Software: Practice and Experience

WILEY

RESEARCH ARTICLE

Early Response Framework for Accident Detection and Prevention Through Multi-Zone Fog-Cloud Collaboration for Safety-Critical Applications

Moumita Mishra[1] | Soumya Kanti Ghosh[2] | Bhargab Maitra[3] | Rajkumar Buyya[4]

[1]Ranbir And Chitra Gupta School of Infrastructure Design and Management, Indian Institute of Technology Kharagpur, West Bengal, India | [2]Department of Computer Science & Engineering, Indian Institute of Technology Kharagpur, West Bengal, India | [3]Department of Civil Engineering, Indian Institute of Technology Kharagpur, West Bengal, India | [4]School of Computing and Information Systems, The University of Melbourne, Melbourne, Australia

Correspondence: Moumita Mishra (moumitamishra15@kgpian.iitkgp.ac.in)

Received: 31 January 2025 | **Revised:** 15 May 2025 | **Accepted:** 23 May 2025

Funding: The authors received no specific funding for this work.

Keywords: accident prediction | accident prone area | early response accident detection | energy efficiency | fog computing | machine learning | QGIS software

ABSTRACT

Background: Early response systems and efficient resource allocation are vital for ensuring timely and reliable accident detection and prevention, especially in high-density urban environments such as Kolkata, India. Traditional cloud-based systems often face challenges in latency, energy consumption, and execution cost.

Methods: This study proposes an Early Response, Zone-Based Accident Detection and Resource Allocation framework utilizing a multi-tier fog–cloud architecture. Historical accident data from 2017 to 2023 are used to identify accident-prone zones. Advanced machine learning models, including Decision Tree, Random Forest, and XGBoost, are employed to predict accident counts. Optimal emergency routes to nearby ambulance centers, police stations, and hospitals are calculated using a shortest-path algorithm. The framework is implemented and evaluated using the iFogSim2 simulator. Spatial analysis and visualization are performed using Quantum Geographic Information System (QGIS).

Results: The proposed architecture shows significant performance improvements, including a 16.77% reduction in energy consumption and an 89.66% reduction in execution cost. Latency and execution times are improved by 11%–20% and 12.08%, respectively. The predictive models demonstrate high accuracy in forecasting accident counts, supporting efficient emergency resource deployment.

Conclusion: The zone-prioritized fog computing framework enhances emergency response efficiency by minimizing latency and energy consumption while offering scalable, real-time accident detection capabilities. The integration of spatial visualization and predictive modeling provides policymakers and urban planners with critical, actionable insights to improve road safety across metropolitan regions. This solution offers a promising direction for safety-critical applications in smart city environments.

1 | Introduction

Worldwide, road safety is a critical public health concern, profoundly affecting societal stability and human welfare. The World Health Organization's 2018 Global Status Report on Traffic Safety indicates that more than 1.35 million deaths occur each year as a result of traffic accidents. Developing nations, such as India, account for about 90% of these deaths, underscoring an urgent necessity for extensive safety measures. In India, traffic accidents cause more than four hundred fatalities per day, highlighting the need for sophisticated solutions for effective mitigation [1, 2].

The proliferation of mobile devices, sensors, and terminal nodes has dramatically increased the demand for efficient resource allocation in real-time applications, especially in fields requiring rapid data processing and low latency [3]. Despite their enhanced functionalities, modern mobile devices often face constraints such as lower-speed processors and limited battery life, which create significant challenges in maintaining both performance and energy efficiency. Some approaches, like instruction-level parallelism, power control, and runtime voltage scaling, have been proposed to mitigate these issues. However, these advanced hardware technologies are often cost-prohibitive, especially for budget-constrained projects. This limitation is particularly relevant for Internet of Things (IoT) applications, where multiple devices continuously generate and transmit vast amounts of data, placing further strain on energy and computational resources.

Cloud computing has become a popular solution for enhancing device performance by offloading computations to remote servers. However, this introduces new challenges, such as increased energy consumption and execution delays due to the need for constant data transmission between mobile devices and cloud servers. This is especially problematic for latency-sensitive applications like accident detection, where an early response alert is crucial. In scenarios such as smart cities, autonomous driving, and transportation systems, cloud-based approaches can struggle to meet the demands of low-latency services and energy efficiency.

Fog computing, a decentralized computing infrastructure, addresses these challenges by distributing computational power and application services closer to the data source [4]. This approach allows for the processing and analysis of data locally or regionally, significantly reducing the amount of data that needs to be sent to the cloud for processing [5]. Fog computing is particularly advantageous for latency-sensitive IoT applications, such as real-time gaming, streaming, and, most importantly, accident detection and prevention systems [6]. By processing data at the edge of the network—closer to where it is generated—fog computing reduces response times, minimizes energy consumption, and enhances system efficiency.

Various tasks in accident detection and prevention systems can be efficiently executed using fog computing. Traffic monitoring involves continuous processing of data streams from cameras and sensors at fog nodes, which can detect anomalies like sudden stops, high traffic density, or accidents, and send early response alerts. For accident detection, fog instances process sensor data from vehicles and roadside units to quickly identify accidents and trigger emergency protocols such as notifying nearby ambulance centers, police stations, and hospitals. Weather and environmental data processing at fog nodes helps assess road conditions and predict accident-prone scenarios, improving safety through more accurate alerts. Additionally, fog nodes enable spatio-temporal analysis of historical accident data, correlating it with real-time traffic and environmental factors to predict high-risk areas and allocate resources efficiently, ensuring faster responses in accident-prone zones.

In fog computing, resource allocation refers to the efficient distribution of available computational resources (CPU, memory, bandwidth) without the need for additional hardware [7, 8]. This is crucial for achieving high performance, maintaining user satisfaction, and ensuring optimal resource utilization. However, fog computing faces its own challenges, including resource management, scalability, energy consumption, job scheduling, and load balancing [9]. Managing these challenges is essential, especially for mission-critical systems like accident detection, where delays can have severe consequences.

In contrast to cloud-based systems, where all data is transmitted to and processed in centralized data centers, fog computing processes data closer to the edge. This significantly reduces latency and energy consumption, making it a more efficient option for real-time accident detection and prevention systems [10]. In such systems, accident detection requires immediate emergency responses, and relying solely on cloud architecture can introduce unacceptable delays [11]. Several studies have demonstrated the effectiveness of fog computing in reducing latency, energy consumption, and bandwidth usage in IoT networks [12], highlighting its potential for real-time data processing.

Additionally, fog-based frameworks have been successfully applied in health monitoring systems [13, 14], where they have improved response times and resource utilization. These benefits can be transferred to accident detection systems, where fog computing's decentralized architecture ensures faster medical responses by processing critical data closer to where accidents occur [15].

The paper proposes an early response, zone-specific machine learning based accident detection and dynamic resource allocation framework utilizing the iFogSim2 [16]. It identifies accident-prone areas. We have used crash data from 2017 to 2023 for accident prediction. The system incorporates the machine learning models (linear regression, lasso regression, ridge regression, support vector regression, KNN regression, decision tree regression, random forest regression, and XGBoost regression) in the cloud to predict accident-prone zones using the crash data. Out of all models, Decision Tree Regressor, Random Forest Regressor, and XGBoost Regressor have demonstrated good results with high r2 scores. Additionally, the edge layer integrates the shortest path algorithm to identify the nearest ambulance centers, police stations, and hospitals, and the system can leverage a mobile gateway or SMS gateway to send immediate emergency notifications. We have used QGIS software to visualize the accident distribution of hotspots. In the iFogSim2 framework, fog agents act as intermediaries between terminal nodes (e.g., sensors, cameras) and the fog server manager, enabling localized data processing and resource management. This hierarchical deployment model reduces latency, energy consumption,

and execution costs while enhancing the speed and efficiency of emergency responses. By processing data closer to the source, our system has the potential to save lives by reducing the time it takes to detect accidents and respond to emergencies.

1.1 | Motivation and Contribution

In safety-critical applications like road safety, a rapid detection and prevention system is essential to efficiently mitigate accident risks and respond to accidents in an early response. Road safety systems require low-latency, high-reliability solutions that quickly detect potential risks, forecast accident-prone zones, and activate emergency actions. Traditional centralized models frequently exhibit deficiencies owing to elevated latency and resource limitations, highlighting the necessity for distributed frameworks such as fog computing. Fog-based architectures provide the decentralization of processing near the data source, hence enhancing quick detection and reaction, which improves overall safety and may save lives.

The key *contributions* of this paper include:

- *Early Response, Zone-based Accident Detection and Resource Allocation*: We propose and design a multitier architecture within a fog computing framework, designed to enable efficient, early response accident detection across multiple geographic zones. This architecture supports dynamic resource allocation, optimizing computational resources to address high-demand areas and reduce response times.
- *Accident Prediction using Historical Data in the Cloud with Machine Learning*: We develop and deploy cloud-hosted machine learning models that predict accident-prone areas using historical data. This predictive capability facilitates proactive accident management by preemptively identifying high-risk zones.
- *Fog-based Simulation for Enhanced Efficiency*: Using the iFogSim framework [17], we simulate and validate our multitier fog computing architecture. This simulation demonstrates the efficiency of our design in terms of latency reduction and energy optimization within a fog-cloud integrated environment, highlighting the scalability and sustainability of the system.

The remainder of this paper is organized as follows. Section 2 reviews recent advancements in fog computing applications, specifically within accident detection and prevention systems. In Section 3, we introduce our proposed early response accident detection and prevention mechanism, detailing its architecture and algorithms. Section 4 provides a comprehensive performance evaluation, including hotspot analysis, predictive modeling results, and simulation outcomes. Finally, Section 5 concludes the paper, discussing the implications of our findings and potential directions for future research.

2 | Related Work

Fog computing is increasingly recognized as a valuable extension of cloud computing, offering the benefit of decentralized data processing close to the source, which is particularly advantageous for latency-sensitive and resource-intensive applications such as accident detection and prevention systems [18, 19]. By minimizing the need to offload tasks to centralized cloud servers, fog computing reduces latency and bandwidth usage, making it suitable for real-time IoT networks where immediate responses are essential [20, 21]. Previous studies have highlighted the success of fog-based frameworks in healthcare, demonstrating significant improvements in response times and resource utilization [22]. For example, SVM has been utilized to predict the severity of aircraft damage [23], and various transportation resource allocation strategies have been explored to enhance performance based on latency and energy efficiency [24, 25]. Authors developed an AI-based stock price prediction framework using serverless cloud computing [26–28]. Authors have used federated learning techniques for crop prediction. Real-time data processing frameworks integrating fog and cloud environments have been proposed to improve resource allocation [29], including approaches using learning automata for optimization [30]. Systems such as fog-based pedestrian and vehicle (PV) alert systems have been developed to enhance road safety by alerting pedestrians and drivers in real-time [31, 32]. To maintain quality of service (QoS) and experience (QoE), some studies have employed multipath transmission protocols for efficient road accident management [33]. The authors explored spatio-temporal multimodal data for accident count prediction in Kolkata city, highlighting the effectiveness of integrating spatial, temporal, and contextual factors for improved forecasting accuracy and used association rule mining and machine learning techniques to unveil hidden patterns in traffic data, contributing to the development of intelligent transportation systems [34, 35].

In smart transportation, fog-cloud-IoT frameworks employing machine learning algorithms have been deployed to support data-driven insights [36, 37], while architectures integrating RNN and LSTM models have focused on predicting faults in fog computing systems proactively [38]. Hybrid fog-cloud solutions for real-time transportation systems [39] and smartphone applications using cloud-fog environments to recommend safe driving speeds [40] are among the innovations presented. Other studies have reduced traffic data streams through clustering techniques like DBSCAN and the OPTICS algorithm [41], while fog-cloud setups have been applied to smart city and traffic management scenarios [42, 43].

Moreover, hierarchical fog-cloud architectures have been proposed for post-disaster management [44], ensuring prompt alerts and efficient response mechanisms, such as ambulance dispatches upon abnormality detection [45]. In healthcare, cloud-fog-edge frameworks have been developed to improve data management and resource allocation [46, 47], and similar architectures have been used in disaster management [48].

Despite these advances, many existing fog computing systems do not consider zone-based priority resource allocation, often treating all fog nodes uniformly. This neglects the opportunity to prioritize accident-prone zones dynamically. Our proposed system fills this gap by implementing a zone-based, priority-driven resource allocation framework that processes data according to accident frequency in specific areas. This ensures that high-risk

TABLE 1 | Summary of existing works.

Paper (author, year)	Zone-based	Local processing	Early response	Forecasting	Collaborative
Dhingra et al., 2021	×	✓	✓	×	×
Peixoto et al., 2021	×	✓	✓	×	×
Atiq et al., 2023	×	✓	✓	×	×
Saini et al., 2023	×	✓	✓	×	×
Lin et al., 2023	×	✓	×	×	×
Ebrahim et al., 2024	×	✓	✓	×	×
Mukherjee at al., 2024	×	✓	×	×	×
Dey et al., 2024	×	✓	×	×	×
Verma et al., 2025	×	✓	×	×	×
Wang et al., 2025	×	✓	×	×	×
Sunku et al., 2025	×	✓	×	×	×
Proposed work	✓	✓	✓	✓	✓

TABLE 2 | Regression results (2017–2020 dataset) using grid search with weather features.

Model name	MAE	RMSE	R2 score
Linear Regressor	15.03	17.90	0.004
Lasso Regressor	15.01	17.89	0.003
Ridge Regressor	13.03	17.09	0.003
SVR	12.57	19.24	0.040
KNN Regression	14.06	17.43	0.007
Decision Tree	1.09	6.09	0.86
Random Forest Regressor	2.34	5.98	0.91
XGBoost Regressor	1.09	4.77	0.93

TABLE 3 | Regression results (2021–2023 dataset) using grid search with weather features.

Model name	MAE	RMSE	R2 score
Linear Regressor	20.67	25.11	0.006
Lasso Regressor	20.61	26.10	0.003
Ridge Regressor	20.50	25.07	0.002
SVR	17.07	23.25	0.127
KNN Regression	16.23	20.60	0.36
Decision Tree	7.06	12.18	0.81
Random Forest Regressor	6.30	9.08	0.90
XGBoost Regressor	2.30	4.07	0.92

zones receive immediate attention, improving response times and reducing delays.

Additionally, most existing solutions rely on a uniform fog or cloud architecture. In contrast, our system integrates both cloud and fog components with an additional layer of foglets and fog agents. This hierarchical, multitier architecture allows real-time processing of roadside sensor data across four distinct zones. By strategically deploying sensors and actuators, our system adapts dynamically to zonal traffic conditions and accident patterns, optimizing energy usage and enhancing system responsiveness for high-priority incidents.

TABLE 4 | Actual vs. predicted accident counts.

Segment-id	Actual accident point	Predicted accident point
Si	22	22.20952
Sj	9	9.07
Sk	23	23
Sl	12	12.64
Sm	7	6.98
Sn	38	38.49
Sp	32	32
St	15	14.25

In summary, while existing research (refer Table 1) highlights the potential of fog computing for early response applications, there is a gap in addressing zone-based resource allocation and multi-tier architectures. Our work provides a scalable, zone-prioritized solution for accident detection, combining localized processing with dynamic resource allocation to outperform traditional cloud-only and uniform fog computing models (see Tables 2–6).

3 | Proposed Early Response Framework and Methodology

The proposed four-layer multizone foglet-based accident detection paradigm for the transportation sector is illustrated in Figure 1. A motivating scenario of the SMS alert message passing scenario has also been provided. Figure 2

Figure 3 provides a flow diagram that outlines the working model of the proposed system.

TABLE 5 | Entity table 1.

Physical entity	Processing capability (MIPS)	RAM (MB)	Uplink bandwidth	Downlink bandwidth	Level
Cloud	10000	2048	100000	100000	0
Foglet	1000	1024	10000	10000	1
fog instance	500	512	5000	5000	2
Actuator (LED)	400	256	2500	2500	3
Sensor (Camera)	200	128	1000	1000	3

TABLE 6 | Network latency between entities.

Source	Destination	Latency (in milliseconds)
F1C1	FogDev1	4
F1C2	FogDev1	4
F1C3	FogDev1	4
F1C4	FogDev1	4
Actuator1	FogDev1	4
F2C1	FogDev2	4
F2C2	FogDev2	4
F2C3	FogDev2	4
F2C4	FogDev2	4
Actuator2	FogDev2	4
F3C1	FogDev3	4
F3C3	FogDev3	4
F3C4	FogDev3	4
Actuator3	FogDev3	4
F4C3	FogDev4	4
F4C4	FogDev4	4
Actuator4	FogDev4	4
FogDev2	Foglet1	8
FogDev3	Foglet2	8
Foglet1	Cloud	10
Foglet2	Cloud	10

3.1 | System Architecture and Methodology

The proposed paradigm comprises four layers that work together to provide an efficient and scalable accident detection system:

- *Terminal Nodes (Layer 3)*: Sensors and cameras are deployed across different geographic zones to collect data related to potential accidents. These terminal nodes continuously gather real-time information and send it to the edge layer.
- *Fog Instances (Layer 2)*: Fog instances, located closer to the source of data, process and evaluate accident alerts. They play a key role in ensuring low-latency responses by handling urgent local processing.
- *Foglets (Layer 1)*: Foglets aggregate and process data from multiple fog instances. They perform deeper analysis and manage resources among fog instances, ensuring efficient task distribution and processing.
- *Cloud Servers (Layer 0)*: Cloud servers provide large-scale processing and storage capabilities. They handle long-term data analytics, large-scale accident detection algorithms, and scalable computing resources for complex tasks.

3.2 | Mathematical Models and Definitions

The system components are mathematically described as follows:

Sensor Node Sets (S):

$$S = \{S_1, S_2, S_3, \ldots, S_n\} \tag{1}$$

where S_i represents a sensor node that collects accident-related data.

Camera Node Set (C):

$$C = \{C_1, C_2, C_3, \ldots, C_n\} \tag{2}$$

where C_i represents a camera node for capturing accident information.

Zone Type Set (R):

$$R = \{R_E, R_W, R_N, R_S\} \tag{3}$$

where each R_i refers to a specific geographic zone, corresponding to areas where accident data is collected.

Definition 1 (Camera/Sensor/Actuator Node). The set defines a sensor node.

$$\{S_i, R_i, type_i, userID_i, appID_i\}, \quad 1 \le i \le n \tag{4}$$

where S_i is the node ID, R_i is the zone, and $type_i$ refers to the type of application or data being collected.

An actuator node is similarly defined by:

$$\{A_i, R_i, userID_i, appID_i, type_i\}, \quad 1 \le i \le n \tag{5}$$

Fog Instance Set (F):

$$F = \{F_1, F_2, F_3, \ldots, F_k\} \tag{6}$$

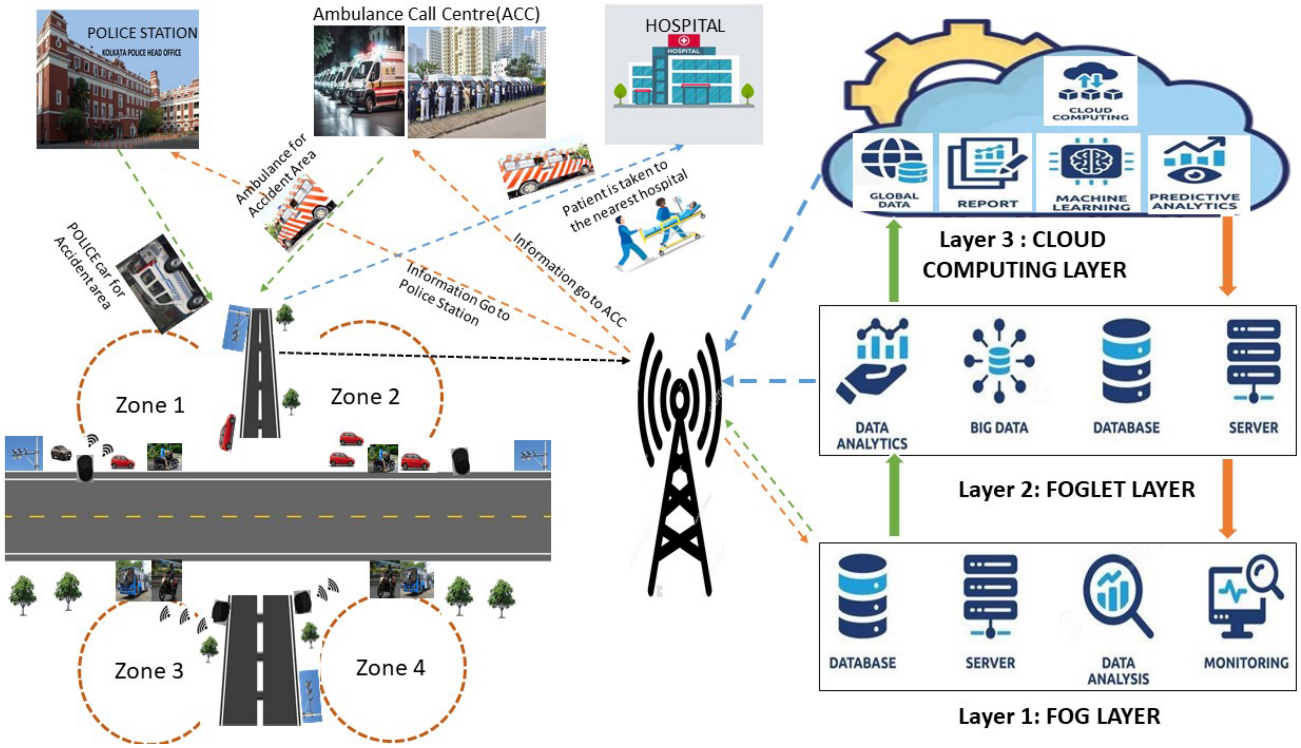

FIGURE 1 | Proposed Four-Layer Multi-zone Foglet-Based Accident Detection Paradigm for the Transportation Sector.

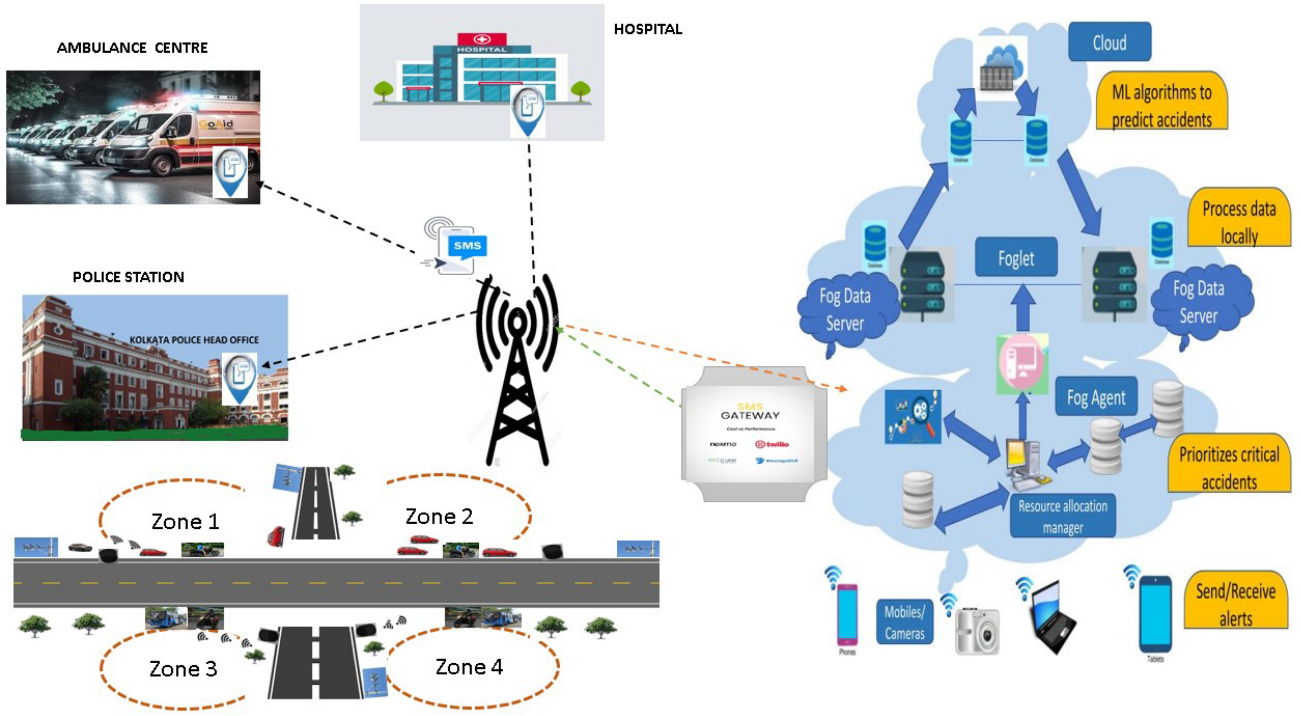

FIGURE 2 | SMS alerting system for identifying accidents in the transportation sector.

where F_i represents a fog instance responsible for processing data from sensors and cameras in the edge layer.

Hardware Specification of Fog Instances (H_n): Hardware specifications for fog instances are defined as:

$$H_n = \{H_{n1}, H_{n2}, \ldots, H_{nk}\}, \quad k \text{ is the number of fog instances} \tag{7}$$

Spatio-Temporal Accident Dataset (ST_A): The spatio-temporal dataset for fog instances is defined as:

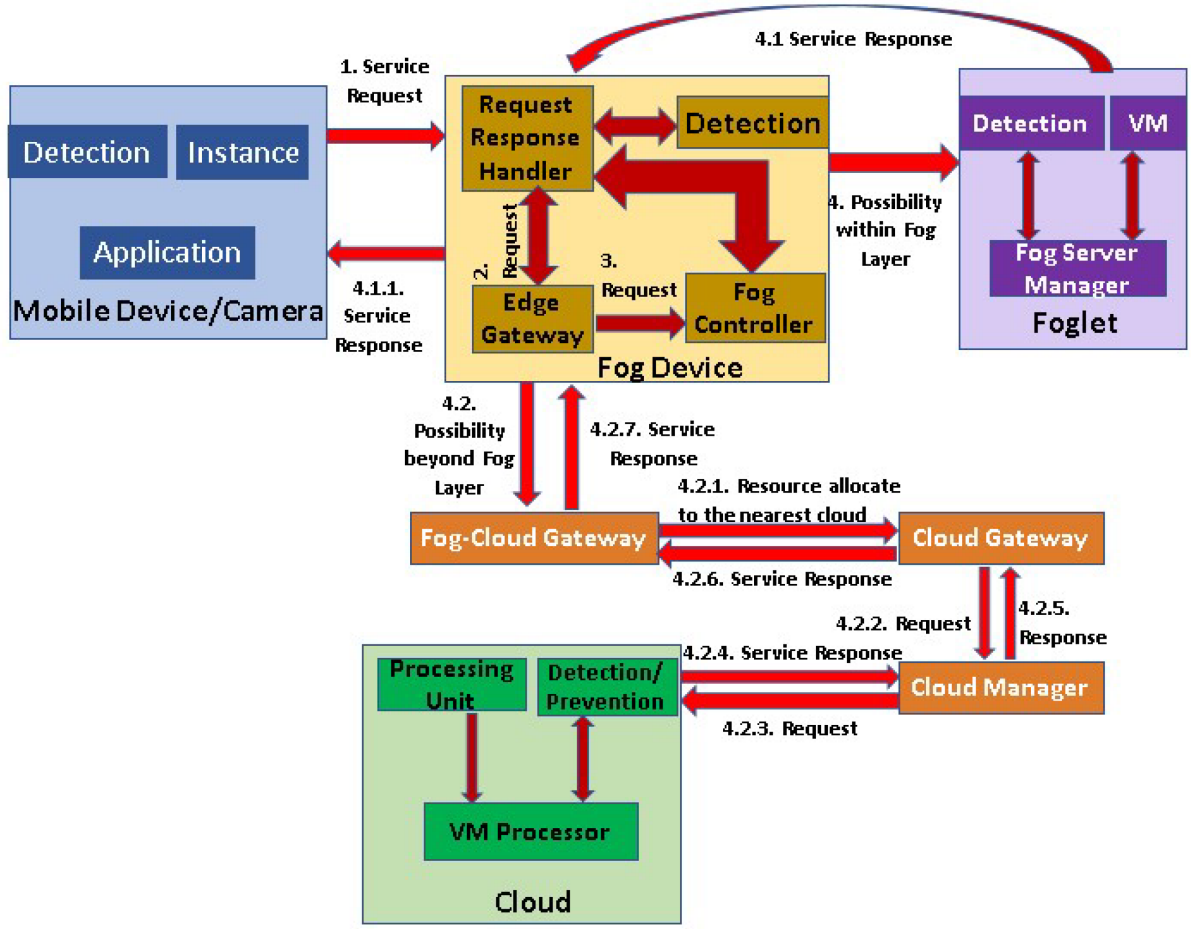

FIGURE 3 | Flow diagram of the proposed framework.

$$ST_{An} = \{ST_{A1}, ST_{A2}, \ldots, ST_{Ak}\},$$
$$k \text{ is the number of fog instances} \tag{8}$$

Definition 2 (Fog Instances). A fog instance is defined by the set:

$$\{F_j, H_{nj}, ST_{An}j, MIPS_j, RAM_j, UpBw_j, DnBw_j, Level_j, BsPw_j, IdlPw_j\}, \quad 1 \le j \le k \tag{9}$$

where F_j represents the fog instance ID, H_{nj} denotes hardware specifications, and $ST_{An}j$ denotes spatio-temporal accident data.

Network Specifications for Fog Instances (N_j): Network parameters are defined as:

$$N_j = \{MIPS_j, RAM_j, UpBw_j, DnBw_j, Level_j, BsPw_j, IdlPw_j\} \tag{10}$$

Mapping from Sensor Nodes to Fog Instances: The mapping from sensor or actuator nodes in Layer 3 to fog instances in Layer 2 is denoted as:

$$Map_{32}^{(\cdot)} : S' \rightarrow F_j \tag{11}$$

where S' is a subset of sensor nodes mapped to fog instance F_j.

Foglet Set (F_l):

$$F_l = \{F_{l1}, F_{l2}, F_{l3}, \ldots, F_{lm}\} \tag{12}$$

where F_{li} represents a foglet managing data from fog instances in its zone.

Hardware Specifications of Foglets (H_f): The hardware specifications for foglets are defined as:

$$H_f = \{H_{f1}, H_{f2}, \ldots, H_{fm}\}, \text{there are} \quad m \text{ foglets} \tag{13}$$

Network Specifications of Foglets (N_f): Network parameters for foglets are defined as:

$$N_f = \{MIPS_p, RAM_p, UpBw_p, DnBw_p, Level_p, BsPw_p, IdlPw_p\} \tag{14}$$

Definition 3 (Foglet). A foglet is defined as:

$$\{F_l^p, H_f^p, N_f^p\}, \quad 1 \le p \le m \tag{15}$$

where F_l^p represents the foglet ID, H_f^p denotes the hardware specification, and N_f^p denotes the network characteristics.

Mapping from Fog Instances to Foglets: The mapping from fog instances in Layer 2 to foglets in Layer 1 is denoted as:

$$Map_{21}^{(\cdot)} : F' \rightarrow F_l^p \tag{16}$$

where F' is a subset of fog instances mapped to foglet F_l^p.

Cloud Computing Instance Set (C):

$$C = \{C_1, C_2, C_3, \ldots, C_r\} \tag{17}$$

where C_i denotes a cloud computing instance responsible for large-scale processing and storage.

Definition 4 (Cloud Server). A cloud server is defined by:

$$\{C_t, \{P_t\}\}, \quad 1 \leq t \leq r \tag{18}$$

where C_t represents the cloud instance ID and $\{P_t\}$ represents the processing unit IDs for that instance.

Mapping from Foglets to Cloud Server: The mapping from foglets in Layer 1 to cloud instances in Layer 0 is denoted as

$$Map_{10}^{(\cdot)} : F_l' \rightarrow C' \tag{19}$$

where F_l' is a subset of foglets and C' is a subset of cloud servers.

Resource Utilization: Resource utilization is defined as

$$Res_{ut} = \frac{Res_{used}}{Res_{available}} \tag{20}$$

where Res_{used} and $Res_{available}$ denote the used and available resources, respectively.

3.3 | Forecasting Future Accident Counts Using Machine Learning

In addition to early response accident detection, the system incorporates a machine learning-based prediction model to forecast future accident counts in specific zones. This section details the prediction algorithm used to anticipate accident occurrences based on historical data and environmental factors.

ALGORITHM 1 | Accident count prediction algorithm.

Require: Historical Accident Data H, Clustered Accident Area Z, Environmental Variables E
Ensure: Predicted Accident Counts P
1: Cluster accident data H into areas Z_i based on geographic proximity.
2: **For each area** Z_i:
3: **for** each time period t **do**
4: Extract historical accident counts and environmental variables for Z_i.
5: Train regression model M using accident counts and variables E.
6: Predict future accident counts $P_i(t+1)$ for the next time period.
7: **end for**
8: **End For return** Predicted accident counts for each zone $P = \{P_1, P_2, \ldots, P_n\}$

3.3.1 | Accident Prediction Algorithm

The following algorithm describes how future accident counts are predicted using regression analysis on historical accident data.

The proposed accident prediction algorithm, shown in Algorithm 1, uses historical accident data and weather features (extracted from an API) to forecast future accident counts. Accident-prone areas are clustered based on geographic proximity, and a regression model is trained for each area. The trained model predicts accident counts for the next period (year), enabling proactive measures to be implemented.

3.4 | Accident Detection and Early Response Algorithm

The accident detection system uses real-time data from sensors and edge devices. When an accident is detected, the system computes the nearest ambulance centers, police stations, and hospitals using a shortest path algorithm. The enhanced algorithm for this process is shown below:

Algorithm 2 outlines how accident detection and notification are managed in a fog-based environment. Sensor data from vehicles or roadside units is continuously evaluated against an accident detection threshold θ. If the sensor data indicates an accident (i.e., the value exceeds θ), the location of the accident is passed to the shortest path algorithm. This algorithm then calculates the nearest ambulance centers, police stations, and hospitals by utilizing a shortest path method (refer Algorithm 3), which leverages the city's weighted road network to determine the quickest possible route for emergency responders. Once the nearest emergency centers are identified, the system sends an automated notification containing essential accident details such as the exact accident location, severity, and time. To ensure prompt and reliable communication with emergency services, the system can employ

ALGORITHM 2 | Accident detection and notification.

Require: Sensor Data S, List of Ambulance Centers A, List of Police Stations P, List of Hospitals H, Accident Detection Threshold θ
Ensure: Notification to the ambulance centers, police stations, and nearest hospital with accident details
1: Initialize accident detection threshold θ.
2: **for** each data point $s_i \in S$ **do**
3: **if** s_i exceeds θ **then**
4: Accident detected at location $loc(s_i)$.
5: Calculate nearest ambulance, police station, and hospital:
6: $h_{\min}^A \leftarrow \texttt{ShortestPath}(loc(s_i), A)$
7: $h_{\min}^P \leftarrow \texttt{ShortestPath}(loc(s_i), P)$
8: $h_{\min}^H \leftarrow \texttt{ShortestPath}(loc(s_i), H)$
9: Send notification to $h_{\min}^A$ with accident details.
10: Send notification to $h_{\min}^P$ with accident details.
11: Send notification to $h_{\min}^H$ with accident details.
12: **end if**
13: **end for**
return Notifications sent to relevant entities.

mobile gateways or SMS gateways, such as Twilio or Nexmo, to send early response alerts directly from the fog nodes to the nearest hospital, ambulance, and police station. This approach minimizes latency, as notifications are sent without needing to route through a central cloud server. In addition, the system can alternatively employ a secure REST API connection to directly communicate with emergency response platforms. This allows for the automated dispatch of messages containing precise accident information to relevant emergency responders, ensuring immediate response initiation.

3.5 | Shortest Path to Nearest Ambulances Centers, Police Stations, and Hospitals Calculation Algorithm

To ensure a quick emergency response, the system calculates the shortest path from the accident site to the nearest hospital using Dijkstra's algorithm. The detailed algorithm is presented below:

Algorithm 3 implements Dijkstra's shortest path algorithm to find the nearest ambulance centers, police stations, and hospitals with the shortest travel time from the accident location. A distance graph G is initialized, where nodes represent geographic points (the accident site, hospitals, ambulance centers, and police stations), and edges represent the distances between them. A priority queue is used to explore the shortest paths iteratively between the accident location and these points. The algorithm efficiently computes the shortest path by continuously updating the distance for each ambulance center, police station, or hospital as new nodes are explored. The nearest ambulance center, police station, and hospital, denoted by $a_{\min}$, $p_{\min}$, and $h_{\min}$, respectively, are then returned.

ALGORITHM 3 | Shortest path calculation algorithm.

Require: Accident Location $loc(s_i)$, List of Hospitals H, List of Ambulance Centers A, List of Police Stations P, Distance Graph G
Ensure: Nearest hospital $h_{\min}$, nearest ambulance center $a_{\min}$, and nearest police station $p_{\min}$

Initialize a priority queue Q.
Set $dist[loc(s_i)] \leftarrow 0$.
Set $dist[h_j] \leftarrow \infty$ for all $h_j \in H$.
Set $dist[a_k] \leftarrow \infty$ for all $a_k \in A$.
Set $dist[p_l] \leftarrow \infty$ for all $p_l \in P$.
Insert $loc(s_i)$ into Q.
while Q is not empty **do**
    Pop the vertex v with the smallest distance from Q.
    for each neighbor u of v **do**
        if $dist[v] + G(v,u) < dist[u]$ **then**
            Update $dist[u] \leftarrow dist[v] + G(v,u)$.
            Insert or update u in Q.
        end if
    end for
end while
Find the nearest hospital $h_{\min} \leftarrow \arg\min_{h_j \in H} dist[h_j]$.
Find the nearest ambulance center $a_{\min} \leftarrow \arg\min_{a_k \in A} dist[a_k]$.
Find the nearest police station $p_{\min} \leftarrow \arg\min_{p_l \in P} dist[p_l]$.**return** $h_{\min}, a_{\min}, p_{\min}$.

Algorithms 2 and 3 work together to ensure a timely response to accidents. When an accident is detected, the system immediately computes the shortest path to the nearest ambulance centers, police stations, and hospitals and sends an automated notification. The architecture supports low-latency processing by leveraging fog computing to handle sensor data at the edge, while the shortest path algorithm ensures that emergency services can respond as quickly as possible.

3.6 | Latency and Energy Consumption

3.6.1 | Latency

The total latency of the system is expressed as:

$$D_{latency} = D_s + D_{ac} + Ds_{tr} + Dac_{tr} + DF_{Fl} + Dproc_{Fl} + DFl_C + D_{prop} \tag{21}$$

3.6.2 | Energy Consumption

The total energy consumption of the system is calculated as:

$$E_{cnsmp} = E_s + E_{ac} + Es_{tr} + Eac_{tr} + EF_{Fl} + E_proc_{Fl} + E_Fl_C + E_{prop} \tag{22}$$

Finally, we formulated an objective function that ties the proposed mathematical models together, ensuring a logical flow from Equations (1–22). This function has formally defined the optimization goals related to energy efficiency, latency reduction, and resource allocation in our multitier fog-cloud framework. Objective Function: We have combined the performance metrics into a single objective function that encapsulates the system's optimization goal.

3.6.3 | The Objective Function Could Be

$$\text{Minimize } Z = \alpha \cdot D_{\text{latency}} + \beta \cdot E_{\text{consumption}} + \gamma \cdot (1 - \text{Res}_{\text{util}}) \tag{23}$$

where: D_{latency} = Total latency; $E_{\text{consumption}}$ = Total energy consumption; Res_{util} = Resource utilization; α, β, γ are Weights indicating the priority of each term.

4 | Performance Evaluation

This section evaluates accident locations to identify high-risk zones by heat map analysis using QGIS. The next step is to examine the performance of the proposed machine learning algorithm for cloud-based decision-making, aimed at forecasting future accident counts. Finally, we evaluate the effectiveness of the proposed system through simulation using iFogSim2 [16] and theoretical analyses conducted using MATLAB.

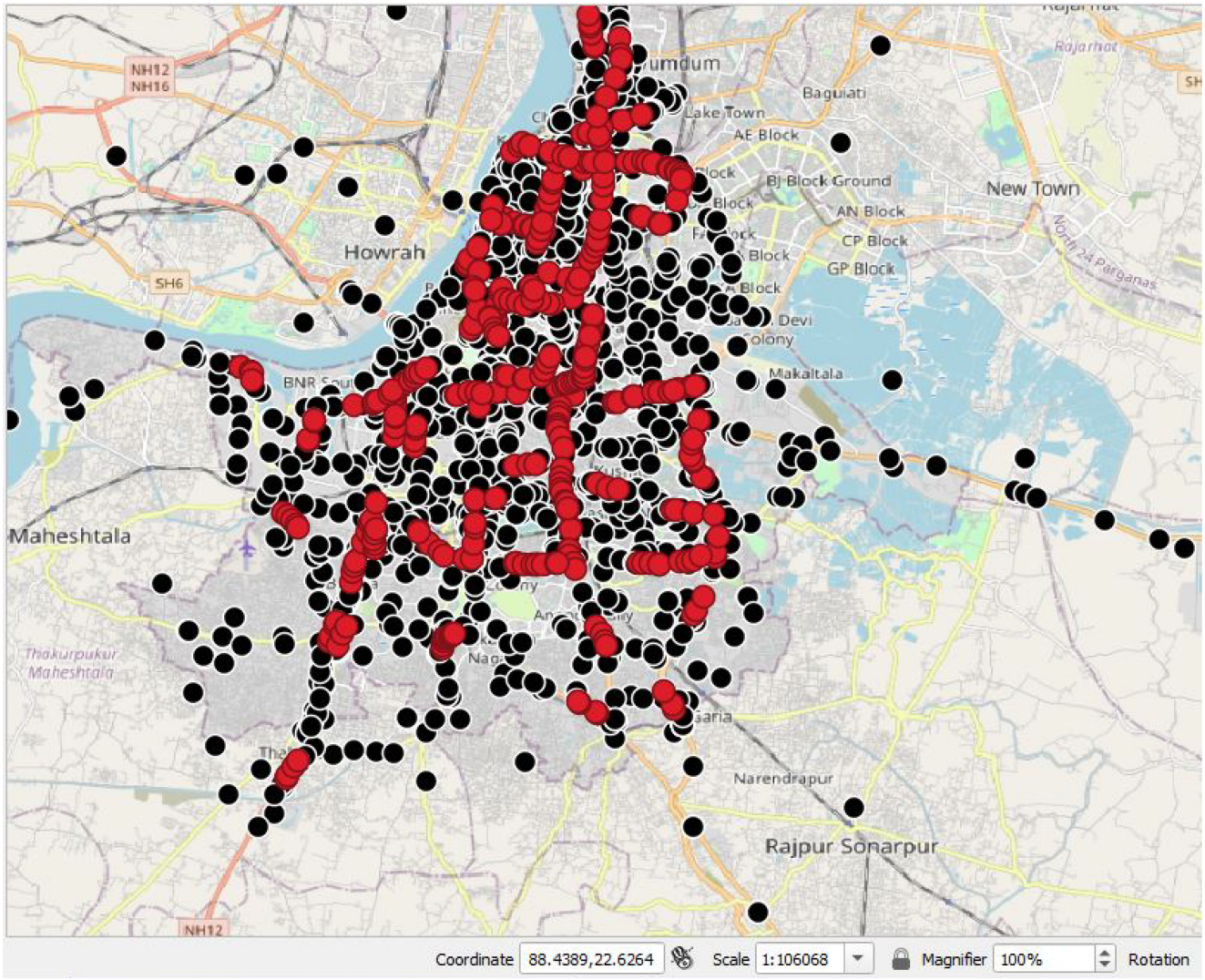

FIGURE 4 | Crash points of study area (Kolkata).

4.1 | Heatmap Analysis Using QGIS

In this analysis, first, we visualized crash points using the open-source software QGIS. A black point indicates the 2017-2020 crash points, and a red point indicates the 2021-2023 crash points (Figure 4). Then, we identified accident-prone zones using combination of historical accident data analysis and spatial hotspot detection techniques in QGIS. Specifically, we employed kernel density estimation (KDE) to visualize high-density accident regions and spatial autocorrelation methods like Moran's I and Getis-Ord Gi* to validate clustering patterns [49]. The heatmap analysis provides insights into the spatial distribution of accident locations, highlighting high-risk areas. Using QGIS, accident locations were visualized based on the data (Figure 5–9). Our analysis indicates that the western section of the study area experiences more accidents than the eastern, northern, and southern sectors. We identified four specific zones with varying accident densities. The highest-risk zone is Golpark–Gariahat on Raja Subhas Chandra Mukherjee Road, near South City Mall. Another high-risk area is around Lady Brabourne College and the intersection of Park Circus, Maa Flyover, Circus Avenue, and Suhrma Avenue. The third zone is Bidhannagar Road, particularly at the Ultodanga crossing. Lastly, the Barabazar MG Road area and Khidirpur Road were identified as additional accident-prone zones. In the identified high-risk areas, we implemented sensors, actuators, and other components to establish a foglet-based framework for early response to accident detection and prevention.

4.2 | Future Accident Count Estimation Using Regression Analysis

Road safety is a primary concern in urban settings. In populated cities like Kolkata, India, preventable fatalities frequently occur, necessitating robust road safety measures. We propose an accident prediction framework that uses multimodal data (historical accident and weather data) collected between 2017–2020 and 2021–2023. After preprocessing the data, roads are segmented into 200-meter lengths, and the number of accidents occurring within each segment is counted. The choice of 200-meter road segments was intentional and driven by the nature of urban road networks. In densely populated urban areas, a 500-meter segment can cover multiple intersections or varying traffic

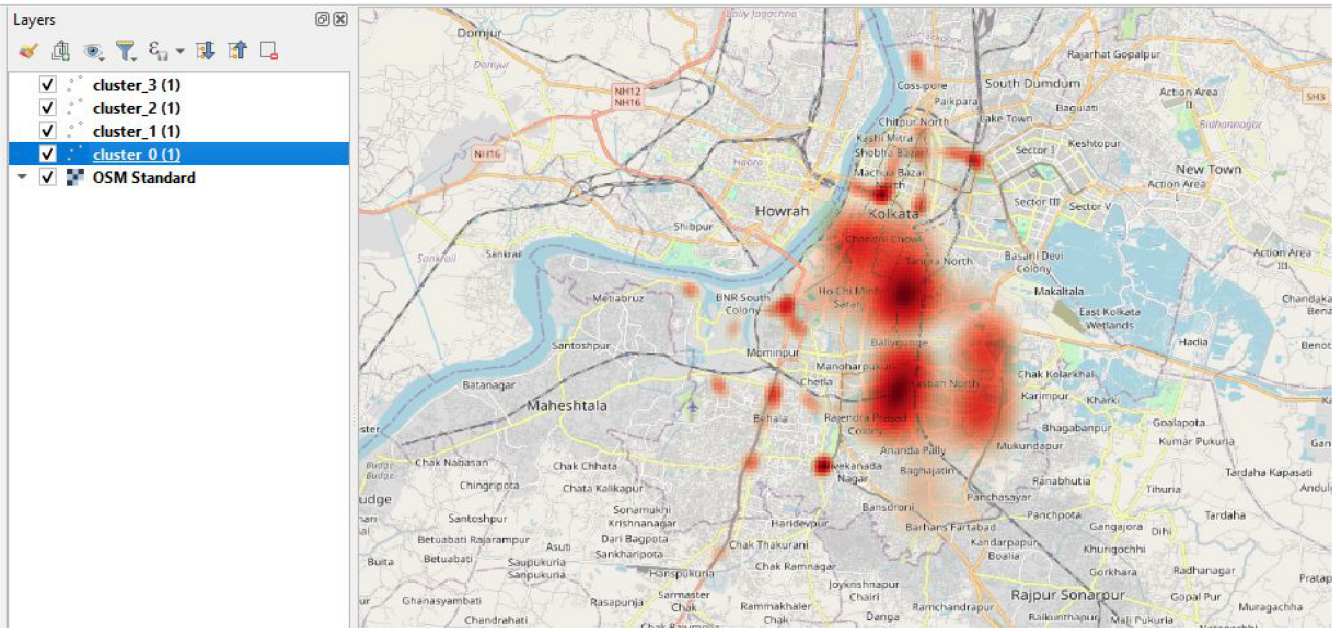

FIGURE 5 | Heatmap of the study area.

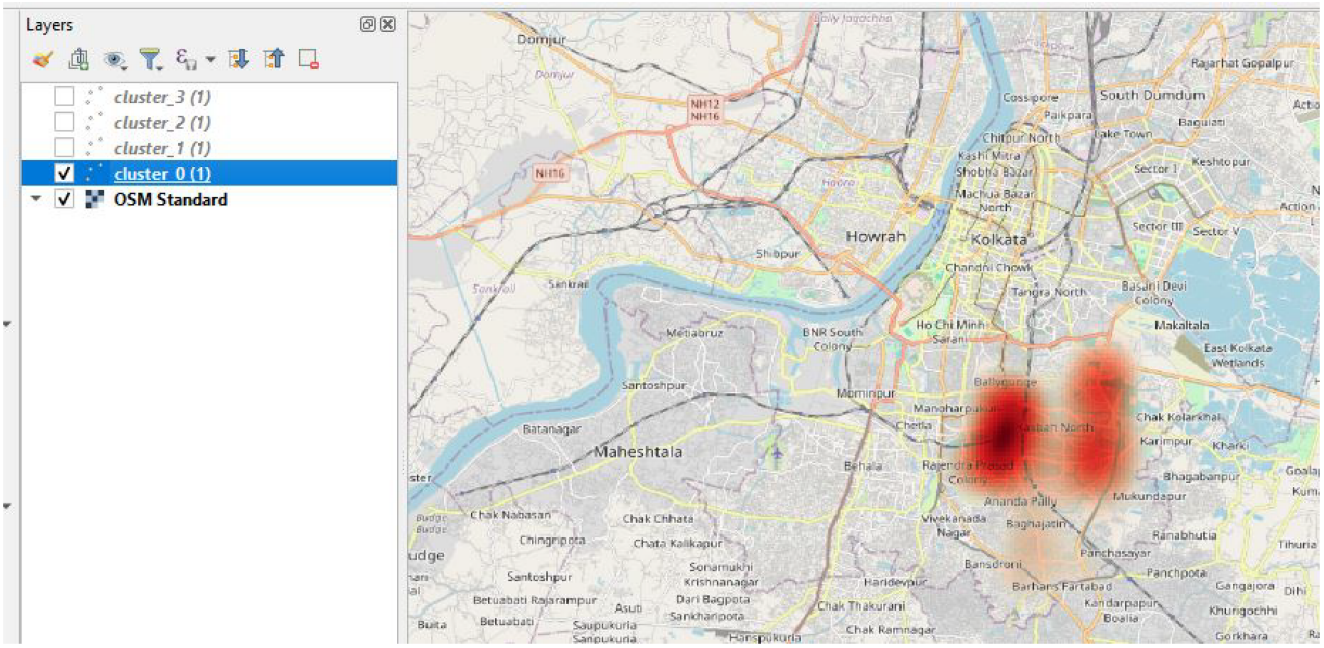

FIGURE 6 | Heatmap zone 1.

conditions, which may dilute the accuracy of accident concentration analysis. According to traffic experts, shorter segments like 200 meters offer a finer spatial resolution, allowing for more precise identification of high-risk zones and better alignment with the dynamic nature of urban traffic patterns. In fact, some urban intersections themselves span close to 500 meters. Therefore, a finer resolution of 200 meters was chosen to capture localized accident patterns more effectively and provide more actionable insights. To identify accident-prone zones, roads were segmented into 200-meter intervals. Each accident point was then mapped to its nearest segment using latitude and longitude coordinates. An iterative segmentation algorithm facilitated this process. Based on the density of accident cluster points in each segment, the data was categorized into four groups: 5–10, 10–20, 20–30, and more than 30 points. Any segment with five or more accident points was classified as an accident-prone zone. We then apply various regression techniques, including Linear Regressor, Lasso Regressor, Ridge Regressor, Decision Tree Regressor, KNN Regression, Random Forest Regressor, Support Vector Regressor (SVR), and XGBoost Regressor. Out of these techniques, XGBoost Regressor showed the highest predictive capabilities with high R2 scores (refer Table 2–4). The aim is to predict future accident counts in any road network segment. For future accident count prediction, a set of features such

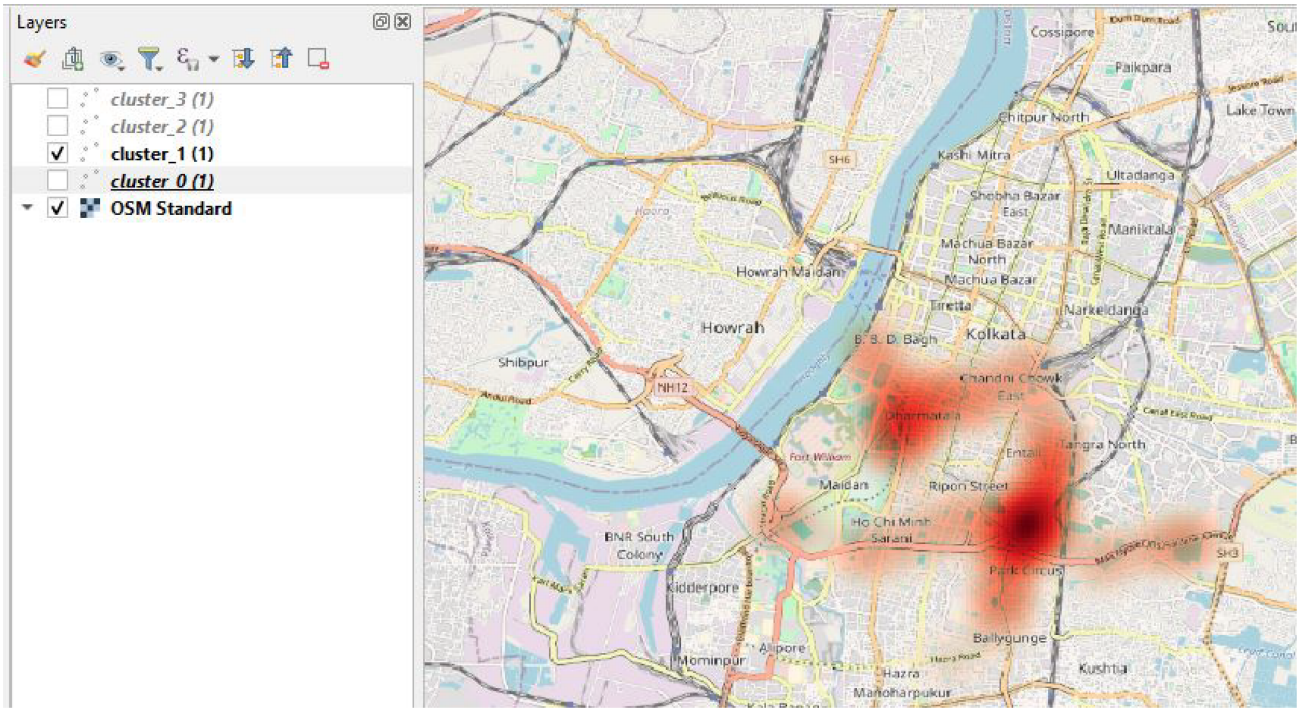

FIGURE 7 | Heatmap zone 2.

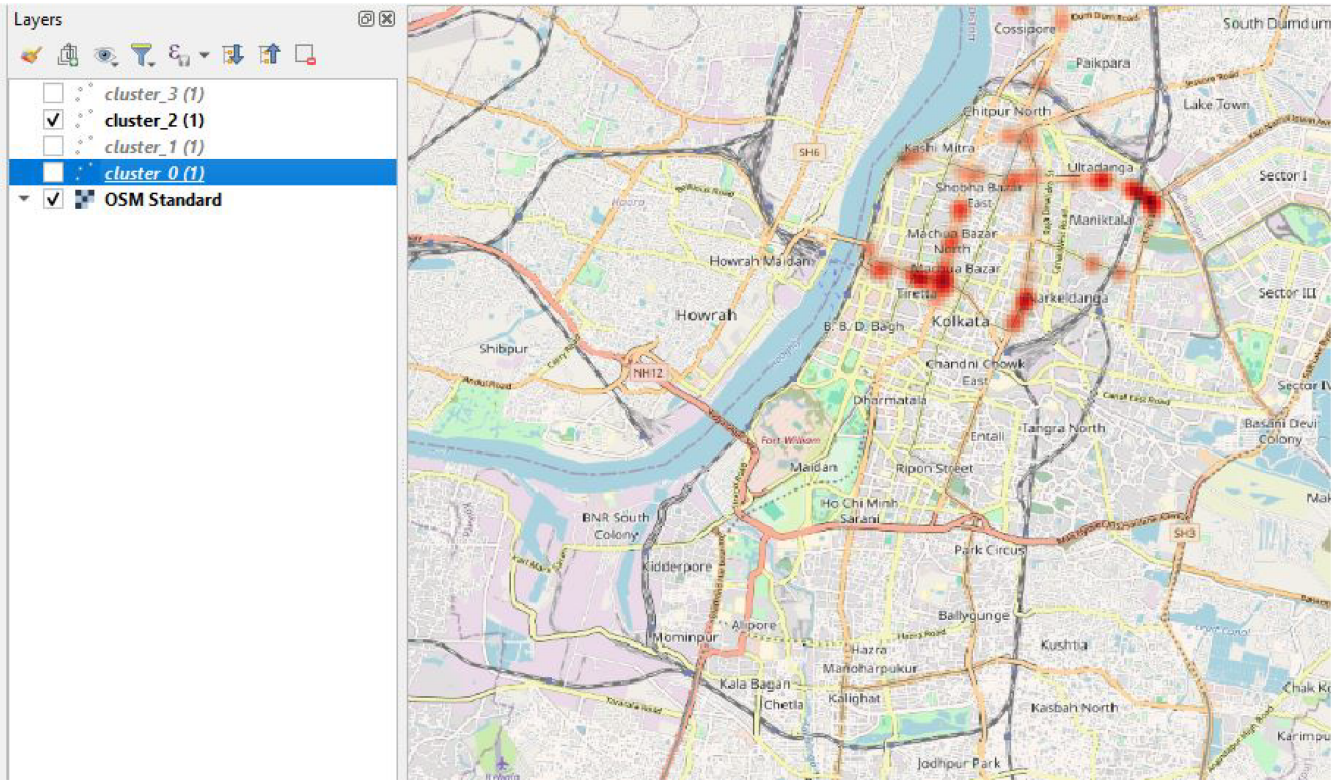

FIGURE 8 | Heatmap zone 3.

as LAT-DEG-MIN, LONG-DEG-MIN, ACC-TIME, MAX-TEMP, MIN-TEMP, APP-TEMP-MAX, APP-TEMP-MIN, PRECIPITATION, and DAYLIGHT-DURATION was used in the regression modeling, with the target variable being the accident count for each location. To evaluate the models' performance, multiple evaluation metrics, including Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R2) scores, were calculated. The dataset was divided into training and testing sets with a 70:30 ratio.

The favorable metric scores of algorithms like XGBoost, Random Forest, and Decision Tree indicate their effectiveness in handling

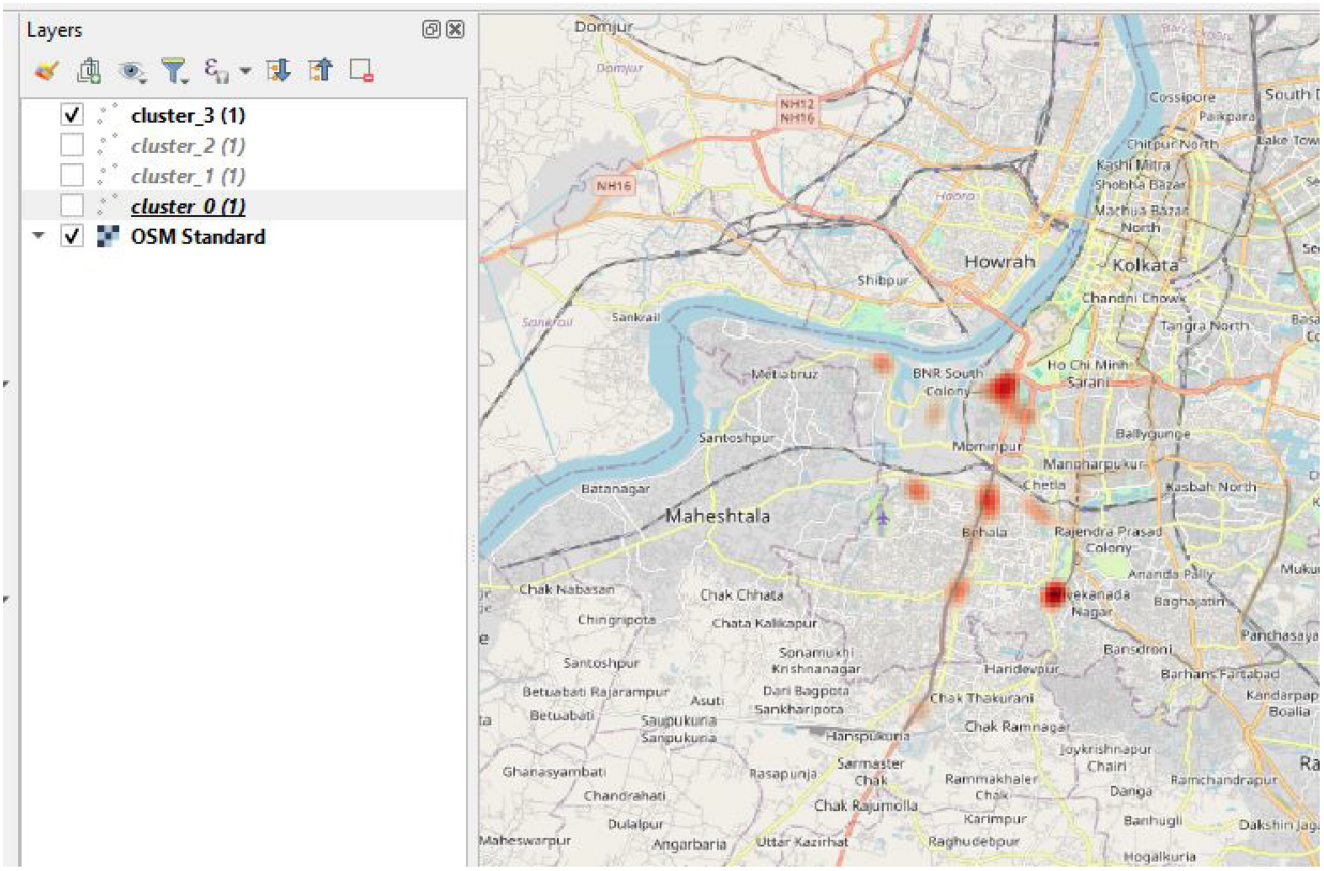

FIGURE 9 | Heatmap zone 4.

the nonlinear and complex relationships between features and target variables. The complexity of the multimodal datasets enhances the models' predictive efficiency. KNN performance declines in high-dimensional data, explaining its suboptimal results. In the case of SVR, the kernel fails to capture the intricate relationships within the multimodal data. Linear, Lasso, and Ridge Regressors produce subpar results due to their limitations in modeling nonlinear separable datasets. Our ML-based accident count prediction method is adaptive, continuously updating its prediction model using real-time accident reports and new traffic data. We have developed an algorithm that not only predicts future accident counts but also identifies the shortest path. The model parameters are periodically fine-tuned with the latest data to ensure high accuracy over time.

A comparison between the predicted results and actual data shows a strong agreement, indicating the model's effectiveness (Figure 10).

4.2.1 | Discussion

MATLAB 2015 was utilized for the theoretical analysis. The data used for the analysis ranged from 100 MB to 1000 MB, encompassing the collection, transmission, and processing of spatiotemporal accident information.

Efficient energy usage is critical in accident detection and prevention systems to ensure continuous and reliable operation. Fog computing, where data processing occurs closer to the source, significantly reduces energy consumption compared to traditional cloud computing methods (Figure 11 and 12). The use of fog agents, when compared to cloud computing, results in substantial energy savings of approximately 16.77% (Figure 13). Implementing fog agents not only improves sustainability but also enhances response times and reduces operational costs, ultimately increasing system efficiency and reliability. This efficiency is particularly crucial for maintaining uninterrupted functionality in remote or resource-constrained environments.

The energy savings achieved through fog computing reach around 3.80%. This reduction ensures longer operational periods, faster data processing, and real-time responsiveness, all of which are vital for maintaining continuous monitoring and quick accident response, especially in environments with limited power resources.

Furthermore, the proposed fog computing architecture achieves a substantial cost saving of approximately 89.66%, highlighting its economic advantages (Figure 14). Fog agents not only reduce delays and improve response times but also minimize operational costs. This cost efficiency is critical for deploying scalable and sustainable accident detection systems, allowing for broader implementation and improved performance across various environments.

Minimizing execution time is essential for the effectiveness of accident detection and prevention systems. The fog computing

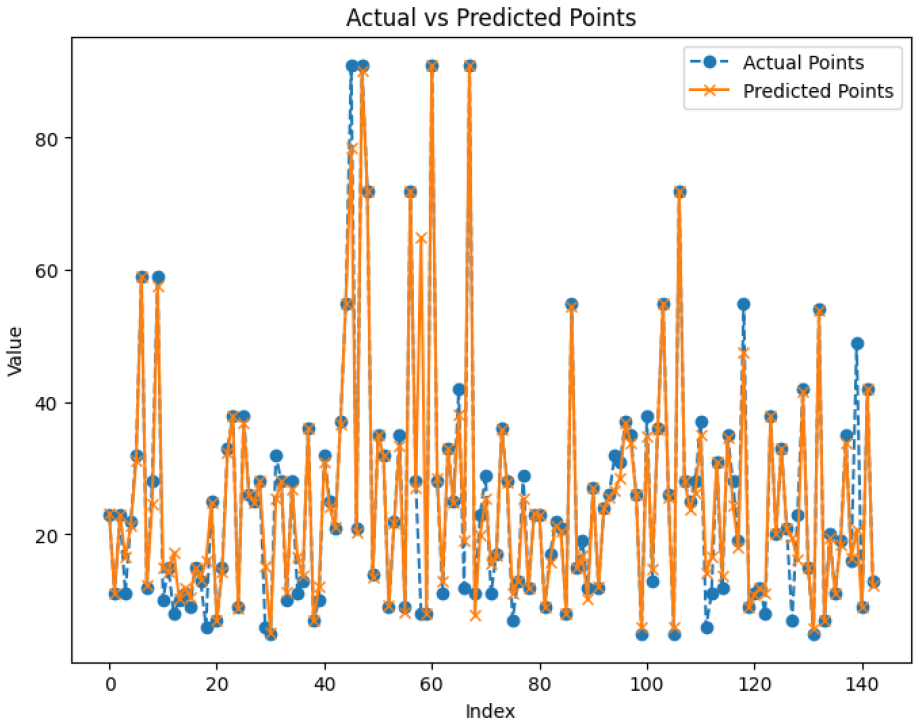

FIGURE 10 | Actual vs predicted accident counts.

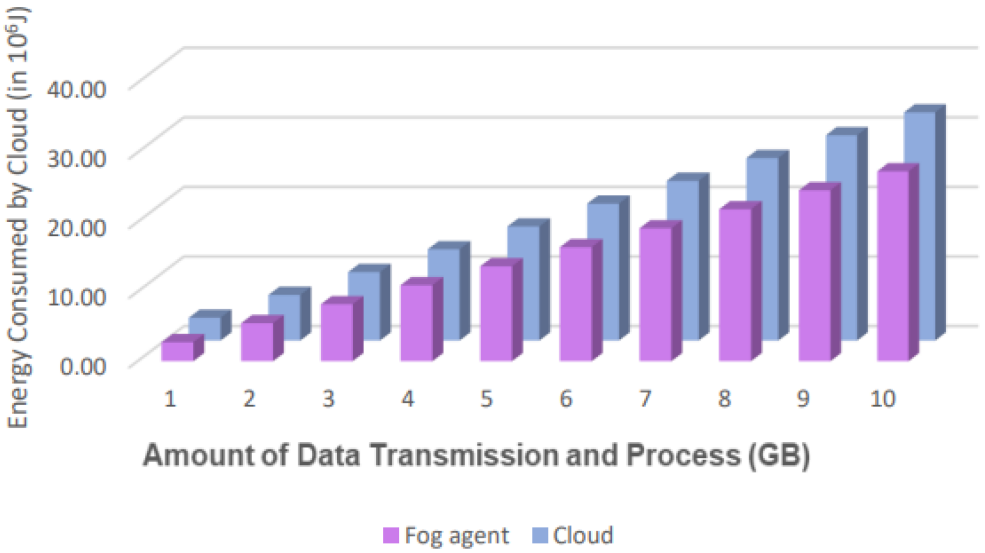

FIGURE 11 | Traditional method using only cloud.

approach results in a reduction of approximately 12.08% in execution time, demonstrating its efficiency (Figure 15). By processing information closer to the source, fog agents facilitate quicker decision-making and faster alerts, which are crucial for preventing accidents and mitigating their impact. This improved response time enhances the overall effectiveness of the system, ensuring timely interventions and increasing safety in different operational environments.

4.3 | Simulation Results

User-defined classes in iFogSim2 were used to represent the proposed system. Initially, we created physical entities and enabled them to interact and communicate with each other. We defined five entities in total: Sensor, Actuator, Fog Agent, fog instance (Foglet), and Cloud. Throughout the experiment, we extensively utilized predefined functions from iFogSim2, such as `createFogDevice()`, `createSensor()`, `createActuator()`, and `addAppEdge()`, to establish the physical topology. To create connections between these entities, user-defined functions like `addArea()` and `addCamera()` were also employed.

`createFogDevices(int userId, String appId)`: This function sets up the physical infrastructure of the fog computing environment, including hierarchical relationships between the cloud, proxy servers, and area routers.

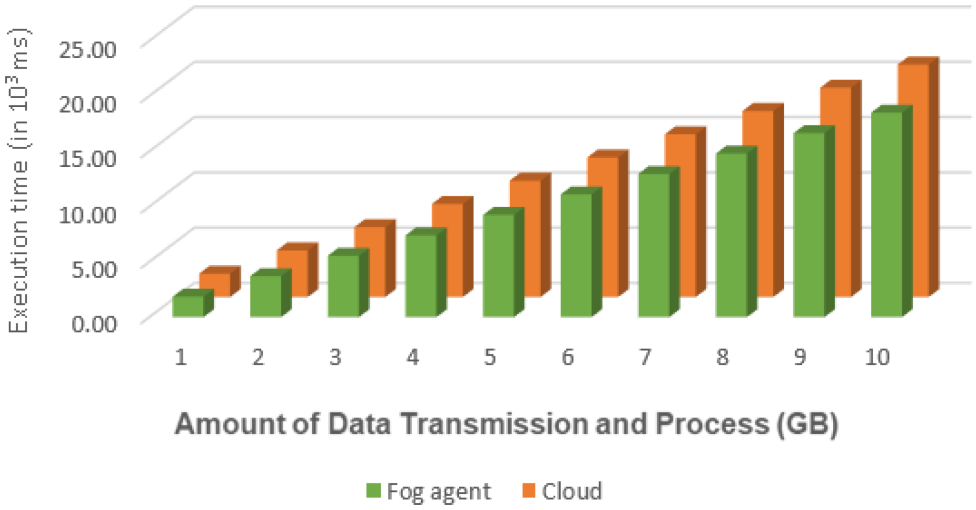

FIGURE 12 | Fog-cloud architecture.

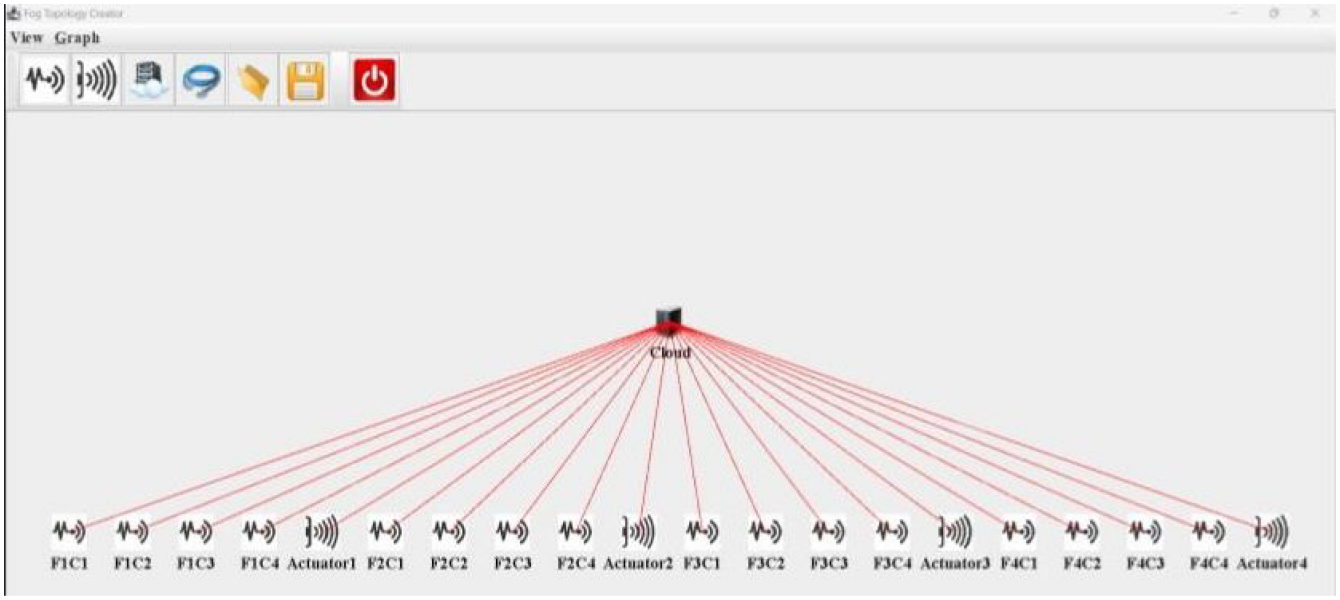

FIGURE 13 | Energy consumption: Fog agent vs cloud.

`addArea(String id, int userId, String appId, int parentId, int proxyIndex, int areaIndex)`: This function creates an area within the physical topology. Each area contains a router connected to multiple smart cameras, allowing for variability in parameters across different areas. It also establishes connections between routers and cameras within an area.

`addCamera(String id, int userId, String appId, int parentId, int cameraIndex, int proxyIndex, int areaIndex)`: This function adds a smart camera to a specific area, setting up the sensor and actuator for that camera. It defines characteristics such as RAM, bandwidth, power consumption, and latency for each smart camera, and creates the necessary sensors and actuators for them.

The modules have specific data dependencies, as illustrated in the workflow diagram of our proposed system. We simulated our system, resulting in the depicted physical topology.

To provide a realistic representation, the experiment measured the end-to-end latency of each device loop. The network latencies between different devices are shown in the Table 6.

4.3.1 | Cloud Layer

The cloud Layer in an accident detection and prevention system serves as the central hub for data aggregation, advanced analytics, and global trend analysis. It collects and stores vast amounts of data from connected fog and edge devices, using this aggregated information to train and refine machine-learning models that detect accident patterns. The cloud leverages complex algorithms and deep learning to analyze historical data, predicting potential accident hotspots/blackspots and times, thereby enabling preemptive measures. Additionally, it distributes updates and policies to lower layers, ensuring that local devices operate with the latest strategies for accident prevention. The cloud also manages resource allocation across the network to maintain timely

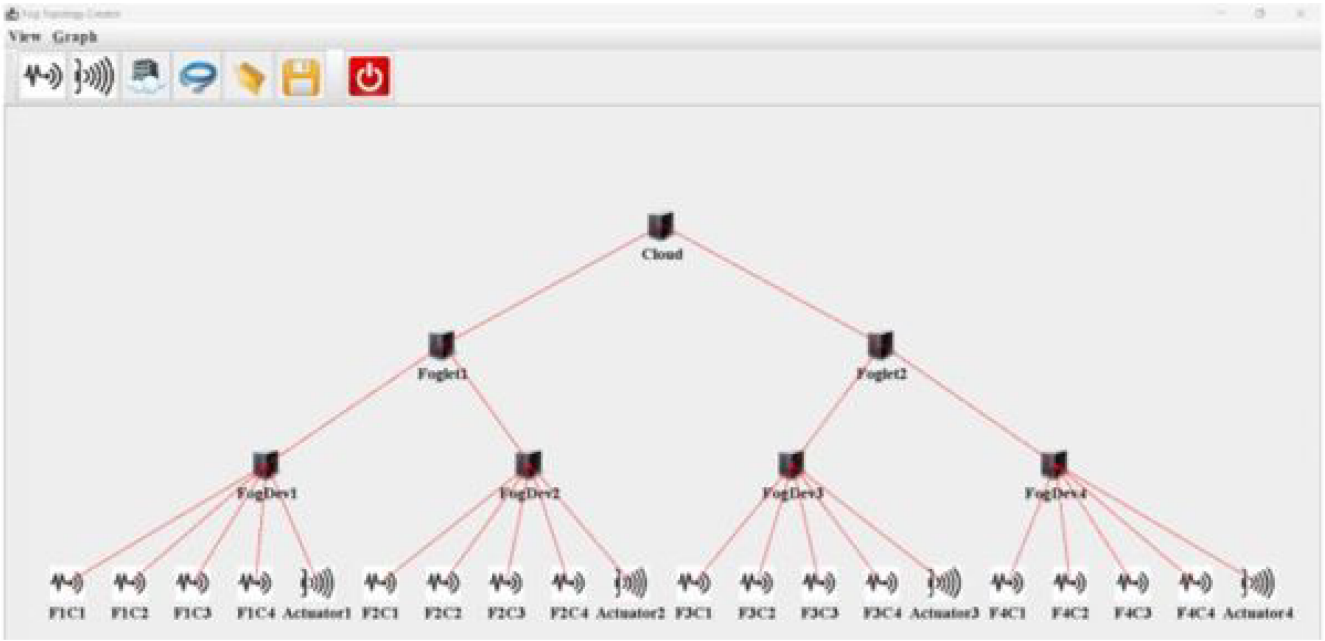

FIGURE 14 | Cost of execution: Fog agent vs cloud.

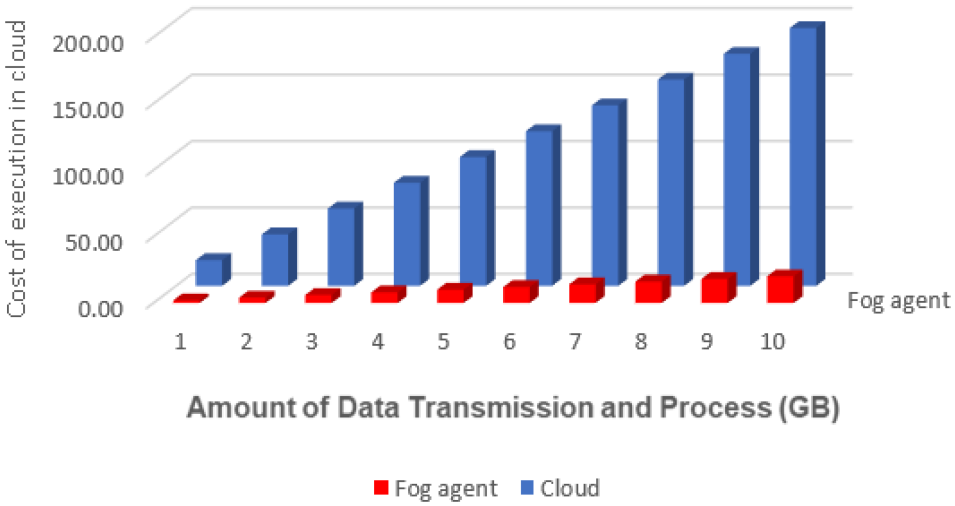

FIGURE 15 | Execution time: Fog agent vs cloud.

responses and prevent system overloads during peak times, making it essential for both comprehensive analysis and centralized management.

4.3.2 | Fog Layer

Due to their localized processing and real-time data analysis, fog nodes are crucial in accident detection and prevention systems. They detect potential accidents through traffic camera feeds and sensor data analysis, send immediate alerts to nearby ambulance centers, police stations, and hospitals, and analyze local data to mitigate risks. This layer is responsible for real-time processing and immediate response actions to enhance the system's efficiency.

4.3.3 | Sensor/Actuator Layer

The Sensor Layer consists of various sensors (such as:F1C1 to F4C4) (refer Table 5) that continuously monitor critical parameters such as vehicle speed, acceleration, distance to obstacles, and environmental conditions with sudden changes or extreme values indicating a potential accident. These sensors collect real-time data and transmit it directly to the fog layer. When an event such as a collision or abrupt stop is detected, actuators take immediate physical actions, such as braking, steering adjustments, or deploying airbags. This layer also provides real-time feedback and assistance to drivers through collision warnings, lane departure alerts, and adaptive cruise control.

In cloud fog systems process on local devices, reducing latency and energy consumption. In a cloud-only system, sensors and actuators are connected to a single server, a hierarchical architecture. However, challenges like network latency and bandwidth limitations may arise.

5 | Conclusions and Future Work

In this paper, we proposed an early response accident detection and prevention system based on a multitier, multizone fog-cloud architecture, designed to enhance efficiency, reliability, and responsiveness in safety-critical applications. By processing accident data closer to the source, our system ensures immediate alerts to the nearest ambulance centers, police stations, and hospitals through mobile gateway or SMS gateway upon accident detection, optimizing emergency response times and coordination. Additionally, our system incorporates a predictive analysis model in the cloud that uses regression analysis on historical accident data to forecast future accident counts, thereby supporting proactive safety measures to help prevent future incidents. Through QGIS-based hotspot analysis, the system identifies high-risk accident zones, enabling targeted resource allocation and safety interventions in the most accident-prone areas. The hierarchical deployment of fog nodes ensures that tasks are efficiently distributed across multiple layers, reducing the computational load on central servers and allowing for more effective resource management. Our results demonstrate significant improvements in energy efficiency (16.77% reduction), execution time (12.08% improvement), and operational costs (89.66% decrease) compared to traditional cloud-only models, making the system both scalable and sustainable. This fog-cloud integrated approach offers a robust and reliable solution for early response accident detection, prevention, and emergency response, especially in densely populated urban environments.

We selected Kolkata as the case study due to its high traffic density, frequent accident occurrences, and diverse urban infrastructure, making it an ideal setting for evaluating the effectiveness of our fog-cloud-based early response framework. Furthermore, we had access to extensive accident datasets from Kolkata's traffic police, enabling us to conduct a robust and data-driven analysis. While this study focuses on Kolkata, our framework is generalizable and can be extended to other urban regions. We plan to extend our implementation to a lab testbed, which will allow us to evaluate the system under controlled yet practical conditions. We also aim to explore possibilities for field deployment (with the help of Kolkata Traffic Police) to further demonstrate the effectiveness and scalability of our approach. Looking forward, future work could focus on optimizing resource allocation through adaptive algorithms that dynamically adjust to traffic patterns and demand, as well as on incorporating machine learning at the fog layer to further enhance the accuracy of accident prediction. Additionally, integrating renewable energy sources for powering fog nodes and implementing advanced data security measures could further increase the system's sustainability and adoption potential. With these advancements, fog computing stands poised to drive innovation in real-time IoT systems across multiple sectors, including healthcare, smart cities, and industrial automation, ultimately improving public safety and operational efficiency.

Author Contributions

Moumita Mishra: conceptualization, methodology, data curation, formal analysis, visualization, validation, writing original draft, writing review and editing. **Soumya Kanti Ghosh:** conceptualization, validation, review, editing and supervision. **Bhargab Maitra:** conceptualization, validation, review and supervision. **Rajkumar Buyya:** conceptualization, validation, writing, review, editing and supervision.

Acknowledgments

The authors are thankful to the Kolkata Traffic Police, West Bengal, India, for sharing the accident database for the present analysis.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The authors have nothing to report.

References

1. M. Guo, B. Janson, and Y. Peng, "A Spatiotemporal Deep Learning Approach for Pedestrian Crash Risk Prediction Based on POI Trip Characteristics and Pedestrian Exposure Intensity," *Accident Analysis & Prevention* 198 (2024): 107493.

2. Z. Zhang, Q. Nie, J. Liu, A. Hainen, N. Islam, and C. Yang, "Machine Learning Based Real-Time Prediction of Freeway Crash Risk Using Crowdsourced Probe Vehicle Data," *Journal of Intelligent Transportation Systems* 28, no. 1 (2024): 84–102.

3. A. Enayet, M. A. Razzaque, M. M. Hassan, A. Alamri, and G. Fortino, "A Mobility-Aware Optimal Resource Allocation Architecture for Big Data Task Execution on Mobile Cloud in Smart Cities," *IEEE Communications Magazine* 56, no. 2 (2018): 110–117.

4. V. Moysiadis, P. Sarigiannidis, and I. Moscholios, "Towards Distributed Data Management in Fog Computing," *Wireless Communications and Mobile Computing* 2018, no. 1 (2018): 7597686.

5. M. Aazam, S. Zeadally, and K. A. Harras, "Fog Computing Architecture, Evaluation, and Future Research Directions," *IEEE Communications Magazine* 56, no. 5 (2018): 46–52.

6. Y. Yang, X. Luo, X. Chu, and M. T. Zhou, *Fog-Enabled Intelligent IoT Systems* (Springer, 2020).

7. H. Wadhwa and R. Aron, "TRAM: Technique for Resource Allocation and Management in Fog Computing Environment," *Journal of Supercomputing* 78, no. 1 (2022): 667–690.

8. B. Jamil, H. Ijaz, M. Shojafar, K. Munir, and R. Buyya, "Resource Allocation and Task Scheduling in Fog Computing and Internet of Everything Environments: A Taxonomy, Review, and Future Directions," *ACM Computing Surveys* 54, no. 11s (2022): 1–38.

9. S. H. H. Madni, M. S. Abd Latiff, Y. Coulibaly, et al., "Resource Scheduling for Infrastructure as a Service (IaaS) in Cloud Computing: Challenges and Opportunities," *Journal of Network and Computer Applications* 68 (2016): 173–200.

10. G. Suciu, V. Suciu, A. Martian, et al., "Big Data, Internet of Things and Cloud Convergence–an Architecture for Secure e-Health Applications," *Journal of Medical Systems* 39 (2015): 141–148.

11. K. A. Khaliq, O. Chughtai, A. Shahwani, A. Qayyum, and J. Pannek, "Road Accidents Detection, Data Collection and Data Analysis Using V2X Communication and Edge/Cloud Computing," *Electronics* 8, no. 8 (2019): 896.

12. M. Mishra, S. K. Roy, A. Mukherjee, D. De, S. K. Ghosh, and R. Buyya, "An Energy-Aware Multi-Sensor Geo-Fog Paradigm for Mission Critical Applications," *Journal of Ambient Intelligence and Humanized Computing* 11 (2020): 3155–3173.

13. A. Hussain, K. Zafar, and A. R. Baig, "Fog-Centric IoT Based Framework for Healthcare Monitoring, Management and Early Warning System," *IEEE Access* 9 (2021): 74168–74179.

14. A. Asghar, A. Abbas, H. A. Khattak, and S. U. Khan, "Fog Based Architecture and Load Balancing Methodology for Health Monitoring Systems," *IEEE Access* 9 (2021): 96189–96200.

15. Q. Abbas and A. Alsheddy, "Driver Fatigue Detection Systems Using Multi-Sensors, Smartphone, and Cloud-Based Computing Platforms: A Comparative Analysis," *Sensors* 21, no. 1 (2020): 56.

16. R. Mahmud, S. Pallewatta, M. Goudarzi, and R. Buyya, "Ifogsim2: An Extended Ifogsim Simulator for Mobility, Clustering, and Microservice Management in Edge and Fog Computing Environments," *Journal of Systems and Software* 190 (2022): 111351.

17. R. Mahmud and R. Buyya, "Modelling and Simulation of Fog and Edge Computing Environments Using iFogSim Toolkit," *Fog and Edge Computing: Principles and Paradigms* (2019): 433–464.

18. H. Trinh, P. Calyam, D. Chemodanov, et al., "Energy-Aware Mobile Edge Computing and Routing for Low-Latency Visual Data Processing," *IEEE Transactions on Multimedia* 20, no. 10 (2018): 2562–2577.

19. A. A. Mutlag, M. K. Abd Ghani, N. Arunkumar, M. A. Mohammed, and O. Mohd, "Enabling Technologies for Fog Computing in Healthcare IoT Systems," *Future Generation Computer Systems* 90 (2019): 62–78.

20. A. Hazra, P. Rana, M. Adhikari, and T. Amgoth, "Fog Computing for Next-Generation Internet of Things: Fundamental, State-Of-The-Art and Research Challenges," *Computer Science Review* 48 (2023): 100549.

21. E. Huaranga-Junco, S. González-Gerpe, M. Castillo-Cara, A. Cimmino, and R. García-Castro, "From Cloud and Fog Computing to Federated-Fog Computing: A Comparative Analysis of Computational Resources in Real-Time IoT Applications Based on Semantic Interoperability," *Future Generation Computer Systems* 159 (2024): 134–150.

22. C. Chakraborty, S. B. Othman, F. A. Almalki, and H. Sakli, "FC-SEEDA: Fog Computing-Based Secure and Energy Efficient Data Aggregation Scheme for Internet of Healthcare Things," *Neural Computing and Applications* 36, no. 1 (2024): 241–257.

23. D. V. Silagyi, II and D. Liu, "Prediction of Severity of Aviation Landing Accidents Using Support Vector Machine Models," *Accident Analysis & Prevention* 187 (2023): 107043.

24. H. U. Atiq, Z. Ahmad, S. K. Uz Zaman, M. A. Khan, A. A. Shaikh, and A. Al-Rasheed, "Reliable Resource Allocation and Management for IoT Transportation Using Fog Computing," *Electronics* 12, no. 6 (2023): 1452.

25. A. Pati, M. Parhi, M. Alnabhan, B. K. Pattanayak, A. K. Habboush, and M. K. Al Nawayseh, "An IoT-Fog-Cloud Integrated Framework for Real-Time Remote Cardiovascular Disease Diagnosis," *Informatics* 10 (2023): 21.

26. H. Wang, V. S. Rajakumar, M. Golec, S. S. Gill, and S. Uhlig, "StockAICloud: AI-Based Sustainable and Scalable Stock Price Prediction Framework Using Serverless Cloud Computing," *Journal of Supercomputing* 81, no. 4 (2025): 527.

27. A. Mukherjee and R. Buyya, "Federated Learning Architectures: A Performance Evaluation With Crop Yield Prediction Application," *Software, Practice & Experience* 55 (2024): 1165–1184.

28. T. Dey, S. Bera, B. Paul, D. De, A. Mukherjee, and R. Buyya, "Fly: Femtolet-Based Edge-Cloud Framework for Crop Yield Prediction Using Bidirectional Long Short-Term Memory," *Software, Practice & Experience* 54, no. 8 (2024): 1361–1377.

29. S. Dhingra, R. B. Madda, R. Patan, P. Jiao, K. Barri, and A. H. Alavi, "Internet of Things-Based Fog and Cloud Computing Technology for Smart Traffic Monitoring," *Internet of Things* 14 (2021): 100175.

30. E. Pourian R, M. Fartash, and J. Akbari Torkestani, "A New Approach to the Resource Allocation Problem in Fog Computing Based on Learning Automata," *Cybernetics and Systems* 55, no. 7 (2024): 1594–1613.

31. M. Abbasi and M. R. Khosravi, "A Robust and Accurate Particle Filter-Based Pupil Detection Method for Big Datasets of Eye Video," *Journal of Grid Computing* 18, no. 2 (2020): 305–325.

32. E. Alemneh, S. M. Senouci, and P. Brunet, "PV-Alert: A Fog-Based Architecture for Safeguarding Vulnerable Road Users," in *2017 Global Information Infrastructure and Networking Symposium (GIIS)* (IEEE, 2017), 9–15.

33. S. Benzerogue, S. Abdelatif, S. Merniz, S. Harous, and L. Khamer, "Multi-Path Transmission Protocol for Video Streaming Over Vehicular Fog Computing Environments," *IEEE Access* 12 (2024): 87199–87216.

34. M. Mishra, S. Ghosh, B. Maitra, and S. K. Ghosh, "Exploring Spatio-Temporal Multimodal Data for Accident Count Prediction: A Case Study of Kolkata," in *2024 IEEE 27th International Conference on Intelligent Transportation Systems (ITSC)* (IEEE, 2024), 1476–1481.

35. M. Mishra, K. Banerjee, S. K. Ghosh, and B. Maitra, "Unveiling Hidden Patterns Using Association Rule Mining and Machine Learning for Intelligent Transportation Systems," in *2024 IEEE 21st India Council International Conference (INDICON)* (IEEE, 2024), 1–5.

36. S. Manimurugan, "IoT-Fog-Cloud Model for Anomaly Detection Using Improved Naïve Bayes and Principal Component Analysis," *Journal of Ambient Intelligence and Humanized Computing* 15 (2021): 1–10.

37. M. Songhorabadi, M. Rahimi, A. MoghadamFarid, and M. H. Kashani, "Fog Computing Approaches in IoT-Enabled Smart Cities," *Journal of Network and Computer Applications* 211 (2023): 103557.

38. N. Venkataraman, "Proactive Fault Prediction of Fog Devices Using LSTM-CRP Conceptual Framework for IoT Applications," *Sensors (Basel)* 23, no. 6 (2023): 2913.

39. D. Lin, M. Yan, L. Kong, R. Quan, and Y. L. Guan, "A Framework of Real-Time Intelligent Transportation System Based on Hybrid Fog-Cloud Computing," *IEEE Communications Magazine* 62, no. 1 (2023): 126–132.

40. M. Asadi, M. Fathy, H. Mahini, and A. M. Rahmani, "An Evolutionary Game Approach to Safety-Aware Speed Recommendation in Fog/Cloud-Based Intelligent Transportation Systems," *IEEE Transactions on Intelligent Transportation Systems* 23, no. 7 (2021): 7431–7440.

41. M. L. M. Peixoto, A. H. Maia, E. Mota, et al., "A Traffic Data Clustering Framework Based on Fog Computing for VANETs," *Vehicular Communications* 31 (2021): 100370.

42. S. Sahil, S. K. Sood, and V. Chang, "Fog-Cloud-IoT Centric Collaborative Framework for Machine Learning-Based Situation-Aware Traffic Management in Urban Spaces," *Computing* 106, no. 4 (2024): 1193–1225.

43. S. Shin, J. Kim, and C. Moon, "Road Dynamic Object Mapping System Based on Edge-Fog-Cloud Computing," *Electronics* 10, no. 22 (2021): 2825.

44. K. Saini, S. Kalra, and S. K. Sood, "Fog-Inspired Framework for Emergency Rescue Operations in Post-Disaster Scenario," *Journal of Supercomputing* 79, no. 18 (2023): 21057–21088.

45. B. K. Dar, M. A. Shah, S. U. Islam, C. Maple, S. Mussadiq, and S. Khan, "Delay-Aware Accident Detection and Response System Using Fog Computing," *IEEE Access* 7 (2019): 70975–70985.

46. Q. V. Khanh, N. V. Hoai, A. D. Van, and Q. N. Minh, "An Integrating Computing Framework Based on Edge-Fog-Cloud for Internet of Healthcare Things Applications," *Internet of Things* 23 (2023): 100907.

47. A. Finogeev, Đ. Finogeev, L. Fionova, A. Lyapin, and K. A. Lychagin, "Intelligent Monitoring System for Smart Road Environment," *Journal of Industrial Information Integration* 15 (2019): 15–20.

48. A. Aljumah, A. Kaur, M. Bhatia, and A. T. Ahamed, "Internet of Things-Fog Computing-Based Framework for Smart Disaster Management," *Transactions on Emerging Telecommunications Technologies* 32, no. 8 (2021): e4078.

49. L. Pregi and L. Novotnỳ, "Spatial Autocorrelation Methods in Identifying Migration Patterns: Case Study of Slovakia," *Applied Spatial Analysis and Policy* 18, no. 1 (2025): 1–21.