iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

EDGE-BASED INTELLIGENT SOFTWARE ARCHITECTURE FOR FAULT DETECTION IN IOT NETWORKS USING MATHEMATICAL MODELS AND MACHINE LEARNING

Olamilekan Samuel Adeniji
ORCID: 0009-0004-3913-4345
https://orcid.org/0009-0004-3913-4345
olamilec@gmail.com
Graduate Student, Department of Computer Science, Crown Polytechnic, Ado-Ekiti
Ekiti State, Nigeria

Ifeanyichukwu Junior Enumah
ifeanyichukwujnr@gmail.com
Graduate Student, Department of Science Education, Delta State University, Abraka
Delta State, Nigeria

Chukwuemeka Dominic Edeh
edehchukwuemekadominic@gmail.com
Graduate Student, Department of Mechatronics Engineering, Federal University Otuoke
Bayelsa State, Nigeria

Mairama Gadzama
mairamagadzama989@gmail.com
Graduate Student, Department of Computer Science, University of Abuja
Federal Capital Territory (FCT), Abuja, Nigeria

ABSTRACT
The rapid growth of Internet of Things (IoT) systems has increased the need for reliable and timely fault detection to ensure system performance, safety, and availability. Many existing fault detection approaches rely on cloud-centric processing, which can introduce latency, increase bandwidth consumption, and depend on continuous network connectivity, limiting their effectiveness in time-sensitive IoT applications. This study proposes an edge-based intelligent software architecture that integrates mathematical modeling with machine-learning techniques to enable accurate, interpretable, and low-latency fault detection closer to data sources. The proposed approach combines residual-based analytical models with data-driven anomaly detection and distributes fault detection tasks between edge and cloud layers to balance responsiveness and computational efficiency. The framework is evaluated through simulation under multiple fault scenarios and varying network conditions. The results show that the proposed hybrid approach improves fault detection effectiveness, reduces detection latency, and lowers communication overhead compared with cloud-only and single-method solutions, while maintaining stable performance as the number of connected IoT devices increases. Overall, the findings indicate that integrating mathematical models and machine learning within an edge-based framework provides a practical and scalable solution for reliable fault detection in modern IoT environments.

Keywords:
IoT fault detection, edge computing, edge AI, machine learning, mathematical modeling, anomaly detection, software architecture

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

1. INTRODUCTION

The Internet of Things (IoT) has become a cornerstone of modern cyber-physical systems, enabling large-scale integration of sensors, actuators, and embedded devices that continuously interact with physical environments. This paradigm has facilitated significant advances in industrial automation, smart cities, healthcare monitoring, energy systems, and intelligent transportation (Sodiya et al., 2024; Li et al., 2024). However, as IoT deployments grow in scale and heterogeneity, ensuring system reliability and operational integrity has become a major challenge, particularly with respect to timely and accurate fault detection (Kauffman et al., 2020; Altarrazi et al., 2023).

IoT faults may arise from sensor degradation, communication failures, hardware malfunctions, software errors, or adverse environmental conditions, leading to inaccurate data, degraded performance, and potential safety risks (Wang, Babulak and Tang, 2021; Santo et al., 2023). In industrial IoT environments, even minor anomalies such as sensor drift or intermittent communication losses can disrupt production processes and cause significant economic losses; undetected faults may compromise safety-critical decisions in healthcare and smart infrastructure applications (Santo et al., 2023; Ortiz-Garcés, Villegas-Ch and Luján-Mora, 2025). Consequently, early, automated, and reliable fault detection is essential for maintaining dependable IoT operation.

Traditional fault detection in IoT systems has largely relied on cloud-centric architectures, in which sensor data are continuously transmitted to centralized cloud servers for analysis. Although cloud platforms provide a high computational capacity and scalability, they introduce several limitations for real-time fault detection, including increased latency, excessive bandwidth consumption, privacy concerns, and single points of failure (Zhou, 2020; Ortiz-Garcés, Villegas-Ch and Luján-Mora, 2025). These drawbacks are particularly problematic for latency-sensitive and mission-critical applications such as industrial control systems and intelligent transportation, where delayed fault detection can lead to system instability or hazardous outcomes (Ouedraogo, 2021; Hong et al., 2024).

Deep learning models, such as recurrent neural networks and auto encoders, have demonstrated strong performance in time-series anomaly detection tasks (Savić et al., 2021; Seba, Gemeda and Ramulu, 2024). However, purely data-driven approaches often suffer from limited interpretability, high data requirements, and reduced robustness when encountering previously unseen fault conditions, which limits their reliability in safety-critical environments (Mansouri and Vadera, 2023; Wadinger and Kvasnica, 2023). Mathematical modeling offers a complementary foundation by providing interpretable, knowledge-driven representations of system behavior and residual-based diagnostics with minimal training data (Steenwinckel et al., 2020; Tordeux et al., 2024). These limitations motivate the integration of mathematical models with machine learning within a hybrid fault-detection framework (Atoui and Cohen, 2021; Wu, Sicard and Gadsden, 2024).

To address these challenges, this study proposed an edge-based intelligent software architecture that integrates mathematical models with machine learning to enable robust, low-latency, and interpretable fault detection in IoT networks. The proposed architecture leverages edge computing for real-time inference and decision-making while utilizing cloud resources for global analytics and model training. By combining analytical rigor with adaptive learning, the proposed hybrid approach aims to improve the fault detection accuracy, reduce false alarms, enhance interpretability, and support scalable deployment across heterogeneous IoT environments.

2. RELATED WORK AND BACKGROUND

This section reviews existing research related to fault detection in IoT networks, edge and fog computing paradigms, machine learning-based anomaly detection, and mathematical modeling approaches. This review highlights the current advances in identifying the limitations of the proposed hybrid edge-based architecture.

2.1 Fault Detection in Iot Networks

Early approaches predominantly relied on rule- and threshold-based mechanisms, where faults were identified when sensor readings exceeded predefined limits. Although computationally efficient, these methods are highly sensitive to noise and environmental variability, leading to frequent false alarms and limited adaptability to dynamic operating conditions (Rhachi et al., 2024).

Data-driven approaches based on machine learning have gained prominence because of the availability of large volumes of sensor data. Supervised learning techniques, such as Support Vector Machines, Decision Trees, and Random Forests, have been applied to classify fault conditions when labeled datasets are available (Rafique et al., 2024). However, labeled fault data are often scarce in real-world IoT deployments, because faults are rare events

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

and are costly to reproduce intentionally (Jung, 2020). To address this limitation, unsupervised and semi-supervised methods, including clustering, Isolation Forests, One-Class SVMs, and autoencoders, have been widely adopted for anomaly detection by learning the patterns of normal operation and flagging deviations (Rhachi et al., 2024).

Deep learning models, particularly Recurrent Neural Networks, Long Short-Term Memory networks, and autoencoders, have demonstrated superior performance for time-series fault detection in IoT systems because of their ability to capture temporal dependencies and nonlinear relationships (Savić et al., 2021; Seba, Gemeda and Ramulu, 2024). Despite their effectiveness, these models often operate as black boxes, offering limited interpretability and making it difficult to justify fault decisions in critical applications (Mansouri and Vadera, 2023). In addition, their performance may degrade when encountering unseen fault types or nonstationary data distributions (Wadinger and Kvasnica, 2023).

2.2 Edge and Fog Computing Paradigms

The limitations of cloud-centric IoT architectures have driven significant interest. Edge computing shifts computation and analytics closer to IoT devices, reducing latency, bandwidth usage, and dependence on a centralized infrastructure (Chen et al., 2020; Zhou et al., 2024). This paradigm is particularly advantageous for fault detection, where rapid response times are essential to prevent cascading failures or unsafe system states (Ortiz-Garcés, Villegas-Ch and Luján-Mora, 2025). These architectural advantages have motivated increasing adoption of edge-based intelligence for time-sensitive IoT applications (Santo et al., 2023; Dong et al., 2024).

Edge-based processing also enhances privacy by minimizing the transmission of sensitive raw data to remote servers (Zhou et al., 2024).

2.3 Machine Learning–Based Anomaly Detection at the Edge

Recent research has increasingly focused on integrating machine learning-based anomaly detection directly at the edge to support real-time fault diagnosis. Edge AI frameworks commonly deploy pre-trained models at edge nodes and perform inference locally while delegating training and optimization tasks to the cloud (Ortiz-Garcés, Villegas-Ch and Luján-Mora, 2025; Reis and Serôdio, 2025).

Recurrent neural networks and autoencoders are frequently adopted for edge-based fault detection because of their effectiveness in handling streaming time-series data (Savić et al., 2021). However, deploying deep learning models at the edge remains challenging owing to the limited hardware resources. Techniques such as model compression, pruning, quantization, and knowledge distillation are commonly used to address these constraints (Gómez et al., 2024).

2.4 Mathematical Modeling Approaches for Fault Detection

Mathematical and model-based fault-detection techniques provide an alternative and complementary approach to data-driven methods. These techniques rely on analytical models derived from physical laws, system dynamics, or statistical relationships to characterize normal system behavior (Tordeux et al., 2024). Residual-based methods, in which deviations between measured and predicted values indicate faults, are widely used in control systems and industrial diagnostics (Jung, 2020).

Statistical models, state-space representations, Kalman filters, and fault tree analysis have been applied to IoT and cyber-physical systems to enable interpretable and explainable fault diagnoses (Steenwinckel et al., 2020; Chinnaiyan, Mary and Mahdal, 2023). These approaches are particularly valuable when fault data are scarce, because they require minimal training data and provide clear causal reasoning. However, developing accurate mathematical models for complex, nonlinear, and evolving IoT systems is often challenging, and may not scale well across heterogeneous deployments.

2.5 Research Gaps and Motivation

The literature reveals several unresolved challenges in the IoT fault detection. First, many edge-based solutions rely primarily on data-driven machine learning models, which, while effective, often lack interpretability and robustness to novel fault conditions. Second, purely model-based approaches, although interpretable, struggle to adapt to the dynamic and complex IoT environments. Third, existing hybrid approaches frequently lack a unified software architecture optimized for resource-constrained edge deployment and scalable operations across heterogeneous IoT networks (Altarrazi et al., 2023).

These gaps motivate the need for an integrated edge-based fault-detection architecture that combines the interpretability and analytical rigor of mathematical models with the adaptability and pattern-recognition capabilities of machine learning. By embedding such a hybrid approach within an edge–cloud collaborative framework, it is

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

possible to achieve low-latency, scalable, and trustworthy fault detection that is suitable for modern IoT applications. Comprehensive surveys further highlight the growing role of intelligent IoT systems and the importance of secure, low-latency, and scalable fault detection solutions (Aouedi et al., 2024).

3. METHODS: EDGE-BASED ARCHITECTURE AND FAULT DETECTION FRAMEWORK

This section presents the proposed edge-based intelligent software architecture for fault detection in an IoT network. The architecture is designed to support real-time fault diagnosis, efficient resource utilization, and scalable deployment by integrating mathematical models and machine learning within an edge–cloud collaborative framework.

3.1 Overall System Architecture

The proposed architecture follows a hierarchical three-layer model consisting of an **IoT Device Layer**, the **Edge Layer**, and the **Cloud Layer**. This layered design ensures that computational tasks are allocated to the most appropriate layer based on latency requirements, resource availability, and data sensitivity. The overall structure and interactions among IoT devices, edge nodes, and cloud services are illustrated in **Figure 1**.

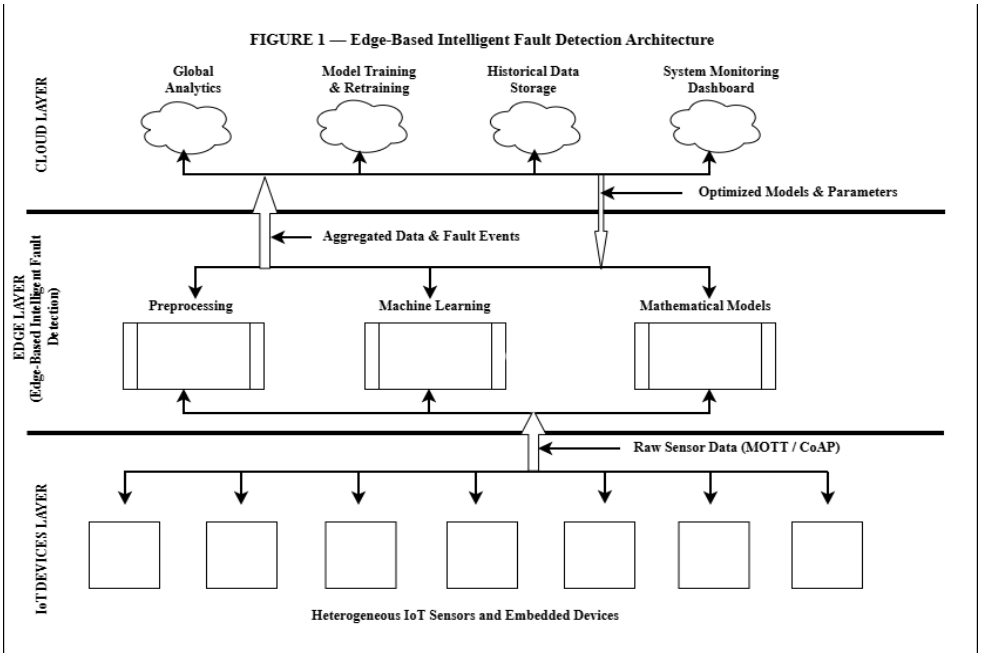

Figure 1. Edge-based intelligent fault detection architecture illustrating hierarchical collaboration between heterogeneous IoT devices, edge nodes performing preprocessing, mathematical modeling, and machine learning–based fault detection, and cloud resources for global analytics, model training, and system monitoring.

3.1.1 Iot Device Layer

The IoT Device Layer comprises heterogeneous sensors, actuators, and embedded devices that are responsible for monitoring physical processes and generating raw data streams. These devices include temperature sensors, pressure sensors, vibration sensors, current meters, and network monitoring components.

Each data sample was timestamped at the source to preserve the temporal consistency, which is critical for time-series fault detection. Communication protocols such as MQTT, CoAP, and lightweight HTTP are used to transmit data securely to the Edge Layer.

3.1.2 Edge Layer

The Edge Layer constitutes the core of the proposed architecture and is responsible for real-time fault detection and decision-making. Edge nodes are typically implemented as gateways, industrial controllers, or microservers located close to the IoT devices.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

Key responsibilities of the Edge Layer include:
Data ingestion and buffering: Receive data streams from multiple IoT devices and temporarily store recent observations.
Data preprocessing: Cleaning, normalizing, and extracting features from raw sensor data.
Local fault detection: Executing mathematical models and machine learning inference to identify anomalies and faults in real-time.
Decision logic and actuation: Triggering immediate corrective actions or alerts based on detected faults.
Edge–cloud communication: Transmitting aggregated data, fault events, and model feedback to the cloud.
By performing local fault detection, the Edge Layer significantly reduces the detection latency and minimizes the dependence on continuous cloud connectivity, which is essential for mission-critical IoT applications (Ortiz-Garcés, Villegas-Ch and Luján-Mora, 2025).

3.1.3 Cloud Layer

The Cloud Layer provides centralized computational resources for global system management, long-term data storage, and advanced analytics. Unlike the Edge Layer, which focuses on real-time operations, the Cloud Layer handles computationally intensive tasks that are not time-critical.
Primary functions of the Cloud Layer include:
Historical data aggregation and storage for long-term analysis and auditing.
Training and retraining of machine learning models using aggregated data from multiple edge nodes.
Model optimization and distribution, including compression and quantization for edge deployment.
Global fault analysis, enabling the identification of system-wide patterns and correlated failures.
System monitoring and management dashboards for operators and administrators.

3.2 Edge–Cloud Interaction Model

The interaction between the Edge and Cloud Layers was designed to balance the local autonomy with global intelligence. This interaction encompasses data exchange, model lifecycle management, and control signaling.

3.2.1 Data Flow Management

Raw sensor data are transmitted from the IoT devices to the Edge Layer, where preprocessing and fault detection are performed. Only essential information, such as aggregated statistics, detected fault events, or sampled data for retraining, is forwarded to the Cloud Layer. This selective data transmission significantly reduces the network bandwidth usage and enhances data privacy (Zhou et al., 2024). The end-to-end data flow and machine learning model lifecycle across the edge and cloud layers are shown in **Figure 2**.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

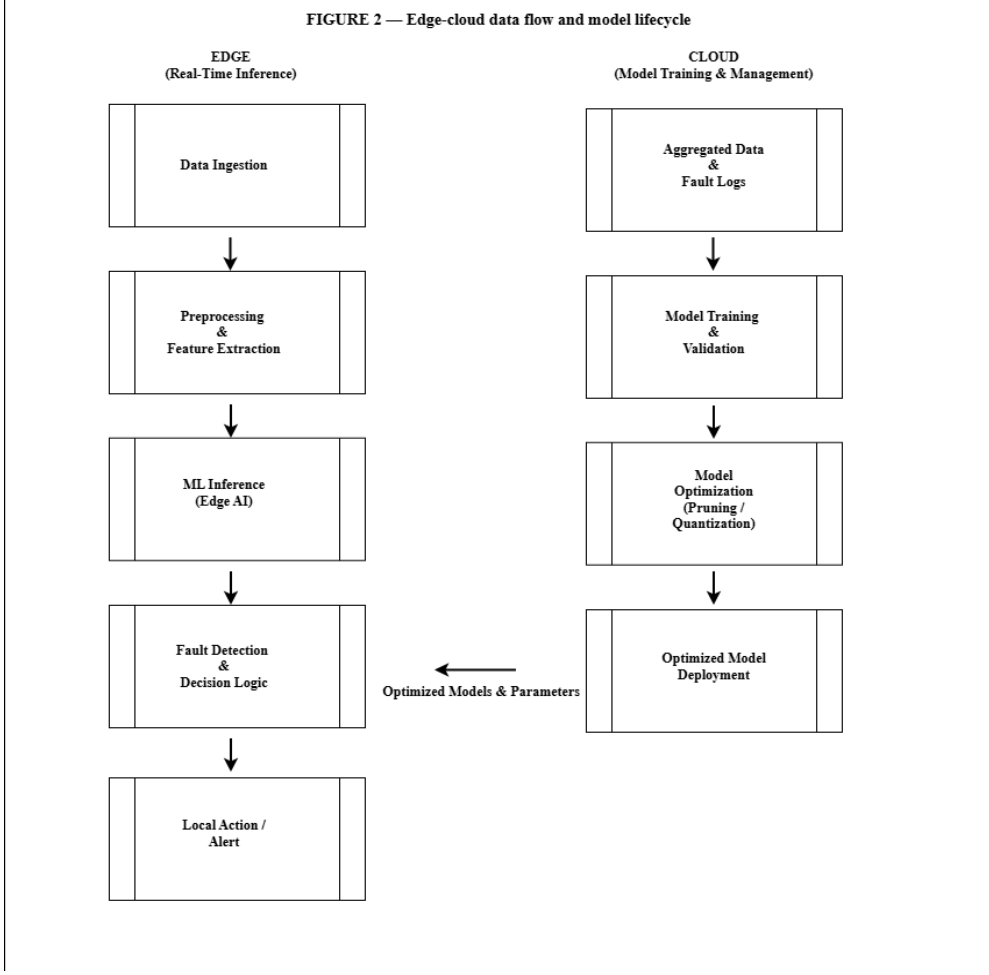

Figure 2. Edge–cloud data flow and machine learning model lifecycle illustrating real-time data processing and fault detection at the edge, selective transmission of aggregated data and fault logs to the cloud, and continuous model training, optimization, and redeployment through optimized models and parameters.

3.2.2 Model Lifecycle Management

Machine-learning models were trained and validated in the Cloud Layer using diverse datasets collected from the IoT network. Once trained, the models were optimized for edge deployment using techniques such as pruning, quantization, and knowledge distillation. The optimized models were then securely deployed to the edge nodes for real-time inference.

Edge nodes continuously monitor the model performance. When performance degradation or concept drift is detected, feedback and selected data samples are sent to the cloud to support the retraining and model updates.

3.2.3 Control and Coordination

While the Edge Layer handles immediate fault responses, the Cloud Layer retains authority over global configuration, system updates, and coordinated actions across multiple edge nodes. This hybrid control mechanism enables fast local reactions while preserving centralized oversight and strategic management.

3.3 Data Acquisition and Preprocessing

Effective fault detection depends on the quality of input data. Data acquisition and preprocessing are primarily executed at the Edge Layer to reduce the noise and computational overhead downstream.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

Data preprocessing at the edge includes noise filtering, missing-value handling, normalization, feature extraction, and sliding-window formation for time-series analysis, optimized for low-latency execution.

3.4 Hybrid Fault Detection Methodology

The proposed fault detection methodology integrates **mathematical models** and **machine learning models** to exploit their complementary strengths. The integration of the mathematical models and machine-learning techniques within the proposed fault-detection pipeline is illustrated in **Figure 3**.

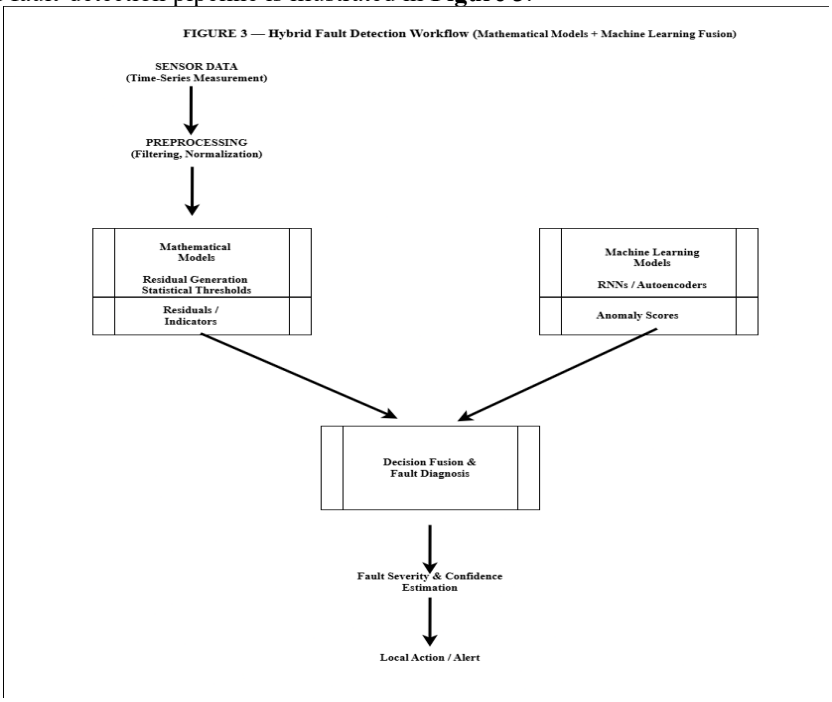

Figure 3. Hybrid fault detection workflow illustrating the integration of mathematical models and machine learning techniques for residual generation, anomaly scoring, decision fusion, and fault diagnosis with severity and confidence estimation.

Mathematical models establish baselines of normal behavior and generate residuals or statistical indicators, offering interpretability and physical consistency. In parallel, machine learning models, such as recurrent neural networks and autoencoders, analyze temporal patterns and identify complex anomalies that may not be captured by analytical models alone.

The outputs of both approaches were fused at the Edge Layer to produce a final fault decision with an associated confidence level.

3.5 Edge-Based Decision Logic

The edge-based decision logic and action-triggering mechanism used to evaluate the fault indicators and initiate responses are shown in **Figure 4**. The decision logic module integrates outputs from mathematical models and machine-learning inference to generate reliable fault diagnoses.

Key functions include:

Multi-criteria evaluation: combining residual thresholds, anomaly scores, and contextual information.

Consistency checks: Increasing confidence when both model types detect a fault.

Context awareness: Reducing false positives by accounting for operational states and environmental conditions.

Prioritization: Classifying faults based on severity and potential impact.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

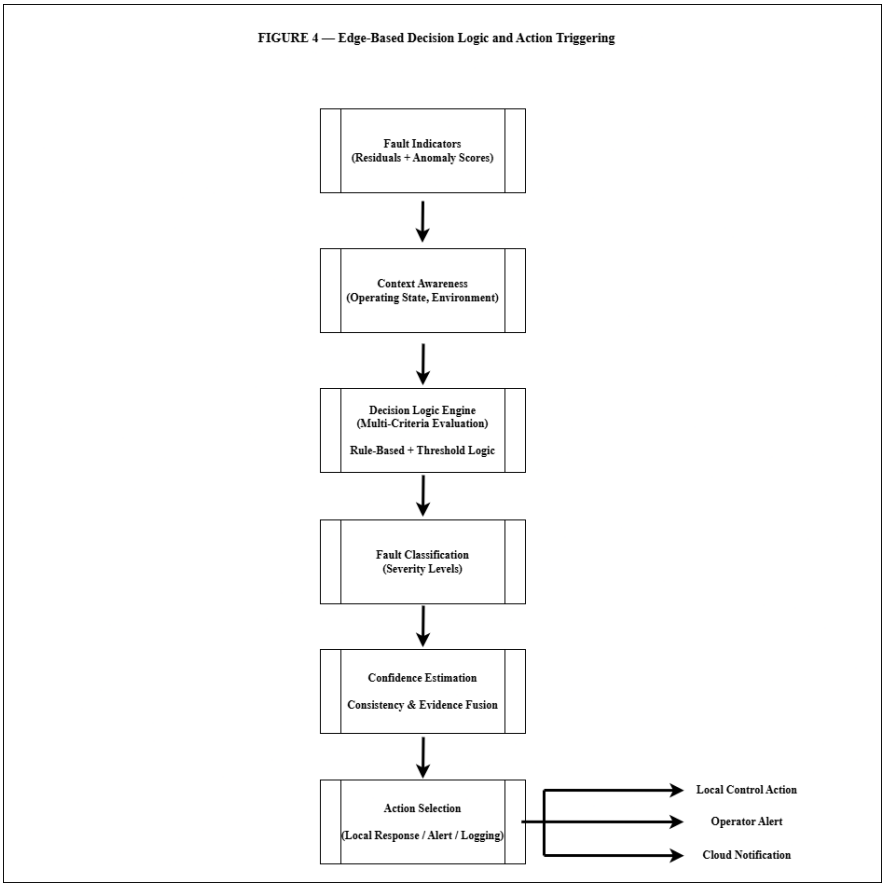

Figure 4. Edge-based decision logic and action triggering mechanism for evaluating fault indicators, contextual awareness, severity classification, and action selection.

3.6 Fault Response and Action Execution

Upon detecting a fault, the Edge Layer initiates appropriate responses based on fault severity. Critical faults may trigger immediate local actions such as isolating faulty components or shutting down unsafe operations. Fewer critical faults are logged and reported to the cloud for further analysis or scheduled maintenance.

Only essential fault information is transmitted to the Cloud Layer, ensuring a rapid response while conserving the network resources.

3.7 Mathematical Modeling and Algorithm Design

This section details the mathematical foundations and algorithmic design of the proposed hybrid fault-detection framework. The integration of analytical modeling with machine learning enables interpretable, adaptive, and resource-efficient fault diagnosis, which is suitable for edge deployment.

3.7.1 System Variables and Assumptions

3.7.1.1 System Variables

Consider an IoT subsystem monitored by an edge node using **S** sensors. At discrete time instant **t**, the sensor observation vector is defined as

$$x(t) = [x_1(t), x_2(t), \dots, xS(t)]$$

where $x_i(t)$ represents the measurement from sensor *i* at time *t*.

Additional variables include:

Fault state, F(t): indicates the system condition at time *t* (healthy or faulty).

Residual, $r_i(t)$: Difference between observed and model-predicted sensor values.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

Anomaly score, A(t): Output of machine learning models representing deviation from learned normal behavior.
Threshold, τ: Decision boundary for residuals or anomaly scores.
Context vector, C(t): Operational and environmental variables used to contextualize sensor behavior.

3.7.1.2 Modeling Assumptions

The proposed framework operates under the following assumptions:
Sensor data may contain noise and missing values, which can be mitigated through edge-level preprocessing.
Both permanent and transient faults may occur with detectable signatures in the time-series data.
Edge nodes possess sufficient computational resources for lightweight model inference.
Machine-learning training was performed in the cloud, whereas inference was executed at the edge.
Communication between layers is secure and fault-tolerant.
These assumptions reflect realistic conditions in modern IoT and Industrial Internet of Things (IIoT) deployments.

3.7.2 Mathematical Fault Detection Models

Mathematical models provide interpretable baselines for normal system behavior and form the first layer of fault detection.

3.7.2.1 Residual-Based Fault Detection

For each sensor *i*, a mathematical model Mi predicts the expected value $\hat{x}_i(t)$. The residual was computed as follows:

$$r_i(t) = x_i(t) - \hat{x}_i(t)$$

Under normal operating conditions, the residuals remained within the statistically defined bounds. A fault is indicated when

$$|r_i(t)| > \tau_i$$

where τ_i is a threshold derived from historical healthy data. Thresholds may be fixed or adaptive, depending on the operational context.

3.7.2.2 Statistical Process Control

Exponentially Weighted Moving Average (EWMA) statistics are applied to detect gradual deviations such as sensor drift:

$$Z(t) = \lambda \cdot \mathrm{r(t)} + (1-\lambda) \cdot \mathrm{Z(t-1)}$$

where $0 < \lambda \leq 1$ controls the sensitivity to recent observations. A fault is declared when Z(t) exceeds the upper or lower control limit. EWMA charts are computationally lightweight and well suited for edge deployment.

3.7.2.3 Evidence Fusion for Robust Diagnosis

To handle uncertainty and conflicting indicators, the outputs from multiple mathematical models or sensors can be fused using evidence-based reasoning. This fusion increases robustness in complex IoT environments, where single indicators may be ambiguous. Such a fusion enables more confident fault decisions, particularly in heterogeneous systems.

3.7.3 Machine Learning Models for Anomaly Detection

Machine learning models complement analytical methods by capturing complex temporal patterns in sensor data.

3.7.3.1 Recurrent Neural Networks

Recurrent Neural Networks, including Long Short-Term Memory (LSTM) variants, are used to model the sequential dependencies in time-series data. The model predicts future sensor values based on historical sequences. A significant deviation between the predicted and observed values indicates anomalous behavior.
Recurrent Neural Network (RNN)-based models are particularly effective in detecting evolving faults that manifest gradually over time.

3.7.3.2 Autoencoder-Based Anomaly Detection

Autoencoders are trained using data from healthy system operations to reconstruct input signals. During the inference, the reconstruction error is computed as follows:

$$A(t) = ||x(t) - \hat{x}(t)||$$

where $\hat{x}(t)$ denotes the reconstructed signal. A high reconstruction error indicates a deviation from normal behavior and suggests a fault. Autoencoders are particularly useful in scenarios where labeled fault data are scarce.

3.7.4 Hybrid Fault Detection Algorithm

The proposed hybrid algorithm integrates outputs from mathematical models and machine-learning inference to produce reliable fault decisions.

Algorithm 1: Hybrid Edge-Based Fault Detection

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

Initialization

1. Load optimized machine learning inference models.
2. Load parameters for mathematical models and thresholds.
3. Set system state to healthy.

Continuous Operation

4. Acquire sensor data and contextual variables.
5. Preprocess data (filtering, normalization, feature extraction).
6. Compute residuals using mathematical models.
7. Evaluate residuals against statistical thresholds.
8. Perform machine learning inference to obtain anomaly scores.
9. Fuse mathematical and ML indicators using decision rules.
10. Validate detection using contextual information.
11. Generate final fault decision and confidence score.
12. Trigger local actions or alerts if a fault is confirmed.
13. Transmit essential fault information to the cloud.

3.7.5 Implementation Considerations

3.7.5.1 Edge Hardware and Software

Edge nodes can be implemented using platforms such as Raspberry Pi or NVIDIA Jetson devices. Lightweight Linux operating systems and containerization technologies (e.g., Docker) support modular deployment and scalability. Optimized inference engines, such as TensorFlow Lite, enable efficient execution of quantized models.

3.7.5.2 Model Optimization

To ensure real-time performance at the edge, machine-learning models undergo optimization techniques, including
Quantization to reduce numerical precision and memory usage.
Pruning to eliminate redundant parameters.
Knowledge distillation to train compact models with minimal accuracy loss.
These techniques significantly reduce computational overhead while maintaining detection performance.

3.7.6 Justification of Hybrid Design

The hybrid approach balances interpretability and adaptability. Mathematical models provide transparent and explainable diagnostics, whereas machine learning models capture complex nonlinear behaviors and adapt to evolving system dynamics. Their integration reduces false positives, improves fault coverage, and enhances trust in automated fault detection decisions.

4. EXPERIMENTAL SETUP

This section describes the experimental environment used to evaluate the proposed edge-based hybrid fault-detection architecture, and presents quantitative results demonstrating its effectiveness in terms of accuracy, latency, scalability, and resource efficiency.

4.1 Experimental Environment

A simulated Industrial IoT (IIoT) environment was developed to emulate a realistic distributed IoT deployment comprising multiple sensors, edge nodes, and a centralized cloud platform.

4.2 Dataset Description

Owing to the limited availability of labeled fault data in real-world IoT systems, the evaluation utilized a hybrid dataset composed of the following:

Normal operation data: Time-series sensor data representing healthy system behavior.

Synthetic fault data: Injected fault patterns based on commonly observed IoT and IIoT failures including
Sensor drift
Sudden spikes
Stuck-at faults
Offset errors
Complete sensor failure

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

The dataset is divided into training, validation, and testing subsets. Machine-learning models are primarily trained on healthy data with limited fault samples, reflecting realistic deployment conditions. The composition of the dataset used for the experimental evaluation is presented in **Table 1**.

Data Category	Description	Proportion (%)
Normal operation	Healthy sensor behavior	70%
Drift faults	Gradual deviation over time	10%
Spike faults	Sudden abnormal sensor	8%
Stuck-at faults	Constant erroneous readings	7%
Sensor failure	Missing or zero-output readings	5%

Table 1. Dataset Composition Used for Experimental Evaluation

4.3 Hardware and Software Configuration

4.3.1 Edge Layer Configuration

Edge nodes were configured using resource-constrained single-board computing platforms running lightweight Linux operating systems with containerized fault-detection services and optimized machine-learning inference engines, consistent with the common edge-deployment practices reported in recent IoT fault-detection studies (Santo et al., 2023; Gómez et al., 2024).

4.3.2 Cloud Layer Configuration

The cloud backend was deployed on a virtualized server with sufficient computational resources to support the model training and analytics.

ML Frameworks: TensorFlow and PyTorch.

Data Storage: Distributed databases for historical sensor data and fault logs.

Orchestration: Container orchestration for model lifecycle management and edge deployment.

4.4 Evaluation Metrics

Performance was evaluated using both fault detection accuracy metrics and operational efficiency metrics.

4.4.1 Accuracy Metrics

Accuracy

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

Precision
Recall
F1-score
False Positive Rate (FPR)
False Negative Rate (FNR)

4.4.2 Operational Metrics

Fault detection latency
Edge CPU utilization
Memory consumption
Network bandwidth usage
Scalability with increasing number of IoT devices

5. RESULTS

5.1 Fault Detection Performance

The hybrid fault-detection approach demonstrated a consistently strong performance across all evaluated fault types. A quantitative comparison of the fault-detection performance across different methods is presented in **Table 2**.

Method	Accuracy (%)	Precision	Recall	F1-score
Mathematical models only	86.4	0.84	0.82	0.83
ML-only (Autoencoder)	91.7	0.89	0.90	0.89
Proposed hybrid approach	96.2	0.95	0.94	0.94

Table 2. Comparison of Fault Detection Performance

The hybrid model outperformed the standalone approach by effectively combining interpretability with adaptive anomaly detection, resulting in fewer false positives and improved recall.

5.2 Latency and Edge Efficiency

The comparative performance of the proposed approach in terms of accuracy, fault detection latency, and scalability is illustrated in **Figure 5**. Local inference at the edge significantly reduces the fault-detection latency compared with cloud-centric processing.

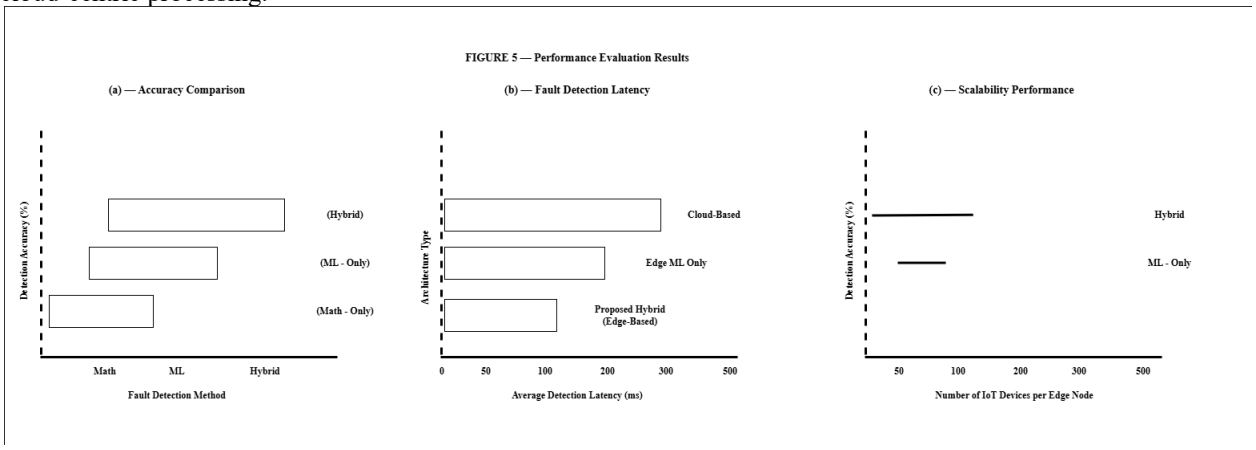

Figure 5. Performance evaluation of the proposed edge-based hybrid fault detection approach, illustrating (a) fault detection accuracy comparison, (b) average fault detection latency across architectures, and (c) scalability performance with increasing numbers of IoT devices.

The average fault-detection latency across different architectures is reported in **Table 3**.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

Architecture Type	Average Latency (ms)
Cloud-based	420
Edge ML-only	180
Proposed hybrid	130

Table 3. Average Fault Detection Latency

5.3 Resource Utilization

The optimized hybrid model maintained low resource overhead, making it suitable for continuous deployment on edge devices. The average computational and memory resource utilization at the edge nodes is summarized in **Table 4**.

Resource Metric	Average Utilization
CPU usage	38%
Memory usage	410 MB
Network bandwidth	~75% reduction

Table 4. Average Resource Utilization at Edge Nodes

5.4 Scalability Analysis

Scalability tests were conducted by increasing the number of connected IoT devices per edge node.
The system maintained a stable fault detection accuracy above 88% with up to 500 devices per edge node, while the latency increased only marginally. **Table 5** summarizes the comparative characteristics of different fault-detection approaches.

Approach Type	Interpretability	Adaptability	Latency	Edge Suitability
Rule-based	High	Low	Low	High
ML-only	Low	High	Medium	Medium
Mathematical only	High	Low	Low	High
Hybrid (Proposed)	High	High	Low	High

Table 5. Comparison of Fault Detection Approaches

6. DISCUSSION

The findings of this study demonstrate that integrating mathematical models with machine learning within an edge-based computing framework provides an effective solution to fault detection challenges in IoT networks. By relocating critical fault-detection tasks from a centralized cloud infrastructure to the network edge, the proposed architecture addresses the key limitations of cloud-centric approaches, including high latency, excessive bandwidth usage, and dependence on continuous connectivity. This shift enables faster and more autonomous responses, which are essential for time-sensitive and mission-critical IoT applications.
The central strength of the proposed approach lies in its hybrid fault detection mechanism. Mathematical models enhance interpretability, physical consistency, and robustness in data-scarce scenarios, whereas machine-learning models offer adaptability and the ability to capture complex temporal and nonlinear patterns in sensor data. The fusion of these complementary techniques improves diagnostic reliability and reduces false alarms compared to standalone methods, a balance that is particularly important in industrial and safety-critical environments, where explainability is required alongside accuracy.
Edge-based deployment further contributes to system efficiency by enabling local preprocessing and selective data transmission, which improves responsiveness and scalability, while reducing communication overhead. The experimental evaluation indicates that the architecture maintains stable performance as the number of connected IoT devices increases, supporting its applicability to large-scale deployments.
Despite these advantages, certain limitations of this study remain to be addressed. The evaluation partly relies on simulated environments and synthetic fault injection, which may not fully capture the diversity and unpredictability of real-world IoT systems. Additionally, although model optimization techniques support efficient edge deployment, resource constraints on low-power edge devices may restrict the complexity of deployable machine learning models. These limitations motivated further validation and optimization in real-world scenarios.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

7. CONCLUSION

This paper presents an **edge-based intelligent software architecture for fault detection in IoT networks** that integrates mathematical modeling with machine learning to achieve a low-latency, scalable, and interpretable fault diagnosis. The proposed architecture adopts a hierarchical edge–cloud collaboration model, where real-time fault detection and decision-making are performed at the edge, whereas computationally intensive model training and global analytics are handled in the cloud.

The main contributions of this research include:

The design of a scalable and resilient edge-based architecture addresses the latency, bandwidth, and reliability limitations of cloud-centric fault-detection.

A hybrid fault detection methodology that combines interpretable mathematical models with adaptive machine learning techniques improves accuracy and reduces false alarms.

A comprehensive experimental evaluation demonstrated a superior fault-detection performance, reduced response time, efficient resource utilization, and strong scalability.

The results confirm that the proposed approach is well suited for modern and industrial IoT environments, particularly those requiring real-time responsiveness and dependable operation. By bridging the gap between model-based and data-driven techniques, this study contributes a practical and trustworthy framework for enhancing the reliability and autonomy of IoT systems.

8. FUTURE WORK

Future research will extend the proposed architecture by incorporating lightweight explainable artificial intelligence techniques at the edge to enhance transparency and support faster root-cause analysis of fault detection (Steenwinckel et al., 2020). Adaptive learning approaches, including online and federated learning, will be explored to enable continuous model updates across distributed edge nodes while preserving data privacy (Reis and Serôdio, 2025). Resource-aware model orchestration strategies will also be investigated to dynamically adjust the model complexity based on the real-time resource constraints of heterogeneous edge devices (Gómez et al., 2024). Enhancing robustness against adversarial threats, such as data poisoning and evasion attacks, remains an important direction for secure IoT fault detection (Aouedi et al., 2024). Finally, large-scale real-world deployments will be conducted to further validate the scalability, robustness, and practical applicability across diverse IoT environments.

REFERENCES

Altarrazi, S. M., et al. (2023). Addressing the faults landscape in the Internet of Things: Toward data-centric and system resilience. *IEEE Internet Computing, 27*(6), 43–51. https://doi.org/10.1109/MIC.2023.3300508

Aouedi, O., et al. (2024). A survey on intelligent Internet of Things: Applications, security, privacy, and future directions. *IEEE Communications Surveys & Tutorials*. https://doi.org/10.1109/COMST.2024.3430368

Atoui, M. A., & Cohen, A. (2021). Coupling data-driven and model-based methods to improve fault diagnosis. *Computers in Industry, 128*, 103401. https://doi.org/10.1016/j.compind.2021.103401

Chen, H., et al. (2020). Edge computing–aided fault detection for traction control systems. *IEEE Transactions on Vehicular Technology, 69*(2), 1309–1320. https://doi.org/10.1109/TVT.2019.2957962

Chinnaiyan, V. K., Mary, G. A. A., & Mahdal, M. (2023). Integrated edge-deployable fault diagnostic algorithm for IoT. *Sensors, 23*(14), 6266. https://doi.org/10.3390/s23146266

Dong, J., et al. (2024). Real-time fault detection for IIoT using edge–cloud collaboration. *Frontiers in Neurorobotics, 18*. https://doi.org/10.3389/fnbot.2024.1499703

Gómez, P. I., et al. (2024). Edge computing method for data-driven anomaly detection. *IEEE Transactions on Industrial Electronics, 71*(10), 13319–13330. https://doi.org/10.1109/TIE.2023.3347839

Jung, D. (2020). Residual generation using physically based grey-box RNNs. arXiv. https://doi.org/10.48550/arxiv.2008.04644

Ortiz-Garcés, I., Villegas-Ch, W., & Luján-Mora, S. (2025). Implementation of edge AI for early fault detection in IoT networks. *Discover Internet of Things, 5*(1). https://doi.org/10.1007/s43926-025-00196-4

Reis, M. J. C. S., & Serôdio, C. (2025). Edge AI for real-time anomaly detection in smart homes. *Future Internet, 17*(4), 179. https://doi.org/10.3390/fi17040179

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

Santo, Y., et al. (2023). Fault detection on the edge and adaptive communication. *Sensors, 23*(7), 3544. https://doi.org/10.3390/s23073544
Savić, M., et al. (2021). Deep learning anomaly detection for cellular IoT. *IEEE Access, 9*, 59406–59418. https://doi.org/10.1109/ACCESS.2021.3072916
Seba, A. M., Gemeda, K. A., & Ramulu, P. J. (2024). Prediction and classification of IoT sensor faults using hybrid deep learning. *SN Applied Sciences, 6*(1). https://doi.org/10.1007/s42452-024-05633-7
Steenwinckel, B., et al. (2020). FLAGS: Adaptive anomaly detection using expert knowledge and ML. *Future Generation Computer Systems, 116*, 30–48. https://doi.org/10.1016/j.future.2020.10.015
Tordeux, A., et al. (2024). System reliability engineering in Industry 4.0. arXiv. https://doi.org/10.48550/arxiv.2411.08913
Wu, Y., Sicard, B., & Gadsden, S. A. (2024). Physics-informed machine learning for anomaly detection. *Expert Systems with Applications*, 124678. https://doi.org/10.1016/j.eswa.2024.124678
Zhou, Z. (2020). *Latency-critical networking*. arXiv preprint arXiv:2006.12345.