Journal of Computer Science

Research Article

Enhancing Dam Safety and Management: Long Short-Term Memory Based Predictive Models for Accurate Alert Forecasting

Nisha C. M and N. Thangarasu

Department of Computer Science, Karpagam Academy of Higher Education, Coimbatore-21, India

Article history
Received: 31-05-2025
Revised: 27-07-2025
Accepted: 16-08-2025

Corresponding Author:
Nisha C. M
Department of Computer science, Karpagam academy of higher education, Coimbatore-21, India
Email: nishacm2008@gmail.com

Abstract: Dam management and early alert systems are critical for effective water resource management. Accurate prediction of dam alert signals facilitates proactive decision-making, thereby aiding in the effective management and reduction of potential risks linked to dam operations. Within this research, Long Short-Term Memory (LSTM) networks are utilized to forecast dam alert signals issued from the dam by leveraging daily parameters, including temperature, dew point, humidity, and other pertinent factors. The study utilizes a dataset of the Malampuzha Dam spanning 10 years, comprising various inputs and the corresponding alert levels. Our objective is to demonstrate the effectiveness of LSTM models in accurately predicting multi-level alert classifications. This is the first application of LSTM for multi-tiered dam alert classification in the Indian context. The LSTM model was trained using optimizers such as Adam, RMSProp, Stochastic Gradient Descent, Adagrad, and Nadam, using learning rates of 0.01, 0.001, and 0.0001, as well as epochs of 50, 100, and 500, and gradient clipping values of 0.5 and 1.0. Evaluation metrics including RMSE (Root mean square error), NSE (Nash-sutcliffe Efficiency), R-squared, and accuracy are employed to assess the model's performance. The LSTM model using the Nadam optimizer achieved high accuracy (99.13%). It was also observed that as the learning rate decreased, the model's accuracy decreased. An appropriate gradient clipping value is found to be 0.5 for the LSTM model.

Keywords: Alert Prediction, Dam Management, Long Short-Term Memory, Nadam Optimizer, Classification Report, Confusion Matrix

Introduction

There are various disasters such as earthquakes, landslides, droughts, floods, and tsunamis, which are unavoidable but cause havoc to human life and property. However, several disasters can be predicted in advance and the severity of destruction can be reduced. Flooding is a significant natural disaster with severe impacts, making long-term flood forecasting crucial for risk management and early warning (Khairudin et al., 2022). Advancing deep learning in water management requires addressing challenges like data privacy, algorithm development, and trustworthiness, with the goal of achieving highly intelligent and autonomous urban water systems (Fu et al., 2022). Dams play a pivotal role in controlling floods and droughts, and effective dam management helps the authorities make better decisions. Dams serve various purposes, including flood control, agricultural use, hydropower generation, water storage for drinking and industrial purposes, and drought mitigation. The application of machine learning and deep learning models has been instrumental in tackling a diverse range of challenges associated with dams, including Reservoir inflow prediction (Banihabib et al., 2020), Reservoir capacity prediction (Dai et al., 2022), Reservoir scheduling problem in hydropower generation (Tang et al., 2022a), streamflow forecasting over reservoir catchment (Liu et al., 2022) and Seepage prediction (Ishfaque et al., 2022).

Issuing dam alerts, even a day or just hours in advance, will undoubtedly assist authorities in taking proactive measures to safeguard the public and prevent disasters when dams are opened. Human-assisted decision-making

in issuing alert signals to the public based on various meteorological parameters may not always be accurate. If adequate warning is not provided prior to the sudden opening of the dam, it could pose a threat to public safety**.** Also, the wrongful issuance of alerts when opening the dam may cause inconvenience to the public as well and property. We have reviewed several studies and it was found that no papers discussed the issue of alert signals generated from the dams in adverse situations.

Related literature like alerts generation in river embankments is studied. The safe operation of water conservation projects and the early detection of any hazards depend on the efficient monitoring of seepage in river embankments and the timely issuance of intelligent alerts as discussed by Shao et al. (2024). A levee seepage intelligent alarm system built on a Bidirectional Long Short-Term Memory (BILSTM) network model was created and put into place to improve the intelligence of seepage alerts and levee safety monitoring. The study conducted at Occidental Mindoro, Philippines by Adrian et al. (2024) discusses the generation of warning alerts based on water levels. A long-range, global system for mobile communication module, an Arduino Uno microcontroller, water level sensors, and temperature-humidity sensors make up the designed flood alarm system. Upon activation and detection of the water level, the sensors will transmit an alert message to the Global System for Mobile communications module, which will then transmit flood alarm messages to the receiver. The response time for these messages should not exceed ten seconds. But the alert generated only gives instantaneous warning and the people does not get enough time to take precautionary measures and artificial intelligence means is not applied in it. Various hardware components like water flow sensor, pressure sensor, ultrasonic sensor, temperature sensor and software like Ardino IDE were implemented by Thirumarai Selvi et al. (2024) for analyzing flood monitoring system. This is more of a hardware-oriented system.

The focus of this research paper is on predicting dam alert levels by utilizing Long Short-Term Memory (LSTM) models, incorporating meteorological and dam-related parameters. Meteorological parameters like rainfall, temperature, humidity, atmospheric pressure, wind speed can cause heavy floods when the levels of the parameters go beyond a certain limit and directly affect the dam operations. Furthermore, dam-related parameters, such as a dam's storage capacity, rainfall in the dam's catchment area, cumulative previous-day rainfall, and water outflow for purposes like irrigation, drinking, and power generation, play a crucial role in determining whether to open the dam's shutters or issue alerts to the public. This paper is an attempt to study the use of one deep learning model, namely LSTM in generating accurate alert signals from the dam by training, and dam related parameters. Different aspects of the LSTM model are studied in the research to find the best optimal solution in the forecasting of the alert signals from the dam. This study presents the first application of LSTM for multi-tier dam alert classification in the Indian context.

Optimizers are used to minimize the discrepancy between the intended output and the existing output; this discrepancy serves as a response indication that determines how much the optimizer should adapt as discussed by Jain et al. (2023). By updating weights appropriately for each input, the network contributes to a reduction in loss. This process continues when the training loop and iteration are repeated (also known as epochs), and each epoch produces an updated weight value that is correct and helps to minimize the loss. The goal is to minimize it to almost zero. LSTM's various hyper parameters are discussed by Kwon et al. (2023) like batch size, dropout rate, learning rate, number of epochs, number of nodes, number of hidden layers, and sequence length. The length of the sequence dictates how much time is spent for learning data at any given moment; a node is crucial in differentiating the features of input patterns; and the dropout rate stops over fitting by arbitrarily removing some of the complete nodes during learning. The amount to be learned at once is determined by the learning rate, the batch size is the amount of data at a time, and an epoch is one complete pass through the entire training dataset. In this paper, we train the model using different optimizers such as Adam, Adagrad, RMSprop, Stochastic Gradient Descent, and Nadam, with the hyper parameters being the learning rate and gradient clipping value. The main objective of the paper is to dive deeply into the LSTM model and to find the best optimizer, learning rate and gradient clipping value so that the LSTM model contributes the best results in predicting the alert signals.

The subsequent sections of the research paper are structured as follows: Literature review section provides an overview of dam-related research conducted by other scholars, exploring diverse machine learning and deep learning models. Materials and method section focuses on the case study of the Malampuzha dam, detailing the data collection process, and includes an in-depth discussion on Long Short-Term Memory (LSTM) and the methodology implemented in this study. Result section and discussion section explain the results obtained from the LSTM model with different epochs, optimizers, learning rates, and gradient clipping values. Lastly, the conclusion section elaborates the overall objectives and results.

Literature Review

The effective use of LSTM in various dam-related studies and flood forecasting system is studied in this section. An LSTM model was effectively used by Le et al. (2019) for flood prediction using daily release and

precipitation at the Hoa Binh station on the Da river situated in Vietnam. The LSTM model effectively predicted precise results in testing and validation phases within the first, second and third days of flood prediction using daily discharge and rainfall. A real-time flood forecasting system was constructed with LSTM and it outperformed conceptual based XAJ model and RNN model across three catchments in China under moist, semi-moist and semi-arid conditions (Liu et al., 2020). Also, the coupled LSTM-KNN model produced better results than single LSTM model while using the evaluation matrices RMSE (Root mean square error), R^2 (coefficient of determination), NSE (Nash Sutcliffe Efficiency) coefficient and VE (Volume error). The success of predicting daily stream flow for the study of hydroelectric turbine efficiency was demonstrated using an LSTM recurrent neural network model, as evidenced in a prior study (Le et al., 2019). The dataset for this investigation was derived from the flow history of the Jirau Hydroelectric Power Plant, located on the Madeira River in Brazil. Nine different architectures were evaluated in the study and fifty LSTM units were considered best prediction with best values of RMSE, MAE and R^2.

Flood forecasting based on temperature and rainfall intensity was studied by Sankaranarayanan et al. (2020) and the research work was carried out on the dataset from 1990 to 2002.The case study was conducted on data of 10 districts of two Indian states Orissa and Bihar. A deep neural network was compared with support vector machine, K-nearest neighbor and naïve Bayes and the studies showed deep neural network performed with better accuracy. Decisions of reservoir management and water pre-release was evaluated in Zarei et al. (2021) with different algorithms like Support vector machine, Regression tree, Genetic Programming and Artificial Neural Network at Dez, Gotvand and Karkheh reservoirs in Iran. The study showed that Support vector machine and Regression tree performed with better accuracy on one month and two months time lag patterns. For predicting water inflow into Zayandehroud dam situated in Iran, Artificial neural network and Support vector machine were used by Babaei et al. (2019) with a variety of nine different input data patterns. The first seven data inputs were monthly inflow into the reservoir with different time lags, the eighth one being time index and the last one consisting of the rainfall at Ghaleh-Ghahrokh station to different monthly time lags. The superior alternative was SVM compared to the ANN model with better values of RMSE and R-squared at training, validation and test processes.

Babaei et al. (2020) introduced the Long Short-Term Memory-Flash Flood framework, designed specifically for the forecasting of flash floods, demonstrating strong performance with qualified rates above 82.7% for peak discharge, 89.3% for peak time, and 84.0% for flood process at lead times of 1–10 hours. It excels in simulating large flood events and highlights the importance of small flood events in model training. The research suggests potential for further integration of hydrological knowledge and addressing rainfall forecast uncertainty in flash flood predictions. A hybrid approach combining the Unscented Kalman Filter (UKF) and a recurrent neural network (NARX model) effectively reduces predictive uncertainty in flood forecasting (Zhou et al., 2020). The recurrent neural network (NARX model) outperforms the static neural network (BPNN model) in producing accurate and stable flood forecasts with reduced time-lag effects.

Kunverji et al. (2021) addressed the critical need for an effective flood prediction system in light of the devastating impact of floods in recent years. In the pursuit of enhancing prediction accuracy, especially for intricate datasets, the study compares a Decision Tree Model with other machine learning algorithms like Random Forest and Gradient Boost, exploring avenues for improved performance. The system is designed with Indian conditions in mind, aiming to provide timely flood warnings to residents and aid in cost-effective government response, including evacuation operations. The study done by Xu et al. (2023) presented a hybrid flood prediction model integrating a Light GBM (Light gradient Boosting machine) learning model with a hydrological-hydraulic model, showing superior performance in predicting inundation depth. It identifies tide levels as the dominant variable. The Light GBM model outperforms other methods and is particularly efficient, offering potential for decision-making in flood mitigation.

The study demonstrates that the developed LSTM model outperforms the standard LSTM for predicting Huanggang Reservoir (Fujian, China) capacity, over 7-day, 14-day, and 30-day periods, with accuracy depending on parameter settings (Dai et al., 2022). It highlights the importance of selecting relevant influencing factors. The paper discusses LSTM and GRU models to predict water levels at the Hangang Bridge Station in South Korea, improving accuracy by analyzing the correlation between water levels and selected hydrological and meteorological data (Park et al., 2022). The results showed that GRU outperformed LSTM in predicting high-water levels, especially when using multivariate input data. This approach is valuable for urban rivers with rapid water level fluctuations, emphasizing the significance of multivariate models for accurate predictions in such scenarios.

Dam displacement prediction is studied by Zhang et al. (2019) on Dongjiang arch dam displacement data and an improved LSTM model was compared with the conventional LSTM, MLP, MLR, BRT and SVM models.

However, only two parameters, water level and temperature, are considered in the study. Wei et al. (2020) considered a combination of Particle Swarm Optimization–Support Vector Machine model, SVM model, and independent regression models for dam displacement prediction, and the combined model reduces the prediction error.

From these studies, we can see that various deep learning models were utilized for general flood forecasting and dam related researches like water inflow calculation, water level generation, dam displacement etc. However, the studies related to the chances of opening the dam based on multiple weather parameters combined with dam related parameters, have not been taken into consideration. The novelty of this research is to check the suitability of LSTM model in predicting the alert levels (No alert/Blue alert/Orange alert/Red alert) from the dam without any human intervention so that people may aware of the dangerous situation of opening the dam in utmost situations. The aim of this research paper is not to compare different machine learning or deep learning models for forecasting alert signals, but to ponder deep into the LSTM model to find the suitable hyper parameters for prediction. This study focuses on identifying optimal hyper parameter values, such as different optimizers, learning rates, and gradient clipping values, suitable for the LSTM model from a different perspective, so that the model can predict alert generation with maximum accuracy.

Materials and Methods

Case Study

The dam chosen for research is the Malampuzha dam as location seen in Figure 1, is the largest reservoir in the Kerala state in India situated on the foot hills of Western Ghats. The Malampuzha dam is around 23.13 square kilometer in area and nearly 10 Km from Palakkad town. It is a multi-purpose concrete gravity dam built on the Malampuzha River, a tributary of the Bharathapuzha River. The Malampuzha dam was constructed in 1955 for the purpose of catering the needs of agriculture, drinking, power generation, Industries and fish farming. Malampuzha is a good case study for time-series modeling since it has decades' worth of documented rainfall, inflow, and storage data. For comparable medium-sized reservoirs, LSTM and other AI models created with Malampuzha data can be used as prototypes.

The control and operation of the Malampuzha Dam involve the monitoring and adjustment of several parameters to ensure efficient water management and safety. The main aim of the study is to issue alert signals from the dam, hours or days in advance to the public so that they can mitigate the flood situation caused by the dam openings.

Fig. 1: Location of the Malampuzha dam

No Alert and Blue Alert indicate safe conditions; Orange Alert signals a warning, and Red Alert represents the most dangerous situation. The dam may be opened at any moment after a Red Alert is issued. If sensors are used, the dangerous situation can only be detected when it occurs. Therefore, the main objective is to issue alerts at least hours or days in advance, so that the authorities and the public can take precautionary measures. The various parameters used for efficient management are reservoir level, maximum storage capacity of the reservoir, inflow of water into the reservoir, outflow of water from the reservoir and giving appropriate alerts (no alert/blue/orange/red) to the public. First warning alert is given as a blue alert at 113meters, second alert is given as an orange alert at 114 meters and third warning is given as red alert at a dam water level of 114.46 meters. The dam related data (from 1st January 2010 to 31st December 2020) are collected from the irrigation department of the Malampuzha division. Various meteorological parameters like temperature, precipitation, atmospheric pressure, wind speed, humidity and dew point directly affects the formation of flood situations and dam management. The daily data for these parameters in the Palakkad region, for the specified time period, were collected from tcktcktck.org, a non-profit organization that provides weather information for over 250.

The first step in creating a model consists of collecting data of relevant parameters and it was acquired from the above sources. We combined the meteorological parameters and dam related parameters into one .csv file. All data was converted into numerical form so that processing is done smoothly. We ensured that labels are correctly represented and do not contain any NaN or infinite values. The next step after data collection is loading and pre-processing of data to required format of the model. Here 14 columns of meteorological and dam related parameters are taken as input variables and alert level consisting values of no alert, blue alert, orange alert and red alert as target label. Loading the dataset included the features as input variables and the target labels. The features and labels separated from the dataset are normalized into a common scale to improve the convergence and performance of the neural network. The alert labels are encoded into numerical values (0, 1, 2, 3) to be used as the target labels for training the model. The encoded labels represent the four alert levels: 0 for No

alert, 1 for blue alert, 2 for orange alert, and 3 for red alert where red alert is the most dangerous one issued. The height of the Malampuzha dam is 115.06 meters and the first warning is issued as blue alert at 113 meters water level of the dam which is considered to be relatively safe. The second warning is issued as orange alert at a height of 114 meters which is a situation just before the dangerous situation. The third and last alert is issued as the red alert at 114.46 meters height which is a very dangerous situation and the dam can be opened at any moment next. Usually, the alert signals are generated from the dam when the water levels in the dam reach at the above discussed levels. The main goal of the study is to predict the possibility of the alert signals by artificial intelligence means so that the forecasting is accurate and in advance.

The Long short- term Memory (LSTM)

The Long Short-Term Memory (LSTM) model as depicted in Figure 2, is a type of Recurrent Neural Network (RNN) that is particularly well-suited for processing and making predictions with sequences of data. The cell state (*Ct*) represents the retention of the network and the input gate decides which information to be stored in it. The forget gate controls what data from the cell state must be discarded or retained. The hidden state (*Ht*) is the output of the LSTM unit which is passed to next stage and the output gate regulates what information is shown as the output. The forget gate (f_t) employs the sigmoid function to assess the extent to which it should discard prior information from the prior cell state (C_{t-1}) built on a combination of the former unseen state (h_{t-1}) and the present input data (x_t) (Kwon et al., 2023).

The forget gate is represented by:

$$ft = \sigma\,(Wf[ht-1, xt] + bf) \quad (1)$$

The input gate it, employs a sigmoid function to regulate the update value:

$$it = \sigma\,(Wi\,[ht-1, xt] + bi) \quad (2)$$

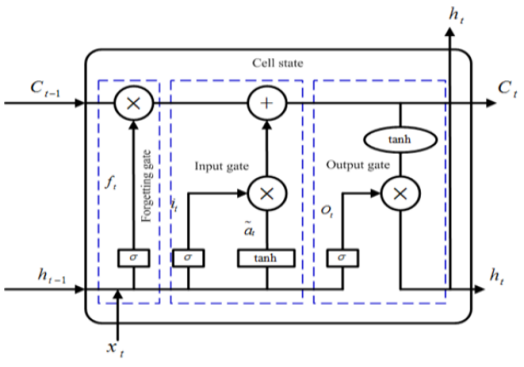


Fig. 2: LSTM architecture (Wang et al., 2022)

Subsequently, a hyperbolic tangent (tanh) layer operates as the second part, generating a vector of new candidate values. The sigmoid function (σ) adjusts weights from both the prior hidden state (h_t−1) and the current input data (x_t) to calculate its value. Similarly, the hyperbolic tangent (tanh) is used to create the candidate cell ($C\tilde{}t$) for updating the new cell, considers adjustments in the weights of the preceding hidden state (h_{t-1}) and the current input (x_t):

$$C\tilde{}t = tan\,h(Wc[ht-1, xt] + bc) \quad (3)$$

The update of the present cell state (C_t) involves a combination of the prior cell state (C_{t-1}) and the candidate cell ($C\tilde{}t$):

$$Ct = ft * Ct-1 + it * C\tilde{}t \quad (4)$$

The sigmoid function, through a weight adjustment of both the previous hidden state (h_{t-1}) and the current input data (x_t), determines the amount of output to be exported from the cell in the form of the output gate (O_t):

$$Ot = \sigma\,(Wo\,[ht-1, xt] + bo) \quad (5)$$

By employing the hyperbolic tangent function, the challenges associated with vanishing and exploding gradients during the update of a particular point state (h_t) can be resolved:

$$ht = ot * \tanh(Ct) \quad (6)$$

The LSTM architecture used in this study consists of a single LSTM layer with 100 units followed by a dense output layer with 4 units (representing the four alert classes) and a softmax activation for multi-class classification. The LSTM layer uses the ReLU activation function to introduce non-linearity. This simple yet effective architecture is well-suited for capturing temporal patterns in the dam's multivariate time-series data.

Methods

An LSTM model with simple architecture is experimented with for the prediction of alert levels. Data pre-processing is the first step to be followed while creating the proposed LSTM model. For that, the input data is reshaped into 3D format (samples, time steps, features) so that it is compatible with the LSTM model architecture. Here, samples represent the number of sequences or samples we are providing for training in each batch and 4014 rows were given to the model. In an LSTM, we typically provide a sequence of data as input and the entire dataset is given as one time step to our model. The number of features is the number of columns

(variables) in our CSV file, and each feature will be treated as a separate input to the LSTM. Since fourteen parameters(temperature in degree Celsius, Dew point in Degree Celsius, Humidity in percentage, Wind speed in Kph, atmospheric pressure in Hg, precipitation in mm, water level of dam in meter, storage of water in dam in Mm3 ,Rain fall(of that day) in mm, Total rainfall (Up to that day) in mm, Discharge of water from dam at LBC in Mm3, Discharge of water from dam at RBC in Mm3, Discharge of Water from dam at spillway in Mm3, total discharge of water from dam in Mm3 and inflow of water to dam in Mm3) are used as input to the model ,the number of features is taken as 14.The target label taken is the alert levels issued from the dam. The target labels are converted from “no alert”, “blue alert”, “orange alert” and “red alert” to numerical values of 0, 1, 2 and 3 respectively. The formatted data is split for training and for testing to evaluate its performance on unseen data. The train test_split function randomly splits the dataset into two parts, and the parameter test size = 0.2 means that 20% of the dataset is set aside for testing, while 80% is used for training. The dataset contained 4,014 rows and 17 columns, of which 3,212 rows were used for training and the remaining 802 rows, were used for testing which is randomly picked. Figure 3 portrays the different stages in the LSTM model.

In the LSTM model for alert prediction, several preprocessing steps are essential to prepare the dataset effectively. First, the dataset is loaded using pandas, and the features (such as temperature, humidity, water level, rainfall, inflow, etc.) are separated from the target labels, which indicate whether a blue, orange, or red alert was given. One of the critical preprocessing steps applied is normalization, where each feature is scaled by subtracting its mean and dividing by its standard deviation. This standardization ensures that all input features have a similar scale, which improves the learning efficiency and convergence of the LSTM model.

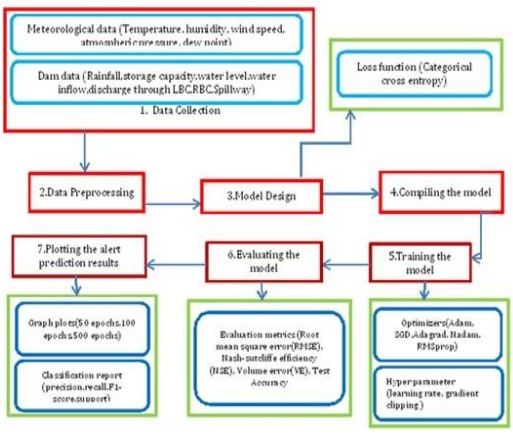


Fig. 3: LSTM Model stages

Missing values can be handled by dropping rows with missing values or by imputing those using statistical methods such as mean or median filling. Finally, since LSTM models expect input data in three dimensions (samples, timesteps, features), the data must be reshaped accordingly. In our dataset, since each row represents a single day's data (i.e., one timestep), an additional dimension is added to reshape the input to meet the required format. These preprocessing steps help ensure the data is clean, scaled appropriately, and structured correctly for effective sequence modelling ing LSTM.

The LSTM architecture design is the next step after pre-processing the data. An LSTM layer with 100 units and ReLU activation function is added. The LSTM model’s accuracy highly depends on the quantity of neurons in the hidden layers (Dai et al., 2022). The input shape is specified based on the reshaped data. A dense output layer with a softmax activation function is used to predict one of the four classes of no alert, blue, orange and red alerts.

Multiple optimizers like Adam, Nadam, RMSprop, SGD, and Adagrad were used to evaluate how different gradient descent strategies affect the learning behaviour of the LSTM model. Adam and Nadam combine momentum and adaptive learning rates, often performing well on complex and noisy datasets. RMSprop is effective in handling non-stationary objectives, common in time-series data like dam alerts. SGD and Adagrad, though simpler, serve as useful benchmarks and sometimes generalize better in certain scenarios, helping to validate the robustness of the model.

The optimizers are implemented with gradient clipping to prevent exploding gradients which can lead to NaN loss. Gradient clipping values of 0.1, 0.5 and 1.0 are chosen for experimentation along with the optimizers chosen. The next stage after LSTM model design is, compilation with suitable loss function. The loss function used for training the model is sparse categorical cross entropy which is suitable for multi class classification and target variables which are mutually exclusive. After compiling the LSTM model, it is trained with the training dataset. The model is trained using the encoded alert labels (y train and y test) as target labels for the training and testing the dataset. The next step is evaluating the model using different evaluation metrics which is discussed in the next section. The results are plotted with the number of epochs on the X-axis and accuracy percentage on the y-axis for different optimizers. The confusion matrix is also generated for LSTM model with different learning rates. Also, a classification report is generated for plotting results.

Results

We trained the LSTM model with different optimizers like Adam, Adagrad, RMSprop, SGD (Stochastic Gradient

Descent) and Nadam with three different epochs (50 epochs, 100 epochs and 500 epochs). The results of LSTM models with different optimizers and epochs are shown in the tables and in the graph, which clearly indicates that the model improved with increased accuracy as the number of epochs increased. We evaluated the model using the testing data and calculated the evaluation metrics (RMSE, NSE, VE). Root-Mean-Square Error (RMSE) is the ratio of the square deviation between the observed value and its true value, and the number of observations (Tang et al., 2022b). A model with RMSE value approaching 0 is considered to be very effective in predictions. Nash-Sutcliffe Efficiency (NSE) gives the extent of model's ability to predict the variables dissimilar from the mean and a value nearing to 1 shows that the model is predicting the results accurately (Liu et al., 2020). The paper also discusses Volume Error (VE), a key to calculate the model performance and a value approaching 0 is considered to be very near the observed data.

When evaluating the model, we used the original labels (y_test) to calculate the evaluation metrics like RMSE, NSE, R-squared, and VE. The numerical predictions from the model (y_pred_labels) are then compared with these original labels for evaluation. The gradient clipping value is considered here as 0.5 to avoid exploding gradient value problem for all the optimizers which can be experimented with different values for better results.

LSTM With Adam Optimizer

The Adam (Adaptive Moment Estimation) optimizer is a popular optimization algorithm which combines techniques from both Adagrad and RMSProp to optimize the learning process (Soydaner, 2020). Firstly, initialization take place by maintaining several moving average estimators of gradients and squared gradients, which are initialized to zero. The main parameters used are, model parameters to be optimized(θ), learning rate which controls the step size in parameter updates(α), exponential decay rates ($\beta 1$ & $\beta 2$) for the moving averages of gradients and squared gradients, small constant (ϵ) to prevent division by zero. Moments are initialized with mt = 0 (Initial value for the first moment (mean) of gradients) and vt = 0 (Initial value for the second moment (un centered variance) of gradients).The update rules are discussed in Bock and Weis (2019), in which the gradient are calculated using a mini-batch of training examples:

$$gt = \nabla(f(x;\theta), y) \quad (7)$$

The first moment estimate are calculated using the following formula:

$$mt = \beta 1 mt - 1 + (1 - \beta 1) gt \quad (8)$$

The second moment estimate are calculated using the following formula:

$$vt = \beta 2 vt - (1 - \beta 2) gt 2 \quad (9)$$

Correct bias in first and second moments are computed by:

$$\hat{}mt = mt(1 - \beta 1t)\ vt = vt / (1 - \beta 2t) \quad (10)$$

parameters θ are updated using the corrected estimates:

$$\hat{}\theta t + 1 = \theta t - \alpha\ \hat{}mt / \sqrt{vt} + \varepsilon \quad (11)$$

The LSTM model with the adam optimizer is tested and the results are given in Table 1. To visualize the model's performance, graph plots are used as evaluation metrics as in Figure 4. The graph plot given above visualizes the proposed LSTM model of Adam optimizer with 50 epochs, 100 epochs and 500 epochs respectively. The accuracy level has increased in a consistent manner, and the model performed the prediction of alert levels with an accuracy of 98.63% over 500 epochs.

Table 1: Evaluation metrics of LSTM model with adam optimizer

	Adam with 50 epochs	Adam with 100 epochs	Adam with 500 epochs
RMSE	0.27	0.16	0.12
NSE	0.93	0.97	0.986
VE	0.073	0.04	0.018
Test Accuracy	95.52 %	97.38%	98.63%

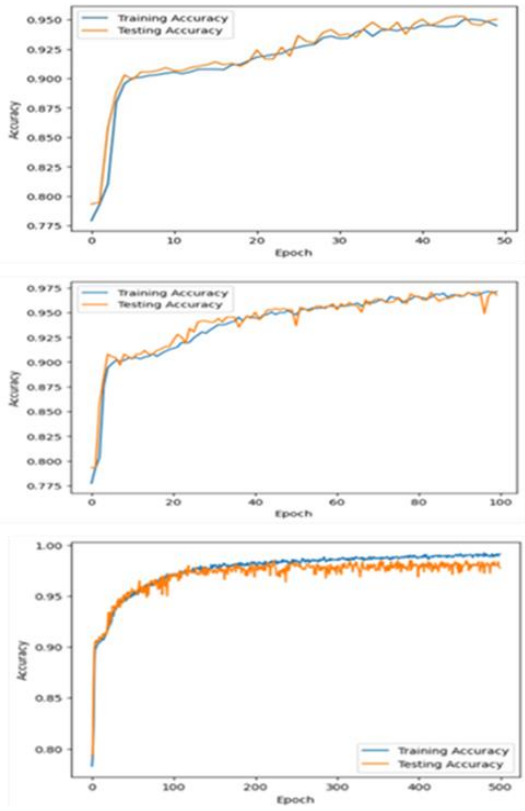


Fig. 4: Graph plots of LSTM model with adam optimizer with 50 epochs, 100 epochs and 500 epochs

LSTM With Adagrad Optimizer

The adagrad (adaptive gradient algorithm) is an optimization algorithm that adjusts learning rates for each parameter based on their historical gradients, providing smaller updates for parameters with frequent changes and larger updates for parameters with significant changes in results. Unlike other optimizers which use a uniform learning rate for all parameters, Adagrad's adaptive learning rates are well-suited for dealing with sparse data by identifying and assigning different rates to essential parameters, thereby enhancing stochastic gradient descent efficiency (Halgamuge et al., 2020). The equation of the Adagrad optimizer is:

$$\theta t+1,i=\theta t,i-n/Gt,i+\varepsilon.gt,I \quad (12)$$

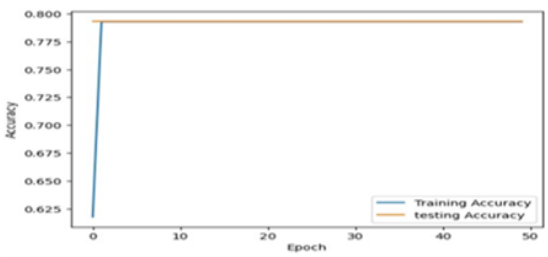
where $\theta t,i$ is the parameter i at time step t, η is the learning rate Gt, i is the accumulated sum of squared gradients for parameter i up to time step t and ϵ is a small constant to avoid division by zero. The outcome of the

LSTM model implemented with the Adagrad optimizer is given in Table 2.

The RMSE values provided (1.11, 1.11, 0.59) suggest that the model's predictive error decreased significantly from 1.11 to 0.59 as the number of epochs increased from 50 to 500. The provided NSE values (-0.219, -0.219, 0.66) suggest that the model initially performed worse than the mean (negative NSE), but as the number of epochs increased, the model's performance improved significantly as in Figure 5, achieving a positive value (0.66) indicating that it outperformed the mean. The provided VE values (0.63, 0.63, 0.23) remain relatively stable across the 50 and 100 epochs, and then slightly decreases at 500 epochs.

Table 2: Evaluation metrics of LSTM model with adagrad optimizer

Metrics	Adagrad with 50 epochs	Adagrad with 100 epochs	Adagrad with 500 epochs
RMSE	1.11	1.11	0.59
NSE	-0.219	-0.219	0.66
VE	0.63	0.63	0.23
Test Accuracy	79.32%	79.32%	89.29%



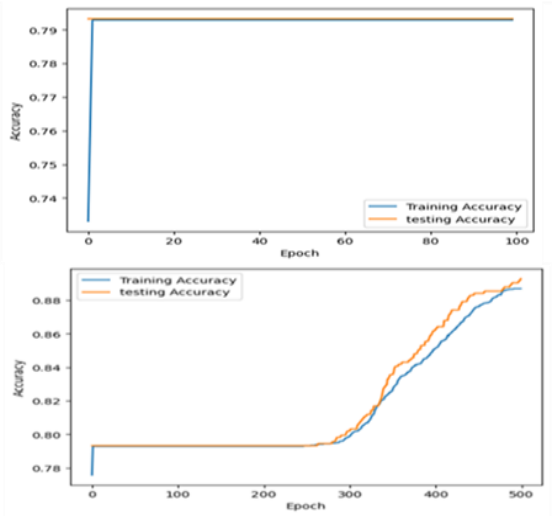


Fig. 5: Graph plots of LSTM model with adagrad optimizer of 50 epochs, 100 epochs and 500 epochs

LSTM With RMSprop Optimizer

RMSProp (Root Mean Square Propagation) is crucial for effectively training neural networks with distance measures and Gaussian activation functions, especially when plain mini-batch gradient descent or momentum yield slow or no convergence, even for shallow networks (Kurbiel and Khaleghian, 2017).

The learning rate for each weight is adjusted by dividing it by this moving average, allowing for more efficient and stable training. The update equations for the parameter θij is:

$$\hat{v}ij = \gamma vij + (1-\gamma)(gijt)2 \quad (13)$$

$$\hat{\theta}ij = \theta ij - \eta vij + \epsilon \cdot gijt \quad (14)$$

The LSTM model is implemented with RMSprop optimizer and results obtained are given in Table 3.

Table 3: Evaluation metrics of LSTM model with RMSprop optimizer

Metrics	RMSprop with 50 epochs	RMSprop with 100 epochs	RMSprop with 500 epochs
RMSE	0.28	0.19	0.149
NSE	0.92	0.96	0.978
VE	0.08	0.03	
	0.04		
Test Accuracy	95.02%	97.76%	
	96.76%		

The decreasing trend in RMSE and VE, along with the increasing trend in NSE and test accuracy, indicates the model's refinement in predictive accuracy, as the epochs increase as shown in Figure 6. This suggests that the model learned more from the data and improved efficiency, and classification capabilities with additional training, its predictive and satisfactory performance significantly over time.

LSTM With SGD Optimizer

A key optimization technique for training models in deep learning and machine learning is stochastic gradient descent (SGD). This variation of gradient descent uses a smaller random selection of the data (mini-batch) to compute the gradient in order to update model parameters, as opposed to computing the gradient of the complete dataset (batch). By adding randomness to the parameter updates, this method can increase convergence and improve escape from local minima.

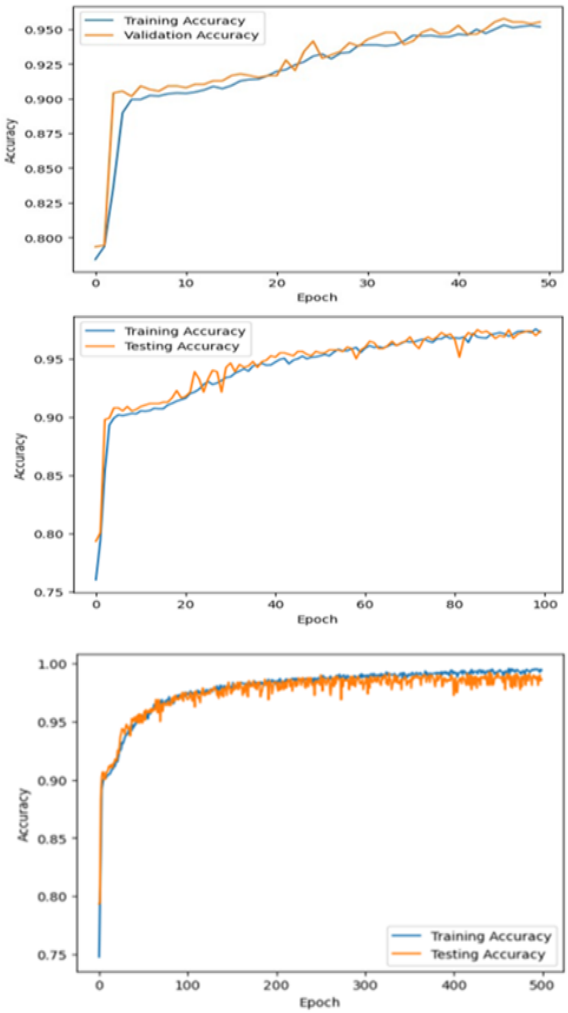


Fig. 6: Graph plots of LSTM model with RMSprop optimizer of 50 epochs, 100 epochs and 500 epochs

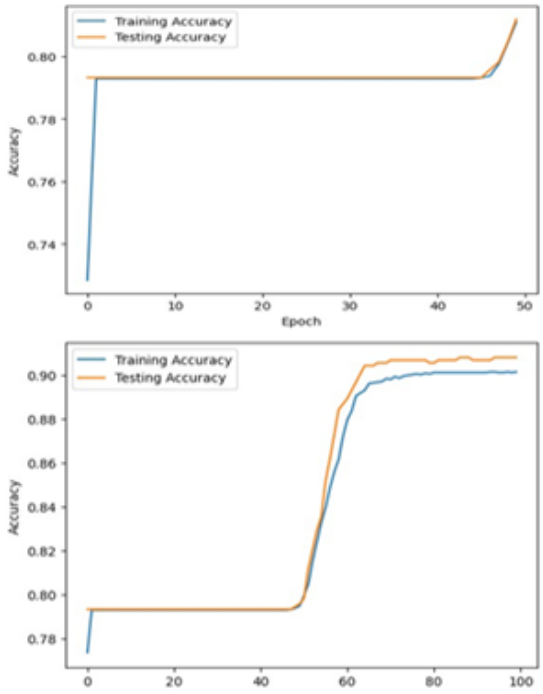
Soydaner (2020) discusses the theoretical working of the SGD. The initialization of SGD starts with an initial guess for the model parameters σ. Then the entire dataset is split into smaller mini-batches containing a subset of the training examples. Next, iterations are performed over each mini-batch to compute the gradient of the loss function with respect to the current mini-batch of examples, and the model parameters σ are updated using the computed gradient. The update rule for each parameter σ is:

$$\theta \leftarrow \theta\eta\nabla\theta L(f(x(i);\theta), y(i)) \quad (15)$$

Where, η (learning rate) controls the step size of the update. Then the last step is convergence in which the iterative process is repeated until convergence criteria are met. The LSTM model trained with the Stochastic Gradient Descent optimizer shows significant improvement in classification tasks as the number of training epoch increases as in Table 4 and Figure 7. The decreasing trend in RMSE, VE, along with the increasing trend in NSE, test accuracy, indicates that the model learned more from the data and improved its performance significantly through additional training epochs when using the SGD optimizer in forecasting alert signals.

Table 4: Evaluation metrics of LSTM model with SGD optimizers

Metrics	SGD with 50 epochs	SGD with 100 epochs	SGD with 500 epochs
RMSE	1.039	0.48	0.117
NSE	-0.068	0.77	0.99
VE	0.562	0.18	0.018
Test Accuracy	81.2%	90.78%	98.63%



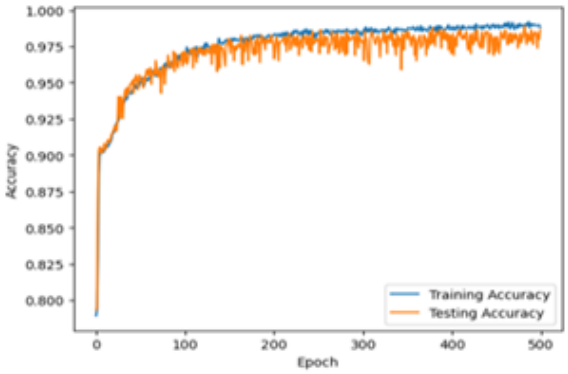


Fig. 7: Graph plots of LSTM model with SGD optimizer of 50 epochs, 100 epochs and 500 epochs

LSTM With Nadam Optimizer

Nesterov Accelerated Gradient (NAG) is applied to adam optimizer to generate Nadam (Nesterov accelerated adaptive moment estimation) optimizer (Wang and Cao, 2017) .The Nadam update equations are:

$$mt = \beta_1 m_{t-1} + (1 - \beta_1)_t \quad (16)$$

$$\tilde{v}t = \beta_2 v_{t-1} + (1 - \beta_2)_t^2 \quad (17)$$

Where is the gradient at time step t with respect to the parameters, and vt are the first and second moment estimates respectively, $\beta 1$ and $\beta 2$ are exponential decay rates and ϵ is a small constant.

The evaluation metrics provided in Table 5 are for an LSTM (Long Short-Term Memory) model trained using the Nadam optimizer across different numbers of epochs. Each metric serves as an indicator of the model's performance in classification tasks. The Nadam optimizer with 500 epochs achieved the highest accuracy of 99.13%. A steady improvement in accuracy can be observed in Figure 8.

LSTM Model With Different Learning Rates

After finding the best optimizer, the next hyper parameter chosen is the learning rate. Here, we have trained and evaluated the LSTM models with three different learning rates: 0.01, 0.001, and 0.0001. We selected the Nadam optimizer with 50 epochs and the evaluation metrics for each learning rate are presented in the table.

Apart from different types of optimizers, the number of epochs and the learning rate the LSTM model can be experimented with by changes in gradient clipping value of optimizer and types of architecture. The result in Table 6 clearly indicates that as the learning rate decreased the prediction accuracy of the model also decreased. The results indicate that the LSTM model accomplished the best enactment with a learning rate of 0.01, obtaining the lowest RMSE, MAE, and highest NSE and accuracy.

Table 5: Evaluation metrics of LSTM model with Nadam optimizer

Metrics	Nadam with 50 epochs	Nadam with 100 epochs	Nadam with500 epochs
RMSE	0.271	0.16	0.09
NSE	0.93	0.975	0.99
VE	0.07	0.033	0.012
Test Accuracy	95.52%	97.51%	99.13%

Table 6: Evaluation metrics of LSTM model with different learning rates

Learning rate	RMSE	MAE	NSE	Accuracy
0.01	0.17	0.029	0.97	97.13%
0.001	0.25	0.05	0.94	95.64%
0.0001	0.48	0.14	0.78	90.66%

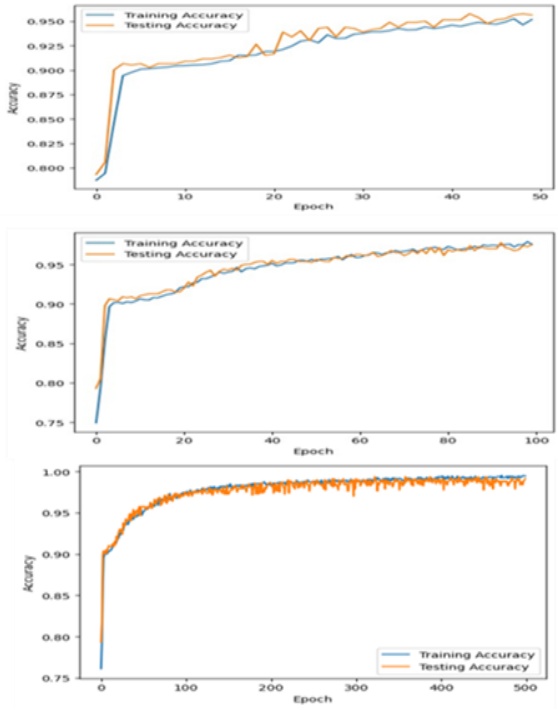


Fig. 8: Graph plots of LSTM model witsh Nadam optimizer of 50 epochs, 100 epochs and 500 epochs

The confusion matrices in Table 7 have four rows and four columns representing the alert levels 0 (No alert), 1 (Blue alert), 2 (Orange alert) and 3 (Red alert) respectively in each row and column. The diagonal elements of the confusion matrix depict the true positive cases for each alert level. The confusion matrix with learning rate 0.01 shows that 637 instances of No alert, 36 instances of blue alert, 14 instances of orange alert and 93 instances of red alert were predicted correctly. In the case of blue alerts, 7 instances were wrongly predicted as No alert and 5 instances were wrongly predicted as orange alert. Only 5 instances of orange alerts were wrongly

classified as blue alert and red alerts. We can see that only 1 instance of red alert is wrongly predicted as orange alert. The matrix of the LSTM model with different learning rates is given below:

	No alert ↓	Blue alert ↓	Orange alert ↓	Red alert ↓
No alert →	637	0	0	0
Blue alert →	7	36	5	0
Orange alert →	0	5	14	5
Red alert →	0	0	1	93

Table 7: Confusion matrix of LSTM model (optimizer = Nadam, epochs = 50)

Learning rate 0.01	Learning rate 0.001	Learning rate 0.0001
[[637 0 0 0] [7 36 5 0] [0 5 14 5] [0 0 1 93]]	[[636 1 0 0] [4 36 3 5] [0 10 4 10] [0 0 2 92]]	[[631 6 0 0] [14 10 0 24] [3 7 0 14] [1 6 0 87]]

In addition to the confusion matrix, classification report containing the values of precision, recall, F1- score and support can be used to determine the accuracy of the proposed LSTM model as indicated in Tables 8-10 respectively. The first four rows of the classification report show 0 for No alert, 1 for blue alert, 2 for orange alert and 3 for red alert. Precision and Recall is specified in Hinojosa et al. (2024) as described below:

$$\text{Recall} = \frac{TP}{(TP + FP)} \tag{18}$$

$$\text{Precision} = \frac{TP}{(TP+FN)} \tag{19}$$

Table 8: Classification report (learning rate = 0.01, Nadam optimizer)

	Precision	Recall	F1-Score	Support
0	0.99	1.00	0.99	637
1	0.88	0.75	0.81	48
2	0.70	0.58	0.64	24
3	0.95	0.99	0.97	94
accuracy			0.97	803
macro avg	0.88	0.83	0.85	803
weighted avg	0.97	0.97	0.97	803

Table 9: Classification report (learning rate = 0.001, Nadam optimizer)

	Precision	Recall	F1-Score	Support
0	0.99	1.00	1.00	637
1	0.77	0.75	0.76	48
2	0.44	0.17	0.24	24
3	0.86	0.98	0.92	94
accuracy			0.96	803
macro avg	0.77	0.72	0.73	803
weighted avg	0.95	0.96	0.95	803

Table 10: Classification report (learning rate = 0.0001, Nadam optimizer)

	precision	recall	f1-score	support
0	0.97	0.99	0.98	637
1	0.34	0.21	0.26	48
2	0	0	0	24
3	0.7	0.93	0.79	94
accuracy			0.91	803
macro avg	0.5	0.53	0.51	803
weighted avg	0.87	0.91	0.89	803

TP represents true positive instances, FN represents false negative instances, FP represents false positive instances and TN represents true negative instances. The F1-score and accuracy are explained in Ahmed et al. (2021) as given below:

$$\text{F1}-\text{score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{20}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{21}$$

The arithmetic mean of each class-specific assessment metric is known as the macro average. By accounting for the proportion of each class-specific sample size in the overall sample size, the weighted average improves upon the macro average (Jianan et al., 2024).

The classification reports show that higher learning rates (especially 0.01) with the Nadam optimizer yield the best performance, achieving high accuracy (97%) and strong precision, recall, and F1-scores across all alert levels, particularly for No alert (0) and red alert (3). At a moderate learning rate (0.001), performance slightly drops, especially for orange alerts (2), with recall falling to 17%. At the lowest learning rate (0.0001), the model struggles significantly, especially with Blue (1) and orange alerts (2), indicating under fitting. Thus, 0.01 is the optimal learning rate in this case for accurately predicting critical dam alerts.

LSTM Model With Different Gradient Clipping Value

The LSTM model is implemented using the SGD optimizer with different gradient clipping values. For 500 epochs, when the gradient clipping value changed from 0.5 to 0.1, the accuracy level reduced to 95.14% from 98.63%. Then the gradient clipping value changed to 1.0 with 500 epochs with SGD optimizer, the accuracy level was 94.77% which clearly indicates that the best option for gradient clipping value is 0.5 as shown in Table 11 and Figure 9.

Table 11: Evaluation metrics of LSTM model (SGD optimizer with different gradient clipping values)

	Gradient clipping value 0.5	Gradient clipping value 0.1	Gradient clipping value 1.0
RMSE	0.117	0.28	0.30
NSE	0.99	0.922	0.911
VE	0.018	0.078	0.087
Test Accuracy	98.63%	95.14%	94.77%

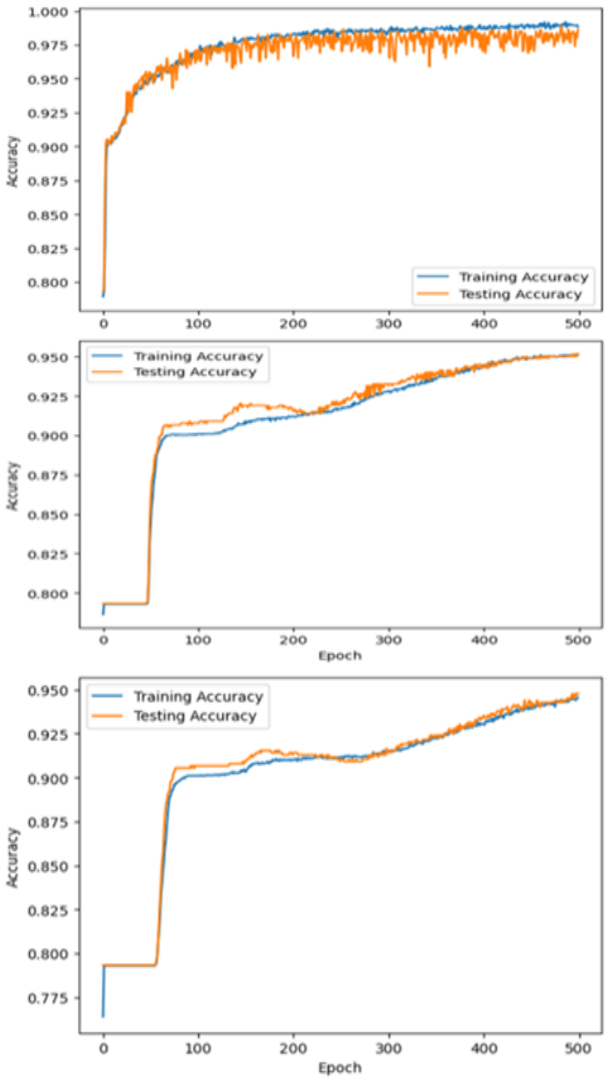


Fig. 9: Graph plots of LSTM model with SGD optimizer with different gradient clipping values

Discussion

The existing literature does not provide details about issuing alert signals from the dam based on various weather parameters and dam related parameters. The main intention of the alert signals is to give prior information about the chances of opening dam shutters and make them aware of the severity of dam related floods. This research employs a decade-long dataset from the Malampuzha Dam, encompassing multiple input parameters along with their corresponding alert categories (No Alert, Blue, Orange, and Red). The primary aim is to illustrate the capability of LSTM models in precisely forecasting multi-level alert classifications. To the best of our knowledge, this represents the first implementation of LSTM for multi-tier dam alert classification within the Indian setting. This paper takes data of fourteen parameters for the consecutive ten years to train and test the LSTM model as discussed in the methods section. The different alerts, No Alert, Blue Alert, Orange Alert, and Red Alert, are predicted using the LSTM model with various optimizers and hyper parameters. A red alert is a very dangerous situation, and it is issued just a few hours before the dam's release. Therefore, the number of red alert occurrences is lower compared to other alerts.

Different evaluation metrics like RMSE, NSE, VE and test accuracy are employed to find the best optimizer and hyper parameters. The results section clearly demonstrates that LSTM model with Nadam optimizer performs better with an accuracy of 99.13%, lowest RMSE (0.09), NSE (0.99), VE (0.012) with 500 epochs of training. When learning rate is taken 0.01, the model performed with higher accuracy compared to learning rates of 0.001 and 0.0001 and best gradient clipping value is 0.5 compared to gradient clipping values of 0.1 and 1.0. Classification reports and graph plots are effectively organized for comparative study.

Advance knowledge of dam shutter openings through alert levels can certainly help government authorities take effective steps to reduce the severity of flood-related damage. If the public is informed hours in advance, they can relocate movable property and household animals. The study is based on the dataset of the Malampuzha Dam and can certainly be applied to forecasting alert signals issued for other dams.

Limitations

One of the limitations of the model is that images or videos related to weather parameters are not included. It is relatively impossible to collect image or video data of meteorological parameters over many years to train the model. This raises the question of whether we can train the model and obtain accurate results. Temporal data leakage risk may happen when building or testing a temporal model, information from the future can influence the model's training phase. Further research with other deep learning models can be conducted to identify the most suitable model for alert prediction. Predictions can be shared with emergency control centers

via dashboards, mobile alerts, or public warning systems. If certain thresholds are exceeded, automatic alerts can be triggered to initiate protocols such as evacuation, public announcements, or alerts to first responders. The research can also be extended with IoT integration, enabling different government departments to take effective steps for disaster management.

Conclusion

Effective dam management is a crucial task, and improper management can affect the lives and properties of thousands of people, as well as the economy of the country as a whole. Before opening the dam, precautions are to be given to the public as alerts (no alert/blue/orange /red) so that people are aware of the dangerous situation and take preventive steps. We have constructed an LSTM model with 14 parameters to accurately predict dam openings by issuing alert signals, using 10 years of meteorological and dam-related daily data from the Malampuzha Dam, Palakkad. The main objective of the article is not to compare different deep learning models but to dive into the LSTM model to find the best solution in forecasting the alert signals from the dam. The LSTM models are implemented with different optimizers like adam, adagrad, SGD, RMSprop and Nadam. The LSTM models are also experimented on with different learning rates and gradient clipping values. The LSTM model with Nadam optimizer and 500 epochs showed remarkable accuracy over other four optimizers used. This paper dives deeply into the workings of the LSTM model, utilizing its architecture, hyper parameters, loss function, learning rates, and gradient clipping values to find the best option for efficient dam management. The results are estimated using the evaluation metrics RMSE, MSE, VE and test accuracy. The results are plotted using graph plots for different epochs. Classification reports are utilized to determine the accuracy of the models. Overall, the LSTM model with Nadam optimizer with learning rate 0.01 and gradient clipping value 0.5 is found to be a suitable choice for alert signal prediction in flood-like situations.

Acknowledgment

The authors would like to thank the Irrigation Department, Malampuzha for providing accessible to ten years dam related data. The authors would also like to thank tcktcktck.org for providing accessible to meteorological data.

Funding Information

No funding was received from any organizations to assist with the preparation of this manuscript.

Author's Contributions

Nisha C. M: Conceptualization, data curation, Investigation, methodology, Formal analysis, interpretation of result and writing -original draft.

N. Thangarasu: Conceptualization, Supervision, Validation and Writing-Review and edited.

Ethics

Ethical approval was not required for this study as it used anonymized operational data.

Data Availability Statement

Data cannot be made publicly available; readers should contact the corresponding author for details.

Conflict of Interest

The authors declare there is no conflict of interests.

References

Adrian, A. P. N., & Enriquez, M. D. (2024). Community-based flood alert system using long-range technology for Brgy. *San Agustin, San Jose, Occidental Mindoro. Mindoro Journal of Social Sciences and Development Studies (MJSSDS)*. 1(2), 27-34. https://journal.omsc.edu.ph/index.php/mjssds/article/view/10

Ahmed, M., Mumtaz, R., & Hassan Zaidi, S. M. (2021). Analysis of water quality indices and machine learning techniques for rating water pollution: a case study of Rawal Dam, Pakistan. *Water Supply*, *21*(6), 3225–3250. https://doi.org/10.2166/ws.2021.082

Babaei, M., Moeini, R., & Ehsanzadeh, E. (2019). Artificial Neural Network and Support Vector Machine Models for Inflow Prediction of Dam Reservoir (Case Study: Zayandehroud Dam Reservoir). *Water Resources Management*, *33*(6), 2203–2218. https://doi.org/10.1007/s11269-019-02252-5

Babaei, M. E., Bandari, R., & Valipour, M. (2020). Improving Daily Peak Flow Forecasts Using Hybrid Fourier-Series Autoregressive Integrated Moving Average and Recurrent Artificial Neural Network Models. *AI*, *1*(2), 263–275. https://doi.org/10.3390/ai1020017

Bock, S., & Weis, M. (2019). A Proof of Local Convergence for the Adam Optimizer. *2019 International Joint Conference on Neural Networks (IJCNN*, 1–8.

Dai, B., Wang, J., Gu, X., Xu, C., Yu, X., Zhang, H., Yuan, C., & Nie, W. (2022). Development of Modified LSTM Model for Reservoir Capacity Prediction in Huanggang Reservoir, Fujian, China. *Geofluids*, *2022*, 1–14. https://doi.org/10.1155/2022/2891029

Fu, G., Jin, Y., Sun, S., Yuan, Z., & Butler, D. (2022). The role of deep learning in urban water management: A critical review. *Water Research*, *223*, 118973. https://doi.org/10.1016/j.watres.2022.118973

Halgamuge, M. N., Daminda, E., & Nirmalathas, A. (2020). Best optimizer selection for predicting bushfire occurrences using deep learning. *Natural Hazards*, *103*(1), 845–860. https://doi.org/10.1007/s11069-020-04015-7

Hinojosa, L., M. C., Braet, J., & Springael, J. (2024). Performance Metrics for Multilabel Emotion Classification: Comparing Micro, Macro, and Weighted F1-Scores. *Applied Sciences*, *14*(21), 9863. https://doi.org/10.3390/app14219863

Ishfaque, M., Dai, Q., Haq, N. ul, Jadoon, K., Shahzad, S. M., & Janjuhah, H. T. (2022). Use of Recurrent Neural Network with Long Short-Term Memory for Seepage Prediction at Tarbela Dam, KP, Pakistan. *Energies*, *15*(9), 3123. https://doi.org/10.3390/en15093123

Jain, A. K., Rao, Dr. P., & Sharma, Dr. K. V. (2023). Pragmatic Assessment of Optimizers in Deep Learning. *International Journal of Computer Science and Network Security*, *23*(10), 115–128. https://doi.org/10.22937/IJCSNS.2023.23.10.15

Jianan, G., Kehao, R., & Binwei, G. (2024). Deep learning-based text knowledge classification for whole-process engineering consulting standards. *Journal of Engineering Research*, *12*(2), 61–71. https://doi.org/10.1016/j.jer.2023.07.011

Kunverji, K., Shah, K., & Shah, N. (2021). A Flood Prediction System Developed Using Various Machine Learning Algorithms. *SSRN Electronic Journal*, *6*, 1–6. https://doi.org/10.2139/ssrn.3866524

Kurbiel, T., & Khaleghian, S. (2017). Training of Deep Neural Networks based on Distance Measures using RMSProp. *ArXiv*, 1–6. https://doi.org/10.48550/arXiv.1708.01911

Kwon, Y., Cha, Y., Park, Y., & Lee, S. (2023). Assessing the impacts of dam/weir operation on streamflow predictions using LSTM across South Korea. *Scientific Reports*, *13*(1), 9296. https://doi.org/10.1038/s41598-023-36439-z

Le, X.-H., Ho, H. V., Lee, G., & Jung, S. (2019). Application of Long Short-Term Memory (LSTM) Neural Network for Flood Forecasting. *Water*, *11*(7), 1387. https://doi.org/10.3390/w11071387

Liu, J., Yuan, X., Zeng, J., Jiao, Y., Li, Y., Zhong, L., & Yao, L. (2022). Ensemble streamflow forecasting over a cascade reservoir catchment with integrated hydrometeorological modeling and machine learning. *Hydrology and Earth System Sciences*, *26*(2), 265–278. https://doi.org/10.5194/hess-26-265-2022

Liu, M., Huang, Y., Li, Z., Tong, B., Liu, Z., Sun, M., Jiang, F., & Zhang, H. (2020). The Applicability of LSTM-KNN Model for Real-Time Flood Forecasting in Different Climate Zones in China. *Water*, *12*(2), 440. https://doi.org/10.3390/w12020440

Khairudin M, N., Aris, T. N. M., & Zolkepli, M. (2022). *In-depth review on machine learning models for long-term flood forecasting*. *100*(10), 3360–3378.

Park, K., Jung, Y., Seong, Y., & Lee, S. (2022). Development of Deep Learning Models to Improve the Accuracy of Water Levels Time Series Prediction through Multivariate Hydrological Data. *Water*, *14*(3), 469. https://doi.org/10.3390/w14030469

Sankaranarayanan, S., Prabhakar, M., Satish, S., Jain, P., Ramprasad, A., & Krishnan, A. (2020). Flood prediction based on weather parameters using deep learning. *Journal of Water and Climate Change*, *11*(4), 1766–1783. https://doi.org/10.2166/wcc.2019.321

Shao, Z., Mei, X., Xue, M., Li, J., & Tang, H. (2024). Intelligent alarm system for river embankment seepage based on BILSTM. *Scientific Reports*, *14*(1), 23822. https://doi.org/10.1038/s41598-024-75125-6

Soydaner, D. (2020). A Comparison of Optimization Algorithms for Deep Learning. *International Journal of Pattern Recognition and Artificial Intelligence*, *34*(13), 2052013. https://doi.org/10.1142/s0218001420520138

Thirumarai Selvi, C., Sankara Subbramanian, R. S., Muthu Krishnan, M., & Gnana Priya, P. (2024). IoT-Enabled Flood Monitoring System for Enhanced Dam Surveillance and Risk Mitigation. *International Research Journal of Multidisciplinary Technovation*, *6*(3), 144–153. https://doi.org/10.54392/irjmt24311

Tang, J., Yang, R., Yuan, G., & Mao, Y. (2022). Time-Series Deep Learning Models for Reservoir Scheduling Problems Based on LSTM and Wavelet Transformation. *Electronics*, *11*(19), 3222. https://doi.org/10.3390/electronics11193222

Wang, J., & Cao, Z. (2017). *Chinese text sentiment analysis using LSTM network based on L2 and Nadam*. 1891–1895. https://doi.org/10.1109/icct.2017.8359958

Wang, S., Yang, B., Chen, H., Fang, W., & Yu, T. (2022). LSTM-Based Deformation Prediction Model of the Embankment Dam of the Danjiangkou Hydropower Station. *Water*, *14*(16), 2464. https://doi.org/10.3390/w14162464

Wei, B., Chen, L., Li, H., Yuan, D., & Wang, G. (2020). Optimized prediction model for concrete dam displacement based on signal residual amendment. *Applied Mathematical Modelling*, *78*, 20–36. https://doi.org/10.1016/j.apm.2019.09.046

Xu, K., Han, Z., Xu, H., & Bin, L. (2023). Rapid Prediction Model for Urban Floods Based on a Light Gradient Boosting Machine Approach and Hydrological–Hydraulic Model. *International Journal of Disaster Risk Science*, *14*, 79–97. https://doi.org/10.1007/s13753-023-00465-2

Zarei, M., Bozorg-Haddad, O., Baghban, S., Delpasand, M., Goharian, E., & Loáiciga, H. A. (2021). Machine-learning algorithms for forecast-informed reservoir operation (FIRO) to reduce flood damages. *Scientific Reports*, *11*(1), 24295. https://doi.org/10.1038/s41598-021-03699-6

Zhang, J., Cao, X., Xie, J., & Kou, P. (2019). An Improved Long Short-Term Memory Model for Dam Displacement Prediction. *Mathematical Problems in Engineering*, *2019*(1). https://doi.org/10.1155/2019/6792189

Zhou, Y., Guo, S., Xu, C.-Y., Chang, F.-J., & Yin, J. (2020). Improving the Reliability of Probabilistic Multi-Step-Ahead Flood Forecasting by Fusing Unscented Kalman Filter with Recurrent Neural Network. *Water*, *12*(2), 578. https://doi.org/10.3390/w12020578