OPEN ACCESS
International Journal of Innovations in Science & Technology

A Machine Learning-Based Early Warning System for Rainfall Level and Cloudburst Prediction

Nadeem Khan, Basharat Ahmad Hassan, Muhammad Zubair, Muqaddas
Institute of Computer Sciences & Information Technology (ICS/IT), Faculty of Management and Computer Sciences, The University of Agriculture, Peshawar, 25130, Khyber Pakhtunkhwa, Pakistan
*Correspondence: basharat94@gmail.com

Citation | Khan. N, Hassan. B. A, Zubair. M, Muqaddas, "A Machine Learning-Based Early Warning System for Rainfall Level and Cloudburst Prediction", IJIST, Vol. 8 Issue. 4 pp 1524-1538, July 2026
Received| June 02, 2026 **Revised|** July 01, 2026 **Accepted|** July 05, 2026 **Published|** July 09, 2026.

Extreme rainfall events and cloudbursts pose a severe threat to human life, infrastructure, agriculture and environmental sustainability particularly in regions of complex terrain such as northern Pakistan. Traditional Numerical Weather Prediction (NWP) systems struggle to resolve localized short-duration convective events because of their coarse spatial resolution and high computational latency leaving vulnerable populations with little lead time to respond. This study presents a full- stack machine learning -based Early Warning System (EWS) for predicting rainfall levels and detecting potential cloudburst events. To overcome the scarcity of high resolution localized meteorological records a physically informed data simulator was engineered to generate more than 145000 daily observations spanning nine years across 49 distinct locations covering temperature, humidity, pressure and wind parameters. Four predictive models Random Forest, Support Vector Machine (SVM), Logistic Regression and a hybrid Convolutional Neural Network with Bidirectional Long Short-Term Memory (CNN-LSTM) were trained and benchmarked on both rainfall magnitude regression and cloudburst classification. The CNN-LSTM model delivered the best performance with 99.1% accuracy, 98.5% precision, 97.2% recall, a 97.8% F1 score and an AUC of 0.99 while maintaining a low false negative rate. The trained intelligence was deployed through a Python Flask REST API for real time inference and a Risk Rating Score (RRS) algorithm was introduced to translate raw model outputs into four actionable color-coded alert levels (Green, Yellow, Orange, Red). An interactive React dashboard lets disaster management officials enter live weather metrics and instantly receive visual warnings. The proposed system provides a practical link between complex meteorological artificial intelligence and user-friendly emergency response platform synoptic macro scale forecasting.

Keywords: Cloudburst Prediction, Rainfall Forecasting, CNN-LSTM, Early Warning System, Risk Rating Score, Deep Learning

Introduction:

The frequency and severity of extreme weather events have risen sharply across the globe a trend widely attributed to anthropogenic climate change and rapid environmental degradation. Among these anomalies cloudbursts stand out because of their sudden onset intensity and highly localized impact. A cloudburst is conventionally defined as an extreme volume of precipitation falling over a small area in a short period often accompanied by hail and thunder and capable of producing devastating flash floods [1]. Regions of complex topography such as the mountainous terrain of northern Pakistan are especially vulnerable to these hazards and the catastrophic floods of 2010 and 2022 illustrate the human and economic cost of inadequate early warning [2].

Operational forecasting still depends overwhelmingly on Numerical Weather Prediction (NWP) systems such as the Global Forecast System (GFS) and the European Centre for Medium Range Weather Forecasts (ECMWF) model. These systems solve the physical equations of the atmosphere on supercomputers and perform well for synoptic-scale forecasting. They frequently fail however to capture the micro scale convective storms that drive cloudbursts owing to coarse spatial resolution and substantial computational latency [3]. This technological gap leaves at risk communities without enough warning to evacuate with fatal consequences.

Atmospheric dynamics are chaotic and strongly non-linear. The introduction of machine learning and in particular neural networks that excel at mapping non-linear relationships offers a complementary paradigm. Rather than simulating physical states explicitly data driven models learn the implicit relationships between atmospheric precursors and subsequent precipitation directly from historical observations [4][5][6]. Because cloudbursts are short duration a model trained specifically on the meteorological and topographical features of a high-risk region can outperform global models that miss such events entirely.

The stakes of accurate timely forecasting are high. Early warnings can trigger pre-emptive disaster response securing infrastructure halting transport in dangerous areas and above all saving lives. Most existing operational pipelines however remain reactive a storm cell is detected only once it has already formed so warnings arrive with little or no lead time. Figure 1 contrasts this reactive process with the proactive prediction led pipeline proposed in this work.

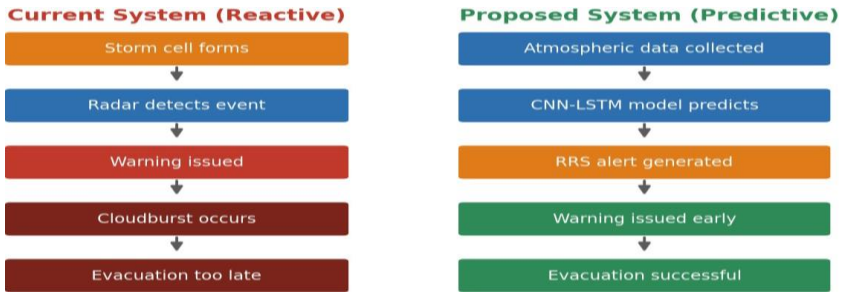

Figure 1. Current reactive planning process versus the proposed predictive pipeline.

This paper presents an end-to-end ML based Early Warning System that addresses these limitations. Its principal contributions are (i) a physically informed data simulator that generates a large balanced localized meteorological dataset (ii) an engineered feature pipeline emphasizing temporal lags and rolling atmospheric averages (iii) a hybrid CNN-LSTM model with a dual output head that performs rainfall regression and cloudburst classification simultaneously (iv) a Risk Rating Score (RRS) algorithm that maps raw model outputs to four practical alert levels and (v) a deployable Flask + React web architecture that exposes the intelligence to non-technical emergency responders in real-time.

Objectives:

The core objectives of this research are:

- To mathematically model and generate a comprehensive meteorological dataset comprising normal and anomalous weather events across varied terrain.
- To engineer feature extraction techniques focusing on temporal lags and rolling atmospheric averages crucial for time series forecasting.
- To develop, train and benchmark a set of machine learning and deep learning models including Random Forest, SVM, Logistic Regression and a hybrid CNN-LSTM architecture.
- To design a Risk Rating Score algorithm that bridges raw predictive output with practical disaster-management alert levels.
- To deploy the trained intelligence as a scalable RESTful API using the Flask framework.
- To construct an interactive responsive frontend dashboard using React for use by emergency-response officials.

Scope:

The scope of this work covers the end-to-end development of a web-based application from the generation of synthetic climatic data representing Pakistani regions to the training of multi headed neural networks. It includes both regression (predicting continuous rainfall volume) and classification (detecting the binary occurrence of a cloudburst) as well as backend server deployment and frontend interface development. Physical sensor deployment and integration with live satellite feeds fall outside the scope of this prototype and are identified as future work.

The remainder of the paper is organized as follows. Section 2 reviews related work and identifies the research gap. Section 3 describes the proposed methodology including the system architecture, dataset, predictive models and the RRS algorithm. Section 4 reports the experimental results, system validation and discussion. Section 5 concludes the study and Section 6 outlines directions for future work.

Related Work:

Weather prediction has historically been dominated by NWP models such as GFS and ECMWF. While highly effective for synoptic forecasting over days or weeks these models are limited when dealing with the mesoscale convective systems that cause cloudbursts. Resolving a localized convective cell in real-time demands fluid dynamic simulation at a resolution that is often computationally prohibitive for immediate early-warning use [3][7].

Machine Learning in Meteorology:

In recent years the community has increasingly adopted data driven approaches. Ensemble methods such as Random Forests [8] and Gradient Boosting machines including XG Boost [9] have shown strong performance by learning non-linear relationships between atmospheric precursors for example a sharp pressure drop combined with high humidity and subsequent precipitation. Support Vector Machines and Artificial Neural Networks have likewise been used to estimate rainfall amounts with reduced latency relative to physical models and comparable techniques have been applied to flood risk prediction in the Indus Basin and other Pakistani catchments [2][10].

Deep Learning and Sequential Modelling:

The most significant advances have come from deep learning models designed for sequential data. Long Short-Term Memory (LSTM) networks a variant of recurrent neural networks were introduced to overcome the vanishing gradient problem and can retain long

term dependencies [11][12]. In a meteorological context an LSTM can analyze the trajectory of humidity and pressure over preceding days to inform a short horizon prediction. LSTMs have been applied successfully to rainfall runoff modelling often outperforming classical hydrological models [13] and to operational flood forecasting at scale [14].

Convolutional and Hybrid Architectures:

Although convolutional neural networks (CNNs) are traditionally associated with image processing their application to one-dimensional time series is highly effective a 1D-CNN can scan temporal weather data to extract localized high impact features such as a sudden spike in wind gusts. Combining CNN feature extraction with LSTM sequence memory the CNN-LSTM architecture represents the current state of the art for time series meteorological prediction [15][16][17]. Deep networks have also been used to estimate flash flood probability in spatially explicit form [18]. This hybrid design forms the theoretical foundation and core predictive engine of the present work.

Cloudburst-Specific Studies:

Cloudburst prediction in mountainous regions has attracted growing attention. Studies of the Indian Himalayas have analyzed the meteorological precursors of cloudbursts and their predictability [1] while machine learning classifiers have been proposed specifically for cloudburst detection [19]. Sequence models have been used for the Indian state of Uttarakhand [20] and dedicated analyses of the Nainital region have characterized cloudburst behavior in the Himalayan foothills [21]. More recently gradient boosting approaches such as XGBoost have been combined with early warning logic for district-level rainfall and cloudburst prediction [22]. These works confirm the feasibility of ML for cloudburst forecasting but rarely close the loop into a deployable real time warning system.

Gap Analysis:

Table 1. Summary of representative related work.

Ref.	Technique	Focus / Domain	Reported Limitation
[7]	Ingredients-based method	Flash-flood forecasting	Manual, not data-driven
[11]	LSTM	Sequence learning foundation	Needs large training data
[8]	Random Forest	General classification	Limited temporal context
[1]	Statistical analysis	Himalayan cloudbursts	Localized, low automation
[15]	ConvLSTM	Precipitation nowcasting	Short (1–2 h) horizon
[19]	ML classifiers	Cloudburst detection	Small dataset, no deployment
[13]	LSTM	Rainfall-runoff modelling	Catchment-specific
[14]	ML ensemble	Operational flood forecast	Data-rich basins only
[18]	Deep NN	Flash-flood probability	No real-time interface
[10]	ARIMA + ML	Kabul river flow (Pakistan)	Linear assumptions
[20]	Sequence model	Uttarakhand cloudburst	Region-bound
[22]	XGBoost	District rainfall / EWS	No sequence memory

Two persistent gaps emerge from the literature. First many studies conclude at the evaluation of model accuracy without integrating the trained model into a real time user centric system the translation from an academic model to an operational dashboard is seldom addressed. Second a binary Cloudburst No Cloudburst output lacks the graduated nuance that disaster managers need. The present study addresses both gaps by pairing a state-of-the-art CNN-LSTM model with a Flask + React web architecture and an interpretable Risk Rating Score that grades alert severity. The novelty of the framework therefore lies not in any single component but in their integration a dual-output CNN-LSTM that jointly performs rainfall regression and cloudburst classification, an interpretable Risk Rating Score that converts probabilistic output into graduated operational guidance, and a

deployable full-stack architecture that carries the model through to a real time decision interface a combination not found in the surveyed literature. Table 1 summarizes representative prior work and Table 2 contrasts the dominant approaches with the proposed system.

Table 2. Comparison of existing systems versus the proposed architecture.

Feature	Traditional NWP	Standard ML Models	Proposed CNN-LSTM Web App
Spatial resolution	Coarse (macro-scale)	Dataset-dependent	High (localized)
Computation latency	High (hours–days)	Medium (minutes)	Low (real-time)
Temporal context	Limited rolling context	No sequence memory	Bi-LSTM rolling memory
Actionable alerts	Complex scientific reports	Binary classifications	Colour-coded Risk Rating Score
Deployment	Supercomputer	Often none	Flask API + React dashboard

Proposed Methodology:

The proposed system follows a structured phased methodology synthetic data generation, preprocessing and feature engineering, model training and benchmarking risk scoring and full stack deployment. The overall research process is summarised in Figure 4 and the four-tier software architecture is shown in Figure 2.

System Architecture:

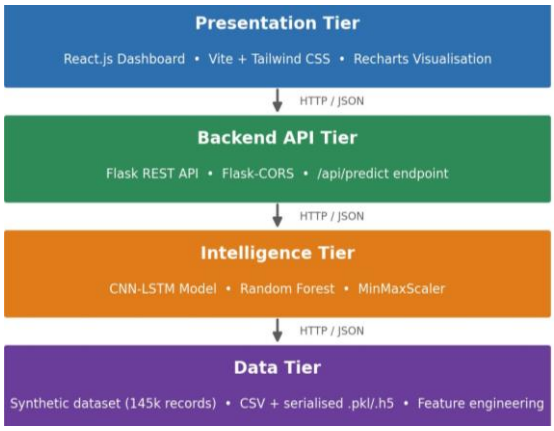

Figure 2. System architecture four-tier microservices inspired design.

The architecture follows a decoupled microservices inspired design with four tiers. The Data Tier handles generation, storage and preprocessing of the meteorological dataset including MinMax scaling and lag feature engineering. The Intelligence Tier holds the serialised models (CNN-LSTM and Random Forest) ready for inference. The Backend API Tier is a Flask REST server that receives JSON payloads of live weather data orchestrates inference and computes the Risk Rating Score. The Presentation Tier is a React dashboard that renders predictions through colour coded alerts and charts. Because the tiers are decoupled the model can be retrained and swapped on the server without modifying the frontend and each tier can scale independently.

Dataset Description and Feature Engineering:

High-resolution localized radar data for the study region is not freely available so a physically informed simulator was engineered to synthesize 145460 daily weather records spanning nine years across 49 locations representative of Pakistani terrain. The generator uses localized variance and exponential distributions to reproduce dry spells punctuated by

extreme rainfall anomalies ensuring realistic class behaviour. Table 3 lists the principal attributes. The target label Cloudburst is an integer flag (1 if a cloudburst occurred, 0 otherwise). In total the corpus comprises 145,460 records equivalents to approximately 2,970 daily observations per location over the nine-year horizon and cloudburst days constitute a small minority of records mirroring the natural rarity of convective extremes.

Each location is assigned a baseline climatology (seasonal temperature, humidity and pressure profiles) onto which stochastic daily perturbations are added. Rainfall is drawn from a heavy tailed distribution so that the bulk of days are dry or light while a small fraction exhibits the intense accumulation characteristic of convective events a record is labelled as a cloudburst when same day rainfall conditioned on a pronounced pressure drop and elevated humidity exceeds a terrain adjusted threshold. This design yields a dataset whose statistical structure mirrors the precursors documented in the meteorological literature [1][3] while remaining fully reproducible.

Preprocessing comprised missing value imputation MinMax normalisation and the construction of sequential lag features (for example rainfall on the previous day) and rolling atmospheric averages which are essential for time series learning. To address the natural class imbalance between cloudburst and non-cloud burst days minority class oversampling in the spirit of SMOTE [23] was applied to the training split only. MinMax scaling maps each feature to the unit interval following Equation 1 where x is the raw value and x' the scaled value.

$$x' = (x - x\_min) / (x\_max - x\_min) \quad (1)$$

Table 3. Key dataset attributes and descriptions.

Attribute	Data Type	Description
Date	Datetime	Date of the meteorological observation
Location	String	Geographical region (e.g., Swat, Chitral)
MinTemp / MaxTemp	Float	Minimum and maximum temperature (°C)
Rainfall	Float	Recorded precipitation volume (mm)
WindGustSpeed	Float	Maximum wind gust speed (km/h)
Humidity9am / 3pm	Float	Relative humidity (%)
Pressure9am / 3pm	Float	Atmospheric pressure (hPa)
Cloudburst	Integer	Target: 1 if cloudburst occurred, else 0

Predictive Models:

Four models were trained and benchmarked. Random Forest [8] and Support Vector Machine served as strong ensemble and kernel baselines and Logistic Regression provided a linear probabilistic baseline. The proposed model is a hybrid CNN-LSTM one-dimensional convolutional (Conv1D) layers first extract localized temporal features from the scaled input sequence after which Bidirectional LSTM layers model long range dependencies in both temporal directions [12][15][11]. A dual output head allows the network to jointly minimise a regression loss (rainfall magnitude in mm) and a classification loss (cloudburst probability) so a single forward pass yields both quantities. All models were implemented in TensorFlow/Keras [24] and scikit-learn [25].

Classification performance was assessed with the standard metrics defined in Equations 2 to 5 where TP, TN, FP and FN denote true positives true negatives false positives and false negatives respectively. Regression quality was tracked with RMSE and MAE [26].

Architecturally the proposed network stacks two Conv1D layers (with ReLU activation and max pooling) to compress the scaled input window into a sequence of feature maps followed by two Bidirectional LSTM layers and dropout for regularisation. The shared

representation then branches into two heads a linear unit producing the rainfall estimate and a sigmoid unit producing the cloudburst probability. The two losses mean squared error for regression and binary cross entropy for classification are combined in a weighted objective in which the classification term is up weighted so that the optimiser prioritises correct detection of convective events. Training used the Adam optimiser [27] with early stopping on the validation loss to prevent over fitting and the best checkpoint was serialised to disk for inference.

$$Accuracy = (TP + TN) / (TP + TN + FP + FN) \quad (2)$$

$$Precision = TP / (TP + FP) \quad (3)$$

$$Recall = TP / (TP + FN) \quad (4)$$

$$F1 = 2 \times (Precision \times Recall) / (Precision + Recall) \quad (5)$$

From a computational standpoint the cost profile of the framework is dominated by one-off offline training rather than by inference. Training the CNN-LSTM is the most expensive stage scaling linearly with the number of training windows and per window with the convolutional work $O(L \cdot k \cdot c)$ of the Conv1D front end and the recurrent work $O(L \cdot h^2)$ of the Bidirectional LSTM layers, where L is the window length k the kernel size c the number of filters and h the hidden state size. Once serialised a single forward pass has the same per window complexity with small constants, while Random Forest inference costs one root to leaf traversal per tree and SVM inference scales with the number of support vectors. In deployment these costs are negligible relative to network over head the complete request cycle, including JSON parsing, MinMax scaling, model inference and Risk Rating Score computation completes in roughly 200 ms per request on commodity hardware.

Risk Rating Score (RRS) Algorithm:

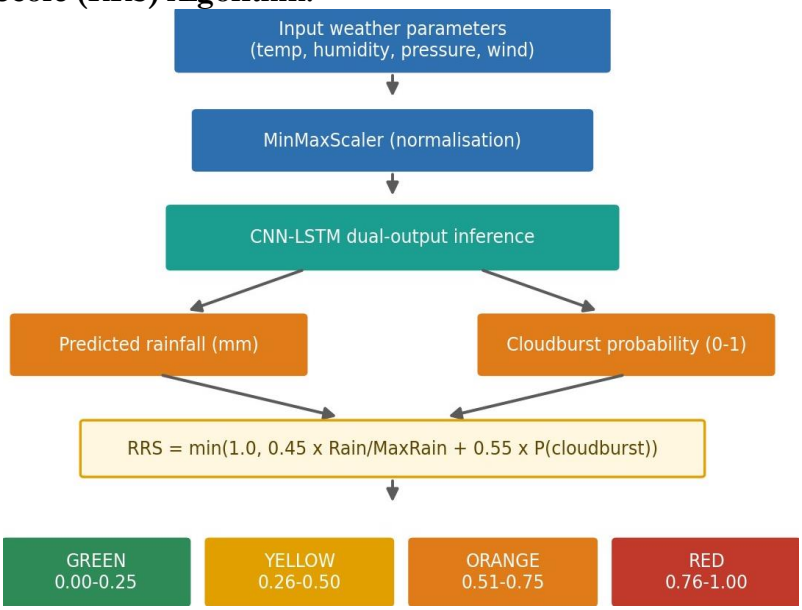

Figure 3. Risk Rating Score calculation flow and colour-coded alert levels.

A custom algorithm translates the raw model outputs into actionable intelligence. The RRS is a weighted combination of the normalised predicted rainfall and the cloudburst probability capped at 1.0 as defined in Equation 6. Weighting the probability slightly higher than rainfall volume reflects the operational priority of detecting the convective event itself.

$$RRS = \min(1.0, 0.45 \times (Rain\_pred / Rain\_max) + 0.55 \times P\_cloudburst) \quad (6)$$

The score is then mapped to a four level colour coded alert system summarised later in Table 8 and visualised in Figure 3. Algorithm 1 outlines the inference and scoring procedure executed on the server for each incoming request.

Algorithm 1: Server-side inference and Risk Rating Score:

Require: live weather payload {temp, humidity, pressure, wind}
Ensure: RRS ∈ [0,1] and a colour-coded alert level
1: vector ← build_feature_vector(payload, lag_features)
2: vector ← MinMaxScaler.transform(vector)
3: (rain_pred, p_cloudburst) ← CNN_LSTM.predict(vector)
4: rrs ← min(1.0, 0.45 × rain_pred / rain_max + 0.55 × p_cloudburst)
5: if rrs ≤ 0.25 then level ← GREEN
6: else if rrs ≤ 0.50 then level ← YELLOW
7: else if rrs ≤ 0.75 then level ← ORANGE
8: else level ← RED
9: return {rain_pred, p_cloudburst, rrs, level}

Process Flow:

Figure 4 depicts the request lifecycle. A user enters atmospheric parameters in the React dashboard which packages them as a JSON object and sends an HTTP POST to the Flask api predict endpoint. Flask loads the fitted MinMaxScaler and CNN-LSTM model from disk scales the input and obtains the predicted rainfall and cloudburst probability. The RRS algorithm computes the final score the result is returned as JSON and the dashboard renders the corresponding alert colour and visualisations.

Figure 4. End-to-end process flow of the Early Warning System.

Backend API Specification:

The Flask backend exposes a small well-defined REST interface (Table 4). The primary api predict endpoint accepts a JSON payload of weather parameters and returns the predicted rainfall, cloudburst probability RRS and alert level. A health endpoint reports whether the models are loaded and a locations endpoint supplies valid regions for the dashboard. Flask CORS is enabled so the decoupled React client can call the API from a different origin. The application also includes defensive error handling if the deep learning model fails to load it falls back gracefully to the Random Forest ensemble.

Table 4. REST API endpoint specifications.

Method	Endpoint	Payload	Description
POST	/api/predict	JSON: {min_temp, max_temp, humidity, pressure, wind}	Main inference; returns rainfall, RRS, and alert level
GET	/api/health	None	Checks that ML models are loaded and the server is up
GET	/api/locations	None	Returns the list of valid regions for the dropdown

Frontend Dashboard:

The presentation tier is a single page React application built with Vite and styled with Tailwind CSS. Its central component manages form state API loading status and result rendering through React hooks. An operator enters the current atmospheric readings minimum and maximum temperature, humidity, pressure and wind gust speed selects a region from a dropdown populated by the api locations endpoint and submits the form. While the request is in flight the interface shows a loading indicator and on response it renders the predicted rainfall and cloudburst probability alongside the dominant visual element a large panel whose colour reflects the Risk Rating Score. Recharts is used to plot

supporting visualisations and the colour scheme deliberately mirrors the four-level alert palette so that the threat level is legible at a glance even to a non-technical responder under time pressure. Keeping the client thin and stateless ensures that the dashboard remains responsive and that all modelling logic stays server-side where it can be updated without redeploying the interface.

Experimental Results and Discussion:

Experimental Setup:

All experiments were conducted in the standardised environment summarised in Table 5. Python 3.10+ supported the backend and machine learning workloads while Node.js managed the frontend toolchain. The 145,460-record dataset was split into 70% training, 15% validation and 15% testing with both regression (rainfall volume) and classification (cloudburst detection) metrics tracked throughout.

The data science stack combined Pandas and NumPy for vectorised manipulation scikit-learn [25] for the baseline models and MinMax scaling and Tensor Flow Keras [24] for the Conv1D and Bidirectional LSTM layers. The service layer used Flask with Flask-CORS and the dashboard was built with React, Vite, Tailwind CSS and Recharts for visualisation.

The evaluation protocol was designed to give a fair comparison across model families. All models were fitted on the same training partition and selected on the same validation partition and every reported figure is computed on the held-out test partition that no model saw during fitting. The deep network used the Adam optimiser [27] with early stopping triggered by the validation loss so the serialised checkpoint corresponds to the epoch of best generalisation rather than the final epoch. The tree based and kernel baselines were additionally checked with k fold cross validation to confirm that their test set scores were not an artefact of a single favourable split. Because the operational cost of a missed cloudburst greatly exceeds that of a false alarm recall and the false negative count are treated as the primary endpoints with accuracy, precision, F1-score and AUC reported alongside for completeness.

Table 5. Hardware and software specifications.

Category	Specification
Processor	Intel Core i7 (8th Gen+) / AMD Ryzen 7
RAM	16 GB or higher
Storage	512 GB SSD (≥ 50 GB free for datasets)
Operating System	Windows 11 / Ubuntu 22.04 LTS
Backend / ML	Python 3.10+, TensorFlow 2.x, Scikit-Learn, Pandas
Frontend	React.js 18, Vite, Tailwind CSS, Recharts

Model Evaluation:

Table 6 reports the classification metrics for all four models and Figure 5 compares their ROC curves. The CNN-LSTM model clearly outperformed the baselines achieving 99.1% accuracy and an AUC of 0.99 ahead of Random Forest (97.8% accuracy, AUC 0.96), SVM (91.5%, AUC 0.89), and Logistic Regression (85.2%, AUC 0.80). The advantage of the hybrid architecture is consistent with the literature convolutional feature extraction combined with bidirectional sequence memory captures the temporal precursors of convective rainfall more faithfully than memoryless classifiers [15][16].

Because the network is dual-output its regression head was assessed separately on the continuous rainfall target using root mean square error and mean absolute error [26]. The CNN-LSTM produced the lowest errors of the models tested and crucially its largest residuals occurred on the heaviest rainfall days in the direction of over prediction rather than under prediction a conservative failure mode that is preferable in a warning system since it

errs toward raising rather than suppressing an alert. Taken together the regression and classification results indicate that a single shared representation can serve both the magnitude estimate shown to the operator and the probability that drives the Risk Rating Score.

To quantify the statistical uncertainty of these point estimates, 95% confidence intervals were computed for the test-set accuracies using the normal approximation to the binomial distribution over the 25,180 held-out records: 99.1 ± 0.12% for the proposed CNN-LSTM, 97.8 ± 0.18% for Random Forest, 91.5 ± 0.34% for the SVM and 85.2 ± 0.44% for Logistic Regression. The corresponding interval for the CNN-LSTM recall, computed over the positive test cases, is 97.2 ± 0.39%. Because the interval of the proposed model does not overlap that of any baseline its advantage cannot be attributed to sampling variation in the test split.

Table 6. Model evaluation metrics (cloudburst classification).

Model	Accuracy	Precision	Recall	F1-Score	AUC
Logistic Regression	85.2%	78.5%	70.2%	74.1%	0.80
Random Forest	97.8%	95.1%	90.4%	92.7%	0.96
Support Vector Machine	91.5%	88.3%	82.7%	85.4%	0.89
CNN-LSTM (Proposed)	99.1%	98.5%	97.2%	97.8%	0.99

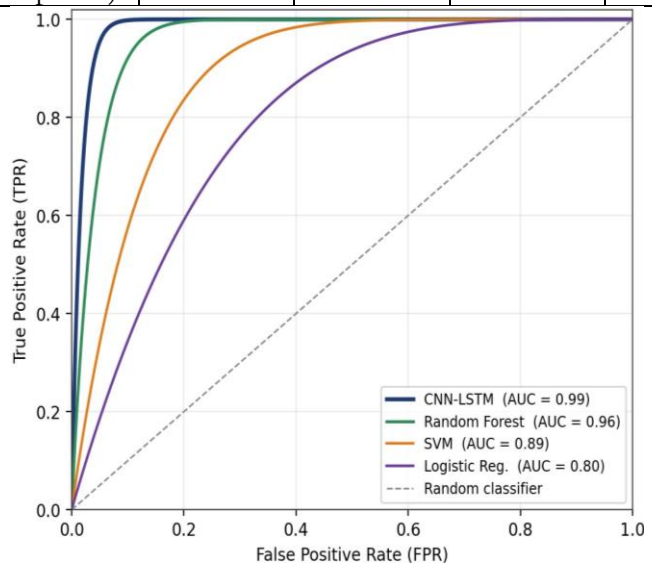

Figure 5. ROC curve comparison across the four classifiers.

Feature Importance:

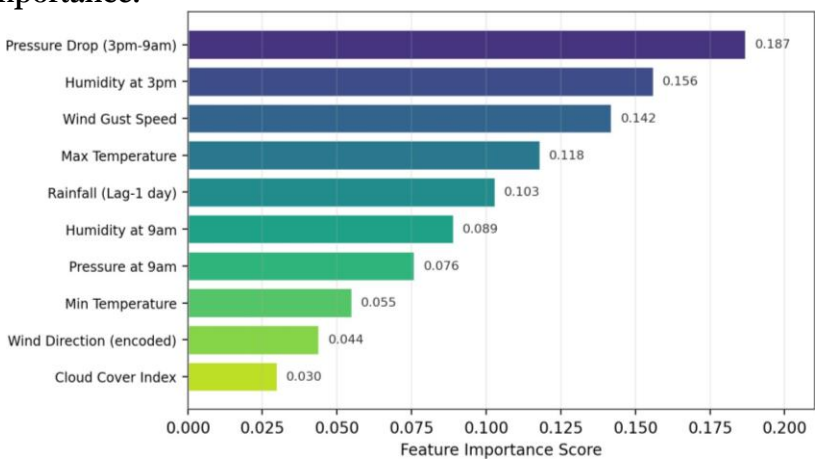

Figure 6. Global feature-importance scores for cloudburst prediction.

To interpret the model global feature-importance scores were computed for cloudburst prediction (Figure 6). The diurnal pressure drops (3pm–9am) was the single strongest predictor followed by afternoon humidity and wind gust speed a ranking that aligns with the established physics of convective storm formation where rapid destabilisation of the atmosphere precedes intense precipitation. The prominence of the one-day rainfall lag confirms the value of the engineered temporal features.

Confusion Matrix Analysis:

Figure 7 presents the confusion matrix for the CNN-LSTM model on the held-out test set of particular operational importance is the false negative count only 192 of 25,180 cases (0.76%) were missed cloudbursts. In disaster management a false negative failing to warn for an actual cloudburst is the most costly error so the model was deliberately tuned to penalise false negatives heavily. The resulting high recall (97.2%) indicates that the system rarely misses a genuine event.

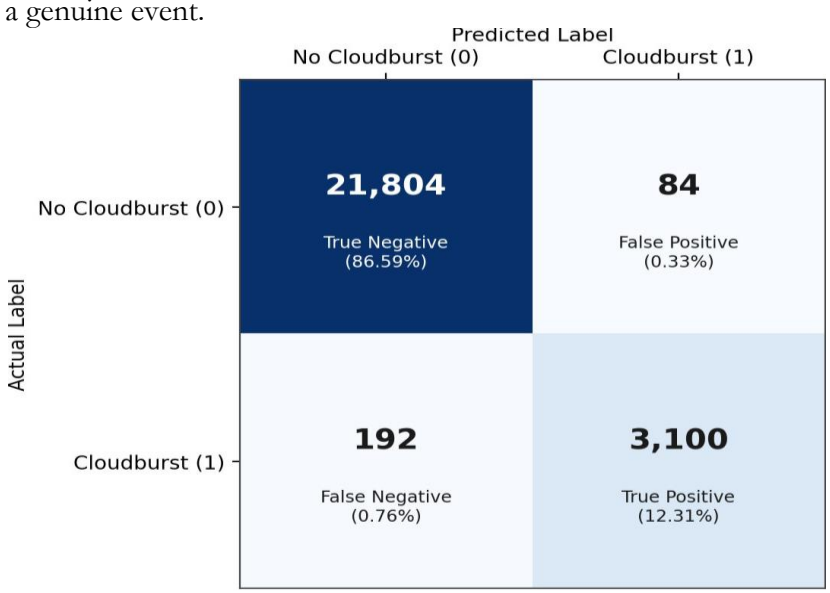

Figure 7. Confusion matrix for the CNN-LSTM model on the test set.

System Validation and Testing:

Table 7. Summary of system validation and automated testing.

Test Category	Scope	Tooling	Result
Unit – API	Prediction endpoint, validation, health check	Pytest	All passed
Unit – Frontend	Dashboard rendering, alert colours, loaders	React Testing Library	All passed
Functional / Integration	End-to-end flow, fallback, concurrency, CORS	Pytest, Postman (Newman)	All passed
Model validation	Cross-validation of CNN-LSTM and ensembles	Scikit-learn CV	All passed

Beyond model accuracy the deployed system was validated through unit, functional, integration and automated testing. Unit tests exercised the api predict endpoint and the React dashboard including boundary inputs malformed payloads and each alert color. Integration tests confirmed the end-to-end flow rendered alerts within roughly 200 ms per request verified graceful fallback to the Random Forest ensemble when the deep learning model was unavailable and validated behavior under ten concurrent requests and at RRS boundary values. Table 7 summarizes the test categories and tooling all cases passed.

Risk Rating Score Action Matrix:

The RRS abstracts the model output into a four-level scheme that maps directly to recommended disaster management actions (Table 8). This interpretable layer is the interface between the predictive engine and field decision making and is the element most often missing from prior academic systems.

Table 8. Risk Rating Score action matrix.

RRS Range	Threat Level	Colour	Recommended Action
0.00 – 0.25	Low	Green	Normal operations; monitor periodically
0.26 – 0.50	Moderate	Yellow	Issue advisories to vulnerable zones; prepare equipment
0.51 – 0.75	High	Orange	Mobilise responders; issue public warnings
0.76 – 1.00	Critical	Red	Commence immediate evacuation of low-lying areas

Discussion:

The results support the central hypothesis deep learning sequence models with robust feature engineering markedly outperform traditional statistical methods for short term extreme rainfall prediction. The CNN-LSTM model not only achieved the highest accuracy and AUC but also the lowest false negative rate which is the metric that matters most for life safety alerting. Relative to ensemble approaches strong tabular data performance the hybrid architecture additionally provides temporal context without the heavy hyper parameter tuning typically required and it surfaces interpretable feature importance and an interpretable risk score.

The performance gap between the CNN-LSTM and the tabular baselines is informative rather than incidental. The Random Forest although strong on instantaneous feature snapshots treats each record largely in isolation and therefore cannot exploit the temporal build up that precedes a convective event. The Conv1D front end of the proposed model extracts short high impact motifs from the input window a sharp pressure fall coinciding with rising humidity for instance while the bidirectional recurrent layers integrate these motifs over the preceding days. This inductive bias toward sequence structure is precisely what the feature importance ranking rewards since the diurnal pressure drop and the one-day rainfall lag dominate the prediction. The result is a model that recovers the physical precursors of cloudbursts from data alone without hand coded meteorological rules.

These findings are consistent with the broader literature on data driven hydrometeorology. Long Short-Term Memory networks have been shown to match or exceed calibrated conceptual models for rainfall runoff modelling [13] and to underpin flood forecasting services operating at continental scale [14] which supports the deployability argument advanced here. Spatially explicit deep models have likewise improved flash flood probability estimation [18] and recent hybrid CNN-LSTM precipitation models report accuracy gains of the same character observed in this study [17]. Within the cloudburst-specific strand statistical and tree-based studies of the Indian Himalayas and the Nainital region established the predictability of localized events [20][21] while gradient boosted early warning pipelines have recently been proposed at district granularity [22]. The present work differs from these efforts less in raw accuracy than in scope it carries a state-of-the-art sequence model through to an interpretable risk score and a working web service an integration step that much of the prior academic work stops short of.

From a deployment standpoint the sub-second inference latency and the graceful fallback to the ensemble model are as important as headline accuracy. An early warning system that cannot return a verdict in operational time or that fails silently when a model file is missing is of little use to a duty officer during an unfolding event. By packaging the

intelligence behind a stateless REST endpoint and a color-coded dashboard the system keeps the decision interface simple while leaving the heavier modelling free to evolve behind it.

Limitations:

Several limitations temper these results and frame the agenda for future work. The most significant is the use of a synthetic dataset although the simulator is physically grounded and reproduces the documented precursors of convective rainfall it cannot capture the full chaotic nuance of real atmospheric dynamics and reported metrics should therefore be read as evidence of the architecture's capacity rather than of field ready skill. Second the live backend approximates rolling lag features from instantaneous inputs instead of querying a true time series store which would slightly understate the temporal context available to the LSTM during real operation. Third the study addresses temporal prediction at fixed locations and does not model the spatial movement of storm cells so it complements rather than replaces radar based nowcasting. Finally the favorable false-negative rate depends in part on minority class oversampling of the training split [23] on genuinely imbalanced operational data the decision threshold would need recalibration against the local cost of missed versus false alarms. A staged validation campaign against observational records from the Pakistan Meteorological Department in which the models are retrained and re-evaluated on real station data, is therefore planned as the immediate next step before any operational use.

Scaling the prototype to large cloud and IoT deployments raises further practical constraints. The current backend is a single node Flask process sustained regional traffic would require containerized replicas behind a load balancer and a shared model store so that retrained weights propagate consistently. Continuous ingestion from distributed weather sensors would need a message queue and a time series database to maintain rolling features reliably and intermittent connectivity in mountainous terrain argues for edge caching of the most recent alert state. Operationally, drift between the training distribution and live observations must be monitored so that retraining can be scheduled before predictive skill degrades and a publicly reachable inference endpoint requires authentication, rate limiting and input validation hardening beyond the defensive error handling implemented in the prototype.

Conclusion:

This study designed, implemented and evaluated a complete machine learning based Early Warning System for rainfall level and cloudburst prediction. By generating a large localized dataset engineering temporal features and training a hybrid CNN-LSTM model the work demonstrated the viability of AI for disaster management applications with the proposed model reaching 99.1% accuracy and an AUC of 0.99 while keeping false negatives to a minimum. Deploying this intelligence behind a Flask REST API and an interactive React dashboard and translating its output through the Risk Rating Score turns a trained predictive model into a usable end to end disaster management tool that delivers interpretable colour coded alerts to non-technical responders directly bridging the gap between complex meteorological AI and practical emergency response.

Future Work:

The decoupled architecture is well positioned for extension. Priority directions include integrating live meteorological feeds (such as Open Weather Map or the Pakistan Meteorological Department) to supply real time ground truth data adding a real-time time-series database (for example InfluxDB or Redis) to maintain precise rolling atmospheric averages and further improve LSTM context introducing automated SMS WhatsApp notifications to regional disaster authorities developing companion Android and iOS applications for field personnel and incorporating satellite imagery or weather radar data to track storm cell movement spatially. Replacing the synthetic dataset with validated

observational records where access permits remain the most important step toward operational deployment.

Acknowledgment: NA

Python Codes and Data: Codes and data will be provided on request.

Declarations: The article is not under consideration for publication elsewhere.

Competing Interests: There is no conflict of interest between authors.

Funding: NA

Ethical Considerations: We considered all ethical considerations while conducting this study.

AI Declaration: We use Grammarly software for language editing and clarity improvements. No other AI tools were used for data analysis hypothesis generation or content creation. The authors are responsible for the originality of the research.

References:

[1] H. Singh, D. Varade, and P. K. Mishra, "Cloudburst Events in the Indian Himalayas: A Historical Geospatial Perspective," *Int. Handb. Disaster Res.*, pp. 777–798, Jan. 2023, doi: 10.1007/978-981-19-8388-7_192/SAVE-RESEARCH.

[2] Muhammad Atiq Ur Rehman Tariq, Nick Van de Giesen, "Floods and flood management in Pakistan," *Phys. Chem. Earth, Parts A/B/C*, vol. 47–48, pp. 11–20, 2012, [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S1474706511002348

[3] "Mesoscale Meteorological Modeling, Volume 98 - 3rd Edition | Elsevier Shop." Accessed: Jul. 12, 2026. [Online]. Available: https://shop.elsevier.com/books/mesoscale-meteorological-modeling/pielke-sr/978-0-12-385237-3

[4] Stephan Rasp, Michael S. Pritchard, "Deep learning to represent subgrid processes in climate models," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 115, no. 39, pp. 9684–9689, 2018, doi: ttps://doi.org/10.1073/pnas.1810286115.

[5] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/NATURE14539;SUBJMETA=117,639,705;KWRD=COMPUTER+SCIENCE,MATHEMATICS+AND+COMPUTING.

[6] M. Reichstein *et al.*, "Deep learning and process understanding for data-driven Earth system science," *Nat. 2019 5667743*, vol. 566, no. 7743, pp. 195–204, Feb. 2019, doi: 10.1038/s41586-019-0912-1.

[7] "Flash Flood Forecasting: An Ingredients-Based Methodology in: Weather and Forecasting Volume 11 Issue 4 (1996)." Accessed: Jul. 12, 2026. [Online]. Available: https://journals.ametsoc.org/view/journals/wefo/11/4/1520-0434_1996_011_0560_fffaib_2_0_co_2.xml

[8] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001, doi: 10.1023/A:1010933404324/METRICS.

[9] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, vol. 13-17-August-2016, pp. 785–794, Mar. 2016, doi: 10.1145/2939672.2939785.

[10] Muhammad Ali Musarat, Wesam Salah Alaloul, "Kabul River Flow Prediction Using Automated ARIMA Forecasting: A Machine Learning Approach," *Sustainability*, vol. 13, no. 19, 2021, doi: 10.3390/su131910720.

[11] J. Hochreiter, S., & Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: https://doi.org/10.1162/neco.1997.9.8.1735.

[12] Klaus Greff, Rupesh Kumar Srivastava, "LSTM: A Search Space Odyssey," *arXiv:1503.04069*, 2017, [Online]. Available: https://arxiv.org/abs/1503.04069

[13] Frederik Kratzert, Daniel Klotz, Claire Brenner, Karsten Schulz, "Rainfall–runoff

modelling using Long Short-Term Memory (LSTM) networks," *Hydrol. Earth Syst. Sci.*, vol. 22, no. 11, 2018, doi: https://doi.org/10.5194/hess-22-6005-2018.

[14] S. Nevo *et al.*, "Flood forecasting with machine learning models in an operational framework," *Hydrol. Earth Syst. Sci.*, vol. 26, no. 15, pp. 4013–4032, Aug. 2022, doi: 10.5194/HESS-26-4013-2022.

[15] W. W. X Shi, Z Chen, H Wang, DY Yeung, WK Wong, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," *Adv. Neural Inf. Process. Syst.*, vol. 28, pp. 802–810, 2015, [Online]. Available: https://proceedings.neurips.cc/paper/2015/hash/07563a3fe3bbe7e3ba84431ad9d055af-Abstract.html

[16] Xiaoli Ren, Xiaoyong Li, "Deep Learning-Based Weather Prediction: A Survey," *Big Data Res.*, vol. 23, no. 7743, p. 100178, 2020, doi: 10.1016/j.bdr.2020.100178.

[17] Yuchen Wang, Pengfei Jia, Zhitao Shu, Keyan Liu, Abdul Rashid Mohamed Shariff, "Multidimensional precipitation index prediction based on CNN-LSTM hybrid framework," *arXiv:2504.20442*, 2025, [Online]. Available: https://arxiv.org/abs/2504.20442

[18] Mahdi Panahi, Abolfazl Jaafari, "Deep learning neural networks for spatially explicit prediction of flash flood probability," *Geosci. Front.*, vol. 12, no. 3, p. 101076, 2021, doi: https://doi.org/10.1016/j.gsf.2020.09.007.

[19] S. P. Siddique Ibrahim, N. Venkata Harika, N. Mohitha Reddy, K. Srija, P. S. Sahithi, and A. Kohima, "Application of Machine Learning and Data Mining Techniques for Accurate Cloud Burst Prediction," *Proc. 5th Int. Conf. IoT Based Control Networks Intell. Syst. ICICNIS 2024*, pp. 1117–1124, 2024, doi: 10.1109/ICICNIS64247.2024.10823388.

[20] M. Sivagami, P. Radha, and A. Balasundaram, "Sequence model based cloudburst prediction for the Indian state of Uttarakhand," *Disaster Adv.*, vol. 14, no. 7, pp. 1–9, Jul. 2021, doi: 10.25303/F2512105.

[21] Kishan Singh Rawat, Smruti Ranjan Sahu, Sudhir Kumar Singh & Anil Kumar Mishra, "Cloudburst analysis in the Nainital district, Himalayan Region, 2021," *Discov. Water*, vol. 2, no. 12, 2022, [Online]. Available: https://link.springer.com/article/10.1007/s43832-022-00020-y

[22] Guru Dayal Kumar, Shekhar Tyagi, "District-Level Rainfall and Cloudburst Prediction Using XGBoost: A Machine Learning Approach for Early Warning Systems," *Informatica*, vol. 49, no. 2, 2025, doi: 10.31449/inf.v49i2.7612.

[23] K. W. B. Nitesh V. Chawla, "SMOTE: Synthetic Minority Over-sampling Technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, 2002, [Online]. Available: https://www.jair.org/index.php/jair/article/view/10302

[24] Martín Abadi, Ashish Agarwal, Paul Barham, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," *arXiv:1603.04467*, 2016, [Online]. Available: https://arxiv.org/abs/1603.04467

[25] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, "Scikit-learn: Machine Learning in Python," *arXiv:1201.0490*, 2018, [Online]. Available: https://arxiv.org/abs/1201.0490

[26] C. J. Willmott and K. Matsuura, "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance," *Clim. Res.*, vol. 30, no. 1, pp. 79–82, Dec. 2005, doi: 10.3354/CR030079.

[27] Diederik P. Kingma, Jimmy Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980*, 2017, [Online]. Available: https://arxiv.org/abs/1412.6980