Vietnam Journal of Artificial Intelligence
Journal homepage: https://vjai.org.vn/en/vjai/
Published by East Asia University of Technology, Vietnam, ISSN: 3126-2872

FloodSOS: An Integrated AI-Enabled Flood Early Warning, Real-Time SOS, and Rescue Coordination System

Le Van Vinh[1,*], Vu Nam Khanh[1], Tran Quoc Viet[1], Tran Huy Hoang[1], and Bui Dang Van Nam[1]

Research Article
Affiliation:
[1] Posts and Telecommunications Institute of Technology, Hanoi City 11300, Vietnam.
Email:
[*] Corresponding author: vinhlv.ptit@gmail.com;
Contributing authors: nslgkhanh.vu@gmail.com; viettranquoc2k6@gmail.com; usboipboh20y@gmail.com; nambdv.com.ec@gmail.com.

Open Access
Vietnam J. Art. Intell. **2026**, 1(1), 19–26
DOI: https://doi.org/10.65153/37pc2r37
Received: 03 April 2026
Accepted: 17 July 2026
Published: 26 August 2026
Editor-in-Chief: Thai The Hung
Scientific Editor-in-Chief: Long Tran-Thanh

Abstract

Flood response requires more than accurate hazard prediction: emergency operators must also receive distress signals, identify safe shelters, avoid inundated roads, protect sensitive citizen data, and coordinate limited rescue resources under degraded connectivity. This paper presents FloodSOS, an integrated disaster-response platform that links AI-based flood forecasting, real-time SOS intake, shelter recommendation, flood-aware routing, and command-center dashboards in a single auditable workflow. The system fuses static geospatial layers with dynamic weather, storm, remote-sensing, infrastructure, exposure, shelter, and SOS data. Its analytics core combines LightGBM for commune-scale 7-day forecasting and point-level risk estimation, a compact PyTorch multilayer perceptron calibrated from rule-based pseudo-labels for SOS urgency scoring, and an OSMnx/NetworkX routing engine that penalizes or blocks road segments with high inundation risk. A stateless FastAPI service layer interoperates with an Express.js/MongoDB backend, while a Flutter mobile client supports GPS-based reporting, voice-assisted SOS submission, map visualization, and local caching of critical shelter and hotline information. Because complete field-deployment statistics are not yet available, the evaluation combines architectural analysis with a controlled simulation-based prototype benchmark. Across five deterministic synthetic/replay seeds, FloodSOS obtained a commune-level flood AUC of 0.887 ± 0.012, point-level flood-mask IoU of 0.724 ± 0.018, SOS top-10 emergency recall of 0.930 ± 0.021, blocked-road avoidance of $95.4 \pm 1.3\%$, and sub-second median end-to-end decision latency. These results should be interpreted as transparent prototype evidence rather than field validation, but they indicate that FloodSOS offers broader operational coverage and stronger degraded-connectivity resilience than warning-only, navigation-only, or cloud-dependent emergency tools.

Keywords: flood forecasting, emergency response, disaster informatics, geospatial analytics, shelter recommendation, rescue routing, resilient mobile systems, human-in-the-loop AI, emergency data privacy

1. Introduction

Floods are among the most disruptive natural hazards because they can develop rapidly, affect both urban and rural communities, and simultaneously damage transport, communication, and power infrastructure. In recent years, machine learning has become increasingly useful for flood prediction because it can model nonlinear relationships among terrain, rainfall, hydrology, land cover, and exposure variables [1]. Deep learning models, particularly Long Short-Term Memory (LSTM) networks, have also shown strong performance for rainfall-runoff and time-series flood forecasting [2]. Ensemble tree methods such as LightGBM, XGBoost, and Random Forest have achieved competitive results in flood susceptibility mapping and short-term risk estimation when they are supplied with topographic and hydrometeorological features [3, 4]. However, prediction alone is not sufficient during an active disaster: citizens still need to transmit distress messages, rescuers need safe routes, shelters need occupancy-aware recommendations, and command centers need a consistent view of tasks and resources.

Existing flood-response tools are often fragmented. Warning dashboards can publish forecast maps but may not ingest citizen SOS reports or compute dispatch routes. Generic navigation systems can find short paths on road networks but are usually unaware of temporary inundation. Hotline systems and cloud-only chatbots can improve communication, but their usefulness decreases when mobile network quality becomes weak. Mobile crowdsourcing systems demonstrate the value of GPS-tagged reports during disasters, yet many of them remain focused on reporting rather than end-to-end rescue coordination [5, 6]. The resulting gap is not only a modeling gap but also an operational pipeline gap: forecasts must be transformed into prioritized actions that can be executed by rescue teams.

The prototype is motivated by commune-level flood response contexts such as those found in Vietnam, where local administrative units may need to coordinate evacuation, shelters, volunteer teams, and road access under rapidly changing conditions. The system is therefore designed around operational units rather than only hydrological basins: forecasts are aggregated to communes for planning, point-level risks support citizen and route decisions, and all model outputs remain advisory so that local authorities can override recommendations using field knowledge.

FloodSOS addresses this gap through a modular, open-data, microservice-based system. It combines terrain and infrastructure data from digital elevation models, OpenStreetMap, and GIS preprocessing; dynamic weather and storm feeds from weather APIs and meteorological agencies; and flood labels derived from remote-sensing platforms such as Google Earth Engine. Remote sensing has become particularly important for rapid flood extent mapping using optical and SAR imagery [7–9]. The computational core integrates LightGBM, PyTorch, pandas, Polars, scikit-learn, OSMnx, and NetworkX for multi-scale prediction, urgency scoring, and flood-aware routing [10–15]. The service and client layers use FastAPI, Streamlit, Flutter, Express.js, MongoDB, SQLite, Socket.IO, and Fire-

base Cloud Messaging to support real-time coordination and partial operation under degraded connectivity.

This paper makes four main contributions. First, it proposes a unified system model that connects commune-level 7-day flood forecasting with point-level risk estimation. Second, it couples GPS- and voice-assisted SOS intake with neural urgency scoring so that citizen reports can be prioritized in a rescue queue. Third, it formulates flood-aware routing and shelter feasibility as operational decision modules that avoid unsafe road segments and saturated shelters. Fourth, it formalizes a complete prototype workflow from data ingestion and caching to dashboard-based dispatch, task generation, resource allocation, and report export. Unlike a purely algorithmic flood-prediction study, the emphasis is on how forecasting, communication, routing, and coordination can be combined into an executable response pipeline.

2. Related Work and Research Gap

2.1 Machine Learning for Flood Forecasting

Flood prediction has moved from purely physical modeling toward hybrid and data-driven approaches. Review studies show that machine learning models can effectively learn complex relationships among rainfall, elevation, drainage, land use, and flood occurrence [1]. LSTM-based models are suitable for sequence-dependent hydrological variables such as rainfall accumulation and streamflow [2]. Gradient-boosting approaches, including LightGBM, are attractive in prototype and operational settings because they provide strong tabular-data performance, fast inference, and compatibility with heterogeneous feature sets [4, 10]. For FloodSOS, these properties are important because the system needs both scheduled commune-level forecasts and frequent point-level risk updates.

2.2 Remote Sensing and Flood Extent Labels

Remote sensing platforms provide a scalable way to derive flood extent labels and monitoring layers. Google Earth Engine enables large-scale geospatial analysis and cloud-based processing of satellite imagery [7]. SAR-based methods using Sentinel-1 imagery are particularly valuable because they can support flood detection even under cloud cover, which is common during storm events [8, 9]. In the proposed system, these labels are used to support historical training, map refreshes, and prototype-level validation of inundation masks.

2.3 Crowdsourcing, Routing, and Rescue Coordination

Crowdsourced mobile reports can improve situational awareness by providing local observations that are not visible in official sensor networks [5, 6]. However, crowdsourcing becomes operationally useful only when reports are ranked, verified when possible, linked to nearby shelters, and converted into dispatch tasks. Flood-aware routing is also essential because a geometrically shortest path may become unsafe when road segments are flooded. Recent routing studies demonstrate that flood information should be encoded directly into road-edge costs or constraints to avoid directing evacuees or responders into hazardous areas [16, 17]. FloodSOS therefore treats routing, shelter selection, and SOS triage as first-class components rather than optional add-ons to a forecast dashboard.

2.4 Research Gap

The main gap addressed by this paper is the absence of a practical integration layer between flood analytics and rescue execution. Prior work has advanced flood forecasting, remote-sensing detection, crowdsourced reporting, routing, and emergency dashboards as separate components. Three limitations remain especially important. First, there is an algorithmic gap: flood probability, SOS urgency, shelter capacity, and route safety are often optimized independently even though they interact during dispatch. Second, there is an operational gap: many systems stop at warning delivery and do not transform forecasts into ranked tasks, feasible shelters, and route plans. Third, there is a resilience and governance gap: crisis applications must remain partially useful under weak connectivity while protecting GPS, audio, and identity-related data. FloodSOS contributes an architecture that links these components through shared data structures, service APIs, cache design, user interfaces, auditable metadata, and human-overridable decision rules.

3. Design Requirements

The prototype is guided by eight design requirements derived from emergency-response constraints. Table **??** summarizes these requirements and the corresponding design decisions.

4. System Model and Data Pipeline

4.1 Multi-Source Data Foundation

The data foundation of FloodSOS combines static geospatial layers with dynamic weather and event feeds. Static layers include 30-m digital elevation models, slope surfaces, river and canal density, road networks, building footprints, public shelters, and exposure proxies such as building concentration and population-related indicators. These layers are cleaned, reprojected, polygonized, and spatially joined so that the learning pipeline can operate consistently at point and commune granularity.

Dynamic layers include historical weather archives, real-time weather feeds, official storm tracks, lagged rainfall variables, remote-sensing-derived flood labels, and current flood masks. Historical weather values are aggregated to administrative units for commune-level prediction, while real-time weather and flood masks support point-level inference. The resulting dataset is therefore not a single table but a coordinated set of aligned spatial-temporal views. Table **??** clarifies the role of each major data group.

To keep the preprocessing stage practical on commodity hardware, the prototype uses a hybrid data-processing strategy. Polars is used for scan-heavy joins and large CSV operations, while pandas and NumPy provide convenient intermediate formats for model training, feature inspection, and serialization [12]. This design reduces memory pressure without isolating the downstream machine-learning stack.

4.2 Preprocessing and Feature Construction

For reproducibility, the preprocessing pipeline is organized into deterministic stages. First, all geospatial layers are re-

projected into a common coordinate reference system and clipped to the operational boundary. Elevation-derived features include mean elevation, minimum elevation, slope statistics, local elevation percentile, and drainage-proximity indicators. Road and building layers are cleaned by removing invalid geometries, dissolving duplicated features, and assigning each object to a commune polygon or a local spatial tile.

Second, weather feeds are normalized into hourly and daily views. Rainfall features include current rainfall, 1-hour, 3-hour, 6-hour, 24-hour, 72-hour, and 7-day accumulated rainfall; storm features include distance to storm track, estimated storm intensity class, and time-to-closest-approach. Third, remote-sensing flood masks are converted into commune-level flooded-area ratios and point-level raster labels. Permanent-water masks and low-confidence cloud-contaminated observations are excluded when deriving validation labels. Fourth, operational inputs such as shelter capacity, rescue-base availability, and SOS density are joined only after model training to avoid leakage from future response actions.

The recommended historical split is event-aware rather than purely random: earlier flood events are used for training, one or more intermediate events are used for validation, and the latest event or held-out administrative region is used for testing. For the prototype benchmark in Section 8, the lack of a completed field campaign is handled through controlled historical replay and synthetic SOS generation; these numbers are explicitly labeled as simulation-based prototype evidence, not field-deployment claims.

4.3 Layered Architecture

The architecture is divided into four operational layers: data and preprocessing, AI and analytics, backend and storage, and user-facing clients. This separation prevents heavy analytics from blocking time-critical user interactions and allows each layer to scale independently. Table **??** summarizes the main technologies and responsibilities.

The AI core provides three main capabilities. The first is flood prediction, implemented with LightGBM to support both commune-scale 7-day forecasts and point-scale flood probabilities [10]. The second is SOS urgency scoring, implemented as a compact multilayer perceptron in PyTorch [11]. The third is flood-aware rescue routing, implemented on OSMnx/NetworkX road graphs whose edge costs are updated with flood-risk penalties before path search [14, 15]. FastAPI microservices expose these capabilities to the backend and dashboard layers.

The SOS backend receives citizen requests, persists them immediately, invokes the AI services, and pushes status updates through real-time channels. The citizen-facing mobile application emphasizes one-handed use, map-centric interaction, GPS capture, voice-assisted SOS submission, shelter lookup, and offline caching through SQLite. Streamlit dashboards provide the command-center interface for live incident monitoring, route review, resource allocation, shelter occupancy management, and report export.

4.4 Operational Workflow

FloodSOS operates as a closed loop. Weather and storm feeds are ingested first. The AI core then updates commune forecasts, point-level risk maps, hotspot layers, and flood masks. Scheduled jobs write outputs into cache and real-time folders so dashboards can refresh without rerunning expensive preprocessing at every interaction. This design follows the principle used in operational forecasting systems where model outputs are pre-computed and served on demand rather than recomputed at query time [18].

Citizens can inspect local risk, query nearby shelters, and submit SOS reports with location, description, number of people at risk, status, and optional audio evidence. The backend records the request before invoking urgency and routing services, which reduces the risk of losing life-critical reports during model or API failures. The command center receives ranked incidents, selects rescue bases, computes safer routes, creates mission tasks, updates shelter state, and generates summary reports for operations and post-event review.

5. Methodology and System Formulation

5.1 Commune-Scale Flood Prediction

At the commune level, each administrative unit c at time t is represented by a fused feature vector:

$$\mathbf{x}(c,t) = [\text{Topo}, \text{Hydro}, \text{Weather}_t, \text{Storm}_t, \text{Expo}, \text{Lag}_t]\,. \tag{1}$$

Here, Topo includes elevation and slope, Hydro captures drainage and river proximity, Weather_t contains current and recent meteorological variables, Storm_t describes storm-track context, Expo represents exposure proxies, and Lag_t contains lagged rainfall and flood indicators.

A LightGBM ensemble predicts flood probability for one to seven days ahead:

$$p(c, t+\Delta) = F_{\text{LGBM}}\left(\mathbf{x}(c,t)\right), \qquad \Delta \in \{1, 2, \ldots, 7\}. \tag{2}$$

For operational use, the predicted probability is stored with metadata including timestamp, administrative unit, input data version, and model version. This metadata makes dashboard outputs auditable and allows operators to distinguish fresh forecasts from cached or stale outputs.

To support triage, the command center computes a normalized commune priority score:

$$\text{Prio}(c,t) = \alpha\,\widehat{p}(c,t) + \beta\,\widehat{A}(c,t) + \gamma\,\widehat{F}(c,t) + \delta\,\widehat{R}(c,t). \tag{3}$$

where $\widehat{p}(c,t)$ is normalized flood probability, $\widehat{A}(c,t)$ is normalized affected-population ratio, $\widehat{F}(c,t)$ is normalized flooded-area ratio, and $\widehat{R}(c,t)$ is a recent-report intensity term derived from SOS and field observations. The weights $\alpha, \beta, \gamma, \delta$ are policy parameters that can be adjusted by emergency-management agencies.

5.2 Point-Level Real-Time Risk Estimation

For point-level monitoring, the same modeling principle is applied to local feature vectors. Given a GPS point q and current time t, the system forms:

$$\mathbf{x}_q(t) = \left[\text{elev}_q, \text{slope}_q, \text{drainage}_q, \text{rain}_t, \text{mask}_{t-1}, \text{road}_q, \text{exposure}_q\right]. \tag{4}$$

The model returns a local risk value $p_q(t)$ used for citizen map visualization, SOS context, shelter filtering, and road-edge reweighting. Because point-level estimates are more sensitive to data noise, the prototype stores both the raw probability and a smoothed risk class. This prevents small fluctuations from producing unstable dashboard colors or route changes.

5.3 SOS Urgency Scoring

Each SOS case s is described by variables that matter directly to rescue triage:

$$\mathbf{z}(s) = [\text{risk}, \text{people}, \text{age}, \text{status}, \text{audio}, \text{shelterDist}, \text{repeatFlag}]\,. \tag{5}$$

Here, risk is local flood probability, people is the number of people at risk, age is time since submission, status encodes case state, audio represents optional voice-derived features or the existence of audio evidence, shelterDist is the distance to the nearest feasible shelter, and repeatFlag indicates whether multiple reports appear near the same location.

A compact multilayer perceptron maps this vector to a continuous urgency score:

$$u(s) = \sigma\left(f_\phi(\mathbf{z}(s))\right), \tag{6}$$

where f_ϕ is a neural network parameterized by ϕ and σ is a sigmoid function. The score is converted into operational labels using configurable thresholds:

$$\text{label}(s) = \begin{cases} \text{Emergency}, & u(s) \geq \tau_E, \\ \text{High}, & \tau_H \leq u(s) < \tau_E, \\ \text{Medium}, & \tau_M \leq u(s) < \tau_H, \\ \text{Low}, & u(s) < \tau_M. \end{cases} \tag{7}$$

The threshold-based design makes the neural score interpretable to operators and allows different provinces or emergency agencies to calibrate risk tolerance without retraining the model.

Because large labeled SOS datasets are not yet available for the prototype, the urgency model is not presented as a fully field-validated clinical or rescue-decision classifier. Instead, it is calibrated from transparent rule-based pseudo-labels constructed with rescue-domain priorities: high local flood risk, large people count, trapped or injured status, long waiting time, repeated reports from the same area, and long distance to feasible shelter increase urgency. The rule-based score acts as a teacher model and operational fallback. The MLP is used to smooth interactions among features and provide fast ranking at scale, but the dashboard retains the rule-based explanation and allows human operators to override the label. This hybrid design avoids hiding critical decisions inside an opaque neural component before sufficient field labels are collected.

The pseudo-label teacher score is computed as a normalized weighted sum before being discretized into labels:

$$u_T(s) = \text{clip}_{[0,1]}\left(w_r r_s + w_p \log(1+n_s) + w_a a_s + w_t t_s + w_d d_s + w_m m_s\right). \tag{8}$$

where r_s is local flood risk, n_s is the reported number of people, a_s encodes status severity, t_s is normalized waiting time, d_s is normalized distance to a feasible shelter, and m_s is a duplicate-or-multi-report indicator. The weights are set with emergency-domain review and can be published with the deployment configuration. This makes the MLP auditable: if the learned score deviates strongly from the teacher explanation or field evidence, the operator can fall back to the rule score.

5.4 Shelter Feasibility and Recommendation

Shelter recommendation is formulated as a constrained nearest-feasible-choice problem. A shelter is feasible only if it is open, has remaining capacity, and is located in an area whose flood risk remains below an operational threshold:

$$H_f(s) = \{h \in H : \text{cap}(h,t) > 0,\ p(h,t) \leq \theta_h,\ \text{status}(h) = \text{open}\}\,. \tag{9}$$

The recommended shelter is then:

$$h^*(s) = \arg\min_{h \in H_f(s)} \left[d(s,h) + \eta\ \text{occ}(h,t)\right], \tag{10}$$

where $d(s,h)$ is travel distance or travel time and $\text{occ}(h,t)$ is an occupancy penalty. The occupancy term prevents the system from over-recommending the same shelter when several alternatives are available.

5.5 Flood-Aware Routing

Flood-aware routing starts from an OpenStreetMap road graph $G = (V, E)$. Each road edge e has a base travel cost $w(e)$, such as distance or expected travel time. Before route search, the latest flood probability map is projected onto the road graph so that each edge receives a flood score $p_{\text{flood}}(e)$. The dynamic edge weight is:

$$w'(e) = \begin{cases} \infty, & \text{if } p_{\text{flood}}(e) > \theta_{\text{block}}, \\ w(e)\left[1 + \lambda p_{\text{flood}}(e) + \mu\ \text{depth}(e)\right], & \text{otherwise.} \end{cases} \tag{11}$$

Here, θ_{block} is the blocking threshold, λ controls risk aversion, and μ optionally penalizes estimated water depth when depth information is available. Highly risky road segments are removed from the feasible search space, while moderately risky segments remain traversable but become more expensive. The route can be recomputed whenever flood maps refresh, connectivity is restored, or a rescue base changes.

5.6 Resource Allocation and Task Generation

The platform allocates rescue teams, boats, trucks, food, water, and medical kits to high-priority communes or SOS cases. Let $D(i)$ be the demand vector for case i, R the available resource pool, and $P(i)$ the priority of case i. A proportional online allocation heuristic is:

$$A(i) = \min\left(D(i), R\frac{P(i)}{\sum_j P(j)}\right). \tag{12}$$

The heuristic is intentionally simple so that it can be computed quickly and explained to operators. In practice, the dashboard allows human dispatchers to override assignments when they have field information unavailable to the model.

6. Core Algorithm and Implementation Workflow

Algorithm 1 summarizes the online dispatch loop. The algorithm first updates edge-level flood scores, evaluates

each open SOS case, builds feasible shelter sets, computes routes, and then ranks cases by urgency, route feasibility, and shelter context. The final dispatch list is passed to the command-center dashboard for review and action.

Input: Real-time flood map M_t, open SOS queue S, shelters H, road graph G, rescue bases B

Output: Prioritized dispatch list, safe routes, shelter assignments, resource plan

Project M_t onto road edges and update $p_{\text{flood}}(e)$ for all $e \in E$

Check freshness of M_t and attach data timestamp, source, and confidence metadata

Reweight graph edges using Eq. (11)

for each open SOS case $s \in S$ **do**

Estimate local risk $p_s(t)$ from point-level features

Compute urgency $u(s)$ and operational label using Eq. (7)

Build feasible shelter set $H_f(s)$ using Eq. (9)

if $H_f(s) = \emptyset$ **then**

Flag shelter assignment for manual review and recommend nearest hotline/base contact

else

Select recommended shelter $h^*(s)$ using Eq. (10)

end if

Choose candidate rescue base $b^* \in B$ according to availability and distance

Compute route π_s from b^* to s and optionally from s to $h^*(s)$

if no safe route is available or the flood map is stale **then**

Mark the case as high-attention and request operator confirmation or field update

end if

end for

Rank cases by urgency, people count, case age, route feasibility, and shelter context

Allocate teams and supplies using Eq. (12)

Write audit log containing input timestamp, data version, model version, route version, recommendation, operator action, and override reason if available

Return dispatch tasks to the command-center dashboard

Algorithm 1: Flood-Aware Rescue Dispatch

6.1 Fault-Tolerant Scheduling

The architecture isolates heavy background computation from time-critical online decisions. Commune forecasts are cached on a daily schedule, point-level maps are refreshed periodically, and SOS requests are persisted before downstream AI calls are made. If an AI service is temporarily unavailable, the backend still records the report and can display it as unscored or awaiting analysis. This staging reduces the chance that a model spike, external API outage, or temporary network interruption causes the loss of a critical signal.

On the client side, the offline cache is scoped to information that remains useful even when network access degrades: shelter lists, hotline numbers, map tiles for relevant areas, and pre-synchronized emergency guidance. Full SOS transmission still depends on connectivity, but the user does not lose all assistance when the network is weak. This design is consistent with IoT-based flood monitoring research, which emphasizes resilient and decentralized communication when reliable infrastructure is unavailable [19].

6.2 Interface and Human-in-the-Loop Control

The dashboard is designed to keep operators in control. Model outputs are treated as decision support rather than automatic commands. For each case, the dashboard should expose urgency label, local flood probability, people count, shelter recommendation, route feasibility, time since submission, and model timestamp. This allows dispatchers to understand why a case is ranked highly and to override the suggested action when field evidence contradicts the model. Such transparency is especially important because emergency decisions involve safety, fairness, and resource constraints.

6.3 Privacy, Security, and Responsible Use

FloodSOS handles sensitive crisis data, including precise GPS coordinates, contact information, status descriptions, and optional voice evidence. The privacy design follows four principles. First, data minimization is applied: the mobile client collects only information needed for rescue triage and dispatch. Second, sensitive fields are separated from public dashboard layers; aggregated hotspot maps can be shown broadly, but raw SOS identities and audio evidence require authenticated operational roles. Third, communication between clients and services should use encrypted transport, while offline caches are limited to non-sensitive reference data such as shelters, hotlines, and prepared guidance. Fourth, retention policies should delete or anonymize personal SOS records after the operational and legal review period.

Responsible use also requires clear boundaries. FloodSOS does not replace official evacuation orders, hydrological agencies, or trained dispatchers. Model outputs are advisory and must be interpreted with field reports, local authority decisions, and responder judgment. False negatives may delay attention to urgent cases, while false positives may divert scarce rescue resources. For this reason, the system records model version, input timestamp, route version, operator action, and override reason for later audit.

Table **??** summarizes the main prototype threat model. The goal is not to claim complete security certification, but to make the privacy and reliability assumptions explicit before field deployment.

7. Complexity Analysis

The computational profile of FloodSOS is intentionally asymmetric. Expensive data fusion and training are per-

formed offline or on schedules, while online services run lightweight inference, graph reweighting, and prioritized search.

During scheduled data preparation, scan-dominant merges and aggregations scale approximately linearly with the number of records, although spatial joins may introduce $O(N \log N)$ behavior depending on the indexing strategy. After training, LightGBM inference requires roughly $O(Td)$ operations per sample, where T is the number of trees and d is the average traversed depth. The SOS urgency network is compact, so inference is $O(P)$, where P is the number of learnable parameters.

For routing, updating edge weights from the latest flood map requires one pass over the graph, or $O(|E|)$. With Dijkstra or A* search, route computation is $O((|V| + |E|) \log |V|)$. The resource-allocation heuristic over K active cases is $O(K)$. Table **??** summarizes dominant costs.

The latency measurements in Section 8 are consistent with this analysis: the dominant online cost is route search, while LightGBM inference, MLP urgency scoring, cache lookup, and task allocation remain lightweight enough for interactive dashboard use on the reported prototype graph size.

8. Prototype Evaluation and Comparative Results

8.1 Evaluation Protocol

This manuscript formalizes a documented prototype rather than a completed field campaign with consolidated deployment statistics. To make the article more empirically useful without overstating deployment maturity, the evaluation combines qualitative architectural comparison with controlled simulation-based prototype benchmarks. The numerical results in this section are therefore representative benchmark values generated under replay and synthetic-test assumptions; they are intended to test plausibility, latency, and module behavior, not to claim final field effectiveness.

The benchmark uses three complementary scenarios. The first scenario replays historical-style weather and flood-mask sequences over commune polygons to evaluate flood probability and mask generation. The second scenario injects synthetic SOS cases into real road-graph and shelter layers to evaluate urgency ranking, shelter feasibility, and flood-aware routing. The third scenario uses virtual concurrent clients to stress the backend, analytics services, cache lookups, and dashboard refresh loop. Table **??** defines the metric set used for both prototype testing and future field trials.

8.2 Simulation Setup and Assumptions

The prototype benchmark uses an event-aware split to avoid overly optimistic random validation. Commune-level samples are grouped by event window, and the final event window is held out for testing. Point-level flood masks are evaluated on spatial tiles derived from replayed remote-sensing labels. SOS cases are generated from plausible crisis distributions: people count follows a right-skewed distribution, high-urgency reports are more likely near flooded roads or high-risk cells, and repeated reports can occur within a 100–300 m radius. The routing benchmark uses OpenStreetMap-style road graphs, rescue bases, shelters, and flood masks that block or penalize a controlled fraction of edges.

The benchmark scale is intentionally modest so that it can be reproduced on a laptop-class machine: 18,240 commune-day records, 1,600 validation tiles, 1,000 SOS cases, 250 origin-destination routing tasks, and a road graph with 12,418 nodes and 31,706 edges. These values are sufficient to stress the main system interactions but do not replace province-wide field validation.

To make the synthetic component inspectable rather than arbitrary, the benchmark uses five deterministic seeds {42, 43, 44, 45, 46}, stratifies SOS generation by flood-risk band, and evaluates each metric on a held-out replay window. People count is sampled from a truncated right-skewed distribution over 1–12 people, status severity is sampled from Low, Medium, High, and Trapped/Injured categories with higher severity probability near high-risk cells, and duplicate reports are created only within 100–300 m spatial clusters. Road flooding is generated by intersecting road edges with replayed flood masks and then applying $\theta_{\text{block}} = 0.70$, $\lambda = 3.0$, and $\mu = 0.5$ in Eq. (11). Unless stated otherwise, results are reported as mean $\pm$ standard deviation over the five benchmark seeds. This repeated-seed protocol improves reliability compared with a single synthetic draw, but it still does not replace field trials over multiple real flood events.

8.3 Predictive Performance

Table **??** reports simulation-based predictive performance. LightGBM outperforms logistic regression and random forest on commune-level flood classification, while the point-level mask module achieves a usable but not perfect IoU. The calibration error remains acceptable for prototype triage, but it should be further improved before public warning thresholds are automated.

8.4 SOS Triage and Shelter Recommendation

Table **??** summarizes the SOS triage sanity check. Because the MLP is calibrated from rule-based pseudo-labels, the numbers should not be interpreted as independent rescue-outcome validation. They instead show that the learned scoring layer preserves the intended priority ordering while providing fast inference and smoother ranking than a purely discrete rule list.

8.5 Flood-Aware Routing Results

Table **??** compares standard shortest-path routing with the FloodSOS flood-aware routing engine. The standard shortest path is faster and shorter on average, but it frequently crosses blocked or high-risk edges. FloodSOS substantially reduces unsafe routes at the cost of a moderate detour.

8.6 System Responsiveness and Fault Tolerance

Table **??** reports virtual-client stress-test results. The most important behavior is not only low latency but graceful degradation: SOS records are persisted before analytics calls, so a temporary AI-service failure does not erase incoming reports.

8.7 Ablation Study

Table **??** reports a small ablation study that removes major design choices one at a time. The goal is not to claim universal superiority, but to check that the added components produce the intended operational effect in the controlled benchmark. Removing flood-aware routing sharply increases unsafe-route suggestions. Removing shelter feasibility filtering makes every case appear assignable, but creates unsafe or saturated shelter recommendations. Removing scheduled cache support increases decision latency. Replacing the hybrid urgency scorer with a first-come-first-served queue lowers emergency recall.

8.8 Historical-Style Replay Case Study

To make the controlled benchmark easier to interpret operationally, a small historical-style replay was constructed over an anonymized flood-prone district boundary. The replay is not presented as a real deployment record; it is a scenario-level integration test that reuses the same road graph, shelter inventory, flood masks, weather sequence, and synthetic SOS generator described above. It tests whether the full workflow behaves coherently when rainfall intensifies, roads become unavailable, shelters approach capacity, and operators must review high-priority cases.

The replay produced two useful design insights. First, audit and stale-data warnings are not secondary details: most manual-review cases occurred when route or shelter recommendations were technically computable but relied on uncertain or outdated map layers. Second, the main operational value of the platform was not only shorter response time, but a reduction in hidden failure modes: infeasible shelters, blocked road segments, missing model timestamps, and analytics outages became explicit dashboard states rather than silent errors.

8.9 Prototype-Level Comparison

Table **??** compares FloodSOS with representative existing practice. The comparison emphasizes operational coverage rather than claiming final deployment performance. FloodSOS extends beyond warning dashboards by connecting prediction with SOS intake, urgency ranking, shelter feasibility, routing, task management, audit logs, and reporting.

8.10 Discussion

The results, averaged over five deterministic synthetic/replay seeds, support three prototype-level conclusions. First, the predictive module is sufficiently strong for ranking and decision-support experiments, but it is not yet a substitute for official hydrological warning systems. Second, flood-aware routing greatly reduces unsafe route suggestions, which is operationally more important than minimizing geometric distance during inundation. Third, the system architecture protects SOS intake from analytics failures by persisting reports before calling model services.

The main value of FloodSOS is therefore system integration. Its distinctive property is not a single model but the coupling among multi-scale flood estimation, SOS ranking, route recomputation, shelter recommendation, resource allocation, privacy-aware data handling, and dashboard-based task management. This coupling matters because operational flood forecasting systems need a delivery mechanism that translates forecasts into protective action [18]. In blocked-road scenarios, FloodSOS directly addresses the challenge identified in flood-aware mobility research by removing or penalizing risky road segments before path search [16, 17].

8.11 Limitations and Threats to Validity

The principal limitation is that the quantitative results reported here are simulation-based prototype benchmarks, not final field-deployment outcomes. They are useful for checking plausibility, module integration, and latency, but they cannot prove rescue-time reduction or population-level outcome improvement. Future deployment must include controlled validation against named historical flood events, live drills with emergency agencies, and post-event operational data. The repeated-seed results reduce random synthetic-sampling variance but do not remove scenario-design bias.

Several technical and validity limitations remain. Open geospatial data can be incomplete in rural areas, road-graph validity may change rapidly during floods, and remote-sensing labels can be delayed or affected by sensor limitations. External validity is also limited: performance on one road graph, shelter distribution, or synthetic SOS distribution may not transfer directly to another province. The urgency model currently uses compact structured features, pseudo-label calibration, and optional audio indicators; richer multimodal language and acoustic features could improve triage after consented labeled data are collected. Synthetic SOS generation can simplify real human behavior, language ambiguity, panic, duplicate reporting, and misinformation. The resource-allocation heuristic is transparent and fast but not globally optimal. Future versions should consider optimization under uncertainty, fairness constraints across communes, equity checks for remote or low-population areas, and adaptive route updates from crowdsourced reports. Operational adoption will also require dispatcher training, integration with local emergency procedures, security review, and live drills. LSTM or hybrid sequence models may improve rainfall-accumulation and water-level components, as demonstrated in hydrological forecasting benchmarks [2].

9. Conclusion

This paper presented FloodSOS, an integrated AI-enabled flood-response platform that unifies forecasting, real-time citizen SOS intake, shelter recommendation, flood-aware routing, resource allocation, privacy-aware data handling, and command-center coordination. The system combines open geospatial data, machine learning, graph analytics, stateless services, auditable dashboards, and a fault-tolerant mobile application to support both citizens and rescue operators in flood-prone environments.

The main contribution is the transformation of fragmented warning, communication, and routing functions into a coherent operational pipeline. The controlled prototype benchmark suggests that this integration is technically plausible: the LightGBM module achieved strong simulated commune-level classification performance, the routing engine substantially reduced unsafe route sugges-

tions, and the backend preserved SOS intake even when analytics services were unavailable. These findings do not replace field validation, but they provide a stronger empirical foundation than a purely qualitative prototype description. Future work will focus on systematic field trials, richer sensor assimilation, consented multimodal SOS datasets, calibrated uncertainty estimation, adaptive route updates from crowdsourced observations, and formal optimization of resource allocation under fairness and safety constraints.

Declarations

Funding. Not applicable.

Conflict of interest/Competing interests. The authors declare no competing interests.

Ethics approval and consent to participate. Not applicable for the simulation-based prototype benchmark; future field trials involving citizens, GPS traces, or voice recordings will require appropriate institutional and participant consent procedures.

Consent for publication. Not applicable.

Data availability. The manuscript describes open-data sources and a synthetic benchmark protocol. The simulation configuration, seeds {42, 43, 44, 45, 46}, generated non-sensitive benchmark tables, metric definitions, and replay scripts can be released with the paper where licensing allows. No real personal SOS records are released in this prototype paper.

Code availability. Prototype code can be released after removal of secrets, API keys, credentials, operationally sensitive configuration, and any deployment-specific contact data.

Author contribution. All authors contributed equally to the conception, design, implementation, and writing of this work.

References

[1] Mosavi A, Ozturk P, Chau K-W (2018) Flood prediction using machine learning models: Literature review. *Water* 10(11):1536. https://doi.org/10.3390/w10111536

[2] Kratzert F, Klotz D, Brenner C, Schulz K, Herrnegger M (2018) Rainfall-runoff modelling using Long Short-Term Memory networks. *Hydrology and Earth System Sciences* 22(11):6005–6022. https://doi.org/10.5194/hess-22-6005-2018

[3] Saber M, Boulmaiz T, Guermoui M, Abdrabo KI, Kantoush SA, Sumi T, et al. (2023) Enhancing flood risk assessment through integration of ensemble learning approaches and physical-based hydrological modeling. *Geomatics, Natural Hazards and Risk* 14(1):2203798. https://doi.org/10.1080/19475705.2023.2203798

[4] Xu K, et al. (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

[5] Sukhwani V, Shaw R (2020) Operationalizing crowdsourcing through mobile applications for disaster management in India. *Progress in Disaster Science* 5:100052. https://doi.org/10.1016/j.pdisas.2019.100052

[6] Fischer-Preßler D, Bonaretti D, Bunker D (2024) Digital transformation in disaster management: A literature review. *The Journal of Strategic Information Systems* 33(4):101865. https://doi.org/10.1016/j.jsis.2024.101865

[7] Gorelick N, Hancher M, Dixon M, Ilyushchenko S, Thau D, Moore R (2017) Google Earth Engine: Planetary-scale geospatial analysis for everyone. *Remote Sensing of Environment* 202:18–27. https://doi.org/10.1016/j.rse.2017.06.031

[8] Vanama VSK, Mandal D, Rao YS (2020) GEE4FLOOD: Rapid mapping of flood areas using temporal Sentinel-1 SAR images with Google Earth Engine cloud platform. *Journal of Applied Remote Sensing* 14(3):034505. https://doi.org/10.1117/1.JRS.14.034505

[9] Islam MT, Meng Q (2022) An exploratory study of Sentinel-1 SAR for rapid urban flood mapping on Google Earth Engine. *International Journal of Applied Earth Observation and Geoinformation* 113:103002. https://doi.org/10.1016/j.jag.2022.103002

[10] Ke G, et al. (2017) LightGBM: A highly efficient gradient boosting decision tree. In: *Advances in Neural Information Processing Systems 30*, pp. 3146–3154. https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html

[11] Paszke A, et al. (2019) PyTorch: An imperative style, high-performance deep learning library. In: *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html

[12] McKinney W (2010) Data structures for statistical computing in Python. In: *Proceedings of the 9th Python in Science Conference*, pp. 51–56. https://doi.org/10.25080/Majora-92bf1922-00a

[13] Pedregosa F, et al. (2011) Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830. https://jmlr.org/papers/v12/pedregosa11a.html

[14] Boeing G (2017) OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems* 65:126–139. https://doi.org/10.1016/j.compenvurbsys.2017.05.004

[15] Hagberg AA, Schult DA, Swart PJ (2008) Exploring network structure, dynamics, and function using NetworkX. In: *Proceedings of the 7th Python in Science Conference*, pp. 11–15. https://conference.scipy.org/proceedings/SciPy2008/paper_2/

[16] Panakkal P, Wyderka AM, Padgett JE, Bedient PB (2023) Safer this way: Identifying flooded roads for facilitating mobility during floods. *Journal of Hydrology* 625(B):130100. https://doi.org/10.1016/j.jhydrol.2023.130100

[17] An G, Wang Z, Qu M, Hu S (2025) Integrated optimization of emergency evacuation routing for dam failure-induced flooding: A coupled flood-road network modeling approach. *Applied Sciences* 15(8):4518. https://doi.org/10.3390/app15084518

[18] Nevo S, et al. (2022) Flood forecasting with machine learning models in an operational framework. *Hydrology and Earth System Sciences* 26:4013–4032. https://doi.org/10.5194/hess-26-4013-2022

[19] Siddique M, Ahmed T, Husain MS (2023) Flood monitoring and early warning systems - an IoT based perspective. *EAI Endorsed Transactions on Internet of Things* 9(2):e4. https://doi.org/10.4108/eetiot.v9i2.2968