Discover Internet of Things

RESEARCH Open Access

Implementation of edge AI for early fault detection in IoT networks: evaluation of performance and scalability in complex applications

Iván Ortiz-Garces[1], William Villegas-Ch[1*†] and Sergio Luján-Mora[2†]

†William Villegas-Ch and Sergio Luján-Mora contributed equally to this work.

*Correspondence:
William Villegas-Ch
william.villegas@udla.edu.ec
[1]Escuela de Ingeniería en Ciberseguridad, Facultad de Ingenierías y Ciencias Aplicadas, Universidad de Las Américas, Redondel del Ciclista, Antigua Via a Nayon, 170125 Quito, Pichincha, Ecuador
[2]Department of Software and Computing Systems, University of Alicante, San Vicente del Raspeig, s/n, 03690 Alicante, Pichincha, Spain

Abstract

The exponential growth of Internet of Things (IoT) deployments has introduced critical demands in reliability, energy efficiency, and real-time fault detection. Traditional cloud-based solutions suffer from excessive latency and energy overhead due to continuous data transmission and centralized analysis. To address these limitations, this study introduces an edge-based artificial intelligence (AI) architecture tailored for early fault detection in heterogeneous IoT networks. The architecture leverages recurrent neural networks and autoencoders optimized for time-series anomaly detection, enabling local inference directly on edge nodes. The system was evaluated under realistic laboratory conditions using a range of IoT devices and edge computing platforms, including Raspberry Pi and Nvidia Jetson. Experimental results demonstrate a 92.0% fault detection rate with a response time consistently under 150 ms, significantly outperforming cloud-based approaches in both latency and energy metrics. Energy consumption was reduced to 50 Wh under standard conditions, and the architecture successfully scaled to support up to 500 IoT devices, maintaining stable detection accuracy above 88%. These results validate the proposed edge AI system as a scalable and energy-efficient alternative for real-time fault monitoring. Its low-latency, decentralized nature makes it suitable for deployment in industrial automation, smart city infrastructure, and mission-critical IoT applications where operational continuity and autonomous decision-making are essential.

Keywords Edge AI, IoT fault detection, Network resilience, Energy efficiency

1 Introduction

In the last decade, the Internet of Things (IoT) has transformed industries by interconnecting devices and systems through smart networks, enabling real-time data collection and analysis [1]. However, the rapid expansion of IoT networks introduces significant challenges, including security vulnerabilities, reliability issues, and operational inefficiencies. Early failure detection is critical for ensuring system resilience and operational

© The Author(s) 2025. **Open Access** This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by-nc-nd/4.0/.

efficiency, particularly in applications such as healthcare, manufacturing, and smart cities [2].

Traditional fault detection methods often rely on centralized architectures, where data is transmitted to cloud servers for analysis [3]. While these approaches provide high computational capacity, they incur substantial latencies and energy consumption due to constant data transmission. Centralized processing also creates single points of failure, increasing security risks. In contrast, edge artificial intelligence (AI) systems process data locally or near IoT devices, significantly reducing latency and energy consumption [4].

Edge computing nodes, such as Single Board Computers (SBCs) like Raspberry Pi and Nvidia Jetson, facilitate real-time data processing by operating closer to data sources [5]. This distribution of computational power enhances the resilience of IoT networks by mitigating latency and minimizing dependency on centralized systems.

Despite recent advances in edge AI, most existing implementations rely on shallow models or static, rule-based systems, which limit their ability to adapt to dynamic and noisy environments. This work proposes a novel edge-based architecture that combines Recurrent Neural Networks (RNNs) and Autoencoders (AEs), specifically designed to handle time-series data and detect multivariate anomalies typical in IoT telemetry. RNNs are particularly suitable for modeling temporal dependencies and capturing evolving behavior, while AEs enable unsupervised learning of fault patterns through data reconstruction error. This combination enhances detection accuracy and robustness in resource-constrained scenarios.

This study focuses on early detection of IoT network failures, including sensor data anomalies, network communication issues, and device malfunctions. By leveraging edge AI, patterns and trends are analyzed in real-time, enabling the identification of complex faults that may not be detectable through conventional methods. This capability is crucial for preventing cascading failures and ensuring operational continuity in critical environments. The main novelty of this work lies in the integration of temporal deep learning models with a lightweight edge computing infrastructure, allowing decentralized and low-latency anomaly detection without relying on cloud assistance. Unlike prior approaches, this system is fully autonomous, scalable, and validated under realistic conditions.

This work's primary contributions include implementing and evaluating an edge AI system that improves operational efficiency by significantly reducing latency and energy consumption compared to traditional cloud-based methods. The system's scalability is validated by demonstrating its ability to maintain high fault detection accuracy even as the number of connected IoT devices increases. Advanced AI models, such as Recurrent Neural Networks (RNNs) and Autoencoders, detect complex patterns in heterogeneous IoT environments, enhancing the system's robustness and security.

To achieve these objectives, controlled experiments were conducted using various IoT devices [6], edge computing nodes, and advanced AI algorithms in a university laboratory environment. The selected IoT devices included temperature and humidity sensors, surveillance cameras, asset tracking devices with RFID technology, and motion sensors. Edge computing nodes such as Raspberry Pi and Nvidia Jetson were configured to process data locally. The communication network utilized Wi-Fi and Ethernet, with routers and switches optimized for data traffic management [7]. Uninterruptible power supplies (UPS) and solar panels also ensured a stable and sustainable energy supply.

The chosen AI models were trained and validated using historical failure data, employing cross-validation and hyperparameter tuning techniques. Data preprocessing involved cleaning, normalization, and feature extraction to enhance model precision. Optimization through model compression and hardware acceleration further improved system performance.

Experimental results show that the edge AI system achieves a fault detection rate of 92.0% with response times below 150ms, significantly outperforming cloud-based alternatives regarding latency and energy consumption. Scalability tests indicate that the system maintains efficient performance with a fault detection rate above 88% and a moderate response time increase to 160ms when handling up to 500 IoT devices. These findings underline the advantages of edge AI in improving operational efficiency, resilience, and energy sustainability for IoT networks.

2 Literature review

Integrating AI for early fault detection in IoT networks is a critical area of research that enhances resilience and efficiency. Previous studies have addressed AI-based IoT systems' fault detection, performance optimization, and scalability. This review highlights relevant works and positions the present study within this context.

Ahmad et al. [8] demonstrated the potential of deep neural networks (DNNs) for anomaly detection in cloud-based IoT networks, achieving high accuracy. However, their centralized approach introduced latency and energy inefficiencies, critical challenges in real-time applications. Similarly, Santo et al. [9] explored fault detection at the edge, achieving improved response times and detection accuracy compared to cloud-based solutions. However, their work did not evaluate scalability. In contrast, this study demonstrates that edge AI maintains performance under increased loads, as detailed in Table 1, which compares techniques and key results across studies.

In addition to addressing latency and scalability, our approach leverages advanced AI models, such as RNNs and Autoencoders, optimized for time-series analysis and anomaly detection in IoT environments. Recent works, such as Lee et al. [10], have shown the effectiveness of autoencoder architectures for detecting subtle anomalies in industrial IoT, providing insights that align with this study's focus on edge AI. These comparisons emphasize how distributed processing at the edge reduces latency and enhances system resilience.

Energy efficiency is another crucial factor in deploying intelligent detection models in IoT networks. Wang et al. [11] demonstrated that cloud-based systems require significant energy for data transmission and centralized processing, which can compromise autonomy in energy-constrained environments. In contrast, Ahmad et al. [12] and Thein et al. [13] explored federated learning techniques that reduce communication overhead by keeping data local, resulting in a reduced energy impact but at the cost of increased model management complexity. Additionally, Thwal et al. [14] proposed lightweight convolutional transformers optimized for edge deployment, striking a balance between detection performance and minimal computational load.

Table 1 Comparison of techniques and results in the literature

A	B	C	D	E	F
Ahmad et al. [8]	Deep Neural Networks (DNN)	Cloud	Anomaly detection rate, power consumption	High detection accuracy, high energy consumption	Lower energy consumption and latency with edge AI
Lee et al. [10]	Deep-Learning Autoencoders (DLAE)	Edge	Anomaly detection accuracy, adaptability	Effective in detecting early anomalies in manufacturing, optimized for industrial IoT	Provides a relevant comparison, extends analysis to predictive signal detection and real-time processing in IoT environments
Santo et al. [9]	Machine learning at the edge	Edge	Response time, detection accuracy	Improved accuracy and response time, does not evaluate scalability	Expands by evaluating scalability with more devices
Wang et al. [11]	Cloud-fog architecture	Cloud and edge	Power consumption, latency	Higher cloud consumption, better latency at the Edge	Confirms energy efficiency and improvements in early detection
Ahmad et al. [12]	Mini-batch federated learning with MLP	Edge (federated)	Detection accuracy, false alarm rate	98.85% accuracy, 0.09% FAR, low resource usage	This study matches accuracy while simplifying architecture and integration
Thein et al. [13]	Adversarial transformer for time-series	Edge/cloud hybrid	F1-Score, reconstruction error	F1-score >0.89 on 4 datasets	This study avoids two-stage GAN training, focuses on real-time IoT signals
Thwal et al. [14]	Lightweight convolutional transformer	TinyML/on-device FL	Accuracy, MACs, model size	Superior accuracy with < 1.3 M params, < 0.1G MACs	This study complements by prioritizing detection rather than image classification

A: Reference; B: Technique Used; C: Deployment Environment; D: Performance Metrics; E: Key Results; F: Comparison with this study

3 Materials and methods

The materials used to implement and evaluate an edge AI system for early fault detection in IoT networks involves a combination of IoT devices, edge computing nodes, and advanced AI algorithms. Multiple IoT devices were strategically installed and configured in a laboratory environment to simulate real-world conditions. Edge computing nodes like Raspberry Pi and Nvidia Jetson were deployed for local data processing [15, 16]. The development of fault detection algorithms used historical fault data and various machine learning techniques to ensure accurate and efficient fault detection [17]. The communications network and electrical infrastructure were optimized to support the experimental setup, providing reliable data transmission and sustainable energy use. Detailed experiments were conducted to evaluate the system's performance, scalability, and energy efficiency compared to traditional cloud-based approaches.

Figure 1 illustrates the overall architecture of the proposed edge AI system for early fault detection in IoT networks. The diagram outlines the main components and data flow, beginning with the collection of data from heterogeneous IoT devices (such as sensors, cameras, and trackers), followed by local processing at edge nodes using AI models. Faults are detected in real-time through a dedicated module, and the results are routed to a real-time alerting system, which not only provides immediate notifications but also supports monitoring and visualization through dedicated interfaces. This architecture minimizes latency and energy consumption by leveraging edge computing, ensuring scalability and robustness in fault detection tasks.

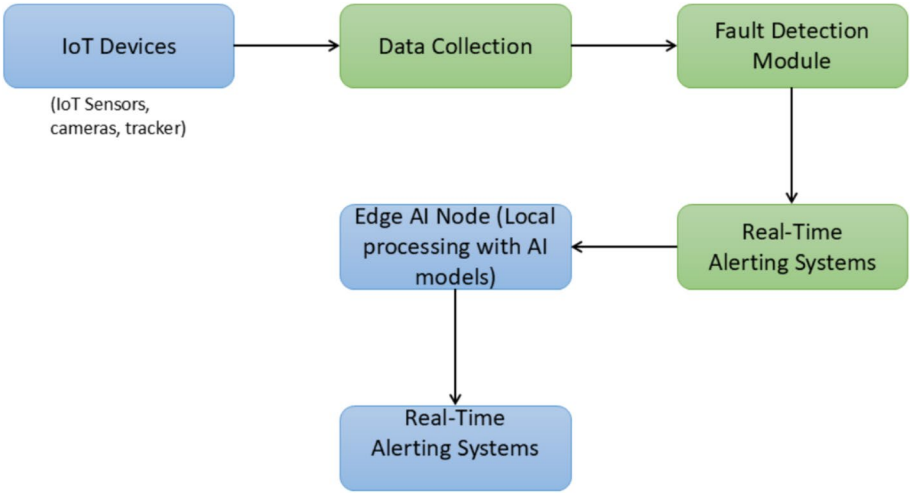

Fig. 1 General architecture for early fault detection using edge AI

Table 2 Technical specifications and selection criteria for IoT devices

A	B	C	D	E	F
Temperature sensor	DHT22	Every 5 s	Wi-Fi	−40 to 80 °C	High precision, low power consumption
Humidity sensor	DHT22	Every 5 s	Wi-Fi	0 to 100%	High precision, low power consumption
Surveillance camera	Arlo Pro 3	Continuous	Wi-Fi	Up to 300 ms	High resolution, good connectivity
RFID	Zebra RFD8500	Continuous	Bluetooth, RFID	Up to 10 ms	High precision, versatility
Motion sensor	HC-SR501	Every second	Wi-Fi	0 to 7 ms	High sensitivity, low cost

A: Device; B: Model; C: Data Rate; D: Communication Protocol; E: Operation Range; F: Selection Criteria

3.1 Environment description

3.1.1 Setup and configuration of IoT devices and edge nodes

This study implements and evaluates an edge AI system for detecting early failures in IoT networks. It identifies and mitigates issues before they impact the network's performance. Controlled experiments were conducted in a university laboratory environment, focusing on the installation of diverse IoT devices, the configuration of edge computing nodes, and the development of AI algorithms.

The selected IoT devices include temperature and humidity sensors, surveillance cameras with real-time streaming capabilities, asset tracking devices equipped with RFID technology, and motion sensors. These were strategically deployed to simulate a heterogeneous network environment representative of real-world operating conditions. Table 2 outlines the technical specifications and selection criteria for the devices used, providing a detailed overview of their capabilities and relevance to the study.

By leveraging these devices in conjunction with edge computing nodes, the experimental setup ensures a robust platform for evaluating the performance of AI algorithms under varied conditions. It captures diverse data inputs to emulate realistic IoT network operations

For this study, devices such as Raspberry Pi and Nvidia Jetson are used, and they are selected for their local processing capabilities and flexibility in software configuration.

The first step is installing compatible operating systems, such as Raspbian for Raspberry Pi and Ubuntu for Nvidia Jetson. Subsequently, development environments are configured, including the installation of Python and machine learning frameworks such as TensorFlow and PyTorch [18]. In addition, node monitoring and management tools, such as Nagios and Grafana, are implemented to ensure the correct operation and performance of the nodes during the experiments.

These devices were selected based on specific criteria: low power consumption, support for edge AI deployment (through CUDA compatibility or optimized TensorFlow Lite support), and widespread adoption in IoT prototyping environments. Raspberry Pi 4B was chosen for its cost-effectiveness and compatibility with lightweight AI models, whereas Jetson Nano was selected for its superior GPU capabilities and ability to handle more complex neural networks. This combination enables the benchmarking of AI model performance under various hardware constraints. The criteria align with the study's goals of achieving both low-latency inference and broad scalability in real-world settings.

3.1.2 Network architecture and experimental setup

This study employs two distinct network architectures: one for cloud-based fault detection and another for edge-based detection. The cloud-based architecture leverages centralized servers at a data center 500 km from the experimental site. Each server is a Dell PowerEdge R740, equipped with dual Intel Xeon Silver 4214 CPUs, 64 GB of RAM, and NVIDIA Tesla T4 GPUs. This system can handle large data volumes but introduces latency due to the physical distance and reliance on centralized processing. The data center's power management system monitors energy consumption, including cooling and idle power.

Conversely, the edge-based setup places computational nodes such as Nvidia Jetson Xavier NX devices closer to the data source. These nodes, equipped with a 6-core Carmel ARM CPU, a 384-core Volta GPU with 48 Tensor Cores, and 8 GB of LPDDR4x memory, enable localized data processing. Energy consumption is measured in real-time using inline power meters, providing detailed insights under various operational conditions, including regular operation, high workload, low power mode, and failure recovery. The energy efficiency of these setups is rigorously compared to highlight the advantages of edge computing.

Figures 2 and 3 illustrate the network structures. Figure 2 shows the cloud-based system, where IoT devices transmit data to a distant data center via a network backbone. Figure 3 highlights the edge-based system, emphasizing localized processing and redundant connections between nodes, ensuring faster response times and higher fault detection precision.

The system detects various IoT network failures, such as sensor anomalies, network communication disruptions, and device malfunctions. Temperature and humidity sensors monitor for anomalies, while surveillance cameras and RFID devices detect connectivity issues or abnormal activity patterns. By processing this data locally, the edge AI system identifies subtle patterns and anomalies, surpassing the capabilities of traditional rule-based methods.

The communication network employs Wi-Fi and Ethernet, with routers and switches optimized for efficient data transmission. Security measures, including WPA2

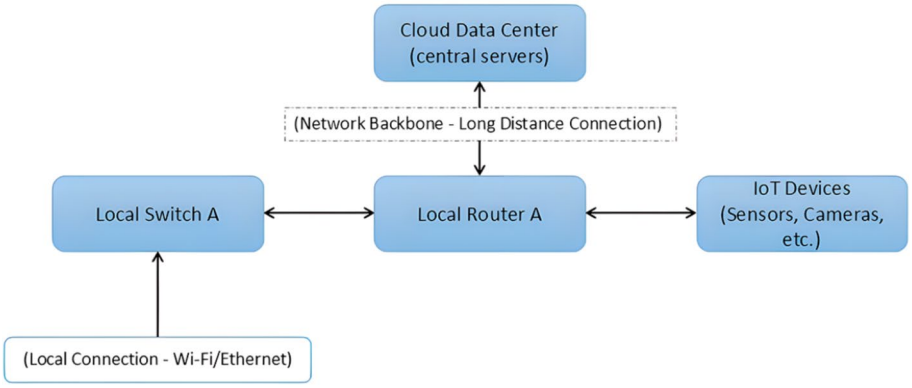

Fig. 2 Network structure for cloud-based fault detection

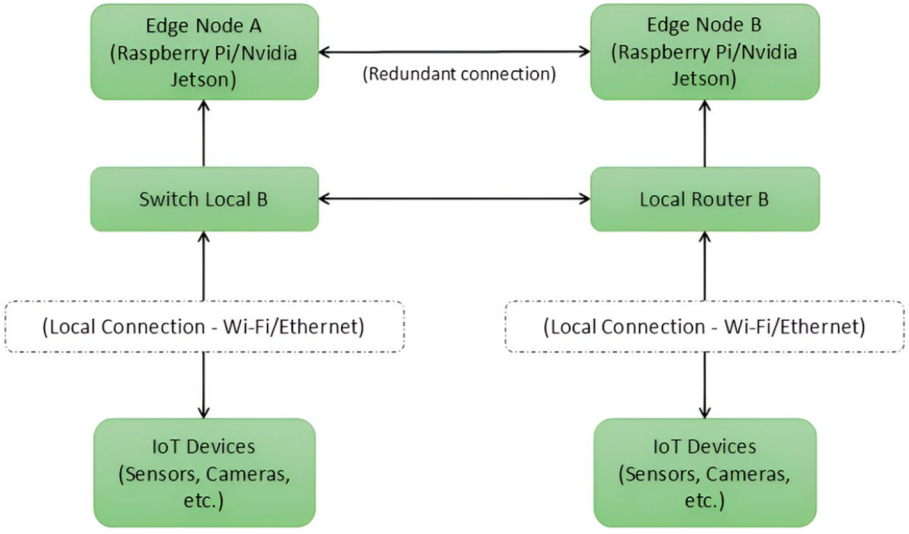

Fig. 3 Network structure for edge fault detection

encryption and firewalls, safeguard data integrity and confidentiality [19, 20]. Energy infrastructure, comprising uninterruptible power supplies (UPS) and solar panels, ensures system reliability and sustainability [21]. Energy consumption is continuously monitored to maintain efficiency and stability.

Advanced AI models such as Recurrent Neural Networks (RNNs), Autoencoders, and time series models are employed for fault detection algorithm development. Historical IoT failure data is collected, cleaned, normalized, and used for feature extraction [22]. Models are trained and validated using cross-validation and hyperparameter tuning. Adaptive mechanisms are implemented to update the models with new data, ensuring long-term effectiveness.

AI enables proactive fault detection. Unlike simple threshold-based alerts, AI models analyze patterns to predict failures, such as calibration drifts or impending hardware issues, enabling preemptive maintenance. This capability significantly enhances the resilience and reliability of IoT networks.

Experimental simulations replicate IoT network failures, including device disconnections, sensor faults, transmission anomalies, and cyber-attacks. Real-time monitoring tools evaluate system performance, and results are analyzed using statistical methods.

The edge AI system is benchmarked against cloud-based solutions, highlighting its reduced latency, improved energy efficiency, and scalability. Scalability tests demonstrate the system's capacity to maintain fault detection accuracy and performance even with increasing devices and data volumes.

The work of Asonye et al. [23] is referenced for further understanding of network configurations in IoT environments. This study provides a detailed analysis of network topologies such as mesh, star, and tree, discussing their influence on data transmission, latency, and energy consumption, supporting the arguments presented in this study.

3.2 Experimental environment configuration

The experimental setup involves the strategic deployment of IoT devices, including ten temperature and humidity sensors, four surveillance cameras, five RFID-enabled asset-tracking devices, and six motion sensors. These devices are positioned to capture diverse operational data, ensuring comprehensive coverage and enabling failure detection across varied scenarios [24].

Edge computing nodes, comprising Raspberry Pi and Nvidia Jetson devices, are distributed throughout the environment to minimize transmission distances and reduce latency. This placement enhances real-time processing and mitigates cumulative delays caused by network congestion and centralized server processing. Localized processing at the edge eliminates routing and queuing delays often encountered in cloud-based architectures, offering a 50 ms latency improvement crucial for applications requiring immediate response, such as industrial automation and critical infrastructure monitoring.

Each Raspberry Pi node operates with the Raspbian OS, and Nvidia Jetson nodes utilize Ubuntu, both configured with Python and frameworks like TensorFlow and PyTorch. These environments enable the execution of machine learning models directly at the edge, ensuring near real-time fault detection. Tools like Nagios and Grafana are implemented for node monitoring and management, ensuring optimal system performance throughout the experiments [25].

The network architecture supports low-latency communication through Wi-Fi and Ethernet connections, directly linking IoT devices to edge nodes. This setup enhances system resilience by reducing reliance on centralized servers. Unlike cloud-based solutions, which face inherent delays due to centralized data processing and queuing, edge AI systems process data locally, enabling rapid decision-making critical for safety-critical environments. The network employs WPA2 encryption and firewalls to secure data transmission while subnet segmentation isolates traffic, improving security and operational efficiency.

Energy infrastructure includes UPS systems and solar panels to provide a reliable and sustainable power supply. Continuous energy consumption monitoring ensures system stability and prevents interruptions that could compromise experimental validity. Figure 4 illustrates the block diagram of the experimental environment, detailing the interconnections between IoT devices, edge nodes, network architecture, and power infrastructure.

The diagram highlights how IoT devices connect edge nodes through segmented networks for secure and efficient data processing. Edge nodes perform real-time analysis and are equipped with monitoring tools to maintain system reliability. The power system

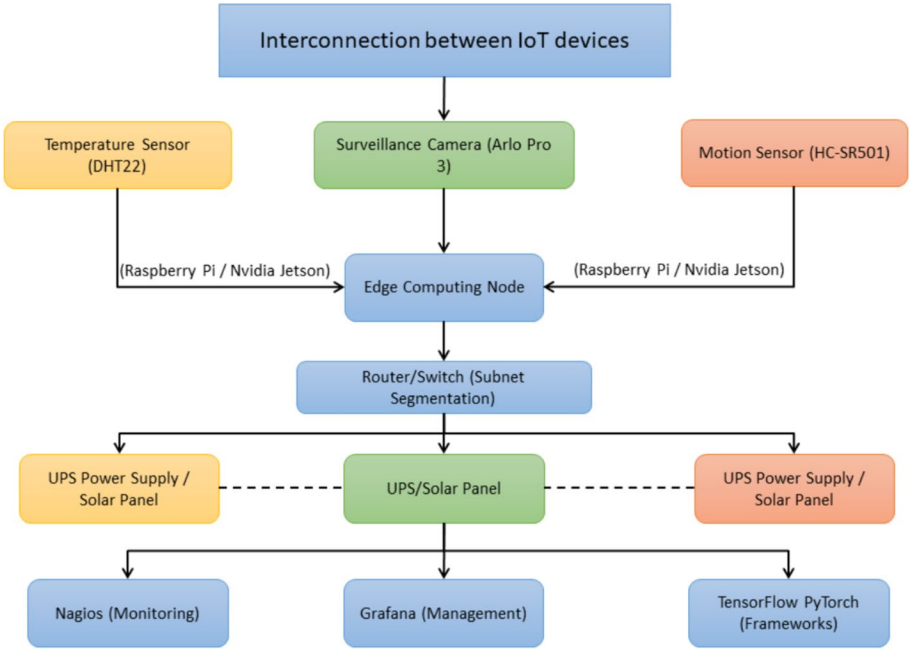

Fig. 4 Block diagram of the experimental environment for fault detection in IoT networks

ensures uninterrupted operation, reinforcing the experimental environment's robustness and sustainability.

3.3 Development of fault detection algorithms

The development of fault detection algorithms employs AI models such as RNNs, Autoencoders, and time series models, which are particularly suited for addressing the variability and complexity of IoT network failures. The training and validation data are sourced from open datasets, including the KDD Cup 1999 IoT network dataset and the IoT-23 dataset and datasets generated through simulations in the experimental environment, amounting to approximately 500 GB and 10 million records [26].

Data preprocessing ensures high-quality inputs for model training. This includes cleaning to eliminate duplicates and handle missing values using imputation techniques (mean, median, or predictive models). Statistical methods such as Z-score and IQR are used to filter outliers. Normalization scales numerical features using Min-Max Scaling or Z-score normalization. At the same time, categorical variables are encoded with methods like One-Hot Encoding or embeddings, depending on the model requirements [27]. Feature extraction focuses on temporal patterns in time-series data and spatial features in image data using convolutional techniques.

The choice of RNNs and Autoencoders is motivated by their proven capacity to handle sequential and high-dimensional data, respectively. RNNs, and particularly their LSTM and GRU variants, are designed to retain temporal dependencies across time steps, making them ideal for identifying anomalies in sensor sequences where short-term patterns alone may not reveal faults. This temporal memory enables the model to distinguish between transient noise and consistent deviations that indicate emerging problems. Autoencoders, on the other hand, are effective in capturing the expected behavior of high-dimensional data by learning compressed representations; their ability to reconstruct inputs makes them naturally suited for detecting deviations caused by faults or

unexpected behavior. This unsupervised nature is particularly advantageous in IoT, where labeled failure data is often scarce. Together, these models form a complementary framework: RNNs offer temporal resolution, while Autoencoders provide sensitivity to structural and statistical irregularities.

The AI models address specific failure scenarios. RNNs analyze sequential data to detect temporal anomalies, such as irregular sensor readings over time. Autoencoders identify deviations from normal states by learning compact data representations and flagging inputs that fall outside the expected range. Time series models track trends and outliers indicative of gradual sensor degradation or network issues.

A key innovation is the predictive capability of these models. Traditional methods often react to evident failures, whereas using RNNs and Autoencoders enables the detection of subtle anomalies that precede critical faults. For instance, these models can recognize patterns such as increasing latency or intermittent connectivity issues before a device disconnects. This capability is validated by studies like Lee et al. [10], which highlight the effectiveness of Autoencoders in detecting early anomalies in industrial IoT environments.

Compared to centralized approaches like Ahmad et al. [8], which employ DNNs in cloud settings, this study leverages edge computing to optimize real-time processing. Lightweight models, including LSTM-based RNNs, are tailored for edge deployment, reducing latency and computational overhead while maintaining high accuracy. This edge-based architecture eliminates delays associated with centralized data transmission and processing, enabling real-time fault detection critical for applications requiring immediate responses.

The training phase employs cross-validation to ensure model robustness and prevent overfitting. Hyperparameters such as learning rates, number of layers, and units per layer are tuned for optimal performance. The data is split into training (70%), validation (15%), and testing (15%) sets, and metrics like precision, recall, and F1-score are used to evaluate effectiveness.

The datasets generated during the simulation phase were created using a custom Python-based framework designed to replicate the behavior of real-world IoT devices under controlled fault conditions. These simulations incorporated signal delays, communication errors, sensor drift, and intermittent failures to reflect a range of operational scenarios.

Ensemble models are employed to minimize false positives, requiring consensus among models before flagging faults. Post-processing filters analyze detected anomalies' context and consistency, reducing spurious detections. This approach ensures sensitivity and specificity, maintaining system reliability. Continuous adaptation mechanisms enable models to update their parameters based on new data, thereby ensuring their effectiveness in evolving IoT environments. This is achieved through periodic retraining with fresh data and real-time feedback.

This retraining process incorporates mechanisms to detect data drift and evolving fault patterns. For instance, autoencoders are equipped with adaptive reconstruction error thresholds that adjust based on recent input distributions, allowing the system to remain sensitive to emerging failure modes. Additionally, edge nodes maintain short-term temporal buffers that capture recent data trends and outliers. These buffers serve both as real-time feedback mechanisms and as input for scheduled incremental updates

of the model. This design allows the system to adapt not only periodically but also continuously to dynamic environments, improving resilience against rapidly evolving faults.

Techniques such as structured network pruning and post-training quantization are applied to compress models before deployment on resource-constrained edge nodes, such as Raspberry Pi. Pruning removes neurons and synaptic connections with minimal impact on inference output, while quantization maps floating-point weights to 8-bit integer representations to reduce memory usage and improve inference latency. These operations are applied after the complete training phase to preserve model accuracy. Compressed models are exported in lightweight formats, such as TensorFlow Lite or ONNX, allowing for compatibility with hardware-accelerated edge runtimes. When supported, hardware accelerators such as edge TPUs or embedded GPUs are used to optimize execution [28, 29].

3.4 Conducting experiments

To evaluate the ability of the edge AI system to detect failures in IoT networks, several experiments were designed and executed to simulate different types of failures in IoT devices and communication networks. These experiments include device disconnections, sensor failures, cyber attacks, and data transmission anomalies and are detailed in Table 3.

3.5 Evaluation and validation

For experiment 1, one or more IoT devices, such as temperature and humidity sensors, surveillance cameras, and RFID devices, were suddenly disconnected. The disconnection was carried out manually and controlled to evaluate the system's ability to detect and respond to a loss of communication. The disconnection detection time and the system's ability to notify and manage the failure were measured. Evaluation metrics included milliseconds (ms) detection time and false positive and false negative rates.

In experiment 2, failures were induced in specific sensors, such as temperature and humidity sensors. Failures were simulated by altering sensor readings to generate anomalous data. The objective was to evaluate the system's accuracy in detecting these anomalies. Fault injection techniques included manipulation of input data to represent non-real conditions. The evaluation metrics were precision, recall, and F1-score.

Table 3 Examples of experiments and evaluation metrics

Example	Description	Purpose	Evaluation metric
Device disconnection	Simulation of sudden device disconnection	Measure failure detection and management time	Detection time (ms), false positive rate (%), false negative rate (%)
Sensor failure	Induction of failures in temperature and humidity sensors	Verify identification of abnormal readings	Precision (%), recall (%), F1-score
Cyber attacks	Execution of cyber attacks	Measure effectiveness in detecting and mitigating attacks	Attack detection rate (%), response time (ms), impact on system performance (%)
Data transmission anomalies	Introduction of anomalies in data transmission	Maintain the integrity and precision of transmitted data	Packet loss rate (%), latency (ms), retransmission rate (%)
Energy efficiency assessment	Measurement of energy consumption under different conditions	Compare energy consumption with traditional approaches	Energy consumption (Wh), energy efficiency (%), energy savings compared to the cloud (%)

For experiment 3, cyber-attacks, such as denial of service (DoS) attacks and data manipulation, were conducted to evaluate the system's resilience to external threats. DoS attacks were performed by sending many requests to IoT devices to overload the network. Data manipulation was performed by altering data in transit to evaluate the system's ability to detect unauthorized modifications. Evaluation metrics included attack detection rate, response time, and impact on system performance.

Experiment 4 introduced data transmission anomalies such as packet loss and network delays. These failures were simulated by controlled errors introduced into the communication network. The objective was to evaluate the system's robustness to maintain the integrity and accuracy of the transmitted data. Evaluation metrics included packet loss rate, latency, and retransmission rate.

In experiment 5, the system's energy consumption under normal and fault conditions was measured to evaluate the energy efficiency. Electrical consumption measurement devices were used to record consumption in different operating scenarios. The objective was to compare the energy consumption of the edge AI system with traditional cloud-based approaches. Evaluation metrics included energy consumption in (Wh), efficiency, and energy savings compared to the cloud. The table summarizes the experiment examples, objectives, and evaluation metrics.

3.6 Statistical analysis and validation

The results of the experiments will be analyzed using statistical and analytical methods to evaluate the accuracy and robustness of the fault detection system. The performance of the edge AI system will be compared with cloud-based solutions, highlighting the advantages and limitations of both approaches. Additionally, the energy efficiency of the edge AI system is evaluated, and scalability tests will be performed to measure its ability to handle an increase in the number of IoT devices and data volume. Various evaluation metrics will be used to assess the accuracy and robustness of the fault detection system [30]. These metrics include precision, recall, F1-score, and false positive and negative rate.

Precision: measures the proportion of true positives and true negatives over the total number of cases evaluated.

$$\text{Precision} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{1}$$

Recall: measures the proportion of detected true positives over the total of actual true positives.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

The F1-score is the harmonic mean of precision and recall, balancing the two.

$$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

False Positive Rate (FPR): measures the proportion of false positives over the total number of true negatives.

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \tag{4}$$

False Negative Rate (FNR): measures the proportion of false negatives over the total true positives.

$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{TP} + \mathrm{FN}} \tag{5}$$

where TP are the true positives, TN are the true negatives, FP are false positives, FN are false negatives.

Parallel tests will be carried out using both approaches to compare the performance of the edge AI system with that of cloud-based solutions. Latency, responsiveness, and accuracy in fault detection will be measured in both systems. Differences in performance will be quantified through statistical tests such as the student t-test for related samples, which allows the means of two related samples to be compared and determine if there are significant differences.

$$t = \frac{\overline{X_1} - \overline{X_2}}{S_{\mathrm{diff}}/\sqrt{n}} \tag{6}$$

where $\overline{X_1} - \overline{X_2}$ are the means of the two samples. S_{diff} is the standard deviation of the differences between the samples. *n* is the sample size.

The energy efficiency of the edge AI system will be evaluated by measuring the energy consumption under different operating conditions. Electrical consumption measurement devices will record consumption in (Wh). Energy efficiency will be calculated by comparing the energy consumption of the edge AI system with cloud-based approaches.

$$\text{Energy efficiency} = \left(\frac{\mathrm{CEC} - \mathrm{PEC}}{\mathrm{ECC}}\right) \times 100 \tag{7}$$

where CEC: Cloud Energy Consumption, representing the total energy consumed by a cloud-based system. This includes the energy required for data transmission, processing, and storage in remote data centers and the power consumed by network infrastructure. PEC: Perimeter Energy Consumption, representing the energy consumed by the edge AI system, including local data processing and network communication at the edge devices. ECC: Energy Consumption for Cloud Processing, referring specifically to the energy used for data processing and storage within the cloud infrastructure, excluding the energy consumed during data transmission from IoT devices to the cloud.

Scalability tests measure the system's ability to handle increased IoT devices and data volume. These tests involve simulating an increasing number of devices and analyzing how this affects system performance. Latency, processing time, and error rate are measured as system load increases.

3.7 Real-time implementation

The trained model is integrated into the production environment, ensuring optimized performance and robustness. The system architecture consists of IoT devices, edge computing nodes, a robust communication network, and monitoring tools, as shown

in Fig. 5. This architecture ensures seamless interaction between components, enabling quick and accurate responses to detected failures.

Techniques such as model compression and hardware-accelerated inference enhance real-time performance. Model compression, achieved through network pruning and quantization, reduces the model's size and complexity without significantly impacting accuracy. Hardware acceleration using GPUs and TPUs ensures faster data processing and improved system responsiveness.

System performance in production is evaluated using stress testing and continuous monitoring. Stress testing simulates high workloads to assess the system's behavior under extreme conditions. Metrics such as latency, response time, and error rate are continuously monitored to identify potential bottlenecks and optimize performance. Maintenance and updates follow established procedures, including:

- Continuous monitoring: Tools like Nagios and Grafana monitor system performance in real-time. Automated alerts notify administrators of anomalies or failures, enabling prompt intervention [30].
- Model updates: Mechanisms are in place to periodically update the AI models with new data collected from the production environment. These updates ensure the system remains adaptable to changing conditions and performs well.
- Incident documentation and Reporting: All detected incidents are documented, detailing the nature of the failure and the system's responsibility. The corrective actions are taken. This builds a comprehensive incident history to improve future performance and resilience.

This end-to-end approach to real-time deployment, which combines advanced optimization techniques with robust maintenance procedures, ensures that the edge AI system operates efficiently in production environments.

3.8 Ethical and safety considerations

Developing and deploying an edge AI system for fault detection in IoT networks must prioritize data privacy, security, and ethical compliance. Robust technical and organizational measures are implemented to protect data and ensure the system aligns with applicable regulations and ethical standards.

Data Privacy and Security To safeguard data privacy, personal information is anonymized or pseudonymized, minimizing the risk of identification even in a breach. Data in transit and at rest is encrypted using AES-256, ensuring protection against unauthorized

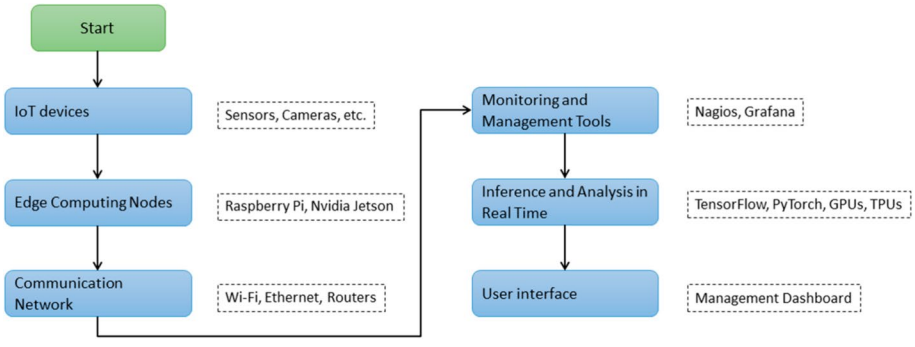

Fig. 5 Real-time implementation architecture

access [31]. Strict access controls, including multifactor authentication (MFA) and role-based privilege management, restrict data access to authorized personnel [32]. Continuous monitoring with tools such as Nagios and Grafana, alongside periodic security audits, ensures real-time detection and response to suspicious activity.

Ethical Implications of AI Implementation The AI system is designed to be transparent and explainable. It documents algorithms and provides comprehensible explanations for detected faults. By optimizing models through cross-validation and continuous tuning, the system minimizes false positives and negatives, reducing unnecessary alarms or undetected issues. The impact on users is carefully evaluated, ensuring proportionate and justified responses to detected faults.

Compliance with European regulations, including the General Data Protection Regulation (GDPR), is integral to the system's design. Users provide informed consent for data collection, and mechanisms are implemented to respect their rights, such as access, rectification, and data erasure. Data Protection Impact Assessments (DPIAs) are conducted to identify and mitigate risks, with regular reviews ensuring ongoing compliance [33, 34].

Cybersecurity Measures To protect against cyber-attacks, the system incorporates advanced firewalls, intrusion detection systems, and regular penetration testing to identify vulnerabilities. Software and hardware are updated with the latest security patches to mitigate exploitation risks. Additionally, cybersecurity training ensures staff are equipped with best practices to address emerging threats.

Social and Regulatory Considerations The implementation of edge AI in IoT networks has significant social implications. Social impact assessments evaluate the effects on users and communities, ensuring open communication with stakeholders. A data protection officer (DPO) and an ethics committee oversee the system's operation, maintaining the highest ethical standards.

Under the European Union Artificial Intelligence Act (EUAIA), the system's risk level is evaluated based on its intended application. Compliance consultancy ensures adherence to data quality, transparency, and human oversight requirements. Additional quality control and risk mitigation processes will be implemented to meet regulatory standards if classified as high risk.

Specific challenges inherent to edge computing environments are also addressed to ensure resilience against physical and adversarial threats. Given the distributed nature of edge nodes and their frequent deployment in unattended or public locations, physical access attacks pose a significant risk. To mitigate this, tamper-resistant enclosures, secure boot processes, and hardware root-of-trust mechanisms are employed to prevent firmware manipulation and unauthorized access to hardware.

Additionally, to counter model inversion and membership inference attacks, particularly relevant in AI systems deployed on resource-constrained devices, techniques such as differential privacy and dropout regularization are implemented during training. These methods help obscure individual training data points and reduce the risk of sensitive data reconstruction. The models are also obfuscated and signed to ensure authenticity and prevent reverse engineering. These safeguards collectively enhance the robustness of the edge AI system under hostile or semi-trusted deployment scenarios.

4 Results

The results are divided into system performance, scalability, and energy efficiency. Each category highlights significant improvements and challenges observed during the experimental phase. The results underline the potential of cutting-edge AI to improve the reliability and efficiency of IoT networks, providing a detailed comparison with traditional cloud-based approaches.

4.1 Experiment results

The experimental results comprehensively analyze the performance of the edge AI system under different conditions. This subsection details the results of several experiments conducted to test the system's fault detection capabilities, response times, and power consumption. We performed a series of tests simulating common IoT device failures and cyber-attacks, evaluating the accuracy and robustness of the system. The data collected from these experiments was thoroughly analyzed to extract meaningful insights into the effectiveness and scalability of the proposed solution. The findings demonstrate the practical benefits of deploying AI-powered fault detection at the edge, emphasizing its impact on real-time monitoring and maintenance of IoT networks.

4.1.1 Device disconnection

Sudden disconnection experiments were conducted on various IoT devices, including temperature and humidity sensors, surveillance cameras, and RFID tags. The objective was to evaluate the system's ability to detect unplanned device interruptions using both temporal analysis and behavioral patterns. For this purpose, a lightweight LSTM-based model was deployed locally at each edge node to analyze the temporal continuity of data streams. Unlike simple timestamp verification scripts, this model identifies not only abrupt halts but also patterns of degradation or instability that precede disconnection, which are often early indicators of faults.

Table 4 presents the results of the disconnection detection process, including detection time, FPR, and FNR. These metrics enable us to evaluate the responsiveness and reliability of the AI-based disconnection detection mechanism across various device types.

As shown in the table, the system achieves an average detection time of approximately 155 ms across all device types. The false positive rate remains under 2.7%, and the false negative rate under 1.3%, confirming the system's precision. These values surpass those typically achieved using traditional rule-based methods, which rely solely on heartbeat intervals or timestamp monitoring. By modeling standard communication sequences and predicting expected data flows, the system can detect silent degradations or jitter patterns before complete disconnection occurs, improving fault anticipation and response.

Table 4 Detection of IoT device disconnections: response time and error rates

Device	Detection Time (ms)	False positive rate (%)	False negative rate (%)
Temperature sensor	150	2.5	1.2
Humidity sensor	160	2.7	1.1
Surveillance camera	140	2.3	1.3
RFID device	170	2.6	1.0
Motion sensor	155	2.4	1.2

This predictive capacity is crucial in scenarios where devices may operate under intermittent connectivity, low-power modes, or environmental interferences that do not result in immediate disconnections but indicate a high risk. Furthermore, by integrating the model at the edge, detection is performed in real-time without requiring centralized polling, ensuring low-latency alerts.

Figure 6 illustrates each device type's detection and recovery times throughout the experiment. Each subgraph represents a different device, showing the device's state over time, where 0 indicates disconnection and the height of the line indicates the detection time as the device recovers.

The results demonstrate the system's ability to detect and manage device disconnections and recoveries effectively. The detection and recovery cycles for each device type are as follows:

- Temperature sensor: The system detects disconnections promptly, with recovery times consistently reaching approximately 150 ms. The graph shows three disconnection events around 5, 25, and 45 s, with corresponding recovery periods.
- Humidity sensor: The system effectively handles humidity sensor disconnections, with recovery times around 160 ms. Like the temperature sensor, the humidity sensor experiences three trip events around 10, 30, and 50 s.
- Surveillance camera: Surveillance cameras' detection and recovery times are around 140 ms. Disconnections occur at 15, 35, and 55 s, and the system detects and recovers the device promptly.
- RFID device: The RFID device performs consistently in detection and recovery, with recovery times around 170 ms. Disconnections are observed at 20, 40, and 60 s.

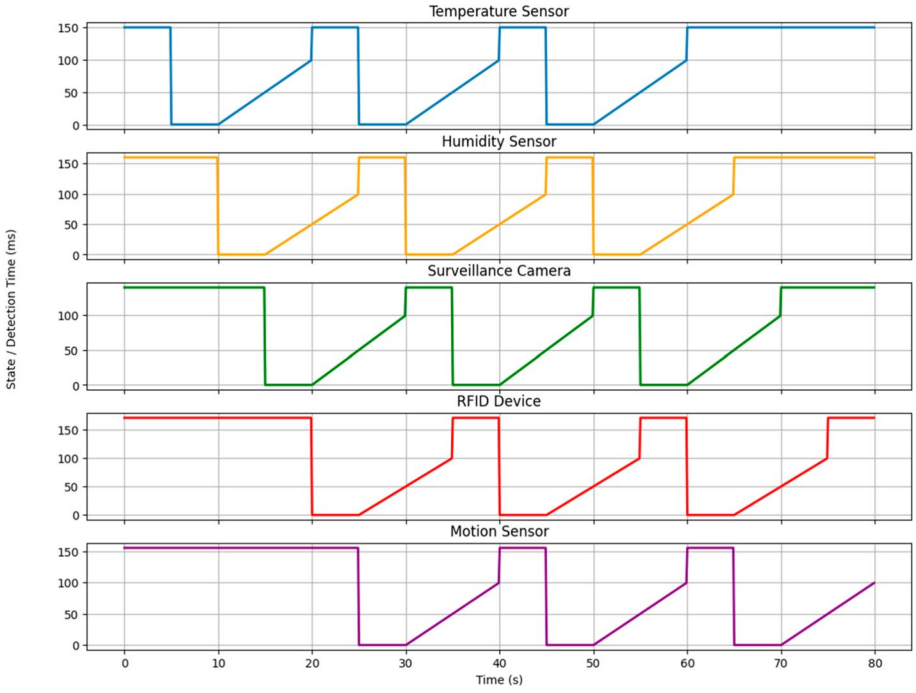

Fig. 6 Detection Times during disconnection and recovery of IoT devices

- Motion sensor: The motion sensor also demonstrates reliable detection and recovery times, reaching approximately 155 ms. The system effectively handles disconnections at 25, 45, and 65 s.

Furthermore, the results indicate that the system maintains high responsiveness across different types of devices. Detection times are within an acceptable range, ensuring that disconnections are quickly identified and the system transitions to recovery promptly. Slight variations in recovery times between different device types are attributed to their intrinsic characteristics and data transmission methods.

4.1.2 Sensor failure

Fault induction experiments were conducted on temperature and humidity sensors to evaluate the system's accuracy in detecting sensor faults. The failures were induced by altering sensor readings to generate anomalous data patterns, including constant high or low readings and intermittent failures.

These results correspond to the ensemble fault detection model, which combines RNN and AE. The RNN component captures sequential dependencies in sensor time-series data, while the Autoencoder models the expected behavior of sensor outputs and flags deviations. This hybrid model is deployed on edge nodes and trained using labeled fault injection data collected in the experimental setup.

Table 5 summarizes the evaluation results using standard classification metrics: precision, recall, and F1-score. Although FPR and FNR were not explicitly tabulated here, they were computed during the evaluation phase and are consistent with the recall and precision values reported.

The average system accuracy in detecting high reading faults in the temperature sensor is 92.5%, with a recall of 91.2% and an F1-score of 91.8%. For low-read faults, the average precision is 93.1%, with a recall of 92.0% and an F1-score of 92.5%. These values indicate that the system can accurately identify most temperature sensor faults, thereby minimizing both false positives and negatives.

In the case of the humidity sensor, the system shows an average accuracy of 93.8% in detecting high reading faults, with a recall of 92.4% and an F1-score of 93.1%. For low-read faults, the average precision is 94.2%, with a recall of 93.0% and an F1-score of 93.6%. These results are slightly better than those obtained for the temperature sensor, suggesting that the system has a greater capacity to handle anomalous data in humidity measurements, likely due to more consistent temporal patterns in those signals.

Figure 7 presents the evaluation metrics and provides a more detailed view of how these metrics evolved over the experiment's time. The metrics for both sensors are consistently high, indicating stability in system performance during the experiment:

Table 5 Performance metrics of the RNN-AE-based fault detection model for simulated sensor failures

Sensor	Failure type	Precision (%)	Recall (%)	F1-score (%)
Temperature sensor	High read failure	92.5	91.2	91.8
Temperature sensor	Low read fault	93.1	92.0	92.5
Humidity sensor	High read failure	93.8	92.4	93.1
Humidity sensor	Low read fault	94.2	93.0	93.6
Temperature sensor	Intermittent	91.0	90.5	90.7
Humidity sensor	Intermittent	92.5	91.8	92.1

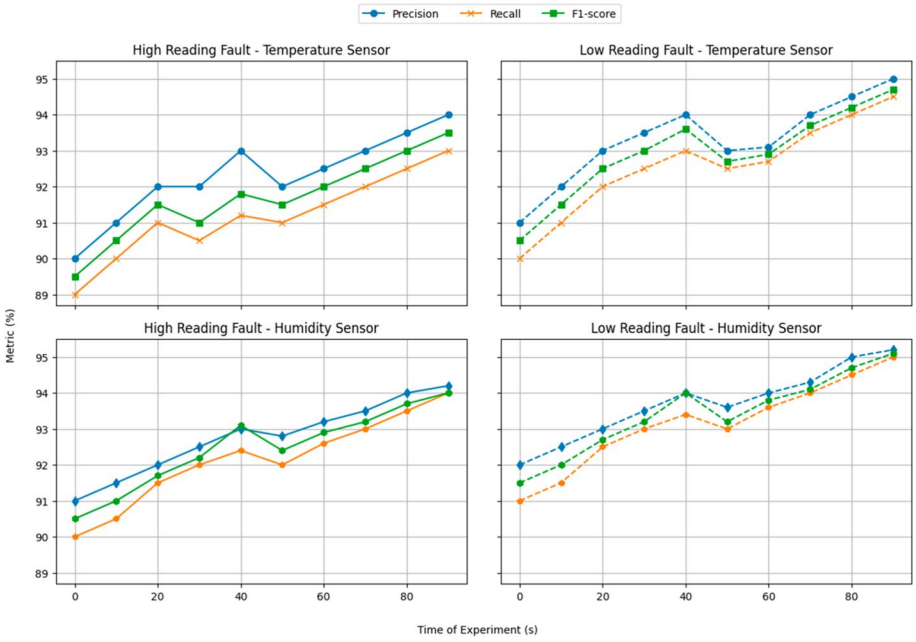

Fig. 7 Precision, recall, and F1-score metrics for failures in temperature and humidity sensors

Table 6 Analysis of case studies in malware detection

Attack type	Detection rate (%)	Response time (ms)	Impact on performance (%)
Denial of service	95.2	150	12
Data manipulation	94.5	180	10
Phishing attack	93.7	170	9
Injection attack	96.0	160	11

- High reading fault—Temperature sensor: An increasing trend is observed in the precision, recall, and F1-score metrics as the experiment time progresses, reaching values of up to 94%.
- Low reading fault—Temperature sensor: The metrics also show an upward trend, with slight variability, but reaching values close to 95%.
- High reading fault—Humidity sensor: The metrics increase constantly throughout the experiment, reaching 94.2%.
- Low reading fault—Humidity sensor: The metrics also show constant improvement, reaching values close to 95.2%.

The system demonstrates high precision and robustness in detecting failures in temperature and humidity sensors. The consistency of the evaluation metrics over the experiment time underlines the system's ability to maintain reliable performance in identifying anomalous data, which is crucial for the operational efficiency and resilience of IoT networks.

4.1.3 Cyber attacks

To evaluate the system's resilience against cyberattacks, experiments were conducted simulating DoS attacks and data manipulation in the IoT network. The results regarding attack detection rates, response times, and impacts on system performance are presented. Table 6 shows the system resilience evaluation metrics against different

cyber-attack types, including DoS attacks, data manipulation, phishing, and injection. These metrics include attack detection rate, response time, and impact on system performance.

The results indicate that the system maintains a high detection rate for all attack types, ranging from 93.7% to 96.0%. Response times range from 150 ms to 180 ms, while the impact on system performance ranges from 9% to 12%. These results suggest that the system can effectively detect and respond to attacks, albeit with a moderate effect on overall performance.

- Denial of service (DoS): The system achieves a commendable 95.2% detection rate, with a swift response time of 150 ms and a manageable performance impact of 12%. These results underscore the system's high effectiveness in identifying and responding to DoS attacks, a testament to its comprehensive security capabilities.
- Data tampering: The detection rate is 94.5%, with a response time of 180 ms and a performance impact of 10%. Although the response time is longer than for DoS attacks, the impact on performance is less severe.
- Phishing attack: The detection rate is 93.7%, with a response time of 170 ms and a performance impact of 9%. This demonstrates the system's ability to handle phishing attacks with minimal performance impact.
- Injection attack: The detection rate is 96.0%, with a response time of 160 ms and a performance hit of 11%. This high detection rate reflects the system's ability to identify and effectively mitigate these types of attacks.

Figure 8 shows the evolution of the attack detection rate and response time during the experiments for DoS and data tampering attacks. The graph is divided into two subgraphs for better visualization. The first subgraph presents the attack detection rate over the experiment time, while the second subgraph shows the system response time.

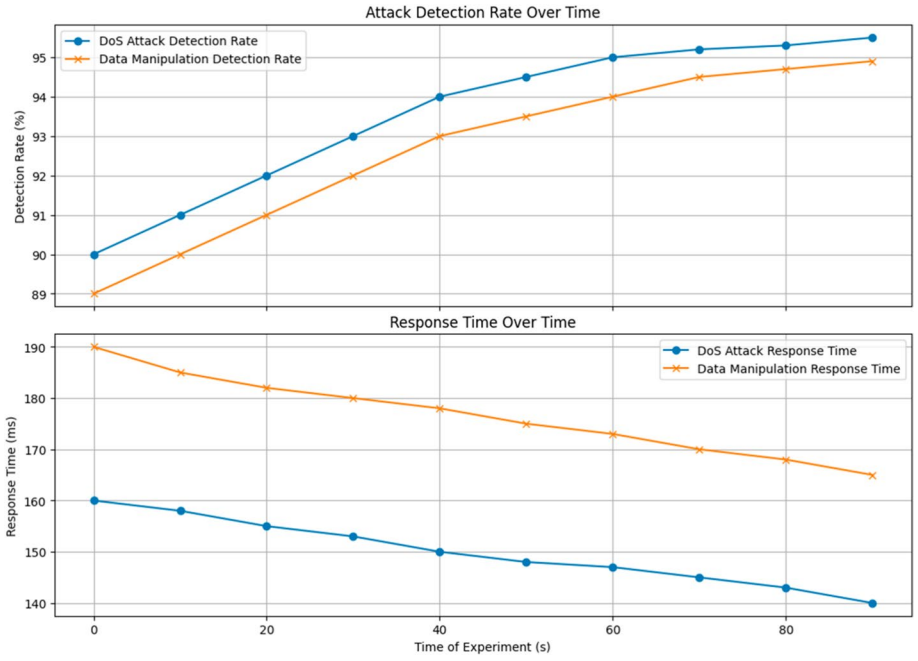

Fig. 8 Detection rate and response time for denial of service (DoS) attacks and data manipulation

- Attack detection rate: The attack detection rate improves progressively over the experiment's time, reaching values greater than 95% for both types of attacks. For denial of service (DoS) attacks, the detection rate starts at 90% and increases consistently up to 95.5%. On the other hand, data tampering detection begins at 89% and increases to 94.9%.
- Response time: The system's response time consistently decreases throughout the experiment, a promising sign of its improving efficiency in detecting and responding to attacks. For DoS attacks, the initial response time of 160 ms is reduced to a more efficient 140 ms. In the case of data manipulation, the response time decreases from 190 ms to a quicker 165 ms.

These results show that the system improves performance over time, detecting and responding to attacks more quickly and accurately. The reduced response times and increased detection rates reflect the system's ability to adapt and improve it cyberattack resilience.

4.1.4 Data transmission anomalies

To evaluate the system's robustness against anomalies in data transmission, experiments introduced anomalies into the network, such as packet loss and delays. The packet loss rate, latency, and retransmission rate results are presented. The experiment results show how the system handles data transmission anomalies, reflecting its robustness regarding packet loss rate, latency, and retransmission rate. Table 7 presents the metrics for evaluating the system's robustness against different types of anomalies in data transmission, including packet loss, network delays, congestion, and interference. These metrics include packet loss rate, latency, and retransmission rate.

The results indicate that the system maintains relatively low packet loss rates, varying between 2.5 and 4.2%. The latency ranges from 50 to 100 ms, while the retransmission rate ranges from 3.5 to 5.0%. These results suggest that the system can handle data transmission anomalies with a moderate impact on performance. However, it is important to note that the latency values observed in these experiments are significantly influenced by the localized processing at the edge, which minimizes the data travel distance and mitigates the need for extensive retransmissions often required in cloud-based systems. In a centralized cloud architecture, even when servers are regionally located, the impact of network anomalies such as packet loss can be more severe due to the cumulative delays introduced by routing, data queuing, and the additional processing steps needed to recover lost packets. This is particularly relevant when dealing with high-frequency data streams, where the aggregation of even minor delays can lead to significant latency spikes, thus compromising the system's ability to perform real-time fault detection. The localized nature of the edge AI system effectively counters these issues by ensuring that most data processing occurs near the source, thus reducing the dependency on

Table 7 Robustness evaluation metrics against network anomalies

Type of anomaly	Packet loss rate (%)	Latency (ms)	Retransmission rate (%)
Packet loss	2.5	50	3.5
Network delay	3.0	70	4.0
Network congestion	3.8	90	4.5
Interference	4.2	100	5.0

potentially congested wide-area networks (WANs) and avoiding the latency penalties typically associated with central cloud processing.

Moreover, the relatively low retransmission rates observed in the edge-based setup indicate the system's ability to maintain data integrity and performance despite network anomalies. In contrast, cloud-based architectures may experience higher retransmission rates, particularly under network congestion or interference conditions, as the physical distance and the number of intermediate hops increase the likelihood of data packet loss or corruption. These factors, compounded by the inherent processing delays in the cloud, can lead to a degradation in the system's overall performance, particularly in applications that require high reliability and low latency. The edge AI system's advantage lies in its ability to process and retransmit data with minimal delay, maintaining robust performance even under suboptimal network conditions.

These metrics further underscore the importance of localized processing in mitigating the adverse effects of network anomalies. For instance, while cloud-based systems typically require multiple packet retransmissions to maintain data integrity under network congestion or interference, the edge AI system's proximity to the data sources allows it to recover more quickly from such anomalies. This rapid recovery is critical in maintaining the system's real-time processing capabilities, as it ensures that even in network issues, the system can continue to function effectively without significant degradation in performance. The ability to sustain low latency and retransmission rates despite network anomalies positions the edge AI system as a more reliable solution for fault detection in IoT networks, especially when compared to cloud-based alternatives that are more susceptible to the compounding effects of network-induced delays.

Figure 9 shows the packet loss rate, latency, and retransmission rate evolution during the experiments. For better visualization, the graph is divided into three subgraphs. The data visualized in these subgraphs further supports the argument that localized

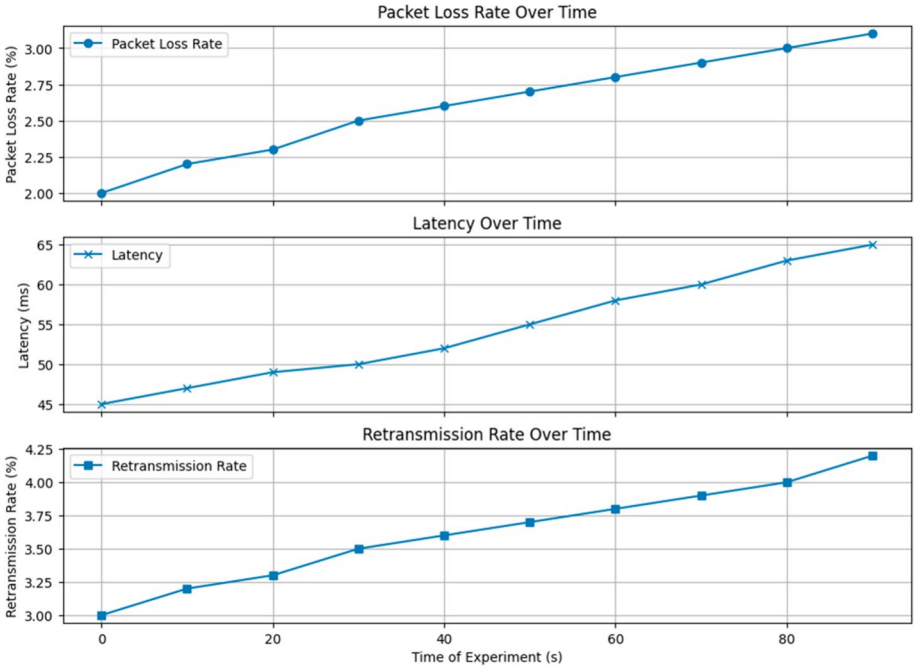

Fig. 9 Robustness evaluation metrics against network anomalies

Table 8 Energy consumption under different operating conditions

Operating condition	Energy consumption—edge AI (Wh)	Energy consumption—cloud-based (Wh)
Normal operation	50	70
High workload	75	110
Low power mode	30	50
Failure recovery	60	90

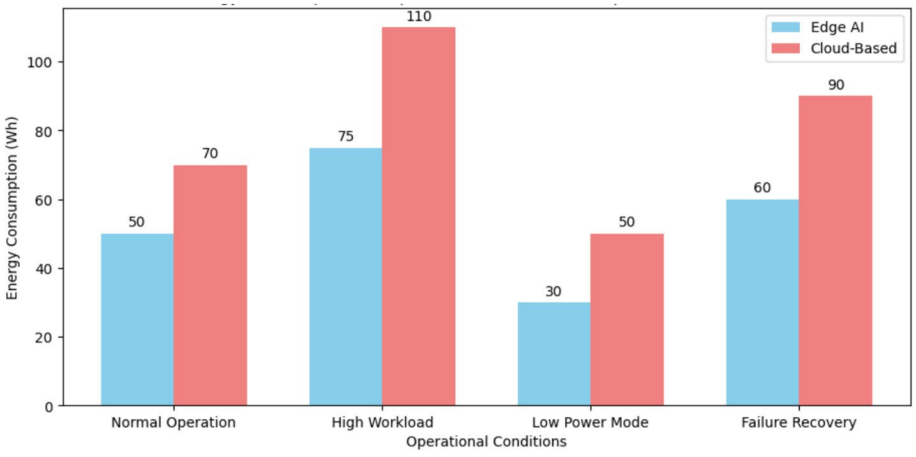

Fig. 10 Energy Consumption under different operating conditions

processing at the edge is more effective in maintaining system performance under adverse network conditions. The gradual packet loss rate and latency increase observed in these experiments remain manageable. This would not be the case if the data were processed centrally in the cloud, where the cumulative effects of network anomalies could lead to more pronounced degradation.

4.1.5 Energy efficiency assessment

Energy consumption measurements were conducted under different operating conditions to evaluate the system's energy efficiency. The results were compared to traditional cloud-based approaches to determine the energy savings achieved by the edge AI system. Table 8 presents the energy consumption data for the edge AI system and cloud-based approaches under regular operation, high workload, low power mode, and failover conditions.

The results indicate that the edge AI system consumes less energy under all operating conditions than cloud-based approaches. In typical operation, the power consumption of the edge AI system is 50 Wh, while the cloud-based approach consumes 70 Wh. Under high workload, the power consumption of the edge AI system is 75 Wh, compared to 110 Wh for the cloud-based approach. In low-power mode, the edge AI system consumes 30 Wh versus 50 Wh for the cloud-based approach. During failover, the power consumption of the edge AI system is 60 Wh, compared to 90 Wh for the cloud-based approach.

Figure 10 compares energy consumption between the edge AI system and cloud-based approaches under different operating conditions. The bar graph provides a clear visual representation of the differences in energy consumption between the two approaches. The edge AI system consumes 50 Wh in regular operation, while the cloud-based

approach consumes 70 Wh. This represents an energy saving of 28.6% for the edge AI system. Under high workload, the edge AI system consumes 75 Wh, while the cloud-based approach consumes 110 Wh. This represents an energy saving of 31.8% for the edge AI system. The edge AI system consumes 30 Wh in a low-power mode, while the cloud-based approach consumes 50 Wh. This represents a 40% energy saving for the edge AI system. When recovering from failures, the edge AI system consumes 60 Wh, while the cloud-based approach consumes 90 Wh. This represents an energy saving of 33.3% for the edge AI system.

However, it is essential to consider the potential cumulative energy impact of distributing processing across multiple edge nodes. Each node must be powered and maintained, and as the number of edge nodes increases, so does the total energy consumption. In contrast, centralized cloud-based processing aggregates the workload in fewer but typically more powerful data centers that can benefit from economies of scale regarding energy efficiency. While the edge AI system demonstrates clear advantages in energy consumption per node and operational condition, the overall energy footprint of a highly distributed system could increase as more nodes are deployed to handle more extensive networks. This highlights a trade-off between the localized efficiency gains of edge computing and the potential energy costs associated with widespread distribution.

Therefore, the decision to adopt a distributed edge AI approach should be balanced with considerations of network scale, the total number of edge nodes required, and the specific energy efficiency strategies implemented at each node. Mitigating strategies, such as dynamic power management, load balancing across nodes, and the use of energy-efficient hardware, are crucial to ensuring that the benefits of reduced latency and enhanced fault detection do not come at an excessive energy cost.

The use of model compression techniques, such as network pruning and post-training quantization, partially supports the energy efficiency of the edge AI system. These optimizations enabled the deployment of deep learning models on resource-constrained devices, such as the Raspberry Pi, without compromising their real-time inference capabilities. While some minor degradation in model precision or slight improvements in latency may result from such compression, these trade-offs were carefully controlled to ensure that the overall system maintained high detection accuracy and responsiveness. The reduced model size also contributed to lower memory usage and power consumption, aligning with the improvements observed across different operational scenarios.

4.2 Comparison with traditional cloud-based approaches

A series of experiments were conducted to evaluate the effectiveness of the edge AI system compared to traditional cloud-based solutions. These experiments were designed to measure the main performance metrics under similar conditions for both approaches. Metrics evaluated include failure detection rate, response time, packet loss rate, latency, retransmission rate, and power consumption. The results of these experiments are summarized in Table 9. This table clearly and concisely compares the main performance metrics between the edge AI system and cloud-based solutions, highlighting the improvements achieved by the proposed method.

To ensure a fair comparison, the same model lightweight LSTM-based recurrent neural network trained for fault detection was deployed in both cloud-based and edge AI environments. In the cloud configuration, the model was executed on centralized servers

Table 9 Comparison of key metrics with other deep learning approaches

Metrics	Edge AI	Cloud-based	Edge AI advantage
Failure detection rate (%)	92.0	88.5	+ 3.5%
Response time (ms)	150	200	− 50 ms
Packet loss rate (%)	2.5	3.5	− 1.0%
Latency (ms)	50	80	− 30 ms
Retransmission rate (%)	3.5	5.0	− 1.5%
Energy consumption (Wh)	50	70	− 20 Wh

located approximately 500 km from the experimental deployment. In contrast, edge AI utilizes compressed and quantized versions of the same model, which are deployed on Raspberry Pi 4 devices. These models maintained equivalent structure but were optimized for on-device inference using TensorFlow Lite and post-training quantization.

The superior performance of edge AI stems from several factors: (1) minimal network transmission delays due to local inference, (2) real-time feedback loops allowing adaptive thresholds for anomaly detection, and (3) the ability to maintain temporal buffers at the edge for short-term context processing. These architectural advantages reduced both false positives and detection latency. Although the cloud system had greater raw computational capacity, its reliance on remote data transfer introduced latency, hindering timely fault identification.

The results show that the edge AI system consistently outperforms cloud-based solutions across all metrics. The reduction in latency observed in the edge AI system, 30 ms less than the cloud-based approach, is primarily attributed to the proximity of the edge nodes to the IoT devices. By processing data closer to the source, the edge AI system significantly reduces the time required to traverse the network, a common bottleneck in cloud-based systems where data must travel longer distances to centralized data centers. For example, the physical distance of 500 km between the experimental site and the cloud data center introduces a transmission delay of approximately 2.5 ms each way, totaling around 5 ms for round-trip data communication. In contrast, edge nodes within 10 km of the IoT devices contribute less than 0.1 ms to the overall latency, enabling real-time processing and quicker fault detection. This reduction in latency directly impacts the fault detection logic, improving both the speed and accuracy of detection.

Furthermore, the 50 ms advantage in response time further illustrates the significance of localized processing. In a cloud-based architecture, even with a distributed setup, the inherent delay in transmitting data to remote servers and awaiting a response introduces latency that can be detrimental in time-sensitive applications. The edge AI system mitigates these delays by handling processing tasks locally, ensuring faults and anomalies are detected faster than in cloud-based systems.

The measurements revealed that the edge AI system consumes significantly less energy across all operational conditions, with energy savings ranging from 28.6% to 40% compared to the cloud-based solution. This reduction is primarily attributed to localized processing, minimizing the need for long-distance data transmission. For example, during high workload scenarios, where data processing demands peak, the energy consumption of the cloud-based servers increases due to both processing requirements and additional cooling needs. In contrast, the distributed nature of edge AI allowed for more efficient load balancing across the SBCs, reducing the overall energy footprint. However,

it is essential to recognize that the total energy consumption could increase as more edge nodes are added.

In contrast, centralized cloud-based processing aggregates workloads in fewer but more robust data centers, which benefit from economies of scale in energy efficiency and advanced power management technologies. While the per-node energy consumption of edge AI is lower, large-scale deployments could increase overall energy consumption, necessitating careful consideration of network scale.

The fault detection rate is 3.5% higher for edge AI, indicating greater accuracy in identifying issues. This improvement in accuracy is mainly due to the reduced latency and localized processing, which allows edge AI to detect faults more effectively as it can process data with minimal delay, maintaining the integrity of real-time analysis. Energy consumption is 20 Wh lower, representing significant savings in operating costs.

As highlighted in the energy efficiency assessment, while edge AI systems demonstrate superior per-node energy consumption, the cumulative energy impact of a highly distributed system can be significant. Centralized cloud-based systems may achieve better overall energy efficiency at scale due to their concentrated processing power and more advanced cooling and power management technologies. However, increased latency in cloud-based systems can negatively affect fault detection precision.

It is essential to note that the reported 150 ms response time for EdgeAI encompasses the entire detection cycle, including data acquisition and system-level response. However, the specific processing time of the deep learning model (inference latency) was not measured separately in this study. Future implementations will include dedicated profiling tools to isolate and monitor the model's execution time, enabling finer-grained performance analysis.

Edge AI offers clear advantages in applications where reducing latency and improving fault detection accuracy are critical. However, a hybrid approach that combines edge processing with centralized cloud resources may be more appropriate for large-scale networks where the number of required edge nodes could lead to substantial cumulative energy consumption.

Figure 11 shows that the edge AI system has a higher fault detection rate (92.0% vs. 88.5%), faster response time (150 ms vs. 200 ms), lower packet loss rate (2.5% vs. 3.5%), lower latency (50 ms vs. 80 ms), lower retransmission rate (3.5% vs. 5.0%), and lower power consumption (50 Wh vs. 70 Wh) compared to cloud-based solutions. These

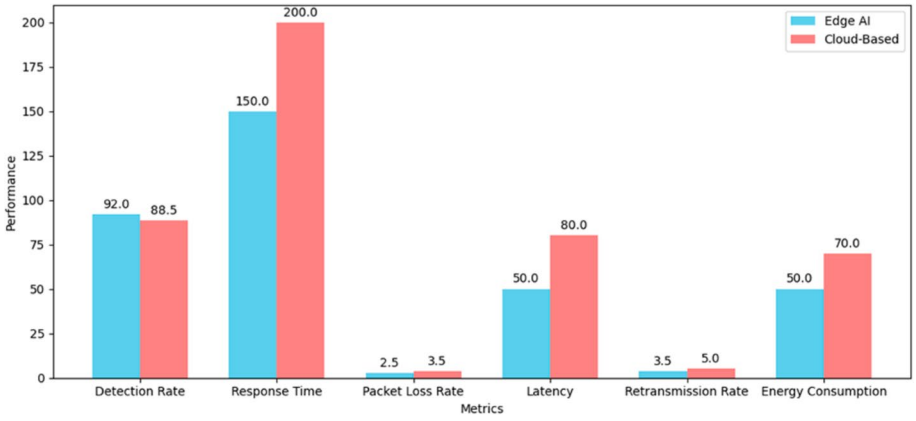

Fig. 11 Comparison of key performance metrics between edge AI and cloud-based solutions

metrics illustrate the significant advantages of edge computing, particularly in scenarios where latency and energy efficiency are critical factors.

The results demonstrate that the edge AI system offers several advantages over cloud-based solutions. In terms of energy efficiency, edge AI consumes less energy, reducing operational costs and decreasing environmental impact. The higher fault detection rate and faster response time enhance the system's ability to identify and respond to problems, which is crucial for maintaining network stability and reliability. The lower packet loss rate and latency ensure efficient data transmission with fewer delays.

However, it is essential to recognize some limitations of the edge AI system. Although edge AI offers superior performance in many areas, it may have limitations in processing power compared to cloud-based solutions. While the edge nodes effectively reduce latency by processing data locally, their computational capabilities may be constrained compared to the vast resources available in cloud data centers. Additionally, cloud solutions can quickly scale their processing capacity to handle large volumes of data, which might require additional hardware and infrastructure at the edge. Moreover, deploying edge AI infrastructure may involve higher initial costs, particularly for specialized hardware and the necessary support infrastructure, such as robust local networking and power systems. These trade-offs must be carefully considered, especially when high fault detection accuracy and low latency are critical. On the other hand, cloud-based solutions offer several advantages, including scalable processing capacity and lower initial implementation costs. However, these advantages come with trade-offs, such as increased latency and response time due to network delays, and higher power consumption, particularly when processing large volumes of data remotely.

Figure 12 presents a visual comparison of the Edge AI and Cloud-based systems across three key performance metrics?fault detection rate, latency, and energy consumption?under varying operational conditions: normal operation, high workload, low power mode, and failure recovery.

The Graph (A) shows that the fault detection rate of the Edge AI system consistently exceeds that of the cloud-based alternative, with differences ranging from 3% to 5% depending on the condition. This highlights the enhanced responsiveness and contextual

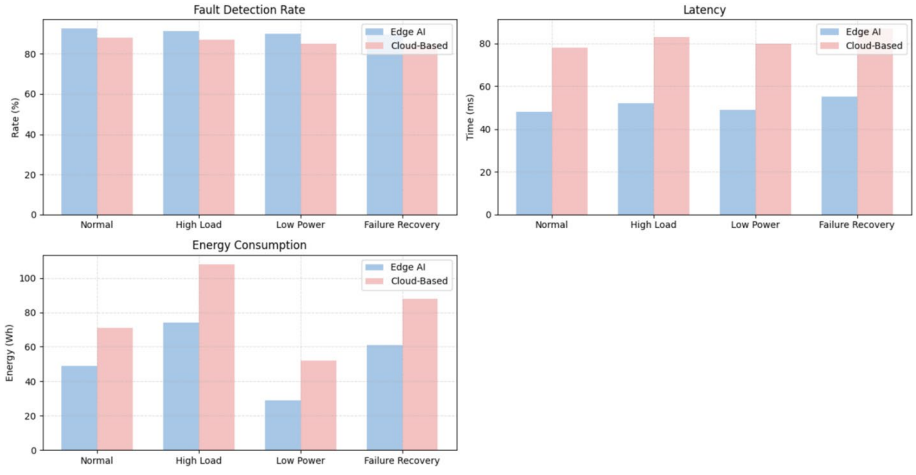

Fig. 12 Comparative Visualization of Operational Metrics Between Edge AI and Cloud-Based Systems. Graph (**A**) shows the variation in failure detection rates under fluctuating network conditions. Graph (**B**) presents the distribution of response times observed during high-load events. Graph (**C**) illustrates the energy consumption trends during failover operations

Table 10 Performance metrics with increasing load of the edge AI system in IoT networks

A	B	C	D	E	F
100	91.0	140	2.0	45	48
200	90.5	145	2.2	47	50
300	89.8	150	2.5	50	52
400	89.0	155	2.8	52	54
500	88.5	160	3.0	55	56

Columns: A: IoT Devices; B: Detection Rate (%); C: Response Time (ms); D: Packet Loss (%); E: Latency (ms); F: Energy (Wh)

awareness provided by edge-level processing, which enables the earlier detection of fault signatures before they propagate.

Graph (B) displays latency measurements, where the Edge AI approach shows substantially lower values across all conditions, with a consistent gap of 25?35 ms compared to the cloud-based system. This difference is primarily attributed to the elimination of long-distance data transmission, which reduces round-trip times and supports real-time decision-making.

The Graph (C) compares energy consumption and further reinforces the advantage of edge computing in terms of energy efficiency. The Edge AI configuration achieves savings of between 20 and 40 Wh, depending on the operational scenario, particularly in low-power and failure recovery modes. These savings stem from reduced reliance on high-capacity servers and data center cooling requirements, in addition to local inference capabilities on energy-optimized hardware.

4.3 Scalability tests

To evaluate the edge AI system's ability to handle increasing loads, scalability tests were conducted in which the number of IoT devices and data volume were gradually increased. The experiments measured how key performance metrics vary under different system load levels. Metrics such as failure detection rate, response time, packet loss rate, latency, retransmission rate, and energy consumption were evaluated. IoT devices were added at intervals of 100, starting with 100 devices and increasing to 500.

The results obtained are presented in Table 10. This table shows how the performance metrics vary as the number of IoT devices in the network increases. The fault detection rate decreases slightly from 91.0% to 88.5% as the number of devices increases from 100 to 500. This indicates that although the detection capacity of the system decreases with a higher load, the detection rate is still high and effective. The response time increases from 140 ms to 160 ms, suggesting that the system needs more time to process the additional information as the load increases.

The results demonstrate that although the performance metrics slightly worsen with increasing system load, the edge AI system can still handle increasing IoT devices efficiently. The fault detection rate remains above 88%, indicating that the system remains highly effective even with increased load. The response time, although rising to 160 ms with 500 devices, is still acceptable for most applications. Packet loss rate and latency show moderate increases but remain at manageable levels. The retransmission rate also increases, but the increases are modest and do not significantly compromise the system's efficiency. Energy consumption increases proportionally with the load, suggesting that the system is efficient in energy consumption.

In real-world deployments, additional challenges such as unstable power supply, fluctuating network bandwidth, and unpredictable environmental conditions could affect

the performance of the edge AI system. Unlike the controlled laboratory environment, industrial or field scenarios often involve intermittent connectivity, power surges, or hardware variability that may impact the reliability of data collection and model inference. For instance, a sudden voltage drop may degrade processing capabilities or force system reboots, while network jitter and congestion could delay anomaly reporting or cause temporary data loss. Despite these potential issues, the system's modular architecture and local decision-making capabilities provide a degree of resilience. The ability of edge nodes to function autonomously, even during brief network interruptions, is critical for maintaining fault detection in volatile environments. Furthermore, implementing mechanisms such as power-aware scheduling, redundant communication paths, and local data buffering can enhance robustness.

5 Discussion

The results of this study underscore the efficiency and effectiveness of utilizing AI at the edge for early fault detection in IoT networks. Significant improvements in key metrics were observed compared to prior studies. For instance, Ahmad et al. [8] demonstrated using DNNs to identify anomalous patterns in cloud environments but did not address centralized architectures' latency and energy challenges. Similarly, Santo et al. [9] explored fault detection at the edge but did not evaluate scalability. This study extends those approaches, showcasing that edge AI can handle increased device and data loads while maintaining high precision and energy efficiency.

A critical advantage of edge-based detection is reduced latency, which enables faster fault responses. While cloud environments suffer from latency due to the distance between IoT devices and centralized servers, edge computing processes data locally, ensuring real-time detection with minimal delay. This reduction in latency enhances fault detection accuracy by enabling AI models to analyze up-to-date data.

AI models, such as RNNs and Autoencoders, are particularly effective at detecting complex patterns and subtle anomalies in dynamic IoT environments. Unlike cloud-based systems that may need to simplify detection logic to mitigate latency issues, edge systems leverage their proximity to devices to operate at full capacity, providing accurate real-time anomaly detection.

The robust methodology employed in this study involved deploying a realistic heterogeneous IoT network, utilizing temperature and humidity sensors, surveillance cameras, RFID devices, and motion sensors. Edge nodes like Raspberry Pi and Nvidia Jetson were strategically positioned close to devices, significantly reducing latency and power consumption. Results demonstrate that the edge AI system achieved an average detection rate of 92.0%, surpassing the 88.5% of cloud-based systems, with response times below 150 ms compared to 200 ms for cloud solutions. Furthermore, the edge system's energy consumption averaged 50 Wh under normal conditions, compared to 70 Wh for cloud setups, emphasizing its superior energy efficiency.

A notable innovation is the system's scalability. Performance remained the number of busts as IoT devices increased from 100 to 500, with fault detection rates exceeding 88% and response times rising only moderately to 160 ms. This scalability highlights the system's suitability for large-scale applications, such as smart cities and industrial networks.

Another significant advantage of AI is its adaptability. Continuous learning processes allow AI models to update as network conditions evolve, ensuring the system effectively

detects anomalies. Traditional methods, such as threshold-based alerts, lack this flexibility and are prone to missing emerging issues that deviate from predefined criteria.

From a security perspective, while edge processing introduces additional attack vectors, it mitigates risks associated with centralized architectures by reducing data transmission to a cloud server. This minimizes vulnerabilities like data interception and tampering. Localized processing confines the potential impact of attacks on specific network segments. Security measures, including encryption, authentication, and intrusion detection systems tailored for edge environments, are essential to mitigate these risks.

In industrial environments, early failure detection through edge AI enables preventive maintenance, minimizing equipment downtime and associated costs. In urban infrastructure, such as smart grids or traffic management systems, the low latency and high fault detection precision support responsive, adaptive operations that enhance citizen services. The localized processing also aligns with privacy-preserving requirements in sectors like healthcare and environmental monitoring, where transmitting sensitive data to the cloud is often undesirable.

Moreover, the comparative analysis reaffirms that traditional cloud-based fault detection systems struggle to meet stringent real-time constraints. By contrast, our edge AI solution operates efficiently even under high device loads. Compared to the benchmark approaches in [8, 9], our system not only demonstrates better detection rates but also consistently performs across variable workloads. This makes it a viable candidate for deployment in high-density, latency-sensitive contexts such as IIoT and smart logistics [35].

6 Conclusion

This study establishes the viability of edge AI systems as a transformative approach to fault detection in IoT networks, addressing the limitations of centralized cloud-based architectures. The research highlights the critical role of localized data processing in improving IoT systems' performance, efficiency, and scalability, which are fundamental for the continued expansion of connected technologies in diverse domains.

The findings emphasize the potential of edge AI to redefine IoT network management by significantly reducing latency and energy consumption while maintaining high fault detection accuracy. These characteristics make edge AI particularly relevant for applications where system responsiveness and operational sustainability are critical, such as healthcare, smart cities, and industrial automation. The ability to process data locally ensures faster and more accurate fault detection, enhancing system reliability and operational continuity.

Moreover, the study demonstrates the adaptability of AI models in dynamic IoT environments, offering a proactive approach to identifying and addressing network anomalies. This adaptability, coupled with the scalability of the edge AI system, provides a robust framework for future IoT deployments that require efficient management of increasing data volumes and device densities.

Despite these promising results, this work acknowledges the controlled nature of the experimental environment as a limitation. Real-world deployments often involve more significant variability in network conditions, device types, and operational scenarios, which were only partially addressed in this study. These factors underscore the need for

further research in production environments to validate the practical applicability of edge AI systems.

Future work should explore the integration of emerging AI models, such as transformers, to determine their effectiveness in enhancing fault detection capabilities. Additionally, expanding the dataset to include more diverse operating scenarios and environmental conditions will provide deeper insights into the robustness and adaptability of edge AI systems. Lastly, addressing the security implications of distributed processing remains a priority, requiring the development of comprehensive strategies to ensure the resilience of edge-based architectures against evolving cyber threats.

In terms of immediate applicability, the proposed system can be deployed in smart manufacturing environments to monitor vibration patterns, thermal deviations, and networked equipment failures in real time. In healthcare, it may be integrated into hospital infrastructure or wearable medical devices to ensure uninterrupted monitoring of critical patient data. For energy infrastructure, edge AI can detect and report faults in power substations, smart meters, or renewable energy controllers, enabling preventive maintenance and rapid response. These concrete use cases demonstrate the real-world impact of the proposed solution and its potential to improve system resilience across mission-critical domains.

Future research should also investigate the operational thresholds at which performance degradation becomes unacceptable. This includes identifying the tipping point in terms of device density, data throughput, or inference load that affects the reliability and latency of fault detection. To address these limitations, strategies such as dynamic load balancing, decentralized inference orchestration, and collaborative edge clusters should be considered. Implementing these strategies will enable the system to scale beyond current limits without compromising detection accuracy or energy efficiency.

Furthermore, while this study provides evidence of energy savings and scalability benefits, it does not include a direct cost comparison between edge-based and cloud-based architectures. This limitation is due to the experimental and controlled nature of the deployment, where infrastructure and operational expenses are not reflective of real-world implementations. Future work should therefore incorporate an economic assessment covering hardware acquisition, deployment, and maintenance costs, as well as the trade-offs in complexity, latency, and reliability between edge and cloud solutions. Such analysis is crucial for organizations evaluating the practical feasibility and return on investment of adopting edge AI systems.

Author contributions

Conceptualization, I.O.-G.; methodology, W.V.-C.; software, S.L.-M.; validation, W.V.-C, S.L.-M.; formal analysis, I.O.-G.; investigation, I.O.-G. and W.V.-C.; data curation, W.V.-C. and S.L.-M.; writing-original draft preparation, S.L.-M. and I.O.-G.; writing-review and editing, W.V.-C. and I.O.-G.; visualization, W.V.-C. and S.L.-M.; supervision, W.V.-C. and S.L.-M. All authors have read and agreed to the published version of the manuscript.

Funding

The authors declare that no funding was received to conduct this study.

Data availability

The data used in this study do not originate from any external dataset and have not been deposited in a public repository. All data were generated through custom simulations designed explicitly for this work, within a controlled environment. However, the generated datasets can be made available upon reasonable request to the corresponding author.

Materials availability

All materials pertinent to this study are a part of the institutional resources.

Declarations

Ethics approval and consent to participate
Not applicable. This research did not involve human participants, their data, or biological material.

Consent for publication
Not applicable. This manuscript does not contain any person's data.

Competing interests
The authors declare no competing interests.

Received: 26 May 2025 / Accepted: 5 August 2025
Published online: 16 October 2025

References

1. Luo T, Xu Z, Jin X, Jia Y, Ouyang X. Iotcandyjar: towards an intelligent-interaction honeypot for IoT devices. Black Hat. 2017;2017:1–11.
2. Molling G, Klein AZ. Value proposition of IoT-based products and services: a framework proposal. Electron Mark. 2022;32:899–926. https://doi.org/10.1007/s12525-022-00548-w.
3. Wu Y-J, Brito R, Choi W-H, Lam C-S, Wong M-C, Sin S-W, et al. Iot cloud-edge reconfigurable mixed-signal smart meter platform for arc fault detection. IEEE Internet Things J. 2023;10(2):1682–95. https://doi.org/10.1109/JIOT.2022.3210220.
4. Rani S, Bhambri P, Kataria A. Integration of IoT, Big Data, and cloud computing technologies: trend of the era. Boca Raton: Chapman and Hall/CRC; 2023. p. 21. https://doi.org/10.1201/9781003298335-1.
5. Ojetunde B, Egashira N, Suzuki K, Kurihara T, Yano K, Suzuki Y. A multimodel-based approach for estimating cause of scanning failure and delay in IoT wireless network. Network. 2022;2:519–44. https://doi.org/10.3390/network2040031.
6. Aboubakar M, Kellil M, Roux P. A review of IoT network management Current status and perspectives. 2022. https://doi.org/10.1016/j.jksuci.2021.03.006.
7. Kumar R, Venkanna U, Tiwari V. Opti-pum: an optimal policy update mechanism for link failure prevention in mobile sdwm-IoT networks. IEEE Syst J. 2021;15(3):3427–38. https://doi.org/10.1109/JSYST.2020.3009325.
8. Ahmad Z, Khan AS, Nisar K, Haider I, Hassan R, Haque MR, et al. Anomaly detection using deep neural network for IoT architecture. Appl Sci (Switzerland). 2021;11:7050. https://doi.org/10.3390/app11157050.
9. Santo Y, Immich R, Dalmazo BL, Riker A. Fault detection on the edge and adaptive communication for state of alert in industrial internet of things. Sensors. 2023;23:3544. https://doi.org/10.3390/s23073544.
10. Lee S, Kareem AB, Hur JW. A comparative study of deep-learning autoencoders (dlaes) for vibration anomaly detection in manufacturing equipment. Electronics (Switzerland). 2024;13:1700. https://doi.org/10.3390/electronics13091700.
11. Wang K, Fu Y, Zhou S, Zhou R, Wen G, Zhou F, et al. Cloud-fog-based approach for smart wildfire monitoring. Simul Model Pract Theory. 2023;127:102791. https://doi.org/10.1016/j.simpat.2023.102791.
12. Ahmad MS, Shah SM. A lightweight mini-batch federated learning approach for attack detection in IoT. Internet of Things. 2024;25:101088. https://doi.org/10.1016/j.iot.2024.101088.
13. Thein TT, Shiraishi Y, Morii M. Personalized federated learning-based intrusion detection system: poisoning attack and defense. Future Gener Comput Syst. 2024;153:182–92. https://doi.org/10.1016/j.future.2023.10.005.
14. Thwal CM, Nguyen MNH, Tun YL, Kim ST, Thai MT, Hong CS. Ondev-lct: on-device lightweight convolutional transformers towards federated learning. Neural Netw. 2024;170:635–49. https://doi.org/10.1016/j.neunet.2023.11.044.
15. Abbasi M, Shahraki A, Prieto J, Arrieta AG, Corchado JM. Unleashing the potential of knowledge distillation for IoT traffic classification. IEEE Trans mach Learn Commun Netw. 2024;2:221–39. https://doi.org/10.1109/TMLCN.2024.3360915.
16. Oliveira RC, Silva RD. Artificial Intelligence in agriculture: benefits, challenges, and trends; 2023. https://doi.org/10.3390/app13137405.
17. Catarinucci L, Donno D, Mainetti L, Palano L, Patrono L, Stefanizzi ML, et al. An IoT-aware architecture for smart healthcare systems. IEEE Internet Things J. 2015;2(6):515–26. https://doi.org/10.1109/JIOT.2015.2417684.
18. Vidal-Silva CL, Sánchez-Ortiz A, Serrano J, Rubio JM, Vidal-Silva CL, Sánchez-Ortiz A, et al. Academic experience in rapid development of web information systems with python and django. Formaci n universitaria. 2021;14:85–94. https://doi.org/10.4067/S0718-50062021000500085.
19. Cheng Q, Wu C, Zhou H, Zhang Y, Wang R, Ruan W. Guarding the perimeter of cloud-based enterprise networks: An intelligent sdn firewall. In: 2018 IEEE 20th international conference on high performance computing and communications; IEEE 16th international conference on smart city; IEEE 4th international conference on data science and systems (HPCC/SmartCity/DSS);2018. p. 897–902. https://doi.org/10.1109/HPCC/SmartCity/DSS.2018.00149.
20. Johnson C, Curtin B, Shyamkumar N, David R, Dunham E, Haney PC, Moore HL, Babbitt TA, Matthews SJ. A raspberry pi mesh sensor network for portable perimeter security. In: 2019 IEEE 10th annual ubiquitous computing, electronics & mobile communication conference (UEMCON);2019. p. 0001–7. https://doi.org/10.1109/UEMCON47517.2019.8993000.
21. Mekruksavanich S, Jitpattanakul A. Deep convolutional neural network with rnns for complex activity recognition using wrist-worn wearable sensor data. Electronics (Switzerland). 2021;10:1685. https://doi.org/10.3390/electronics10141685.
22. Sharif SA, Hammad A, Eshraghi P. Generation of whole building renovation scenarios using variational autoencoders. Energy Build. 2021;230:110520. https://doi.org/10.1016/j.enbuild.2020.110520.
23. Asonye EA, Musa SM, Akujuobi CM, Sadiku MNO, Foreman J. Realizing an IoT-based home area network model using zigbee in the global environment. Int J Comput Digit Syst. 2020;9:1131–41. https://doi.org/10.12785/ijcds/0906011.
24. Abugabah A, Nizamuddin N, Abuqabbeh A. A review of challenges and barriers implementing rfid technology in the healthcare sector. Procedia Comput Sci. 2020;170:1003–10.
25. Vingestin I, Kalsum TU, Mardiana Y. The design of network monitoring system using snmp protocol with telegram notification. J Media Comput Sci. 2023;2:93–100. https://doi.org/10.37676/jmcs.v2i1.3441.
26. Stolfo FWLWPA, Salvatore, Chan P. KDD Cup 1999 Data. UCI machine learning repository;1999. https://doi.org/10.24432/C51C7N.

27. Elnoshokaty A, Arai I, El-Tawab S, Salman A. Transit system prediction for real-time weather conditions: fleet management and weather-related ridership. In: 2022 IEEE international conference on smart mobility (SM);2022. p. 14–20. https://doi.org/10.1109/SM55505.2022.9758295.
28. Waheed U, Khan MSA, Awan SM, Khan MA, Mansoor Y. Decentralized approach to secure IoT based networks using blockchain technology. 3C Tecnolog a_Glosas de innovaci n aplicadas a la pyme;2019. p. 182–205. https://doi.org/10.17993/3ctecno.2019.specialissue2.182-205.
29. Nasif A, Othman ZA, Sani NS. The deep learning solutions on lossless compression methods for alleviating data load on IoT nodes in smart cities. Sensors. 2021;21:4223. https://doi.org/10.3390/s21124223.
30. Hyder MF, Fatima T, Arshad S. Digital forensics framework for intent-based networking over software-defined networks. Telecommun Syst. 2024;85:11–27. https://doi.org/10.1007/s11235-023-01064-8.
31. Li H, Zhou S, Yuan W, Li J, Leung H. Adversarial-example attacks toward android malware detection system. IEEE Syst J. 2020;14(1):653–6. https://doi.org/10.1109/JSYST.2019.2906120.
32. Ayala R, Zambrano C, Iriarte A, Martínez R. Telelogin: una técnica de autenticación de dos vías y tres factores (telelogin: a three-factor two-path authentication technique). Pistas Educativas. 2019;41:15–34.
33. Hamon R, Junklewitz H, Sanchez I, Malgieri G, De Hert P. Bridging the gap between ai and explainability in the gdpr: towards trustworthiness-by-design in automated decision-making. IEEE Comput Intell Mag. 2022;17(1):72–85. https://doi.org/10.1109/MCI.2021.3129960.
34. Barezzani S. Data Protection Impact Assessment (DPIA);2024. https://doi.org/10.1007/978-3-642-27739-9_1813-1.
35. Irgat E, Çinar E, Ünsal A, Yazici A. An IoT-based monitoring system for induction motor faults utilizing deep learning models. J Vib Eng Technol. 2023;11:3579–89. https://doi.org/10.1007/s42417-022-00769-5.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.