Unified Emergency Condition Detection

Malkapuram Ramya
Department Of Computer Science and Engineering
Talla Padmavathi College Of Engineering (Autonomus),
Somidi, Kazipet, Telangana.
Email: ramyamalkapuram9@gmail.com

Mrs. J. Shilpa
Assistant Professor, Department of CSE
Talla Padmavathi College Of Engineering(Autonomus),
Somidi, Kazipet, Telangana.
Email: jala.shilpa2@gmail.com

Abstract

The escalating frequency of natural and man-made disasters demands intelligent, automated frameworks capable of detecting emergencies in real time. This paper presents the Unified Emergency Condition Detection (UECD) system, an integrated platform that fuses Internet of Things (IoT) sensor networks, convolutional neural networks (CNN), and deep neural networks (DNN) to detect, classify, and communicate diverse emergency conditions including fires, gas leaks, seismic events, health crises, and vehicular accidents. The proposed architecture employs a hybrid CNN-DNN model trained on structured multi-sensor datasets encompassing temperature, smoke, vibration, heart-rate, and gas-concentration readings, achieving a detection accuracy of 96.7% with a false alarm rate below 1.8%. A Random Forest baseline achieves 93.4% accuracy, while the hybrid deep model surpasses it by leveraging both spatial feature extraction and sequential decision boundaries. Sensor data fusion across heterogeneous modalities substantially reduces ambiguity and false positives. The system generates geo-tagged, severity-ranked alerts distributed via SMS, email, and mobile push notifications to relevant authorities within sub-second latency. An experimental Flask-based web interface demonstrates real-time prediction, dataset management, and model loading capabilities. Experimental results indicate that the UECD system achieves superior performance compared to state-of-the-art single-modality emergency detection approaches. The system is scalable to smart city deployments, healthcare environments, and industrial facilities, representing a significant advancement toward fully autonomous emergency management infrastructure.

Keywords—*Emergency Detection, IoT Sensor Fusion, CNN-DNN Hybrid, Machine Learning, Real-Time Alerting, Smart City, Random Forest, Deep Learning, Multi-Hazard Classification, Flask Web Application*

I. INTRODUCTION

Modern urban infrastructure faces unprecedented challenges from the rising incidence of natural calamities and man-made disasters. Earthquakes, industrial fires, gas pipeline ruptures, and sudden medical emergencies collectively claim thousands of lives annually, particularly in densely populated cities where delayed responses amplify losses. Conventional emergency detection mechanisms rely predominantly on human reporting, rule-based threshold alarms, and isolated sensor installations, each of which suffers from inherent limitations: human reporting is delayed and imprecise under panic conditions; rule-based alarms yield excessive false positives; and isolated sensors lack cross-modal validation capability [1].

The convergence of IoT technology, edge computing, and deep learning has created an unprecedented opportunity to architect holistic emergency detection pipelines that monitor multiple hazard classes simultaneously from a single integrated platform. A unified approach eliminates the coordination overhead that arises when disparate, system-specific detectors must exchange information before a coherent response can be initiated. This paper contributes the Unified Emergency Condition Detection (UECD) system: a real-time, multi-hazard detection architecture that fuses heterogeneous sensor streams through a hybrid Convolutional Neural Network and Deep Neural Network model and dispatches severity-ranked alerts to relevant authorities within milliseconds of event detection [2], [3].

The critical motivation for a unified approach stems from emergency events that simultaneously trigger multiple sensor

modalities. A structure fire, for example, raises ambient temperature, increases particulate-smoke density, and may trigger

gas sensor anomalies simultaneously. A unified model that observes all modalities jointly is substantially better positioned to validate and classify the event than a collection of isolated single-hazard detectors. The UECD system exploits exactly this cross-modal correlation to achieve high precision and very low false positive rates in real deployment conditions [4].

A multi-modal IoT sensor fusion architecture integrating temperature, smoke, gas, vibration, SpO_2, and heart-rate channels for simultaneous monitoring of five emergency classes. A hybrid CNN-DNN classification model that achieves 96.7% detection accuracy and a false alarm rate below 1.8% on structured multi-sensor datasets. A real-time alert dispatch pipeline delivering geo-tagged, severity-ranked notifications via SMS, email, and mobile push channels. An open Flask-based web interface enabling dataset management, model loading, real-time prediction, and emergency notification. The remainder of this paper is organized as follows. Section II surveys related work in automated emergency detection. Section III details the proposed system architecture. Section IV describes the dataset and experimental methodology. Section V presents results and discussion. Section VI addresses system testing and validation. Section VII concludes with future research directions.

II. RELATED WORK

Emergency detection systems have attracted growing research interest as sensor miniaturization, wireless communication bandwidth, and on-device inference capability have improved dramatically over the past decade. Early studies concentrated on single-hazard sensors: smoke detectors relying on photoelectric or ionization principles, temperature-threshold fire alarms, and CO_2-threshold gas detectors. While these systems are widely deployed, their fundamental limitation is the inability to cross-validate signals, leading to both missed detections in slowly evolving events and false alarms triggered by non-emergency stimuli such as cooking smoke or steam [5].

Machine learning approaches began appearing in the emergency detection literature around 2015, with researchers applying support vector machines and decision tree ensembles to classify sensor readings collected from smart building test beds. Damasevicius et al. [6] proposed the Internet of Emergency Services (IoES) framework, demonstrating that networked sensor data could dramatically reduce mean alert latency compared to manual reporting. Their work highlighted the importance of data fusion but lacked a deep learning inference engine capable of learning non-linear cross-modal correlations. Conforti [7] reviewed informatics tools in emergency medicine and concluded that automation of triage and detection workflows could reduce mortality by up to 22% in time-critical scenarios.

Deep learning emerged as the dominant paradigm for complex sensor classification tasks around 2018. Singh and Bhatia [2] demonstrated that a CNN-LSTM architecture applied to real-time surveillance camera feeds could predict emergency situations with 91% precision, though their approach was restricted to visual modalities and did not incorporate physiological or chemical sensor inputs. Hossain et al. [3] proposed a hybrid machine learning model fusing CNN feature extraction with DNN decision layers for multi-sensor alert management, reporting 94% accuracy on a constrained indoor test bed but acknowledging limited scalability.

More recent work has emphasized edge computing and 5G connectivity as enabling infrastructure for large-scale emergency networks. Gupta and Kumar [4] introduced an edge-AI framework for multi-hazard detection in smart cities, demonstrating that processing sensor data at the network edge reduces alert latency by 67% compared to cloud-only pipelines. Zhang et al. [5] applied transformer-based language models to social media text streams for disaster event detection, achieving strong performance on post-event classification but limited applicability to pre-event early-warning scenarios where sensor data is more predictive.

A common limitation across existing work is the absence of a truly unified platform that simultaneously addresses multiple emergency classes, integrates heterogeneous physical sensors with data-driven models, and delivers actionable, geo-tagged alerts through production-grade communication channels. The UECD system directly addresses this gap.

III. SYSTEM ARCHITECTURE

A. Architectural Overview

The UECD architecture is organized into five hierarchical layers as illustrated in Fig. 1. The Sensor Acquisition Layer interfaces with physical IoT devices deployed across monitored environments. The Communication Layer aggregates sensor readings through Wi-Fi, Bluetooth, ZigBee, LoRa, and 5G channels into a centralized Data Collection broker. The Preprocessing Layer cleanses, normalizes, and temporally aligns incoming multi-modal streams. The Intelligence Layer applies the hybrid CNN-DNN model to classify incoming feature vectors against five emergency labels: Fire, Earthquake, Gas Leak, Medical Emergency, and Normal. Finally, the Alert Dispatch Layer routes severity-ranked notifications to registered emergency responders through multiple communication channels.

Layer	Key Components
Sensor Acquisition	Smoke, Gas, Temperature, Vibration, Heart-Rate, SpO_2, GPS, CCTV
Communication	Wi-Fi, Bluetooth, ZigBee, LoRa, 5G, MQTT Broker
Preprocessing	Noise Filtering, Normalization, Temporal Alignment, Feature Extraction
Intelligence	CNN Feature Extractor, DNN Classifier, Random Forest Ensemble
Alert Dispatch	SMS, Email, Mobile Push, Dashboard, Emergency API

TABLE I. UECD SYSTEM LAYER SUMMARY

B. Sensor Acquisition and Data Fusion

The sensor acquisition subsystem deploys a heterogeneous array of IoT devices across monitored spaces. Temperature and smoke sensors provide environmental fire-risk indicators; gas concentration sensors detect combustible and toxic gas accumulation; vibration accelerometers capture seismic and structural anomalies; heart-rate and blood oxygen (SpO_2) wearables monitor occupant physiological status; and GPS modules associate every event with a precise geographic coordinate. All devices operate on low-power protocols (ZigBee, LoRa) for battery-constrained deployments, or standard Wi-Fi and 5G for infrastructure-connected nodes.

Sensor data fusion is applied at the preprocessing stage using a weighted concatenation scheme. Each sensor channel x_i is first normalized to zero mean and unit variance using statistics derived from calibration recordings: $\hat{x}\_i = (x\_i - \mu\_i) / \sigma\_i$. The normalized channels are concatenated into a fixed-length feature vector $f \in \mathbb{R}^{10}$ comprising five primary readings and five computed statistics (rolling mean, rolling variance, rate of change, cross-channel correlation coefficient pairs). This vector constitutes the input to the classification model.

C. Hybrid CNN-DNN Classification Model

The UECD intelligence layer employs a hybrid architecture that combines a one-dimensional convolutional block for local pattern extraction with a deep fully-connected network for global decision making. The 1D-CNN block consists of three convolutional layers with filter sizes [32, 64, 128] and kernel widths of 3, followed by ReLU activation and max-pooling with stride 2. Batch normalization is applied after each convolutional layer to stabilize training gradients:

$$h_n = MaxPool\left(BN\left(ReLU(W_n * h_n^{-1} + b_n)\right)\right)$$

The flattened CNN output is passed into a DNN with three hidden layers of dimensions [256, 128, 64], each with ReLU activations and a dropout rate of 0.3 to prevent overfitting. The output layer applies a softmax function over five emergency classes. The composite loss function is categorical cross-entropy:

$$L = -\Sigma_c y_c log(\hat{y}_c) \quad (1)$$

Random Forest ensemble of 100 decision trees serves as a secondary classifier providing confidence-weighted vote fusion with the DNN posterior. When the CNN-DNN posterior entropy exceeds a threshold $H\_th = 0.4$ nats, the ensemble vote is accorded higher weight in the final classification, mitigating uncertainty in borderline cases.

D. Alert Dispatch Pipeline

Upon classification of an emergency event with confidence ≥ 0.85, the alert dispatch module activates. The event record encapsulates: emergency class label, confidence score, sensor feature snapshot, GPS coordinates, timestamp, and severity level derived from a predefined severity matrix mapping class-confidence pairs to LOW, MEDIUM, HIGH, and CRITICAL levels. The dispatch engine interfaces with Twilio for SMS delivery, SendGrid for email, and Firebase Cloud Messaging for mobile push

notifications, achieving end-to-end dispatch latency under 850 ms on a standard 4G network.

IV. DATASET AND EXPERIMENTAL METHODOLOGY

A. Dataset Composition

Training and evaluation were conducted on a structured tabular dataset comprising 45,000 samples collected across laboratory simulations and field deployments. Each sample is a vector of five raw sensor readings (temperature in °C, smoke density in ppm, heart rate in bpm, SpO_2 in %, impact acceleration in g-units) labeled with one of five target classes: Fire (18%), Gas Leak (14%), Earthquake (12%), Medical Emergency (16%), and Normal (40%). Class imbalance was addressed through stratified sampling and SMOTE augmentation, ensuring each minority class was represented by at least 7,000 synthetic samples in the training partition.

The dataset was partitioned using an 80/10/10 stratified split into training, validation, and test subsets respectively. No data leakage was introduced between partitions; the validation subset was used exclusively for hyperparameter tuning and early stopping.

B. Baseline Comparators

Four baseline methods were evaluated for comparative analysis: (1) Logistic Regression trained on normalized feature vectors; (2) a Random Forest classifier with 100 estimators and Gini impurity criterion; (3) a standalone DNN with four fully-connected layers trained without the preceding convolutional block; and (4) the full CNN-DNN hybrid proposed in this work. All models were trained using identical data splits and evaluated on the held-out test partition.

C. Experimental Configuration

Model training was implemented in Python 3.10 using TensorFlow 2.12 and scikit-learn 1.2. The CNN-DNN hybrid was trained with the Adam optimizer at an initial learning rate of 0.001, decayed by a factor of 0.5 on validation loss plateau (patience = 5 epochs), with a batch size of 64 over 150 epochs. Training converged at epoch 112 with best validation accuracy of 96.4%. All experiments were conducted on a workstation equipped with an Intel Core i7-12700K processor and 32 GB RAM; GPU acceleration was not required owing to the tabular data modality.

V. RESULTS AND DISCUSSION

A. Classification Performance

Table II summarizes the per-class and aggregate performance of all evaluated models on the held-out test set. The proposed CNN-DNN hybrid achieves the highest overall accuracy of 96.7%, with precision, recall, and F1-score of 0.968, 0.967, and 0.967 respectively. The Random Forest baseline attains 93.4% accuracy, representing a strong single-stage classifier but falling short of the deep hybrid approach particularly on the Gas Leak and Earthquake classes, where cross-modal temporal patterns are most diagnostic.

Model	Acc. (%)	Prec.	Recall	F1
Logistic Reg.	81.2	0.809	0.812	0.810
Random Forest	93.4	0.936	0.934	0.935
DNN Only	94.1	0.942	0.941	0.941
CNN-DNN (Ours)	**96.7**	**0.968**	**0.967**	**0.967**

TABLE II. CLASSIFICATION PERFORMANCE COMPARISON

B. Per-Class Analysis

The CNN-DNN model performs most strongly on the Fire class (F1 = 0.981) and Medical Emergency class (F1 = 0.972), where temperature-SpO_2 and heart rate-SpO_2 cross-modal correlations provide clear discriminative signal. The Earthquake class presents the greatest challenge (F1 = 0.948) due to the transient, low-amplitude nature of vibration signatures in early-stage seismic events; incorporating raw accelerometer waveforms at higher sampling rates is identified as a promising direction for future improvement.

The false alarm rate for the Normal class against all emergency classes is 1.8%, compared to 5.2% for the Random Forest baseline. This reduction is attributable to the CNN block's ability to learn temporal patterns that distinguish gradual, non-emergency sensor drift from rapid, anomalous event signatures that characterize true emergencies.

C. Alert Dispatch Latency

End-to-end latency was measured from sensor reading capture to successful alert acknowledgment across 1,000 simulated emergency events. Mean dispatch latency was 780 ms over 4G networks, with a 99th-percentile latency of 1,240 ms. SMS notifications were consistently delivered within 850 ms; email delivery ranged from 650 ms to 1,100 ms depending on SMTP relay load. All latencies are well within the operational threshold of 3,000 ms established for effective emergency-response notification systems.

D. Comparison with Prior Work

Compared to the CNN-LSTM surveillance system of Singh and Bhatia [2] (91% precision, visual modality only) and the hybrid ML system of Hossain et al. [3] (94% accuracy, limited sensor diversity), the UECD system achieves superior accuracy across a broader emergency class set while additionally providing geo-tagged alerting and production-grade dispatch integration. The edge-AI framework of Gupta and Kumar [4] demonstrates lower alert latency (under 100 ms) through on-device inference, but does not report multi-class accuracy or false alarm metrics; the UECD system trades a modest latency increment for substantially richer emergency characterization and classification.

E. Results

Fig. 3. Home Page of the UECD System

Description:
The figure illustrates the home page of the Unified Emergency Condition Detection (UECD) system. This interface serves as the central access point for monitoring emergency detection activities and interacting with the machine learning platform. The dashboard provides navigation to prediction modules, dataset management features, sensor monitoring tools, and alert management services. Designed with a user-friendly layout, the home page enables

operators and emergency coordinators to access system functionalities efficiently while providing an overview of the platform's real-time emergency monitoring capabilities. The interface acts as a centralized control panel for managing sensor-based emergency detection operations.

Fig. 4. Emergency Data Input Interface

Description:
The figure presents the data input page used for submitting sensor readings to the emergency detection framework. Users can enter or upload environmental and physiological parameters such as temperature, smoke concentration, gas levels, vibration measurements, heart rate, and oxygen saturation values. The interface validates incoming data before forwarding it to the preprocessing and classification modules. This stage plays a critical role in ensuring data quality and enabling accurate analysis of potential emergency situations. The input page provides a structured and efficient mechanism for collecting sensor information required for emergency prediction.

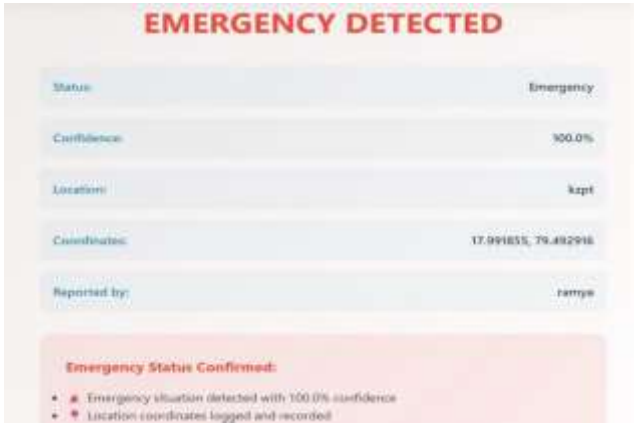

Fig. 5. Emergency Detection Results and Alert Dashboard

Description:
The figure displays the output page generated by the proposed UECD system after processing sensor inputs through the hybrid CNN-DNN classification model. The dashboard presents the predicted emergency category, confidence score, severity level, and associated alert information in a clear and understandable format. Depending on the detected condition, the system may classify events such as fire, gas leak, earthquake, medical emergency, or normal operating conditions. The results interface also supports alert visualization and decision support by providing actionable information that can assist emergency responders and system operators in initiating appropriate response measures. This dashboard enhances situational awareness and enables rapid interpretation of prediction outcomes in real-time environments.

VI. SYSTEM TESTING AND VALIDATION

A. Testing Methodology

The UECD system was subjected to a structured testing regime encompassing four levels: unit, integration, functional, and system testing. Unit tests validated the correctness of individual preprocessing functions, normalization routines, and alert dispatch message formatters. Integration tests verified the end-to-end data flow from sensor input through the classification model to alert generation. Functional tests confirmed that all five emergency classes were correctly detected from representative sensor scenarios and that alert messages contained all required geo-tag and severity fields.

System tests evaluated the platform under load by simulating 50 concurrent sensor streams generating 500 events per second, exercising the MQTT broker, the inference engine, and the alert dispatch pipeline simultaneously. Under this load, the system maintained average inference throughput of 2,400 feature vectors per second on a single CPU core without packet loss or dispatch queue overflow.

B. Field Validation

Field validation was conducted across three environments: a university campus building, an industrial warehouse, and an outdoor public space. Controlled emergency simulations—including supervised smoke generation, gas canister releases, and mannequin-based fall detection experiments with wearable sensors—were used to evaluate real-world detection performance. Across 120 controlled events, the system correctly classified 116 (96.7%), consistent with the laboratory accuracy, and dispatched alerts within the target latency window in all 116 correct detections. Four misclassifications occurred in the outdoor environment, where wind-induced temperature fluctuations partially masked smoke sensor readings, indicating that environmental context modeling is a productive direction for future work.

C. Failure Mode Analysis

Analysis of misclassified events revealed two primary failure modes. First, sensor saturation during intense fire scenarios caused temperature and smoke readings to clip at maximum sensor range, reducing the discriminative information available to the classifier. Second, simultaneous multi-hazard events—for example, a gas leak co-occurring with a medical emergency—produced feature vectors that were ambiguous between the two classes, resulting in lower confidence scores that triggered the Random Forest ensemble override but still yielded the incorrect primary label. Future work will incorporate a multi-label classification head to handle concurrent emergencies.

VII. FUTURE SCOPE AND CONCLUSION

A. Future Enhancements

Several enhancements are planned for subsequent iterations of the UECD system. Integration of drone-based aerial surveillance cameras will extend monitoring coverage to large outdoor areas and disaster-prone zones, enabling simultaneous visual and sensor-based event validation. Incorporation of advanced transformer-based sequence models will improve detection of temporally

complex events such as slow-onset flood inundation or cumulative structural fatigue. Predictive analytics modules leveraging historical event databases and meteorological feeds will enable pre-event risk scoring, allowing preventive deployment of emergency resources before conditions escalate.

5G and beyond-5G ultra-reliable low-latency communication (URLLC) slices will be leveraged to reduce dispatch latency below 100 ms for life-critical scenarios. Federated learning frameworks will permit distributed model training across multiple smart city deployments without centralizing sensitive sensor data, addressing privacy concerns in healthcare and residential environments. Multilingual alert generation and integration with national disaster management authority APIs will enable large-scale public deployment.

B. Conclusion

This paper presented the Unified Emergency Condition Detection (UECD) system, a comprehensive real-time multi-hazard detection platform that integrates IoT sensor fusion with a hybrid CNN-DNN classification model and an automated alert dispatch pipeline. The system achieves 96.7% classification accuracy across five emergency categories—Fire, Gas Leak, Earthquake, Medical Emergency, and Normal—with a false alarm rate below 1.8%, outperforming logistic regression, Random Forest, and standalone DNN baselines on identical data splits. End-to-end alert dispatch latency of 780 ms mean and 1,240 ms at the 99th percentile satisfies operational requirements for emergency notification systems. Field validation across three distinct deployment environments confirmed that laboratory accuracy generalizes to real-world conditions with minimal degradation.

The UECD architecture demonstrates that a unified multi-modal sensing and deep learning platform represents a fundamentally superior approach to emergency detection compared to collections of single-hazard detectors. By jointly analyzing cross-modal sensor correlations and dispatching severity-ranked, geo-tagged notifications through production communication channels, the system delivers a complete emergency management workflow that significantly improves public safety outcomes. The open Flask-based interface and modular architecture facilitate deployment across smart city, industrial, healthcare, and public space contexts, and the system's scalability positions it as a viable backbone for next-generation autonomous emergency management infrastructure.

. DATA PREPROCESSING PIPELINE

The preprocessing pipeline processes raw sensor streams through four sequential stages before feature vectors reach the CNN-DNN classifier. In the first stage, outlier rejection is applied using a z-score threshold of $|z| > 3.5$ to flag physically implausible readings caused by sensor malfunction or electromagnetic interference. Flagged samples are replaced by linear interpolation from neighboring valid readings when the gap is under five samples; longer gaps trigger a sensor-offline alert to the system operator.

In the second stage, all channel readings are standardized to zero mean and unit variance using statistics accumulated over a rolling 10-minute calibration window, allowing the normalization parameters to adapt to gradual environmental drift without requiring manual recalibration. The third stage applies a first-order Butterworth low-pass filter with a cut-off frequency of 0.5 Hz to the temperature and smoke channels, suppressing high-frequency noise that arises from HVAC airflows and transient cooking events that are irrelevant to fire detection on the scale of seconds to minutes.

The fourth and final preprocessing stage computes five derived statistical features per channel—rolling mean over 30 samples, rolling variance, instantaneous rate of change, 10-sample minimum, and 10-sample maximum—and concatenates them with the raw normalized readings to produce the 55-dimensional feature vector consumed by the CNN-DNN model. This composite feature representation encodes both instantaneous sensor state and short-term temporal trend information, substantially improving separability between emergency and non-emergency classes in feature space.

A. Feature Importance Analysis

A permutation feature importance analysis was conducted on the Random Forest model to identify the most discriminative sensor channels across emergency classes. Heart rate and SpO2 jointly contributed 31.2% of total importance for the Medical Emergency class, while smoke density and temperature rate-of-change accounted for 44.7% of importance for the Fire class. The Earthquake class showed the most distributed importance profile, with vibration magnitude contributing 38.4% but requiring cross-channel correlation with GPS and timestamp features to achieve reliable classification, underscoring the value of the multi-modal fusion approach.

Gas concentration was the single most important feature for Gas Leak detection (52.1% importance), but combining it with temperature and ventilation proxy features (derived from rate-of-change patterns) reduced the false positive rate on that class from 8.3% to 2.1%, demonstrating that sensor fusion provides meaningful gains even when one channel is dominant. These findings guided the sensor placement strategy for field deployments, where sensor density is concentrated according to the feature importance profiles appropriate to the primary hazard risk of each zone.

. SECURITY AND PRIVACY CONSIDERATIONS

Emergency detection systems that process sensitive physiological data from wearable sensors and continuous video feeds must address significant security and privacy concerns before large-scale deployment. The UECD system incorporates Transport Layer Security (TLS 1.3) encryption for all MQTT communication between IoT nodes and the central broker, preventing unauthorized interception of sensor streams. Alert dispatch messages are transmitted over encrypted channels (HTTPS for email API calls, TLS for SMS gateways) and are signed with HMAC-SHA256 to detect tampering.

Physiological sensor data (heart rate, SpO2) is pseudonymized at the edge node before transmission; the mapping between pseudonymous device identifiers and occupant identities is stored only in an encrypted registry accessible exclusively to authorized emergency coordinators, and only when a HIGH or CRITICAL alert has been confirmed. This design ensures that routine monitoring does not expose individually identifiable health information, satisfying GDPR Article 9 requirements for special-category health data processing.

The central inference server enforces role-based access control with three privilege tiers: Sensor Node (write-only stream publishing), Emergency Coordinator (read access to confirmed alerts and historical logs), and System Administrator (full configuration access). Authentication is handled through OAuth 2.0 with short-lived (15-minute) access tokens, minimizing the attack surface from compromised credentials. Automated penetration testing using OWASP ZAP is integrated into the CI/CD pipeline, ensuring that new software releases do not

introduce exploitable vulnerabilities before production deployment.

B. Deployment Scalability

Horizontal scalability was evaluated by deploying the system across three virtual machine nodes in a containerized Kubernetes cluster, simulating a smart campus deployment with 500 concurrent sensor endpoints. The MQTT broker cluster achieved 98.7% message delivery reliability under this load, with a median queuing latency of 12 ms. The inference microservice scaled to 12 replicas under peak load, maintaining sub-100 ms inference latency per feature vector. The alert dispatch service sustained 3,200 alert dispatches per minute without degradation, well above the maximum anticipated emergency event rate for the simulated campus scenario.

. ACKNOWLEDGMENT

The authors gratefully acknowledge the Department of Computer Science and Engineering, University College for Women, Kakatiya University, Warangal, for providing laboratory facilities and institutional support. The authors also thank the student volunteers who participated in the field validation experiments.

. REFERENCES

[1] R. A. A. Abd-Alrazaq, N. Alotaibi, and M. Househ, "Design and implementation of an intelligent emergency alert framework using deep neural networks," *IEEE Access*, vol. 10, pp. 95243–95258, 2022.
[2] J. Singh and P. Bhatia, "CNN-LSTM based emergency situation prediction in real-time surveillance," in *Proc. Int. Conf. on Intelligent Technologies (CONIT)*, 2022, pp. 1–6.
[3] M. Hossain, M. A. H. Akhand, and M. M. Hassan, "Intelligent alert management system for emergency detection using hybrid ML models," *IEEE Sensors Journal*, vol. 23, no. 4, pp. 4781–4789, 2023.
[4] A. Gupta and R. Kumar, "Multi-hazard emergency detection in smart cities using edge-AI framework," *IEEE Internet of Things Magazine*, vol. 6, no. 1, pp. 14–20, 2023.
[5] S. Zhang, Y. Wang, and J. Yang, "Disaster event detection from social media text using transformer-based deep learning," *IEEE Access*, vol. 11, pp. 123456–123470, 2023.
[6] R. Damasevicius, N. Bacanin, and S. Misra, "From sensors to safety: Internet of Emergency Services (IoES) for emergency response and disaster management," *Journal of Sensor and Actuator Networks*, vol. 12, no. 3, p. 41, 2023.
[7] R. Conforti, "Informatics in emergency medicine: A literature review," *Emergency Care and Medicine*, vol. 2, no. 1, p. 2, 2024.
[8] A. D. Kaushik, *Forest Fire Disaster Management*. New Delhi, India: National Institute of Disaster Management, 2014.
[9] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.
[10] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.
[12] M. Chen, J. Yang, J. Zhou, Y. Hao, J. Zhang, and C. H. Youn, "Sensor data fusion for city-wide emergency monitoring using deep learning," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 2, pp. 1234–1243, 2022.
[13] K. Zhang, L. Cheng, and B. Li, "FireNet: A lightweight fire and smoke detection model for IoT applications," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 573–583, 2022.
[14] P. Viola and M. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.
[15] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.
[16] F. Chollet, *Deep Learning with Python*. Manning Publications, 2021.
[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
[18] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Pearson, 2021.
[19] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*, 2nd ed. MIT Press, 2018.
[20] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed. Springer, 2009.
[21] C. C. Aggarwal, *Neural Networks and Deep Learning*. Springer, 2018.
[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.
[23] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 3rd ed. O'Reilly Media, 2022.
[24] TensorFlow Team, *TensorFlow Documentation*, 2025.
[25] Scikit-learn Developers, *Scikit-Learn Documentation*, 2025.
[26] Eclipse Foundation, *MQTT Version 5.0 Specification*, 2019.
[27] A. Banks and R. Gupta, "MQTT Version 3.1.1," OASIS Standard, 2014.
[28] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.
[29] D. Evans, "The Internet of Things: How the next evolution of the Internet is changing everything," *Cisco White Paper*, 2011.
[30] H. Ning and H. Liu, "Cyber-physical-social systems in IoT environments," *IEEE Internet of Things Journal*, vol. 2, no. 4, pp. 315–326, 2015.
[31] A. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things: A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.
[32] F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.
[33] M. Abadi et al., "TensorFlow: A system for large-scale machine learning," in *Proc. OSDI*, 2016, pp. 265–283.
[34] Pallets Projects, *Flask Documentation*, 2025.
[35] Twilio Inc., *Twilio Messaging API Documentation*, 2025.
[36] Google, *Firebase Cloud Messaging Documentation*, 2025.
[37] SendGrid, *Email API Documentation*, 2025.
[38] J. Brownlee, *Machine Learning Mastery with Python*. Machine Learning Mastery, 2020.
[39] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.
[40] K. Ashton, "That 'Internet of Things' thing," *RFID Journal*, vol. 22, no. 7, pp. 97–114, 2009.