TREND ANALYSIS AND PREDICTION OF RAINFALL USING DEEP LEARNING MODELS IN THREE SUB-DIVISIONS OF KARNATAKA

ABSTRACT

Precise estimation of rainfall is a crucial and challenging task in environmental science. It involves the use of advanced and powerful models to forecast non-linear and dynamic changes in rainfall. Deep learning, a recently developed method for handling vast amounts of data and resolving complex problems, has proven to be an effective tool for rainfall forecasting. In this study, we applied various deep learning models such as Multilayer Perceptron (MLP), Long Short-Term Memory (LSTM), Bidirectional LSTM (Bi-LSTM), Stacked LSTM, Gated Recurrent Units (GRUs), and a traditional model called Autoregressive Integrated Moving Average (ARIMA), to forecast monthly rainfall data (mm) for three regions of Karnataka: Coastal Karnataka, North Interior Karnataka (NIK), and South Interior Karnataka (SIK). Trend analysis was conducted using the Mann-Kendall trend test (MK test) and the Seasonal Mann-Kendall trend test, along with Sen's Slope Estimator, to determine trends and slope magnitudes. The results showed that deep learning models perform better than traditional methods in forecasting rainfall. The performance of different models was evaluated using forecasting evaluation criteria and found that the LSTM model performed best for Coastal Karnataka, with an RMSE value of 149.45, while the Bi-LSTM model performed best for NIK, with an RMSE value of 32.57, and the Stacked LSTM model performed best for SIK, with an RMSE value of 45.33. Therefore, deep learning models can be effectively used to predict rainfall data with greater accuracy.

Keywords: *Autoregressive Integrated Moving Average (ARIMA), Bidirectional LSTM (Bi-LSTM), Gated Recurrent Unit (GRU), Long Short-Term Memory (LSTM), Multilayer Perceptron (MLP), Stacked LSTM*

INTRODUCTION

Agriculture is a critical component of India's well-being, heavily reliant on precipitation for success. Efficient water resource management can also be supported through agricultural practices. Past precipitation data has enabled farmers to better manage their crops, contributing to the country's economic growth. Accurate rainfall prediction is essential for preventing flooding, which can lead to significant property damage and loss of life. Due to

fluctuations in timing and quantity, precipitation forecasting is challenging for meteorologists. Researchers from various disciplines, such as weather data mining, environmental machine learning, functional hydrology, and numerical forecasting, are working to develop predictive models for accurate rainfall forecasting. The capacity to infer past and future predictions are crucial in addressing these issues. Precipitation involves numerous sub-processes, making it difficult to predict accurately on a global scale. Climate forecasting is essential for all countries worldwide and is provided by meteorological departments, offering significant benefits and services. Early identification of severe weather conditions through precise precipitation forecasting can aid in mitigating natural disaster injuries and damages. In India, numerous rainfall prediction methods are available, with traditional models limited to linear data analysis. Machine learning models can capture nonlinearity in data, but feature extraction issues can pose challenges. Deep learning models, a subset of machine learning, can overcome these issues and adapt to the data to capture volatility better, resulting in improved results. Therefore, the forecasting of rainfall is being carried out for three Karnataka subdivisions, namely Coastal Karnataka, North Interior Karnataka, and South Interior Karnataka.

MATERIAL AND METHODS

2.1 To analyse the trend in monthly rainfall data

Trend analysis is a commonly used tool to detect changes in climatic and rainfall time-series data. With the predicted changes in global climate, trend detection in rainfall data has received significant attention. However, climatic variability, as reflected in rainfall data, can negatively impact rainfall trends. Parametric tests are more commonly used for trend analysis, assuming independent and normally distributed data. However, real-world data may not always meet these assumptions. To address this, more powerful nonparametric tests are used, especially for rainfall data, as they do not require the strict assumptions of parametric tests and can tolerate the effect of outliers in the data. Several nonparametric tests exist in the literature to analyze trends, with Mann-Kendall's (M-K) test being one of the most widely used tests for estimating trends in rainfall data. The M-K test procedure involves:

2.1.1 Mann-Kendall (M-K) test

The Mann-Kendall trend (M-K) test is a non-parametric test, in contrast to the parametric method of trend analysis. It is considered the most suitable test for identifying trends in rainfall data. One of the benefits of using the non-parametric M-K test is that it relies on the

sign of differences rather than the values of random variables, making the trend less influenced by fluctuations and extreme rainfall values. This is especially important in the analysis of weather parameters, which are highly prone to fluctuations (Sanjeevaiah et al., 2021).

H_0: The rainfall data don't show any monotonic trends.

H_1: The rainfall data show monotonic trends (either decreasing or increasing).

The Mann-Kendall statistics 'S' is given as

$$S = \sum_{k=1}^{n-1} \sum_{j=k+1}^{n} sign(x_j - x_k) \quad \ldots [1]$$

where,

$$sign(x_j - x_k) = \begin{cases} +1 & if \quad (x_j - x_k) > 0 \\ 0 & if \quad (x_j - x_k) > 0 \\ -1 & if \quad (x_j - x_k) > 0 \end{cases} \quad \ldots [2]$$

and x_j, x_k are the sequential data values and for all $k, j \leq n$ with $k \neq j$ and n is the length of the rainfall data set.

2.1.2 Seasonal Mann-Kendall test (Seasonal M-K test)

The Seasonal Mann-Kendall (M-K) test is a nonparametric statistical test that evaluates seasonal data for monotonic trends. It was developed by Hirsch, Smith, & Slack in the 1980s and has become widely used in environmental studies. The term "monotonic" refers to a consistent upward or downward trend, while "seasonal" data pertains to data collected for periods that exhibit trends that can be either upward or downward. These periods can include seasons such as spring and summer, as well as time periods such as hours, days, or months.

The Seasonal M-K test is a subset of the Mann-Kendall Trend Test and is used when the data is seasonal. In cases where the data is not seasonal, the Mann-Kendall Trend Test should be used instead. The Seasonal M-K test involves conducting independent Mann-Kendall trend tests on each of the m seasons, where m is the number of seasons. The comparison is made only between data from the same season (Zhang et al., 2016). The overall Seasonal M-K test statistic is obtained by adding the Kendall S statistics from each season.

$$S_k = \sum_{i=1}^{m} S_i \quad \text{... [3]}$$

2.1.3 Sen's slope estimator

Sen's slope estimator is a commonly used method for estimating the magnitude of a trend. It is a nonparametric linear slope estimator that is particularly effective with monotonic data. Unlike linear regression, Sen's slope estimator is less affected by data errors, outliers, or missing data. The method involves calculating the slope (T_i)of each pair of data using the following formula:

$$T_{i=}\frac{(x_j - x_k)}{j-k} \; for\; i = 1,2,\dots,n \quad \text{... [4]}$$

where x_j and x_k are considered as data values at time j and k ($j > k$) correspondingly. The median of these n values of T_i is represented as Sen's estimator of the slope, which is given as:

$$Q_{Med} = \begin{cases} T_{\frac{N+1}{2}} & if\ N\ is\ odd \\ \frac{1}{2}\left(T_{\frac{N}{2}} + T_{\frac{N+2}{2}}\right) & if\ N\ is\ even \end{cases} \quad \text{... [5]}$$

In the end, T_{med} is computed by a two-sided test at 100 $(1-\alpha)$ % confidence interval and then a true slope can be obtained by this non-parametric test. Positive value of Q_{Med} indicates an upward or increasing trend and a negative value of Q_{Med} gives a downward or decreasing trend in the time-series (Mondal *et al*, 2012).

2.2 Forecasting of monthly rainfall data

2.2.1 Autoregressive Integrated Moving Average (ARIMA)

ARIMA model is a generalization of ARMA models which incorporate a wide range of non-stationary timeseries by suitable order of differencing. The simplest example of a non-stationary process which reduces to a stationary one after differencing is 'Random Walk.' A process $\{y_t\}$ is said to follow an Integrated ARMA model, denoted by ARIMA (p, d, q), if $\nabla^d y_t = (1-B)^d \varepsilon_t$ is ARMA (p, q). The model is written as:

$$\varphi(B)(1-B)^d y_t = \theta(B)\varepsilon_t \quad \text{... [6]}$$

where, $\varepsilon_t \sim WN(0,\sigma^2), WN$ indicates white noise, $\varphi(B) = 1 - \varphi_1 B - \varphi_2 B^2 - \cdots - \varphi_p B^p$ and $\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \cdots - \theta_q B^q$. The integration parameter d is a nonnegative integer.

There are four major stages of ARIMA model building. These are identification, estimation, validation, and forecasting. A detailed discussion on various aspects of this approach is given in Box *et al.,* (2007).

2.2.2 Multi-layer Perceptron (MLP)

An artificial neural network (ANN) is a computational model inspired by the brain's information processing and analysis capabilities, capable of solving a wide range of non-linear problems (Nguyen et al., 2019). ANN provides several advantages, such as the ability to perform parallel processing, learning from experience (datasets), and being capable of approximating complex functions with high accuracy without the need for reprogramming. ANN finds wide application in classification and forecasting problems. Multi-layer Perceptron (MLP) is the most popular ANN model used for time-series forecasting problems (Ravi et al., 2017). The MLP structure consists of an input layer, hidden layer, and output layer (Eskov et al., 2019), each containing several processing units or neurons connected via directed links with individual weights.

The mathematical equations of the neural network are presented by Eq. (7).

$$\hat{y} = \sum_{j=0}^{s_o} W_{jk} \cdot \xi\left(\sum_{i=0}^{s_h} W_{ij} \cdot x\right) \quad \text{... [7]}$$

$$\xi(\alpha, x) = \begin{cases} \alpha \cdot (e^x - 1), & \text{if } x < 0 \\ x, & \text{if otherwise} \end{cases} \quad \text{... [8]}$$

where x and $\hat{y}$ are the input and output of the network, respectively. s_0 and s_h are the size of the output layer and hidden layer. W_{ij} are the weights of connection between input and hidden layer, W_{jk} is the weights of connection between hidden and output layer. ξ is the Exponential Linear Unit (ELU) (Clevert *et al.,* 2015) presented by Eq. (8) (in its general form, when $\alpha = 1$). It becomes the Rectified Linear Unit (ReLU) when $\alpha = 0$.

2.2.3 Long Short-Term Memory (LSTM)

Recurrent Neural Networks (RNNs) incorporate Long Short-Term Memory (LSTM), a type of neural network that addresses the challenge of long-term RNN dependencies, where the RNN can only forecast based on current data and is unable to access data stored in long-term memory. LSTM was developed by Hochreiter and Schmidhuber in 1997. RNNs become less effective as the time gap between relevant data points increases. LSTM solves the problem of gradient vanishing and forgetting that typically occurs in traditional RNNs. It has the ability to retain data for an extended period and can also learn to make a one-shot multi-step prediction, which is useful for time-series prediction (Jaiswal *et al.*, 2022). An LSTM neural network unit includes four gates: an input gate, a cell state, a forgotten gate, and an output gate.

Structure of LSTM

The LSTM has a chain structure that includes four neural networks and various memory blocks known as cells.

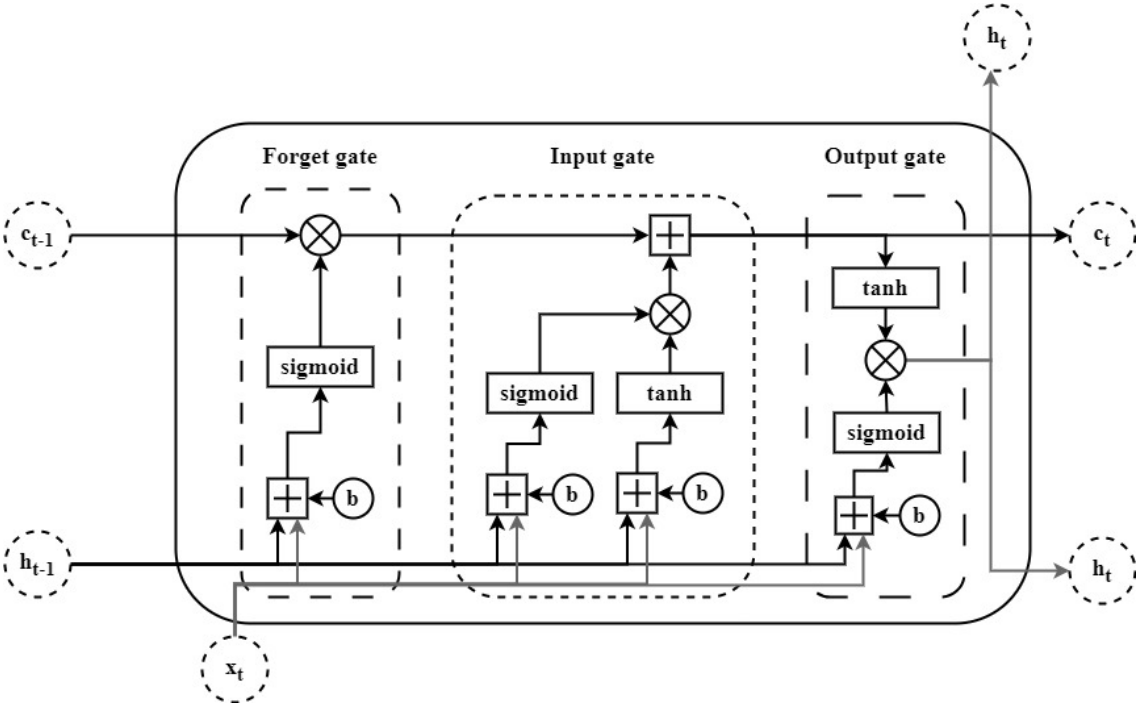

Fig. 1: The long short-term memory (LSTM) architecture

The cells store information, and the gates perform memory manipulations. There are three gates –

1. **Forget Gate:** The forget gate deletes information that is no longer useful in the cell state. Two inputs, $\boldsymbol{x_t}$ (at-the-time input) and $\boldsymbol{h_{t-1}}$ (previous cell output), are fed into the gate and multiplied with weight matrices before bias is added. The result is fed into an activation function, which produces a binary output. If the output for a specific cell state is 0, the information is lost; if the output is 1, the information is saved for future use.

2. **Input gate:** The input gate is responsible for adding useful information to the cell state. First, the information is regulated using the sigmoid function, and the values to be remembered are filtered using the $\boldsymbol{h_{t-1}}$ and $\boldsymbol{x_t}$ inputs, similar to the forget gate. The tanh function is then used to generate a vector with values ranging from -1 to +1 that contains all of the possible values from $\boldsymbol{h_{t-1}}$ and $\boldsymbol{x_t}$. Finally, the vector and regulated values are multiplied to obtain useful information.

3. **Output gate:** The output gate is in charge of extracting useful information from the current cell state and presenting it as output. To begin, a vector is created by applying the tanh function to the cell. The information is then regulated using the sigmoid function and filtered by the values to be remembered via $\boldsymbol{h_{t-1}}$ and $\boldsymbol{x_t}$ inputs. Finally, the vector and regulated values are multiplied and sent as output and input to the next cell.

The main calculation process is described as follows

$$i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1} + W_{ic}C_{t-1} + b_i) \quad \text{... [9]}$$

$$f_t = \sigma\left(W_{fx}x_t + W_{fm}m_{t-1} + W_{fc}C_{t-1} + b_f\right) \quad \text{... [10]}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g\,(W_{cx}x_t + W_{cm}m_{t-1} + b_c) \quad \text{... [11]}$$

$$o_t = \sigma(W_{ox}x_t + W_{om}m_{t-1} + W_{oc}C_{t-1} + b_o) \quad \text{... [12]}$$

$$m_t = o_t \odot h(c_t) \quad \text{... [13]}$$

$$y_t = \varphi(W_{ym}m_t + b_y) \quad \text{... [14]}$$

where c_t represent the memory cell, w and b are the weight matrices and the bias vectors, respectively, x_t denotes the input data at a present time step t, σ is the sigmoid function, and φ is the output activation function.

2.2.4 Bi-directional Long Short-Term Memory (Bi-LSTM)

Bidirectional long-short term memory (Bi-LSTM) is a neural network architecture that processes input data in both directions, either forward (past to future) or backward (future to past), allowing it to capture information from both the past and future contexts. Bi-LSTM extends the traditional LSTM by adding a second LSTM network that processes the input in reverse order. This results in two sets of hidden states, one for each direction, which is concatenated to form the final output. Bi-LSTM is especially useful for tasks where both past and future context is important, such as natural language processing, speech recognition, and image processing (Huang et al., 2022).

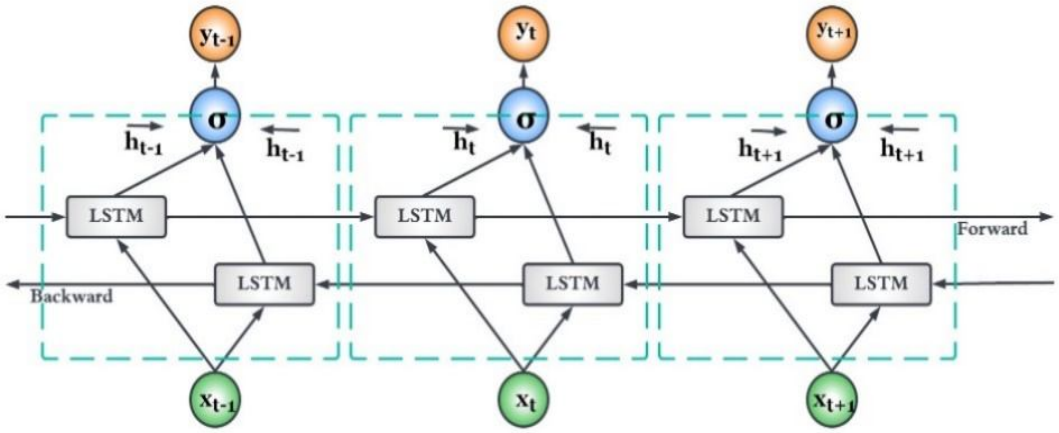

Fig. 2: The Bidirectional long short-term memory (LSTM) architecture

In the above diagram, we can see the flow of information from backward and forward layers. Bi-LSTM is usually employed where the sequence-to-sequence tasks are needed. This kind of network can be used in text classification, speech recognition and forecasting models.

The process can be described as:

$$\vec{h}_t = LSTM\left(x_t, \vec{h}_{t-1}\right) \quad \text{... [15]}$$

$$\overleftarrow{h}_t = LSTM\left(x_t, \overleftarrow{h}_{t+1}\right) \quad \text{... [16]}$$

$$y_t = \delta\left(W_{\vec{h}_y}\vec{h}_t + W_{\overleftarrow{h}_t}\overleftarrow{h}_t + b_y\right) \quad \text{... [17]}$$

where (.) represents LSTM network, ($W_{\vec{h}_y}$ and $W_{\overleftarrow{h}_t}$) represent the weight of the forward and backward LSTM layer at time *t*, respectively. b_y denotes the bias of the output layer, $\delta(.)$ represents the activation function.

2.2.5 Stacked LSTM

The Long Short-Term Memory (LSTM) model is a type of neural network consisting of a single hidden layer of LSTM units, followed by a conventional output layer. To increase the depth of the model, a model extension called Stacked LSTM was introduced. Stacked LSTM has multiple hidden LSTM layers, each with multiple memory cells. The depth of the neural network is increased by the addition of multiple LSTM layers, making it a deep learning technique. The success of deep learning in tackling difficult prediction problems is often attributed to the depth of the network. Stacked LSTM has emerged as a reliable technique for solving difficult sequence prediction problems. In a stacked LSTM architecture, each LSTM layer sends a sequence of values to the layer below, rather than a single value. The output of each time step is an input for all subsequent time steps (Ojo et al., 2019).

Fig. 3: The stacked long short-term memory (LSTM) architecture

2.2.6 Gated Recurrent Unit (GRU)

The gated recurrent unit (GRU) is a gating mechanism used in recurrent neural networks (RNNs) to address the vanishing gradient problem that can arise in traditional RNNs. The GRU has a similar design to the long short-term memory (LSTM) unit, but it does not have an output gate. GRUs employ an update gate and a reset gate to regulate the flow of information, which helps to solve the vanishing gradient problem. The update gate controls the inflow of information into memory, while the reset gate controls the outflow of information from memory. While GRUs and LSTMs have similar designs and can produce comparable results in some cases, GRUs are particularly useful for addressing the vanishing gradient problem and are more effective than LSTMs with smaller datasets. GRUs have been

used in various applications, including polyphonic music modeling, speech signal processing, handwriting recognition, and time series forecasting.

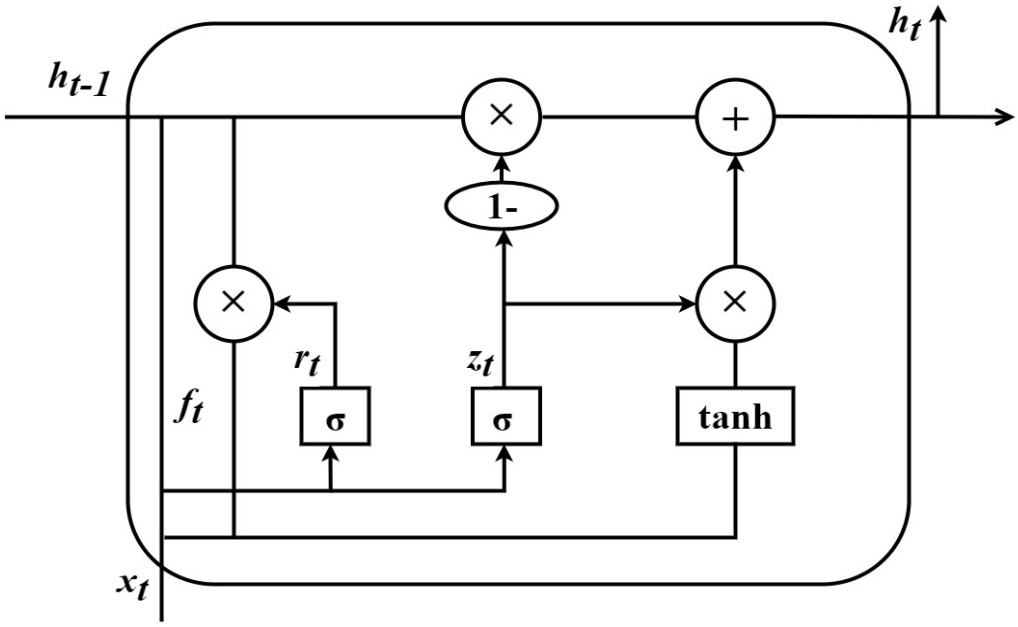

Fig. 4: The Gated Recurrent Unit (GRU) architecture

The process can be described as:

$$Z_t = \sigma(x_t w^z + h_{t-1} U^z + b_z) \quad \text{... [18]}$$

$$r_t = \sigma(x_t w^r + h_{t-1} U^r + b_z) \quad \text{... [19]}$$

$$\widetilde{h_t} = tan(r_t \times h_{t-1} U + x_t W + b) \quad \text{... [20]}$$

$$h_t = (1 - Z_t) \times \widetilde{h_t} + Z_t \times h_{t-1} \quad \text{... [21]}$$

where w^z , w^r , W denote the weight matrices for the corresponding connected input vector. U^z , U^r , U represent the weight matrices of the previous time step, and b_r, b_z and b are bias. The σ denotes the sigmoid function, r_t denotes the reset gate, z_t denotes the update gate, and $\widetilde{h_t}$ denotes the candidate hidden layer.

RESULTS AND DISCUSSION

3.1 Analysis of trend in month-wise rainfall data

The trend in monthly, seasonal, and yearly rainfall data from three meteorological subdivisions of Karnataka was analysed using non-parametric tests, specifically the Mann-

Kendall (M-K) and Seasonal Mann-Kendall tests. The M-K and seasonal M-K test statistics (Tau) values were used to determine the presence of a monotonic trend in the rainfall data. A positive Tau (τ) value indicates an upward or increasing trend, while a negative value suggests a downward or decreasing trend. The magnitude of the trend was calculated using Sen's slope estimator (Q_{Med}), with values close to zero indicating a minimal change in rainfall rate, while greater positive or negative values denote a significant increase or decrease in rainfall rate, respectively.

Before trend analysis, the significance of the serial correlation coefficient was assessed to confirm the presence of serial correlation in all rainfall data series. Kendall's Tau (τ) test revealed substantial serial correlation in the majority of the rainfall data series. To remove the impact of serial correlation on the acceptance or rejection of Kendall Tau (τ), the Seasonal M-K test for variance correction was employed. Python (3.9.0) and R (3.6.1) was used for the rainfall trend analysis.

3.1.1 Coastal Karnataka

Table 1 shows that there is no significant monotonic trend in the monthly rainfall data. The M-K test statistic for February is significant at a 1 percent level of significance, while the remaining months (January, March, April, May, June, July, August, September, October, November, and December) are not significant. The presence of serial correlation between consecutive months may have contributed to this result. The seasonal M-K test statistic value indicates a significant monotonic increasing trend for January, February, March, and April at a 1 percent level of significance, and for July at a 5 percent level of significance. The value is not significant for the remaining months of May, June, August, September, October, November, and December. The positive sign of the seasonal M-K value for January, February, March, and April indicates an upward trend, while the negative sign for July suggests a downward trend. No significant trend is observed for the remaining months.

Table 1: M-K and Seasonal M-K test statistic (Tau) and Sen's slope estimate for Coastal subdivision for monthly rainfall (mm) data

Period	M-K test			Seasonal M-K test			Sen's slope
	Tau	*p*-value	Trend	Tau	*p*-value	Trend	
January	0.15^{NS}	0.09	No trend	**0.15****	<0.01	**Increasing**	0.00
February	**0.27****	0.01	**Increasing**	**0.27****	<0.01	**Increasing**	0.0003

March	0.15^{NS}	0.09	No trend	**0.15***	0.03	**Increasing**	0.02
April	0.12^{NS}	0.17	No trend	**0.12****	<0.01	**Increasing**	0.02
May	-0.03^{NS}	0.75	No trend	-0.03 ^{NS}	0.48	No trend	-0.29
June	0.07 ^{NS}	0.43	No trend	0.07^{NS}	0.24	No trend	1.10
July	-0.15^{NS}	0.08	No trend	**-0.15****	0.00	**Decreasing**	-3.43
August	-0.003^{NS}	0.97	No trend	-0.0^{NS}	0.94	No trend	-0.03
September	0.03^{NS}	0.74	No trend	0.03 ^{NS}	0.56	No trend	0.36
October	0.06^{NS}	0.52	No trend	0.06^{NS}	0.23	No trend	0.42
November	0.04^{NS}	0.70	No trend	0.04^{NS}	0.30	No trend	0.14
December	-0.05^{NS}	0.60	No trend	-0.05 ^{NS}	0.52	No trend	-0.01

NS: Non-Significant, *Significant at 5% level of significance, **Significant at 1% level of significance

3.1.2 NIK Subdivision

For the NIK subdivision, the M-K test statistic values in Table 2 indicate that there is no monotonic trend, as they were non-significant at the 5 percent level of significance for all months. This lack of trend may be attributed to the serial correlation present in the rainfall data from month to month.

Regarding the seasonal M-K test, January, March, April, and August months' test statistic values were significant at a 5 percent level of significance, while February, May, July, and September months' values were significant at a 1 percent level of significance, with the remaining months being non-significant. The positive sign of the seasonal M-K value indicated a monotonic increasing trend for January, February, March, and August, while the negative sign indicated a monotonic decreasing trend for April, May, July, and September. There was no monotonic trend observed for the remaining months of June, October, November, and December.

Table 2: M-K and Seasonal M-K test statistic (Tau) and Sen's slope estimate for NIK subdivision for monthly rainfall (mm) data

Period	**M-K test**			**Seasonal M-K test**			**Sen's slope**
	Tau	***p*-value**	**Trend**	**Tau**	***p*-value**	**Trend**	
January	0.10^{NS}	0.26	No Trend	**0.10***	0.01	**Increasing**	0.00

February	0.12^{NS}	0.18	No Trend	**0.12****	<0.01	**Increasing**	0.0008
March	0.04^{NS}	0.64	No Trend	**0.04***	0.05	**Increasing**	0.01
April	-0.05^{NS}	0.61	No Trend	**-0.05***	0.02	**Decreasing**	-0.06
May	-0.07^{NS}	0.44	No Trend	**-0.07****	0.01	**Decreasing**	-0.17
June	0.03^{NS}	0.77	No Trend	0.03^{NS}	0.05	No trend	0.05
July	-0.13^{NS}	0.15	No Trend	**-0.13****	<0.01	**Decreasing**	-0.50
August	0.06^{NS}	0.50	No Trend	**0.06***	0.01	**Increasing**	0.20
September	-0.09^{NS}	0.32	No Trend	**-0.09****	<0.01	**Decreasing**	-0.50
October	-0.05^{NS}	0.57	No Trend	-0.05^{NS}	0.17	No trend	-0.26
November	-0.04^{NS}	0.66	No Trend	-0.04^{NS}	0.34	No trend	-0.04
December	0.04^{NS}	0.64	No Trend	0.04^{NS}	0.41	No trend	0.001

NS: Non-Significant, *Significant at 5% level of significance, **Significant at 1% level of significance

3.1.3 SIK Subdivision

The monthly rainfall data for the SIK subdivision does not exhibit a consistent trend, as indicated by the non-significant Mann-Kendall (M-K) test statistic values in Table 3, with a 5 percent level of significance. However, the Seasonal M-K test results indicate a significant increasing trend in rainfall during February, March, April, June, and August, with statistical significance at the 1 percent or 5 percent level. In contrast, there is no statistically significant trend observed in the remaining months of January, May, July, September, October, November, and December. The positive Seasonal M-K test values for the months with a significant trend suggest a consistent increase in monthly rainfall over time.

Table 3: M-K and Seasonal M-K test statistic (Tau) and Sen's slope estimate for SIK subdivision for monthly rainfall (mm) data

Period	**M-K test**			**Seasonal M-K test**			**Sen's slope**
	Tau	***p*-value**	**Trend**	**Tau**	***p*-value**	**Trend**	
January	0.02^{NS}	0.15	No trend	0.02^{NS}	0.05	No trend	0.00
February	0.08^{NS}	0.39	No trend	**0.08****	0.01	**Increasing**	0.003

March	0.15^{NS}	0.08	No trend	**0.15****	0.01	**Increasing**	0.08
April	0.09^{NS}	0.32	No trend	**0.09***	0.02	**Increasing**	0.16
May	0.06^{NS}	0.47	No trend	0.06^{NS}	0.31	No trend	0.16
June	0.17^{NS}	0.06	No trend	**0.17****	<0.01	**Increasing**	0.37
July	0.03^{NS}	0.25	No trend	0.03^{NS}	0.16	No trend	0.05
August	0.15^{NS}	0.10	No trend	**0.15****	<0.01	**Increasing**	0.56
September	-0.07^{NS}	0.44	No trend	-0.07^{NS}	0.05	No trend	-0.38
October	0.02^{NS}	0.86	No trend	0.02^{NS}	0.70	No trend	0.08
November	-0.03^{NS}	0.75	No trend	-0.03^{NS}	0.29	No trend	-0.07
December	-0.04^{NS}	0.65	No trend	-0.04^{NS}	0.41	No trend	-0.03

NS: Non-Significant, *Significant at 5% level of significance, **Significant at 1% level of significance

3.2 Forecasting of rainfall for three subdivisions

3.2.1 Data Description

This study utilizes monthly rainfall data (measured in millimeters) from three subdivisions of Karnataka, India, including Coastal Karnataka, North Interior Karnataka (NIK), and South Interior Karnataka (SIK). The data were obtained from the public repository "data.world" and cover the period from January 1901 to December 2015, with a total of 1381 observations for each series. Visual inspection of the time plots in Figures 5, 6, and 7 suggests that all three series exhibit non-stationarity, which is confirmed through statistical tests for stationarity presented in Table 4. To construct and evaluate the models, the series is divided into training and testing sets, with the training set comprising 1105 months of observations used for model development and in-sample prediction, while the remaining 276 observations are reserved for post-sample prediction testing.

This study aims to identify the most effective deep learning model for forecasting rainfall and compare its performance to that of other statistical forecasting models, including ARIMA. The ARIMA modeling is implemented using R version 3.6.1 software, while the deep learning models are developed using Python 3.9.0 with the Keras and Tensorflow libraries. The software is executed on a system with the following specifications: a Ryzen 7 5700U

CPU with a clock speed of 1.8 GHz, 8.0 GB of RAM, and AMD Radeon Graphics running the Windows 11 operating system.

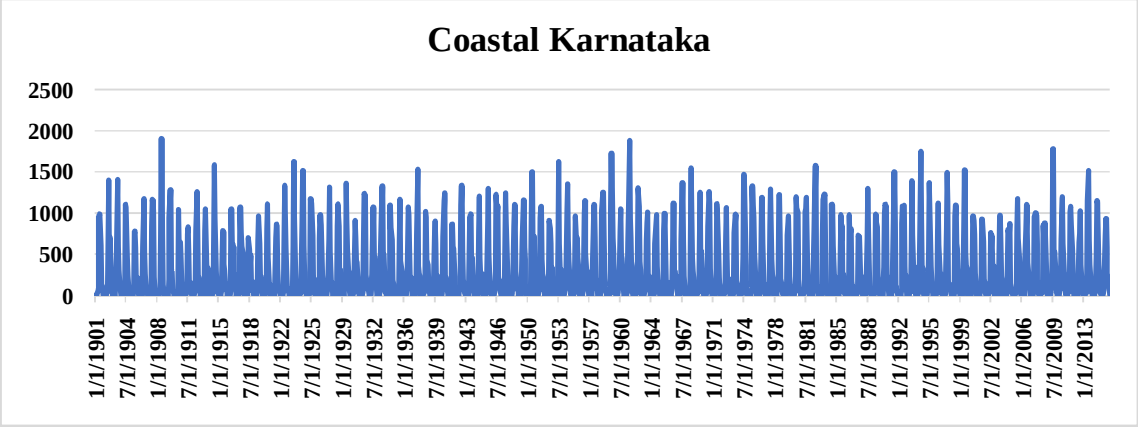

Fig. 5: Time plot of rainfall (mm) of Coastal Karnataka

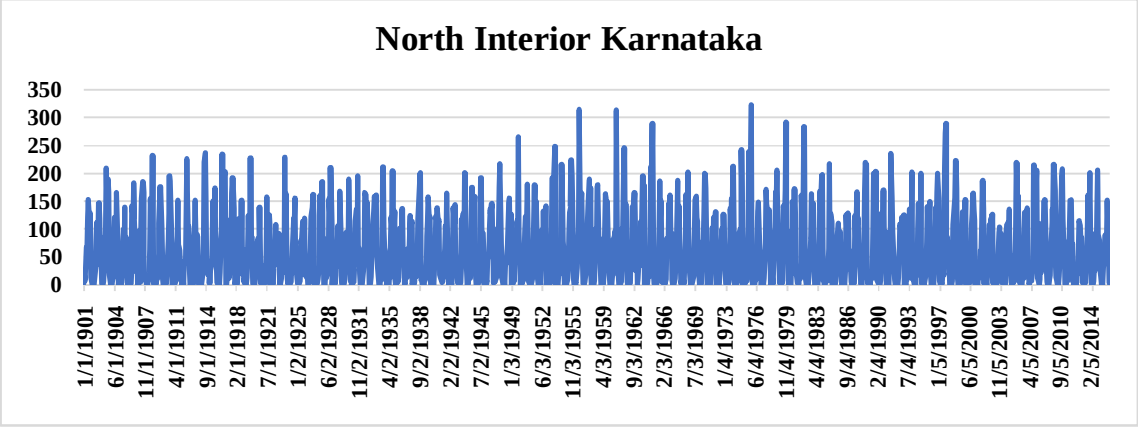

Fig. 6: Time plot of rainfall (mm) of North Interior Karnataka

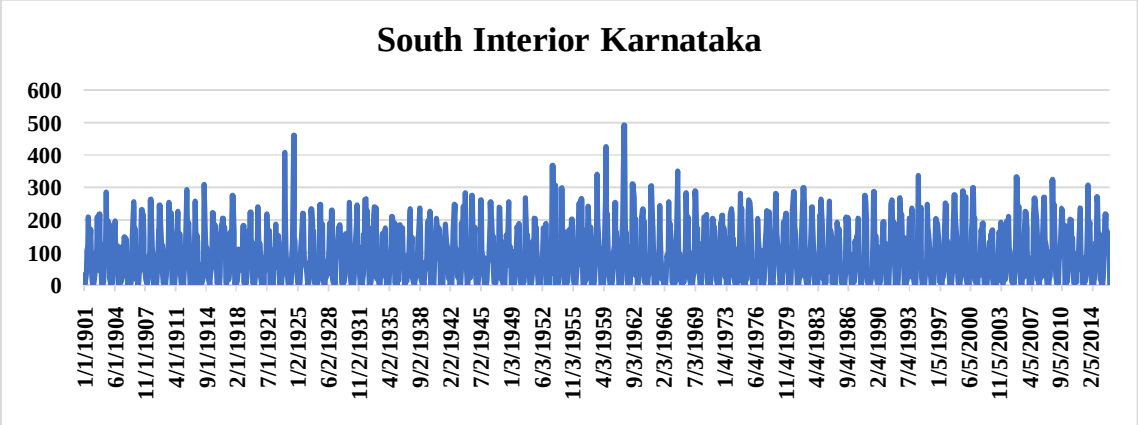

Fig. 7: Time plot of rainfall (mm) of South Interior Karnataka

3.2.2 Test for stationarity

A time series is considered stationary if it exhibits constant mean and variance over time. The Augmented Dickey-Fuller (ADF) test is commonly used to assess the stationarity of a data series. The null hypothesis of the ADF test is that the time series contains a unit root, indicating non-stationarity. The ADF statistic returns a negative value, with greater negative values indicating stronger evidence against the null hypothesis. The results of the ADF tests are presented in Table 4, confirming that all series under investigation exhibit non-stationarity. The tests were conducted using the *adf.test* function from the *tseries* package in the R software.

Table 4: Results for stationarity using ADF test

Subdivisions of Karnataka	Test Statistic	*p*-value	Remarks
Coastal Karnataka	-3.08	0.21	Non-stationary
NIK	-2.86	0.30	Non-stationary
SIK	-1.69	0.15	Non-stationary

3.2.3 Data pre-processing and normalisation

Data pre-processing and normalisation of data series are required so that deep learning models can be fitted effectively and extrapolated without bias. The normalisation

technique rescales a data series' values between 0 and 1 without altering its shape. The following equation is used to normalise both series:

$$X'_t = \frac{X_t - X_{min}}{X_{max} - X_{min}}$$

where X_{min}, X_{max} and X_t are the minimum, maximum and observation at time t, respectively and X'_t is the rescaled value. In python software, we have used Min-Max Scaler function of the Scikitlearn package for this purpose.

3.2.4 Implementation of forecasting models

Following the confirmation of non-stationarity and normalization of each series, the proposed models are developed and fitted as follows: The data is first pre-processed and normalized before being divided into training (80%) and testing (20%) sets. The training set is utilized for model construction, while the testing set is used to evaluate its performance. Each series is fitted individually using ARIMA, MLP, LSTM, Bi-LSTM, Stacked LSTM, and GRU models, and their performance is assessed using the Root Mean Square Error (RMSE) metric. The required model parameters and hyperparameters are detailed in Table 5 and Table 6. The model with the lowest RMSE value among all fitted models is deemed to perform better.

The ARIMA model was fitted using the optimal combination of parameters as determined by Table 5. While this model can capture linear components in the data, it does not account for nonlinearity. Therefore, other nonlinear models were explored.

After estimating the models, forecasts were generated for the testing data set. The RMSE was used to evaluate the effectiveness of rainfall predictions made by various models, including ARIMA, MLP, LSTM, Bi-LSTM, Stacked LSTM, and GRU. For Coastal Karnataka, NIK, and SIK, respectively, the LSTM, Bi-LSTM, and Stacked LSTM models produced lower RMSE values than other competing models. Among all the models, ARIMA had the highest RMSE, indicating that it was the least accurate model.

Deep learning models are capable of sequence learning, enabling them to effectively capture the nonlinearity present in the rainfall dataset. Consequently, the LSTM, Bi-LSTM, and Stacked LSTM models were found to be the best-fitted models for Coastal Karnataka, NIK, and SIK, respectively. These models provide predicted values that are very close to the actual values, as shown in Table 7.

Table 5: Parameters of ARIMA model

Subdivisions of Karnataka	*p*	*d*	*q*
Coastal Karnataka	2	0	1
NIK	2	1	1
SIK	1	2	1

Table 6: Hyperparameters used for all deep learning models

Hyperparameters	
Activation function	ReLU
Optimizer	Adam
Loss function	Mean Squared Error
Batch size	10
Epochs	100

Table 7: Comparison of prediction performances of different models

Sl. No.	Models	RMSE value		
		Coastal Karnataka	**NIK**	**SIK**
1	**ARIMA**	304.6457	47.69623	65.21994
2	**MLP**	163.9448	33.55378	66.5592
3	**LSTM**	**149.4526**	33.21547	46.47722
4	**Bi-LSTM**	168.7255	**32.57809**	46.84162
5	**Stacked LSTM**	158.7145	36.55438	**45.3386**
6	**GRU**	162.826	32.8662	46.71207

Discussion

This study aimed to develop and evaluate various models for predicting rainfall in three distinct regions of India: Coastal Karnataka, North Interior Karnataka (NIK), and South Interior Karnataka (SIK). The models utilized include ARIMA, MLP, LSTM, Bi-LSTM, Stacked LSTM, and GRU. The performance of each model was evaluated using the RMSE metric, and the model with the lowest RMSE value was deemed the most accurate. The data was first pre-processed and normalized before being split into training and testing sets. The training set was utilized for model construction, while the testing set was used to evaluate the performance of the models. The ARIMA model was fitted using the optimal combination of parameters as determined by Table 5. However, since this model only captures linear components, the researchers explored other nonlinear models to capture nonlinearity in the

data. The results indicated that the LSTM, Bi-LSTM, and Stacked LSTM models provided the most accurate predictions for rainfall in Coastal Karnataka, NIK, and SIK, respectively. These models are capable of sequence learning, which enables them to effectively capture the nonlinearity present in the rainfall dataset. In contrast, ARIMA had the highest RMSE value among all the models, indicating that it was the least accurate model. On the whole, the LSTM, Bi-LSTM, and Stacked LSTM models were found to be the best-fitted models for rainfall prediction in Coastal Karnataka, NIK, and SIK, respectively. These models provide accurate predictions and can be utilized to improve agricultural planning, water management, and disaster preparedness in these regions. However, further research is needed to examine the generalizability of these models to other regions and datasets and to evaluate their performance using additional metrics.

CONCLUSION

In this study, we compared the performance of deep learning models (LSTM, Bi-LSTM, Stacked LSTM, and GRU) with the traditional ARIMA model for modeling and forecasting monthly rainfall data in three subdivisions of Karnataka. Our results showed that deep learning models outperformed other models in capturing nonlinear relationships in the data. This can be attributed to their ability to model sequential and time series data effectively, but training these models requires expertise and time. These findings highlight the potential of deep learning models in predicting monthly rainfall data, and future research may explore the use of other models or additional meteorological variables to improve forecast accuracy.

REFERENCE

Basha, C. Z., Bhavana, N., Bhavya, P., and Sowmya, V. (2020). Rainfall prediction using machine learning & deep learning techniques. In *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*, 92-97.

Bhuyan, M. D. I., Islam, M. M., & Bhuiyan, M. E. K. (2018). A trend analysis of temperature and rainfall to predict climate change for northwestern region of Bangladesh. *American Journal of Climate Change*, **7(2)**: 115-134.

Box, G. E. P., Jenkins, G. M. and Reinsel, G. C. (2007). Time-series analysis: forecasting and control. 3rd ed. Pearson Education, India.

Cho, K., Van Merrienboer, B., Bahdanau, D. and Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint* arXiv*:*1409.1259. https://arxiv.org/abs/1409.1259

D. A. Clevert, T. Unterthiner, and S. Hochreiter (2015). Fast and accurate deep network learning by exponential linear units (elus), *ArXiv Prepr*. arXiv: 1511.07289.

Hernández, E., Sanchez-Anguix, V., Julian, V., Palanca, J., and Duque, N. (2016). Rainfall prediction: A deep learning approach. In *International conference on hybrid artificial intelligence systems*, 151-162. Springer, Cham.

Huang, X., Li, Q., Tai, Y., Chen, Z., Liu, J., Shi, J. and Liu, W. (2022). Time series forecasting for hourly photovoltaic power using conditional generative adversarial network and Bi-LSTM. *Energy*, **246**: 123403.

Jaiswal, R., Jha, G. K., Kumar, R. R., and Choudhary, K. (2022). Deep long short-term memory based model for agricultural price forecasting. *Neural Computing and Applications*, **34(6):**4661-4676.

MONDAL, A., KUNDU, S. AND MUKHOPADHYAY, A., 2012, Rainfall trend analysis by Mann-Kendall test: A case study of north-eastern part of Cuttack district, Orissa. *Int. J. Geology, Earth Environ. Sci.*, **2(1):**70-78.

Ojo, S. O., Owolawi, P. A., Mphahlele, M., and Adisa, J. A. (2019). Stock market behaviour prediction using stacked LSTM networks. In *2019 International Multidisciplinary Information Technology and Engineering Conference (IMITEC)*, 1-5.

Sanjeevaiah, S. H., Kodandarama, S. R., Mohankumar, T. L., Lingaraj, H., Manjunatha, M. H., Sowmya, D. V., Nagesha, L., and Sheshshayee, M. S. (2021). Understanding the temporal variability of rainfall for estimating agro-climatic onset of cropping season over south interior Karnataka, India. *Agronomy*, **11(6):** 1135.

T. Nguyen, T. Nguyen, B.M. Nguyen, and G. Nguyen (2019). Efficient time-series forecasting using neural network and opposition-based coral reefs optimization, *Int. J. Comput. Intell. Syst.* **12:**1144-1161.

V. M. Eskov, V. F. Pyatin, V. V. Eskov, and L. K. Ilyashenko (2019). The heuristic work of the brain and artificial neural networks, *Biophysics (Oxf)*, **64:**293-299.

V. Ravi, D. Pradeepkumar, and K. Deb (2017). Financial time series prediction using hybrids of chaos theory, multi-layer perceptron and multi-objective evolutionary algorithms, *Swarm Evol. Comput*. **36**136-149.

Zhang, Y., Cabilio, P., and Nadeem, K. (2016). Improved seasonal Mann–kendall tests for trend analysis in water resources time series. In *Advances in time series methods and applications*, 215-229. Springer, New York, NY.