Rainfall Data Assisted Flood Alert System using Deep Learning and IoT

Sunil Khatri[1, a] and Rajani P. K. [2, b], Roshan Mathew[3, c], Atharva Walawalkar[4, d], Pratham Patil[5, e]

[1]*Research Scholar, Dept of E.X.T.C Engg, Pimpri Chinchwad College of Engineering, Savitribai Phule Pune University, Pune, India.*
[1]*Assistant Professor, Thakur College of Engineering and Technology, Mumbai, India.*
[2]*Associate Professor, Dept of E.X.T.C Engg, Pimpri Chinchwad College of Engineering, Savitribai Phule Pune University, Pune, India.*
[3]*Dept of E.X.T.C Engg, Pimpri Chinchwad College of Engineering, Savitribai Phule Pune University, Pune, India.*
[4]*Dept of E.X.T.C Engg, Pimpri Chinchwad College of Engineering, Savitribai Phule Pune University, Pune, India.*
[5]*Dept of E.X.T.C Engg, Pimpri Chinchwad College of Engineering, Savitribai Phule Pune University, Pune, India.*

[a)] skhatri1909@gmail.com
[b)] rajani.pk@pccoepune.org
[c)] roshan.mathew20@pccoepune.org
[d)] atharva.walawalkar20@pccoepune.org
[e)] pratham.patil20@pccoepune.org

Abstract. In areas prone to floods, accurate prediction and alert systems are extremely essential to help to minimize the consequences of natural occurrences. Examining past meteorological data since 1979, this work proposes a flood alert system using Deep Learning (DL) and Internet of Things (IoT), based on Combining the OpenWeatherMap Application Programming Interface (API) with an Encoder-Decoder Long Short-Term Memory (LSTM) model and alerting to possible flood hazards depending on pre-defined thresholds, the system offers real-time weather updates for users' locations, rainfall forecasts for the next 120 hours. Using a hardware CPU, the Raspberry Pi serves as the model server and manages seamless interactions between the model and APIs so allowing the propagation of flood risk notifications. Matunga East, Byculla, and Dahisar were three models developed for various areas of Mumbai given their high flood risk, heavy population, and national financial center importance. One can extend this approach to other sites as well. The suggested system computes the nearest model using the Haversine formula based on user distance from three sites. This geographical proximity-based method ensures that the offered flood prediction fits the specific rainfall threshold for flooding of the specified area.

INTRODUCTION

In places prone to erratic rainfall and flooding, effective prediction and alert systems are very vital to help to reduce the detrimental effects of such natural catastrophes. Accurate and timely flood forecasting is therefore very important for disaster readiness and response activities since climate change aggravates irregular weather patterns that adversely harm infrastructure, agriculture, and people [21], [22]. Even with advances in meteorological research, traditional forecasting methods occasionally fail to offer early warnings, thereby demanding innovative techniques [23].

The challenge of improving flood prediction and alert systems in the face of increasing climate variability has motivated researchers to explore hybrid and AI-assisted models that combine statistical and deep learning frameworks [24], [25]. Since deep learning, a subset of artificial intelligence (AI), has shown promise in identifying complex patterns in large datasets, it is suited for assessing historical weather data and expected trends. Moreover, the integration of IoT devices provides real-time environmental condition monitoring, thereby boosting the speed and accuracy of flood risk estimations.

A main novelty of this work is the implementation of a data-assisted flood alert system generating timely forecasts using Deep Learning models using Deep Learning models. Unlike traditional forecasting methods, which rely on simplistic statistical models or deterministic algorithms, the proposed methodology derives important insights from massive volumes of historical weather data using the complexity of Deep Learning architectures. Moreover, IoT devices such as the Raspberry Pi help to offer perfect connection between the Deep Learning model and real-time meteorological APIs, so facilitating fast distribution of flood risk warnings to end users.

This paper shows the design and implementation of a new flood prediction and alert system leveraging IoT technology and Deep Learning capability, following earlier exploratory work in AI-driven hazard detection systems [26], [27]. This work uses real-time IoT device data and previous meteorological data to offer better timeliness in forecasting flood disasters. Our system provides end-user timely warnings of flood risk by means of Deep Learning models housed on Raspberry Pi devices and integration of real-time meteorological APIs, therefore supplementing catastrophic preparedness and response activities. The method presented here marks a significant advancement in addressing the problems resulting from erratic rainfall and flooding in vulnerable regions, therefore helping to preserve life and infrastructure against the consequences of climate variability.

LITERATURE SURVEY

Proposed multiple times are LSTM-based models for rainfall prediction and forecasting in coastal and regional environments. Nithyashri et al. [1] reported an IoT-based IoT-based method employing deep reinforcement learning and LSTM networks with 89% accuracy and outperforming existing methods for rainfall forecasting in coastal parts of India. With 97.14% accuracy in rainfall prediction for Bangladesh, Billah et al. [2] surpassed conventional methods using a two-layer LSTM model with Principal Component Analysis (PCA). Using neural networks generating the lowest error (Mean Absolute Error (MAE)=2.20) and best R2 score (0.94), Tiwari and Singh [3] examined many machine learning algorithms for rainfall prediction in Indian states. Rani et al. [4] proposed an IoT-based flood monitoring and alerting system whereby Convolutional Neural Network (CNN) beat Linear Regression and Support Vector Machine (SVM) including rainfall prediction.

LSTM networks have helped time series forecasting in many different domains greatly. Lin et al. conducted transformer-based models in Long-term Time Series Forecasting (LTSF) accuracy better than To minimize recurrent iterations in Recurrent Neural Networks (RNNs), [5] introduced Segment Recurrent Neural Network (SegRNN), which introduces segment-wise iterations and concurrent multi-step forecasting, so reaching significant runtime and memory reductions. Wen and Li [6] shown an LSTM-attention-LSTM model that outperformed conventional models over multiple datasets by showing the efficacy of the attention mechanism in capturing inter-dependencies between time steps. Zhang et al. [7] presented a new encoder-decoder architecture obtaining greater prediction accuracy than corresponding models using Gated Recurrent Unit (GRU) and attention for multivariate time series forecasting.

Some study have looked at hybrid designs and variants of LSTM networks in order to improve performance. Albeladi et al. [8] tested LSTM with Autoregressive Integrated Moving Average (ARIMA) models for time series forecasting to discover ARIMA outperforming LSTM based on assessment criteria including R-squared, Mean Squared Error (MSE), and Mean Absolute Percentage Error (MAPE). Wang et al. [9] proposed Fuzzy Inference-based Long Short-Term Memory (FI-LSTM) by adding fuzzy inference systems into LSTM, which more successfully captures long-term dependencies and non-linear patterns than state-of- the-art models across many datasets. Lindemann et al. [10] addressed the partially conditioned Sequence-to- Sequence (Seq2Seq) LSTM for modeling both short-term and long-term dependencies, classifying LSTM designs for time series prediction into ideal cell state representations and interacting cell states, and so discussed.

LSTM networks have been particularly successful in flood prediction and hydrology applications. Zenkner and Navarro-Martinez [11] developed Bi-directional Long Short-Term Memory (Bi-LSTM) models for temperature, humidity, and wind speed forecasting in London, achieving Root Mean Squared Error (RMSE) of 1.45°C for 24-hour temperature predictions. Li et al. [12] proposed the Synced Sequence Input and Output (SSIO) LSTM architecture for rainfall-runoff modeling, outperforming Sequence Input and a Single Output (SISO) LSTM in prediction accuracy and capturing long-term dependencies without fixed window sizes. Prabuddhi and Seneviratne [13] utilized LSTM for water level and flood prediction in the Deduru Oya river basin in Sri Lanka, with a hybrid LSTM+ARIMA approach achieving an RMSE of 0.0247 and R2 score of 0.8901. Gude et al. [14] used LSTM for river gauge height prediction with uncertainty intervals, outperforming ARIMA and achieving better accuracy with the data sub-selection method for uncertainty estimation.

Several studies have explored the performance of LSTM and its variants for financial time series forecasting. Yadav et al. [15] compared stateful and stateless LSTM models for stock price prediction in the Indian stock market, recommending a stateless LSTM with a single hidden layer for most time series prediction problems. Siami-Namini et al. [16] found that BiLSTM outperformed ARIMA and LSTM models for stock index forecasting, with an average RMSE reduction of 93.11% and 37.78%, respectively, as the bi-directional training in BiLSTM captures additional features.

The literature also covers interpretability and hybrid approaches in time series forecasting with deep learning. Lim and Zohren [17] surveyed encoder and decoder designs, iterative and direct methods for multi-horizon forecasting, and interpretability techniques, suggesting that hybrid models combining statistical and deep learning approaches can outperform pure methods. Varghese and Vanitha [18] performed time series analysis on monthly rainfall data from

Kerala, India, using Statistical Package for the Social Sciences (SPSS) to extract trends and predict future rainfall values using regression coefficients, identifying significant variations in rainfall patterns across districts and seasons. Widiasari et al. [19] compared LSTM and multiple linear regression for flood prediction, with LSTM achieving lower error (3.6% MAPE) due to its ability to handle complex temporal data dependencies better. Marino et al. [20] investigated LSTM-based architectures for building energy load forecasting, with the LSTM-based Seq2Seq architecture excelling at handling variable sequence lengths.

From the reviewed literature, LSTM and its variants consistently emerge as strong candidates for time series prediction tasks related to rainfall and flood forecasting. For instance, BiLSTM models such as those used by Siami-Namini et al. [16] and Zenkner and Navarro-Martinez [11] demonstrate superior performance in capturing bidirectional temporal dependencies. Hybrid models integrating LSTM with statistical methods (e.g., ARIMA) or fuzzy logic [8], [9] further improve forecasting accuracy by modeling both linear and nonlinear trends. While ARIMA-based models may outperform LSTM in certain structured datasets [8], deep learning models show greater adaptability in complex and noisy real-world data [14], [19]. Moreover, the encoder-decoder and attention-based architectures [6], [7] provide enhanced long-range temporal forecasting. In contrast to these approaches, our proposed method distinguishes itself through integration with real-time IoT data and dynamic deployment via Raspberry Pi, offering not only accurate predictions but also practical on-ground alerting capabilities. This places our system closer to real-time operational flood management tools, a gap not fully addressed in many of the reviewed works.

METHODOLOGY

The dataset under analysis encompasses hourly weather observations for Matunga East, Dahisar, and Byculla, commencing from January 1, 1979 to February 8, 2024. It consists of 404,424, 395,400, and 402,082 records across 27 columns respectively, incorporating diverse meteorological parameters such as hourly rainfall, temperature, visibility, dew point, wind speed, and humidity, among others. The range of meteorological parameters used in this study is shown in Figure 1, illustrating the data columns available for analysis.

```
Data columns (total 27 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   dt                   404424 non-null  int64
 1   timezone             404424 non-null  int64
 2   city_name            404424 non-null  object
 3   lat                  404424 non-null  float64
 4   lon                  404424 non-null  float64
 5   temp                 404424 non-null  float64
 6   visibility           254500 non-null  float64
 7   dew_point            404424 non-null  float64
 8   feels_like           404424 non-null  float64
 9   temp_min             404424 non-null  float64
 10  temp_max             404424 non-null  float64
 11  pressure             404424 non-null  int64
 12  sea_level            0 non-null       float64
 13  grnd_level           0 non-null       float64
 14  humidity             404424 non-null  int64
 15  wind_speed           404424 non-null  float64
 16  wind_deg             404424 non-null  int64
 17  wind_gust            12601 non-null   float64
 18  rain_1h              69258 non-null   float64
 19  rain_3h              2290 non-null    float64
...
 25  weather_description  404424 non-null  object
 26  weather_icon         404424 non-null  object
```

FIGURE. 1 *DATA COLUMNS IN THE DATASET*

Categorical variables such as "weather_main" and "weather_description" characterize the basic atmospheric conditions and offer interesting study of the ambient environment around the received meteorological data. Two instruments that can help to extract insights, identify trends, and create forecasts depending on the meteorological data are deep learning models and time-series analysis. By means of validation and consistency checks, missing values and assurances of data quality should be addressed thereby preserving the integrity and dependability of the dataset across the analytical process.

DEEP LEARNING MODEL DEVELOPMENT

The technique is driven by the development of a Univariate Multi-Step Encoder-Decoder LSTM (Long Short-Term Memory) model for rainfall prediction. First starting with data preparation, the development loads historical rainfall data for all three areas and imports required libraries. These well chosen datasets remove extraneous time data, then convert into a datetime format prior to indexing.

Simple imputation methods—where zeros replace missing values in the dataset—as indicated by the dataset source—address missing values within the dataset. After treatment of missing entries, the datasets are placed under strict inspection for structure to ensure data integrity. Following that, using the hourly rainfall observations, one single column—known as "rain_1h"—is extracted from the databases for more investigation. General image of the features and temporal trends of the rainfall data is given by visualizations and statistical summaries. Year-wise rainfall trends visualized over the 40-year period are depicted in Figure 2, highlighting the seasonality and extremes captured in the dataset.

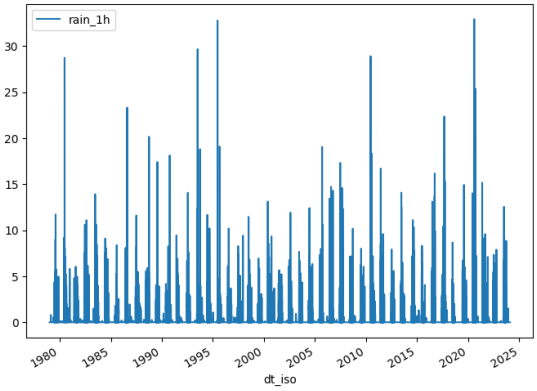

FIGURE. 2 *YEARLY RAINFALLS ACROSS 40 YEARS*

The datasets are partitioned into training, testing, and validation sets, with considerations for temporal dynamics, specifically focusing on monsoon months to mitigate data imbalance. The training set constitutes 80% of the data, while the testing and validation sets each encompass 10%. The process of data splitting ensures the preservation of temporal order and assists in evaluating model generalization.

Next, the datasets undergo pre-processing steps tailored for LSTM model compatibility. This includes scaling of features using MinMaxScaler to constrain values within a specified range, thereby facilitating model convergence and stability during training. The pre-processing pipeline extends to windowing the data, thereby transforming the sequential rainfall observations into input-output pairs suitable for training the LSTM model.

The LSTM architecture employed is structured as an Encoder-Decoder model, adept at capturing temporal dependencies within the data. The model's design incorporates encoder and decoder components, facilitating sequential processing and generation of multi-step forecasts. Hyperparameters, such as the number of hidden units and sequence lengths, are meticulously chosen to balance model complexity and predictive performance.

Model training commences with the compilation of the LSTM model, utilizing the Huber loss function and the Adam optimizer to minimize prediction errors. The training process is monitored using appropriate callbacks, including model checkpointing and early stopping mechanisms to prevent overfitting and ensure model robustness. The Encoder-Decoder LSTM model used for multi-step rainfall forecasting is shown in Figure 3.

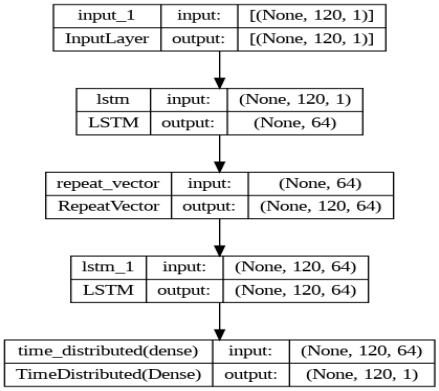

FIGURE. 3 *MODEL ARCHITECTURE*

Evaluation is conducted on the testing set, involving the assessment of model performance metrics. Post-evaluation, the trained model undergoes testing on unseen data to validate its predictive capabilities in real-world scenarios.

MOBILE APPLICATION DEVELOPMENT

The application uses several significant technologies and strategies to achieve its objectives. First it takes use of the Flutter framework and Dart programming language to enable single codebase building of cross-platform mobile apps. Through enabling efficient development and deployment on several platforms, this approach increases accessibility and usability for end users.

Geo-location services are added into the application using the Geolocator package therefore granting access to the device's location data. Once the user's coordinates are established, the program could find out whether the user is near designated flooding-sensitive locations including Matunga East, Dahisar, and Byculla.

Using the Haversine function of an angle, θ, the distances between the user's position and the approved flood-risk zones are calculated. This mathematical approach exactly finds the shortest distance between two locations on a sphere, so providing correct geographical measurements needed for flood prediction. The formula is as follows:

$$haversine\ (\theta) = \ sin^2\left(\frac{\theta}{2}\right) \tag{1}$$

After locating the user and estimating the distances to the specified flood-prone sites, the application begins API calls to acquire pertinent data for flood prediction. Important information about user proximity to each location-based flood risk assessment and forecast rainfall levels comes from these API endpoints.

The application's user interface (UI) is supposed to provide basic information access together with required tools. Using the Google Maps Flutter plugin, the UI displays a map including pre-defined flood-sensitive sites marked by user position. The UI also dynamically adjusts to display projected rainfall for the next several days and real-time flood projections. A snapshot of the mobile application interface with geo-located flood alert markers is presented in Figure 4.

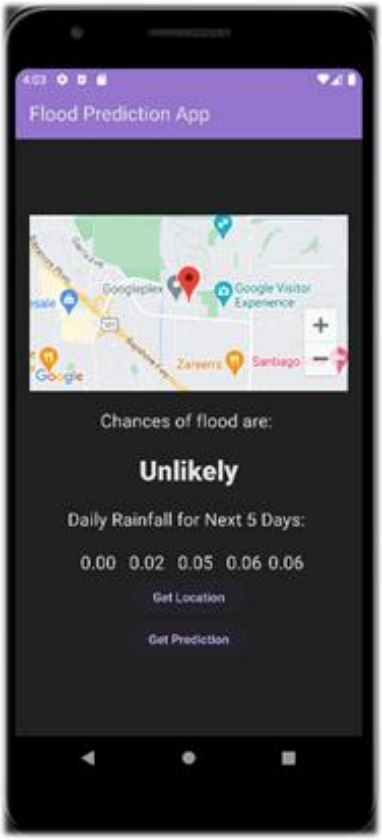

FIGURE. 4 *FLOOD ALERT MOBILE APPLICATION*

The software interaction process from location detection to flood alert delivery is depicted in Figure 5. Error handling mechanisms are implemented to ensure robustness and reliability in API interactions and location services. In case of API call failures or location service issues, the application displays informative toast messages to alert users and maintain transparency regarding potential data discrepancies or service unavailability. The software interaction process from location detection to flood alert delivery is depicted in Figure 5.

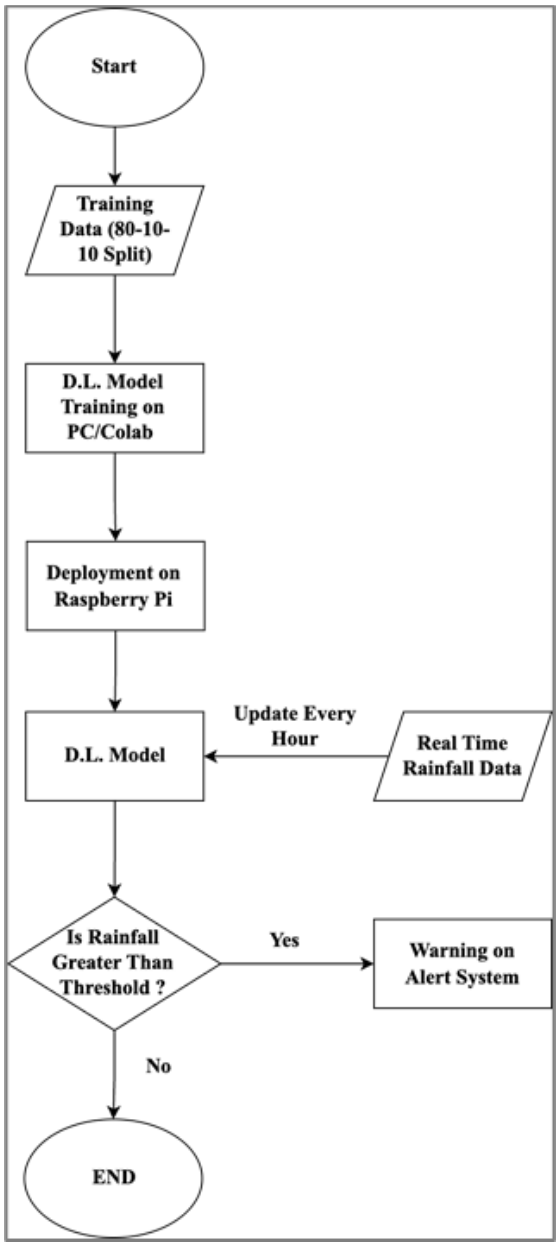

FIGURE. 5 *SOFTWARE WORKFLOW*

DEPLOYMENT OF FLOOD ALERT SYSTEM

The block diagram of the proposed system, illustrating data flow from APIs to end-user alerts, is shown in Figure 6. Methodically installed deep learning models housed on a Raspberry Pi guarantee flawless IoT system functioning and integration. The process begins initially with configuring and setting up the Raspberry Pi hardware to assure appropriate operating system (Raspbian) connection installation. Since the models rely on obtaining real-time weather data from the OpenWeatherMap API, Wi-Fi provides a constant internet connection. Once the hardware was

set up, dependant program installation took front stage. Along with the required libraries—NumPy, Requests, TensorFlow Lite, and FastAPI—this featured pip-based Python installation. To ensure fit and accessibility for IoT ecosystem deployment, Python scripts for data collecting and API development also found place on the Raspberry Pi.

Next the Raspberry Pi housed the pre-trained TFLite models. These models were designed using inference on resource- constrained IoT devices to ensure efficient system functioning. The FastAPI tool then was configured as the model prediction interface. This included outlining API endpoints for the intended weather forecasting sites, Matunga East, Byculla, and Dahisar.

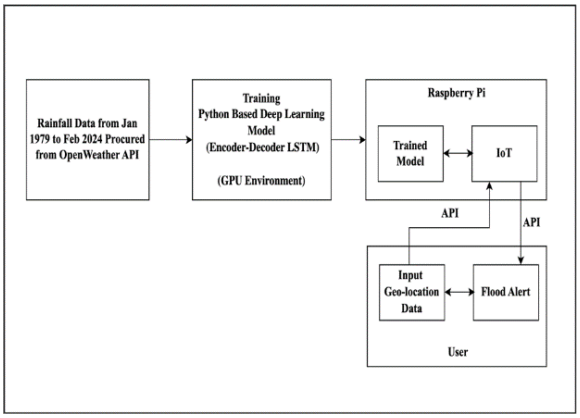

FIGURE. 6 *BLOCK DIAGRAM OF THE PROPOSED SYSTEM*

Along the inference process, the Raspberry Pi receives requests providing latitude and longitude coordinates reflecting the target location for rainfall prediction. Following the request, the Raspberry Pi looks for current weather information for the specified location on the OpenWeatherMap API. This data consists of variables including hourly rainfall, or rain_1h. After that, the obtained data is pre-processed to prepare it for usage as input to the ideal constructed model from an IoT framework.

Pre-processing consists in transforming the data into the appropriate form and scaling it with pre-trained scalers thereby ensuring fit with the models. TensorFlow Lite models loads the data for inference once pre-processed. Using the given data, the models generate forecasts of rainfall intensity over the given 120-hour period.

Following the inference, the model produces important understanding by means of post-processing activities. The forecasts are aggregated to determine daily overall precipitation. Moreover, defined are thresholds to determine whether the anticipated precipitation beyond the recommended level suggestive of flood risk.

Then, sent to the client as JSON-formatted responses via the FastAPI endpoints inside the IoT ecosystem, are the inference results including flood risk assessments and expected rainfall intensities. Clients can access this information by looking for the correct API endpoint with latitude and longitude coordinates corresponding to their desired location.

Moreover, constant data collection and improvements define the reliability of the system inside the IoT ecosystem. Running constantly on the Raspberry Pi, the data collecting script periodically changes the weather data, therefore ensuring the models have access to the most current data for prediction.

RESULTS AND DISCUSSION

Performance evaluation of prediction models determines their dependability and efficiency most of the time for useful applications. This paper explores the prediction performance of three models named Matunga East, Dahisar, and Byculla using significant criteria including Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE as shown in Table 1). These steps coupled with qualitative remarks provide insightful examination of the factors influencing the performance of every model, therefore providing a full knowledge of their predictive ability and their implications for decision-making processes.

Table 1. *Evaluation of models based on metrics*

Model	MAE	MSE	RMSE
Matunga East	0.1725	0.6520	0.8074
Dahisar	0.1709	0.5261	0.7253
Byculla	0.1853	0.6757	0.8220

With a decent MAE of 0.1725, the model linked with Matunga East displays pretty accurate average predictions. Conversely, a higher MSE of 0.6520 and RMSE of 0.8074 suggest that noise or outliers in the dataset affects the general model performance even if individual mistakes might not be significant. More data points mean more mistakes. This suggests prospective challenges in effectively spotting basic trends in ever rising data volume.

The Dahisar model shows the lowest MAE among the three by means of values of 0.1709, thereby indicating improved accuracy in target variable prediction. This model exhibits a relatively low MSE (0.5261) and RMSE (0.7253), which indicates an efficient performance considering the lowest quantity of non-zero values.

By comparison, the Byculla model reveals more errors than Matunga East and Dahisar. With an MAE of 0.1853, MSE of 0.6757, and RMSE of 0.8220 this model produces generally less accurate projections. Rather less training and validation data-points compromise the model's performance.

Analyzing daily rainfall test results for three separate locations—Byculla, Dahisar, and Matunga East—showcases considerable variations in both actual and forecast values across the recorded period. With few spikes pointing to substantial precipitation occurrences, the actual rainfall data in Byculla (Figure. 7) reveal irregular swings from minimum to significant. Conversely, the expected values quite closely correspond with the actual data, therefore demonstrating the capacity of the prediction model to reflect the temporal trends of the rainfall in this area.

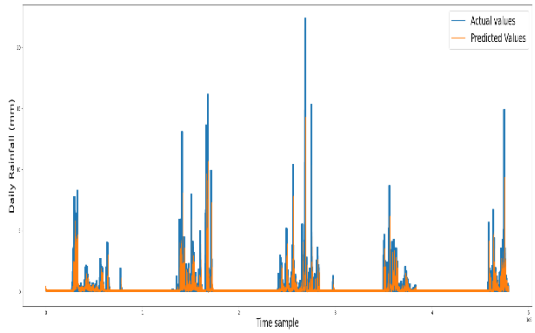

FIGURE. 7 *BYCULLA TIME SERIES PLOT ACTUAL VS PREDICTED*

Although on a considerably smaller scale, Dahisar's daily rainfall patterns (Figure. 8) exhibit comparable chaotic behavior as those of Byculla. The actual rainfall figures indicate regular increases, implying intermittent strong rain events combined with low intervals of precipitation. The expected values almost match the actual data, therefore verifying the model's accuracy in forecasts of Dahisar's rainfall occurrences.

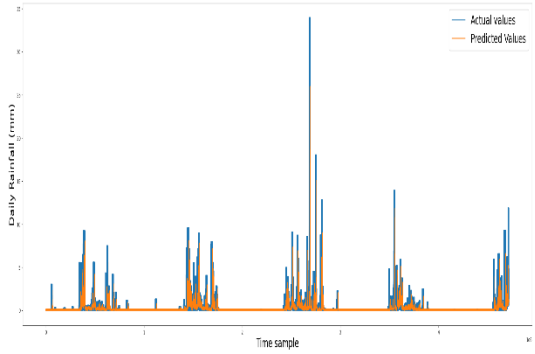

FIGURE. 8 *DAHISAR TIME SERIES PLOT ACTUAL VS PREDICTED*

The daily rainfall dynamics in Matunga East (Figure. 9) exhibit a different pattern from Byculla and Dahisar in more subdued oscillations. Both actual and anticipated values show significantly less variation, suggesting in this region a more consistent precipitation pattern. Still, occasional spikes in the actual rainfall data point to the presence of significant rainfall events, which the forecast model reasonably notes depending on the alignment between actual and projected values.

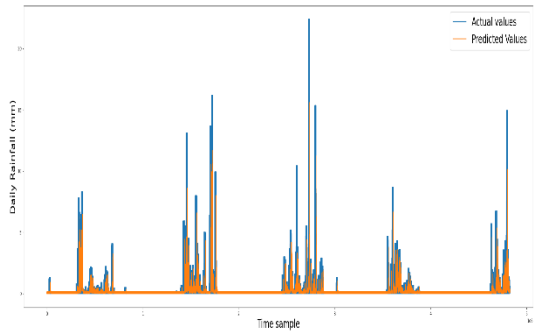

FIGURE. 9 *MATUNGA EAST TIME SERIES PLOT ACTUAL VS PREDICTED*

The study stresses overall the variability in daily rainfall patterns among the three places since the forecast model sufficiently captures the temporal dynamics in every site. These findings emphasize the significance of accurate rainfall prediction models in understanding and minimizing the consequences of climatic variability and extreme weather events on metropolitan surroundings.

CONCLUSION

This work demonstrates how extremely successfully Encoder-Decoder LSTM models change early warning systems for flood prediction. LSTM design has made effective rainfall pattern prediction possible, therefore enhancing the capacity to predict and lower the risks associated with flooding events. Including a classification system based on pre-defined thresholds inside the framework not only helps to foresee quickly but also separates between prospective flood threats and normal conditions. This function strengthens the practical utility of the recommended system by providing more actionable insights for campaigns of catastrophe readiness and response. The integration of the models with a user-friendly flood alert system serves to underline the accessibility and real-time responsiveness of the suggested solution. By means of IoT capabilities on Raspberry Pi devices and linking rainfall estimates to users' locations, the system provides timely distribution of emergency alerts, therefore mitigating the impact of probable floods on sensitive areas. There are several ways the flood prediction and early warning system might be improved. Important areas of focus are geographic expansion to other flood-prone sites worldwide, optimization of system speed and efficiency using cache memory techniques, and use of computational resources for training larger and more sophisticated models. Moreover offering opportunities for improved accuracy, scalability, and worldwide accessibility are lowering prediction thresholds by means of ground truth data integration and moving toward cloud-based and web-based systems. This study essentially underlines the critical requirement of proactive efforts in building resilience and readiness against natural calamities as well as assists flood prediction technologies to evolve. By means of continuous development and growth of this strategy, it is hoped to enable communities all around to better forecast, react to, and minimize the consequences of flooding events, therefore saving lives and safeguarding livelihoods.

ACKNOWLEDGMENTS

The authors would like to thank Mr. Abhimanyu Chauhan, Scientist-C, IMD, Geo-spatial Scientist, Meteorologist for his guidance throughout this project.

REFERENCES

1. J. Nithyashri et al., "IoT based prediction of rainfall forecast in coastal regions using deep reinforcement model," Measurement: Sensors, vol. 29, 2023, Art. no. 100877. [Online]. Available: https://doi.org/10.1016/j.measen.2023.100877
2. M. Billah et al., "Rainfall prediction system for Bangladesh using long short-term memory," Open Computer Science, vol. 12, no. 1, pp. 323–331, 2022. [Online]. Available: https://doi.org/10.1515/comp-2022-0254
3. N. Tiwari and A. Singh, "A novel study of rainfall in the Indian states and predictive analysis using machine learning algorithms," in Proc. Int. Conf. Comput. Perform. Eval. (ComPE), Shillong, India, 2020, pp. 199–204. doi: 10.1109/ComPE49325.2020.9200091.
4. D. S. Rani, G. N. Jayalakshmi and V. P. Baligar, "Low-cost IoT based flood monitoring system using machine learning and neural networks: Flood alerting and rainfall prediction," in Proc. Int. Conf. Innovative Mechanisms

for Industry Applications (ICIMIA), Bengaluru, India, 2020, pp. 261–267. doi: 10.1109/ICIMIA48430.2020.9074928.
5. S. Lin et al., "SegRNN: Segment recurrent neural network for long-term time series forecasting," arXiv preprint, arXiv:2302.02079, 2023.
6. X. Wen and W. Li, "Time series prediction based on LSTM-Attention-LSTM model," IEEE Access, vol. 11, pp. 48322–48331, 2023. doi: 10.1109/ACCESS.2023.3276628.
7. H. Zhang et al., "A novel encoder-decoder model for multivariate time series forecasting," Computational Intelligence and Neuroscience, vol. 2022, Art. ID 5596676, 2022. [Online]. Available: https://doi.org/10.1155/2022/5596676
8. K. Albeladi, B. Zafar and A. Mueen, "Time series forecasting using LSTM and ARIMA," Int. J. Adv. Comput. Sci. Appl. (IJACSA), vol. 14, no. 1, pp. 112–119, 2023. doi: 10.14569/IJACSA.2023.0140133.
9. W. Wang, J. Shao and H. Jumahong, "Fuzzy inference-based LSTM for long-term time series prediction," Scientific Reports, vol. 13, Art. no. 20359, 2023. doi: 10.1038/s41598-023-47812-3.
10. B. Lindemann et al., "A survey on long short-term memory networks for time series prediction," Procedia CIRP, vol. 99, pp. 650–655, 2021. doi: 10.1016/j.procir.2021.03.080.
11. G. Zenkner and S. Navarro-Martinez, "A flexible and lightweight deep learning weather forecasting model," Applied Intelligence, vol. 53, pp. 24991–25002, 2023. doi: 10.1007/s10489-023-04824-w.
12. W. Li, A. Kiaghadi and C. Dawson, "Exploring the best sequence LSTM modeling architecture for flood prediction," Neural Comput. & Applic., vol. 33, pp. 5571–5580, 2021. doi: 10.1007/s00521-020-05334-3.
13. W. A. M. Prabuddhi and B. L. D. Seneviratne, "Long short-term memory modelling approach for flood prediction: An application in Deduru Oya Basin of Sri Lanka," in Proc. Int. Conf. Advances in ICT for Emerging Regions (ICTer), Colombo, Sri Lanka, 2020, pp. 226–231. doi: 10.1109/ICTer51097.2020.9325438.
14. V. Gude, S. Corns and S. Long, "Flood prediction and uncertainty estimation using deep learning," Water, vol. 12, no. 3, Art. no. 884, 2020. doi: 10.3390/w12030884.
15. A. Yadav, C. K. Jha and A. Sharan, "Optimizing LSTM for time series prediction in Indian stock market," Procedia Computer Science, vol. 167, pp. 2091–2100, 2020. doi: 10.1016/j.procs.2020.03.257.
16. S. Siami-Namini, N. Tavakoli and A. S. Namin, "The performance of LSTM and BiLSTM in forecasting time series," in Proc. IEEE Int. Conf. Big Data, Los Angeles, CA, USA, 2019, pp. 3285–3292. doi: 10.1109/BigData47090.2019.9005997.
17. B. Lim and S. Zohren, "Time-series forecasting with deep learning: A survey," Philosophical Transactions of the Royal Society A, vol. 379, no. 2194, 2021. doi: 10.1098/rsta.2020.0209.
18. L. R. Varghese and K. Vanitha, "A time-series based prediction analysis of rainfall detection," in Proc. Int. Conf. Inventive Computation Technologies (ICICT), Coimbatore, India, 2020, pp. 513–518. doi: 10.1109/ICICT48043.2020.9112488.
19. I. R. Widiasari et al., "Context-based hydrology time series data for a flood prediction model using LSTM," in Proc. Int. Conf. Information Technology, Computer, and Electrical Engineering (ICITACEE), Semarang, Indonesia, 2018, pp. 385–390. doi: 10.1109/ICITACEE.2018.8576900.
20. D. L. Marino, K. Amarasinghe and M. Manic, "Building energy load forecasting using deep neural networks," in Proc. IEEE Annu. Conf. Ind. Electron. Soc. (IECON), Florence, Italy, 2016, pp. 7046–7051. doi: 10.1109/IECON.2016.7793413.
21. M. P. Brown and K. Austin, The New Physique, Publisher Name, Publisher City, 2005, pp. 25–30.
22. M. P. Brown and K. Austin, "Title of article," Appl. Phys. Lett., vol. 85, pp. 2503–2504, 2004.
23. R. T. Wang, “Title of chapter,” in Classic Physiques, R. B. Hamil, Ed. Publisher Name, Publisher City, 1999, pp. 212–213.
24. C. D. Smith and E. F. Jones, "Load-cycling in cubic press," in Shock Compression of Condensed Matter–2001, AIP Conf. Proc. 620, M. D. Furnish et al., Eds. Melville, NY: AIP Publishing, 2002, pp. 651–654.
25. B. R. Jackson and T. Pitman, "Load cycling improvement technique," U.S. Patent 6,345,224, Jul. 8, 2004.
26. D. L. Davids, “Recovery effects in binary aluminum alloys,” Ph.D. dissertation, Harvard Univ., 1998.
27. R. C. Mikkelson, Private communication, 2024.