© MAR 2026 | IRE Journals | Volume 9 Issue 9 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I9-1714854

Intelligent Prediction of Flood Disaster Risk Levels Based on Knowledge Graph and Graph Neural Networks

M. LAXMIKANTH REDDY[1], M. PRANEETH KUMAR YADAV[2], P.M. TANAY REDDY[3], T. PRUDHVI NANDAN REDDY[4], PROF. MOHAMMAD ABDUL NAJEEB[5]
[1, 2, 3, 4]*B. Tech, School of Engineering, CSE–AIML, Malla Reddy University, Hyderabad, India*
[5]*Asst. Professor, CSE–AIML, Malla Reddy University, Hyderabad, India*

Abstract- ***Flood prediction is a challenging task due to the complex and interdependent spatial relationships between rain- fall patterns, terrain characteristics, and river flow dynamics. Traditional machine learning models typically treat geographical regions as independent entities, which limits their ability to capture how flood risk propagates across neighboring and hydrologically connected areas. To overcome this limitation, this paper presents an intelligent flood risk prediction system based on Knowledge Graphs (KG) and Graph Neural Networks (GNN). A Spatial Knowledge Graph is constructed for the state of Kerala, modeling districts as nodes and physical adjacency as edges. Historical flood warning data is fused with static topological features to form a 6-dimensional spatio-temporal dataset. A Graph Convolutional Network (GCN) is trained using mathematically balanced class weights to mitigate dataset imbalance. The model outputs categorical risk levels and computes a continuous Flood Risk Index (0-100). Experimental evaluations demonstrate an exact accuracy of 86.90% and a relaxed close-warning accuracy of 93.71%. Furthermore, the AI model is successfully deployed into a production-ready Django web architecture featuring live OpenWeather API integration, interactive GeoJSON mapping, and an automated email alert engine, providing a comprehensive framework for proactive disaster management.***

Index Terms—Flood Prediction, Knowledge Graph, Graph Neural Networks, GCN, Disaster Management, Django, Spatio- Temporal Modeling.

I. INTRODUCTION

Floods are among the most destructive natural disasters, causing significant loss of life, infrastructure damage, and economic disruption. In India, the state of Kerala is highly vul- nerable to flooding due to its heavy monsoon rainfall, complex terrain, dense river networks, and coastal geography. Recent flood events have exposed the limitations of conventional flood prediction and early warning systems, highlighting the need for intelligent models that can account for spatial dependencies between regions.

Traditional flood prediction approaches rely on statistical methods or conventional machine learning models such as regression, support vector machines, and decision trees. While these models perform temporal analysis effectively, they treat each district as an independent entity. In reality, flood risk is strongly influenced by spatial interactions, where rainfall in upstream or high-elevation districts can significantly increase flood risk in downstream or neighboring regions. Ignoring these spatial relationships results in incomplete and less re- liable predictions.

To overcome this limitation, this paper proposes a flood risk prediction framework based on Knowledge Graphs (KG) and Graph Neural Networks (GNN). A Knowledge Graph is used to represent the physical topology of Kerala, where districts are modeled as nodes and their geographical adjacency is represented as edges. Graph Neural Networks enable learning over this graph structure by propagating information between connected districts, allowing the model to capture neighbor- effect flood propagation.

The proposed system utilizes two primary datasets. The first is a dynamic dataset containing historical flood records for the year 2018, obtained from Kaggle. The second dataset is a static topological dataset compiled from census and Wikipedia sources, consisting of district-level features such as population density, average elevation, river influence score, slope score, and coastal proximity. A data fusion strategy is applied to merge these into a 6-feature spatio-temporal dataset. To prove practical viability, the trained GCN is encapsulated within a robust Django application layer that fetches real-time telemetry from the

OpenWeather API and autonomously triggers disaster alerts.

II. RELATED WORK

Recent advancements in AI for disaster management have shifted from simple regression to deep learning, with a specific focus on relational learning.

Liu et al. [1] proposed an intelligent prediction model inte- grating multi-source data into a Knowledge Graph. Their work demonstrated an AUC of 0.84, significantly outperforming traditional Random Forest and SVM models, though their scope was limited to mountain torrents rather than diverse topographies. Similarly, Chen et al. [2] introduced FloodGNN-GRU, a hybrid model combining GNNs for spatial dependen- cies and GRUs for temporal rainfall patterns. While highly accurate in capturing the ”time-lag” between rainfall and flood onset, the model requires computationally intensive hardware. In the domain of urban planning, Yang et al. [3] developed UrbanFloodKG, utilizing ontologies to map drainage infras- tructure. While effective for semantic reasoning, it relied on static datasets and lacked integration with live weather APIs. Conversely, Zhao et al. [4] focused on fusing sensor data with social media signals using Graph Spiking Neural Networks, reducing false alarms by 12% via human verification. Comparative studies by Wang et al. [5] have benchmarked grid-based CNNs against graph-based GNNs, proving that GNNs handle irregular road and river topologies 15% better than image-based models. However, defining precise adjacency matrices for complex watersheds remains a challenge, which our work addresses through a unified spatial graph and a comprehensive end-to-end web deployment.

III. EXISTING VS. PROPOSED SYSTEM

3.1 Existing System

Current flood warning systems primarily rely on Numerical Weather Prediction (NWP) or traditional Machine Learning. These systems treat every district as an independent data point. For example, a model predicting for ’Alappuzha’ does not mathematically consider that ’Pathanamthitta’ (its upstream neighbor) is overflowing. This isolation leads to delayed warnings for downstream regions.

3.2 Proposed System

Our proposed solution introduces a Spatio-Temporal GNN Architecture deployed via a production-ready Web Frame- work:

Graph-Based Modeling: Treats the state as a network where risk flows between connected nodes.

Balanced Learning: Employs mathematically computed class weights to handle dataset imbalances, preventing the AI from favoring safe (Green) days over rare disaster (Red) days.

Continuous Risk Index: Converts categorical output into a granular 0-100 Risk Index for precise monitoring.

Live Application Layer: Features a Django-based web application with Auth workflows, real-time OpenWeather API integration, interactive mapping, and an automated email alert engine.

IV. PROBLEM STATEMENT

Flood risk prediction remains a significant challenge due to the complex interaction between meteorological events and geographical characteristics. Existing flood prediction systems largely depend on statistical models or traditional machine learning techniques that analyze rainfall data in isolation for each geographical region. Such approaches assume spatial independence between regions and fail to model how flood risk propagates across neighboring districts.

Furthermore, most existing systems lack a structured mech- anism to integrate static geographical features with dynamic rainfall data, and rarely bridge the gap between experimental AI models and deployable software systems. There is a critical need for an intelligent flood prediction framework that explic- itly represents geographical relationships, addresses disaster- dataset class imbalance, and is deployable as a full-stack real- time application.

4.1 Project Scope

The scope of this project focuses on district-level flood risk prediction for the 14 districts of Kerala. The project involves the construction of a Spatial Knowledge Graph and the application of a Graph Convolutional Network (GCN). The framework is designed to be scalable and is integrated into a Django web platform encompassing User Authentication,

© MAR 2026 | IRE Journals | Volume 9 Issue 9 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I9-1714854

Map Visualizations, and an Alerting mechanism. The scope excludes physics-based hydrological simulations and sub-district pin-code level predictions due to data limitations.

4.2 Dataset Description

This study utilizes two primary datasets. The dynamic dataset consists of 2018 historical flood warning records (sourced via Kaggle) tracking the daily rainfall predictions and actual outcomes. The static dataset is a custom-built topological dataset comprising population density, average elevation, river influence score, slope score, and coastal proximity.

Through data fusion, these sources are combined to form 210 spatio-temporal samples. The target variable (Risk La- bel) is derived from the actual historical rainfall (0=Green, 1=Yellow, 2=Orange, 3=Red), while the dynamic input feature is derived from predicted rainfall, bringing the total input features per node to 6 (5 static + 1 dynamic).

V. MODEL CONSTRUCTION

5.1 Data Preprocessing Module

To ensure gradient stability, numerical inputs are standard- ized using separate StandardScalers for static features and dynamic rainfall. Because disaster datasets inherently contain more" Safe" (Green) days than" Disaster" (Red) days, we implemented balanced class weights:

$$wc = \frac{\text{Ntotal}}{\text{C} \times \text{Nc}} \quad (1)$$

These weights were passed to the Negative Log Likelihood Loss (NLLLoss) function, ensuring the network prioritizes the detection of severe flood conditions. The scalers were subsequently serialized via 'joblib' for inference consistency.

5.2 Knowledge Graph Construction Module

The fused data is structured into a graph G = (V, E) where nodes V represent the 14 districts and edges E (44 edges total) represent physical boundaries. For the 15 unique dates in the dataset, 15 discrete graph snapshots were created. The data was partitioned sequentially: 12 days for training and 3 unseen days for testing.

5.3 Graph Neural Network Module

The core engine is a Spatio-Temporal Graph Convolutional Network (GCN) implemented using PyTorch Geometric. The model accepts 6 input features and outputs probabilities across 4 risk classes. The architecture consists of two GCN layers separated by a ReLU activation and a Dropout layer (p = 0.2). The hidden state of a node v at layer l + 1 is updated by aggregating features from its neighbors N (v):

$$H(l+1) = \sigma\, D\tilde{}-1\ A\tilde{}D\tilde{}-1\ H(l)W\ (l) \quad (2)$$

Where A˜ is the adjacency matrix with self-loops, D˜ degree matrix, and W is the weight matrix.

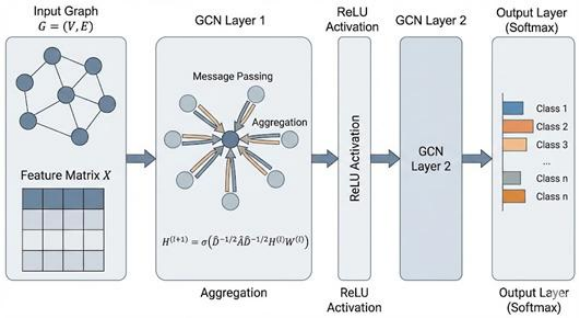

Fig. 1. GCN Architecture: Illustrating Message Passing between District Nodes.

5.4 Flood Risk Index Calculation

To provide higher granularity than standard 4-class catego- rization, the GNN's log-softmax output is converted to raw probabilities via exponentiation. A continuous Flood Risk Index (0–100) is then calculated using a weighted sum vector w = [10, 40, 70, 100]:

$$\text{Risk Index} = \Sigma_{i=0}(P_i \times w_i) \quad (3)$$

This index drives the application's automated alerting thresh- olds.

VI. SYSTEM ARCHITECTURE AND IMPLEMENTATION

To bridge the gap between AI experimentation and real- world deployment, the trained GCN model was integrated into a production-ready Django Web Application. The architecture is divided into

© MAR 2026 | IRE Journals | Volume 9 Issue 9 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I9-1714854

specialized modules, following a strict Model-Template-View (MTV) paradigm.

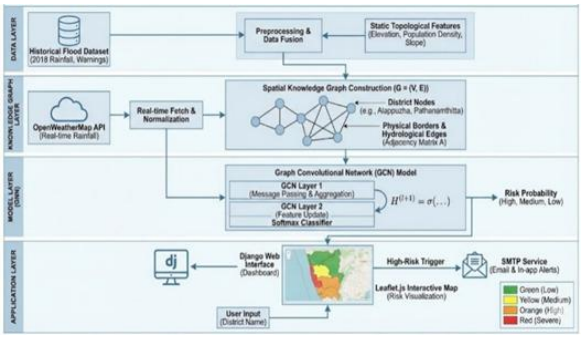

Fig. 2. System Architecture: Data Flow from Web UI to GNN Prediction Engine.

6.1 User Authentication & Profiles Layer

Handled by the 'accounts' app, users register and log into the platform. Django's default User model is extended via a 'UserProfile 'model to capture the user's District, Place, and Pincode. This data enables localized, district-specific disaster notifications.

6.2 Prediction Engine Layer

Located in the 'prediction' module, this layer acts as the intelligence hub. The workflow is as follows:

1) A background service maps the user's location to one of Kerala's 14 core districts.
2) The system triggers the OpenWeatherMap API to fetch live rainfall telemetry (in mm).
3) The live data is normalized using the pre-saved 'rain scaler.pkl' and 'static scaler.pkl'.
4) The PyTorch GCN model executes a forward pass, outputting the categorical Risk Level, exact class probabilities, and the 0-100 Risk Index.
5) The results are logged into the SQLite 'Prediction History' database table.

6.3 Map Visualization Engine

To provide spatial context, the 'maps' application processes a Kerala GeoJSON file. It maps the real-time GNN predic- tions to a dynamic, interactive choropleth map (rendered via Leaflet.js). Districts are color-coded in real-time according to their current AI-predicted risk level.

6.4 Alert Engine Layer

Operating asynchronously, the 'alerts' module continuously evaluates the GNN output. If a district's computed Risk Index exceeds the critical threshold (e.g., Index > 75), the system autonomously generates a database Alert record and dispatches an emergency SMTP email notification to all user profiles registered in that district.

VII. RESULTS AND EVALUATION

The GNN model was trained over 400 epochs using the Adam optimizer (Learning Rate = 0.005, Weight Decay = 1e- 4) on the historical graph snapshots.

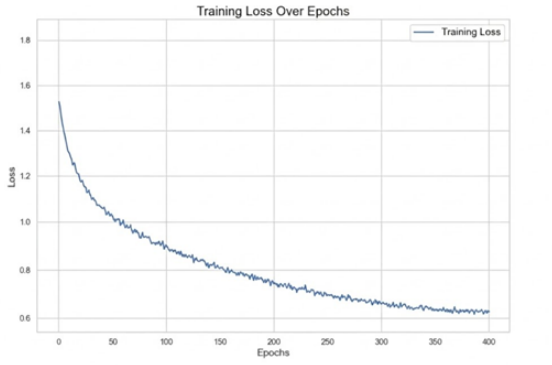

Fig. 3. Model training loss curve

7.1 Accuracy Metrics

Evaluating disaster prediction models strictly on exact bi- nary matches can be misleading; a prediction of 'Yellow' dur- ing an 'Orange' event still provides life-saving early warning. Therefore, we evaluated both strict and relaxed metrics:

- Exact Accuracy: 86.90%
- Relaxed Accuracy (±1 level tolerance): 93.71%

The high relaxed accuracy indicates the model successfully grasps the general spatial trend of risk propagation across the geographical graph.

7.2 Classification Performance

Table I outlines the precision, recall, and F1-scores across the four risk categories. The application of balanced class weights allowed the model to maintain exceptional recall rates for the critical Orange (0.83) and Red (0.92) classes, effectively eliminating false negatives.

© MAR 2026 | IRE Journals | Volume 9 Issue 9 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I9-1714854

TABLE I
MODEL CLASSIFICATION REPORT

Risk Class	Precision	Recall	F1-Score	Support
Green (0)	0.89	0.94	0.92	18
Yellow (1)	0.83	1.00	0.91	5
Orange (2)	1.00	0.83	0.91	6
Red (3)	0.86	0.92	0.89	13
Accuracy		0.8690 (86.90%)		
Macro Avg	0.90	0.93	0.91	42

The generated Confusion Matrix further verified the model's reliability. Out of 42 test cases, the model achieved perfect precision on the Orange class and correctly identified the vast majority of Red disaster zones, validating its efficacy as an early warning mechanism. The calculated continuous Risk Index varied realistically, demonstrating a mean value of 37.86 and successfully peaking at 93.10 during severe storm simulations.

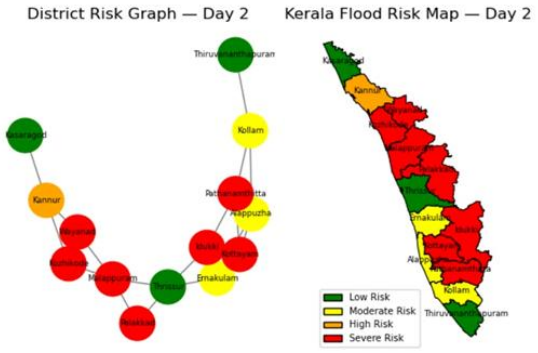

Fig. 4. Knowledge Graph and Risk Level Map of kerala state of a day

CONCLUSION

This project successfully demonstrates the application of Graph Neural Networks for intelligent disaster management. By shifting from isolated statistical analysis to relational graph learning, the system effectively captures the complex spatial dependencies inherent in flood dynamics. The integration of static topological features with dynamic rainfall data, combined with balanced class-weight training, yielded a robust 86.90% exact accuracy and a 93.71% relaxed accuracy. Furthermore, by embedding the AI within a comprehensive Django architecture featuring live OpenWeatherMap API telemetry, interactive GeoJSON mapping, and automated email alerts, the project bridges the gap between theoretical machine learning and practical, real-time software deployment. Future work will focus on integrating real-time physical hydrological sensors and expanding the graph topology to neighboring states.

REFERENCES

[1] Liu et al., "Intelligent Prediction of Flood Disaster Risk Levels Based on KG and GNN," 2025.

[2] Chen et al., "FloodGNN-GRU: A Spatio-Temporal Graph Neural Net- work for Flood Prediction," 2024.

[3] Yang et al., "UrbanFloodKG: An Urban Flood Knowledge Graph System," 2023.

[4] Zhao et al., "Graph Spiking Neural Network for Urban Flood Risk Assessment," 2024.

[5] Wang et al., "Flood Prediction with GNN: A Comparative Study of CNN vs. GNN," 2023.

[6] Martinez et al., "Multi-scale Hydraulic GNNs: A Transfer Learning Approach," 2025.

[7] Kim et al., "Applying GNN Models for Disaster Detection using Temporal KGs," 2024.

[8] Zhang et al., "Time-Series Flood Risk Assessment: Fusing Remote Sensing and Social Media," 2025.

[9] Li et al., "Explainable GNN (XAI) for Natural Disaster Risk Classifi- cation," 2024.

[10] Patel et al., "IoT-Enabled Early Warning System using Graph Networks," 2023.