basins.

In the next phase, the LSTM model was run, and errors were computed. The best result from the ANN using the Conjugate Gradient Algorithm was used for comparison with the LSTM errors, as shown in Table 3.

Table 3. Comparison of errors

Criteria	ANN (Conjugate Gradient Algorithm)	LSTM
RMSE	60	55
MAE	43	42
R	0.86	0.90



(a) ANN (CG) **(b) LSTM**

Fig.7. Prediction results of ANN (CG) and LSTM

The prediction results are shown in Figure 7. Table 3 indicates that LSTM outperformed the best ANN model across all three evaluation criteria, demonstrating superior predictive accuracy. The comparison reveals that LSTM outperforms statistically in terms of prediction performance. Additionally, Figure 7 illustrates the close similarity between the plots of the LSTM predictions and the stream-flow data. Significant differences were noticeable in the bulging and prediction of major peaks when comparing both models; these aspects were more accurately represented by the LSTM model than by the ANN model.

LSTM's ability to retain and discard information is a crucial factor in modelling nonlinear time-series data for prediction purposes. This feature guarantees relatively improved outcomes when predicting river flow based on past data. However, it has been noted that LSTM failed to accurately predict peaks in two instances. This could be attributed to the limited size of the training data and potential unavailability of high-quality data.

IV. Conclusion

Streamflow prediction is essential for the efficient management of water resource projects. With accurate forecasts, project managers can allocate water supply to various needs, such as agriculture, energy production, household consumption, and ecological requirements, while considering trade-offs. Streamflow predictions are particularly crucial for reservoirs with diverse uses, particularly for flood control. Forecasting streamflow is a difficult task because of the complex physical processes guided by multiple unknown factors. The current study

employs an ANN and a deep learning LSTM network with a remember-forget mechanism to regulate the vanishing and amplification of gradients for streamflow prediction. The study revealed that, compared to conventional ANN networks, LSTM proved to be more effective in modelling time series with nonlinear characteristics. The experimental results demonstrate that LSTM predictions outperform those of other evaluation criteria. The following aspects are significant regarding the LSTM model for enhancing streamflow prediction.

i. Some of the peaks were not precisely predicted. The predictions were inaccurate.

ii. The default settings for LSTM models are often adjusted on an ad hoc basis with some parameters remaining unchanged. It is important to conduct more systematic experiments to investigate the effects of these parameters thoroughly.

iii. The accuracy of the flow prediction diminishes when rainfall data are incorporated, which is not typically expected. This raises doubts about the quality of rainfall data.

iv. The available data cover a period of 19 years, on a monthly basis. The amount of data available may not be adequate for capturing fundamental physical phenomena.

v. There is a chance that data have been altered at various points, potentially impacting the precision of the forecast.

Using unbiased data to run the model may lead to enhanced forecast accuracy.

References

[1] D. Mantzaris, S. Trougkakos, K. Vadikolias, E. Priska, And M. Vrizas, "Artificial Neural Networks For Estimation Of Dementias Types," Artificial Intelligence And Applications, Vol. 2014, No. 1, Pp. 74–82, May 2014, Doi: 10.15764/Aia.2014.01006.

[2] R. Qamar And B. Ali Zardari, "Artificial Neural Networks: An Overview," Mesopotamian Journal Of Computer Science, Pp. 130–139, Aug. 2023, Doi: 10.58496/Mjcsc/2023/015.

[3] M. M Mijwil, "Artificial Neural Networks Advantages And Disadvantages," Mesopotamian Journal Of Big Data, Vol. 2021, Pp. 29–31, Aug. 2021, Doi: 10.58496/Mjbd/2021/006.

[4] T. Syed, H. Kim, V. Kakani, And X. Cui, "Spiking Neural Networks Using Backpropagation," Aug. 2021. Doi: 10.1109/Tensymp52854.2021.9550994.

[5] M. Dampfhoffer, T. Mesquida, A. Valentian, And L. Anghel, "Backpropagation-Based Learning Techniques For Deep Spiking Neural Networks: A Survey.," Ieee Transactions On Neural Networks And Learning Systems, Vol. 35, No. 9, Pp. 11906–11921, Sep. 2024, Doi: 10.1109/Tnnls.2023.3263008.

[6] T. Zhang, S. Jia, M.-M. Poo, X. Cheng, Y. Zeng, And B. Xu, "Self-Backpropagation Of Synaptic Modifications Elevates The Efficiency Of Spiking And Artificial Neural Networks.," Science Advances, Vol. 7, No. 43, Oct. 2021, Doi: 10.1126/Sciadv.Abh0146.

[7] P. H. Panicker, A. A. Deshmukh, And S. H. Karamchandani, "Applications Of Ai Techniques Like Machine Learning Methods And Deep Learning Models (Anns) In Emerging Areas: A Review," Springer Nature Singapore, 2022, Pp. 303–312. Doi: 10.1007/978-981-16-6601-8_28.

[8] S. Chen, "Bayes Theory In Application To Medicine, Machine Learning, And Finance," Science And Technology Of Engineering, Chemistry And Environmental Protection, Vol. 1, No. 10, Dec. 2024, Doi: 10.61173/Hxpe5t77.

[9] V. R. Karolis, M. Thiebaut De Schotten, And M. Corbetta, "The Architecture Of Functional Lateralisation And Its Relationship To Callosal Connectivity In The Human Brain," Nature Communications, Vol. 10, No. 1, Mar. 2019, Doi: 10.1038/S41467-019-09344-1.

[10] K. Katyal, J. Parent, And B. Alicea, "Connectionism, Complexity, And Living Systems: A Comparison Of Artificial And Biological Neural Networks." Cornell University, Mar. 14, 2021. Doi: 10.48550/Arxiv.2103.15553.

[11] J. Schmidt-Hieber, "Interpreting Learning In Biological Neural Networks As Zero-Order Optimization Method." Cornell University, Jan. 27, 2023. Doi: 10.48550/Arxiv.2301.11777.

[12] F. Sarfraz, E. Arani, And B. Zonooz, "A Study Of Biologically Plausible Neural Network: The Role And Interactions Of Brain-Inspired Mechanisms In Continual Learning." Cornell University, Apr. 13, 2023. Doi: 10.48550/Arxiv.2304.06738.

[13] M. Yucel, S. M. Nigdeli, And G. Bekdaş, "Artificial Neural Networks (Anns) And Solution Of Civil Engineering Problems," Igi Global, 2019, Pp. 13–38. Doi: 10.4018/978-1-7998-0301-0.Ch002.

[14] G. M. Khan, "Artificial Neural Network (Anns)," Springer, 2017, Pp. 39–55. Doi: 10.1007/978-3-319-67466-7_4.

[15] K. A. Aguilar Cruz, J. D. J. Medel Juárez, And M. T. Zagaceta Álvarez, "Adaptation Of Weights In A Neuron Using An Integrated Filter," Research In Computing Science, Vol. 100, No. 1, Pp. 139–147, Dec. 2015, Doi: 10.13053/Rcs-100-1-12.

[16] Z. Cai And Q. Shen, "Encoding Semantic Priors Into The Weights Of Implicit Neural Representation." Jun. 06, 2024. Doi: 10.48550/Arxiv.2406.04178.

[17] R. Remesan And J. Mathew, "Data Based Rainfall-Runoff Modelling," Springer, 2014, Pp. 151–182. Doi: 10.1007/978-3-319-09235-5_6.

[18] D. Kim, J. A. Chun, And I. Jung, "Comparative Evaluation Of Rainfall-Runoff Modelling Against Flow Duration Curves In Semi-Humid Catchments." Copernicus Gmbh, Apr. 10, 2017. Doi: 10.5194/Hess-2017-138.

[19] M. J. Alexopoulos, M. Šraj, P. Nistahl, N. Bezak, And H. Müller-Thomy, "Validation Of Precipitation Reanalysis Products For Rainfall-Runoff Modelling In Slovenia." Copernicus Gmbh, Dec. 06, 2022. Doi: 10.5194/Egusphere-2022-1279.

[20] N. N. Che Razali, N. S. Widodo, S. I. Hisham, S. Kasim, T. Sutikno, And N. A. Ghani, "Rainfall-Runoff Modelling Using Adaptive Neuro-Fuzzy Inference System," Indonesian Journal Of Electrical Engineering And Computer Science, Vol. 17, No. 2, P. 1117, Feb. 2020, Doi: 10.11591/Ijeecs. V17.I2. Pp1117-1126.

[21] R. Bakış, Y. Bayazıt, D. M. Ahmady, And C. Koç, "Investigation Of Hydropower Generation From Irrigation Dams," Anadolu University Journal Of Science And Technology-A Applied Sciences And Engineering, Vol. 19, No. 2, Pp. 1–5, Jun. 2018, Doi: 10.18038/Aubtda.368219.

[22] R. F. Digna, Y. A. Mohamed, W. Van Der Krogt, S. Uhlenbrook, P. Van Der Zaag, And G. Corzo, "Impact Of Water Resources Development On Water Availability For Hydropower Production And Irrigated Agriculture Of The Eastern Nile Basin," Journal Of Water Resources Planning And Management, Vol. 144, No. 5, Feb. 2018, Doi: 10.1061/(Asce)Wr.1943-5452.0000912.

[23] Z. Chitu Et Al., "Improving Irrigation Scheduling Using Moses Short-Term Irrigation Forecasts And In Situ Water Resources Measurements On Alluvial Soils Of Lower Danube Floodplain, Romania," Water, Vol. 12, No. 2, P. 520, Feb. 2020,

Doi: 10.3390/W12020520.
[24] M. Sarireh, "Management Of Water Resources Infrastructure In Jordan: Dams And Reservoirs," Journal Of Information Systems Engineering And Management, Vol. 10, No. 34s, Pp. 439–450, Apr. 2025, Doi: 10.52783/Jisem.V10i34s.5819.
[25] R. Ragab, O. M. Amer, And R. El-Sherbiny, "The Global Challenges Facing The Water Resources And Irrigation Sector," E3s Web Of Conferences, Vol. 368, P. 03002, Jan. 2023, Doi: 10.1051/E3sconf/202336803002.
[26] A. Elmahdi, "Addressing Water Scarcity In Agricultural Irrigation: By Exploring Alternative Water Resources For Sustainable Irrigated Agriculture," Irrigation And Drainage, Vol. 73, No. 5, Pp. 1675–1683, Apr. 2024, Doi: 10.1002/Ird.2973.
[27] G. K. Sahoo, D. P. Satapathy, S. Samantaray, D. Panda, S. Mishra, And N. Patel, "Streamflow Forecasting Using Novel Anfis-Gwo Approach," Springer Nature Singapore, 2023, Pp. 141–152. Doi: 10.1007/978-981-19-7513-4_13.
[28] Z. M. Yaseen Et Al., "Hourly River Flow Forecasting: Application Of Emotional Neural Network Versus Multiple Machine Learning Paradigms," Water Resources Management, Vol. 34, No. 3, Pp. 1075–1091, Jan. 2020, Doi: 10.1007/S11269-020-02484-W.
[29] K. B. Tadesse, M. O. Dinka, T. Alamirew, And S. A. Moges, "Evaluation Of Seasonal Autoregressive Integrated Moving Average Models For River Flow Forecasting," American Journal Of Environmental Sciences, Vol. 13, No. 5, Pp. 378–387, May 2017, Doi: 10.3844/Ajessp.2017.378.387.
[30] A. A. Akinola And G. A. Okanlawon, "Modelling Moisture Ratio Of Dehydrating Yam Slices Using The Levenberg-Marquardt Back-Propagation Artificial Neural Network Technique," Fuoye Journal Of Engineering And Technology, Vol. 7, No. 4, Dec. 2022, Doi: 10.46792/Fuoyejet.V7i4.923.
[31] "Artificial Neural Networks In Hydrology. Ii: Hydrologic Applications," Journal Of Hydrologic Engineering, Vol. 5, No. 2, Pp. 124–137, Apr. 2000, Doi: 10.1061/(Asce)1084-0699(2000)5:2(124).
[32] S. Nacar, M. A. Hınıs, And M. Kankal, "Forecasting Daily Streamflow Discharges Using Various Neural Network Models And Training Algorithms," Ksce Journal Of Civil Engineering, Vol. 22, No. 9, Pp. 3676–3685, Dec. 2017, Doi: 10.1007/S12205-017-1933-7.
[33] E. Mutlu, H. Hexmoor, I. Chaubey, And S. G. Bajwa, "Comparison Of Artificial Neural Network Models For Hydrologic Predictions At Multiple Gauging Stations In An Agricultural Watershed," Hydrological Processes, Vol. 22, No. 26, Pp. 5097–5106, Sep. 2008, Doi: 10.1002/Hyp.7136.
[34] Y. Hu, L. Yan, T. Hang, And J. Feng, "Stream-Flow Forecasting Of Small Rivers Based On Lstm." Cornell University, Jan. 16, 2020. Doi: 10.48550/Arxiv.2001.05681.
[35] R. I. Abdulkadirov And P. A. Lyakhov, "A New Approach To Training Neural Networks Using Natural Gradient Descent With Momentum Based On Dirichlet Distributions," Computer Optics, Vol. 47, No. 1, Pp. 160–169, Feb. 2023, Doi: 10.18287/2412-6179-Co-1147.
[36] S. N. Endah, F. Maulana, S. I. Nadianada, A. P. Widodo, And M. L. Fariq, "Beyond Back-Propagation Learning For Diabetic Detection: Convergence Comparison Of Gradient Descent, Momentum And Adaptive Learning Rate," Nov. 2017, Pp. 189–194. Doi: 10.1109/Icicos.2017.8276360.
[37] Q. Li, "An Error Based Adaptive Learning Rate Stochastic Gradient Descent Algorithm In Convolutional Neural Network," Applied And Computational Engineering, Vol. 2, No. 1, Pp. 223–231, Mar. 2023, Doi: 10.54254/2755-2721/2/20220515.
[38] J. Bilski, B. Kowalczyk, I. Izonin, J. Smoląg, And K. Grzanek, "Fast Computational Approach To The Levenberg-Marquardt Algorithm For Training Feedforward Neural Networks," Journal Of Artificial Intelligence And Soft Computing Research, Vol. 13, No. 2, Pp. 45–61, Mar. 2023, Doi: 10.2478/Jaiscr-2023-0006.
[39] G. Wu, G. Yuan, And Y. Li, "A Three-Term Conjugate Gradient Algorithm With Quadratic Convergence For Unconstrained Optimization Problems," Mathematical Problems In Engineering, Vol. 2018, No. 2018, Pp. 1–15, Jun. 2018, Doi: 10.1155/2018/4813030.
[40] D. Hopp, "Economic Nowcasting With Long Short-Term Memory Artificial Neural Networks (Lstm)," Journal Of Official Statistics, Vol. 38, No. 3, Pp. 847–873, Sep. 2022, Doi: 10.2478/Jos-2022-0037.