Available online at www.ijournalse.org

Emerging Science Journal

(ISSN: 2610-9182)

Vol. 9, No. 3, June, 2025

AI-Based Architecture and Distributed Processing for the Detection and Mitigation of Spoofing Attacks in IoT Networks

William Villegas [1]*, Iván Ortiz-Garcés [1], Jaime Govea [1]

[1] *Escuela de Ingeniría en Ciberseguridad, Facultad de Ingenierías y Ciencias Aplicadas, Universidad de Las Américas, Ecuador.*

Abstract

The exponential increase in IoT devices within industrial networks has heightened their exposure to cyberattacks, with spoofing attacks posing one of the most critical threats. These attacks exploit communication vulnerabilities, enabling malicious entities to manipulate network traffic and impersonate legitimate devices, compromising system integrity and security. This study aims to develop an AI-driven detection and mitigation system to enhance IoT network security against spoofing attacks. The proposed approach integrates Convolutional Neural Networks with a distributed processing architecture based on Edge nodes, enabling real-time anomaly detection while reducing computational overhead on central servers. The system was tested in four simulated industrial scenarios involving up to 1,000 IoT devices and multiple concurrent attacks to validate its effectiveness. The evaluation included detection accuracy, response time, and system scalability metrics. Results indicate a detection rate of up to 95% under optimal conditions and 88% in high-density environments. Detection and response times ranged from 150 ms to 220 ms and 300 ms to 450 ms, respectively. Additionally, 97% of compromised devices were successfully isolated, with a false positive rate between 3% and 6%. This study introduces a scalable and adaptive AI-based framework, surpassing traditional machine learning techniques in accuracy, efficiency, and real-time applicability for industrial IoT security.

Keywords:
Security;
Spoofing Detection;
Edge Computing;
Convolutional Neural Networks.

Article History:

Received:	10	February	2025
Revised:	11	May	2025
Accepted:	19	May	2025
Published:	01	June	2025

1- Introduction

The rapid expansion of the Internet of Things (IoT) has significantly transformed industrial processes, smart cities, and home automation by enabling large-scale connectivity, automation, and real-time data processing [1, 2]. The number of IoT devices is projected to grow exponentially in the coming years, increasing the complexity of network management and exposing these infrastructures to severe cybersecurity threats. Among the most critical, spoofing attacks allow adversaries to manipulate network traffic by impersonating legitimate devices, threatening data integrity, system security, and operational reliability [3]. These attacks are particularly problematic in industrial environments, where real-time data exchange is essential for automation and safety.

Several studies have proposed different approaches for detecting and mitigating spoofing attacks in IoT networks. Olawale & Ebadinezhad [3] demonstrated that Intrusion Detection Systems (IDS) using Support Vector Machines (SVM) and Convolutional Neural Networks (CNNs) can detect anomalous behavior in healthcare IoT environments. However, their findings indicate that SVMs struggle with complex network topologies, leading to poor generalization under high-load conditions. Similarly, Prathapchandran & Janani [4] implemented a Random Forest-based trust

* **CONTACT**: william.villegas@udla.edu.ec

DOI: http://dx.doi.org/10.28991/ESJ-2025-09-03-026

mechanism (RFTRUST) to detect sinkhole attacks in IoT environments, demonstrating its effectiveness in structured networks. However, their study highlighted computational latency issues, limiting its applicability in real-time industrial settings. Shukla et al. [5] explored anomaly detection in smart grid networks using a linear SVM combined with a blockchain-based model. Still, the approach faced challenges in high-density environments due to resource constraints and increased processing time.

Recent advances in deep learning have introduced CNN-based anomaly detection techniques that outperform classical machine learning models. Abraham et al. [6]. validated the effectiveness of Random Forests (RFs) for anomaly detection in IoT-based smart homes, emphasizing their robustness in structured datasets. However, their study also identified high computational costs and response time limitations, restricting real-time applicability. Kim [7] proposed an optimization method for IoT device deployment but did not address security concerns related to network resilience against adversarial attacks. Alamatsaz et al. [8]. introduced a hybrid CNN-LSTM model for ECG-based arrhythmia detection, demonstrating the capability of deep learning architectures to capture intricate temporal patterns in streaming data. However, their work did not explore real-time detection in distributed IoT security applications, leaving a gap in the literature for adaptive, scalable security solutions in industrial networks.

Despite these advances, existing solutions rely primarily on centralized architectures, which introduce single points of failure, increased bandwidth consumption, and high latency—factors that hinder real-time attack mitigation. This study proposes an AI-driven detection and mitigation system that integrates CNN models with a distributed Edge computing architecture to address these limitations. Unlike traditional approaches, our system distributes computational workloads across Edge nodes, reducing processing overhead on central servers and improving response times. This decentralized approach enables real-time anomaly detection, rapidly mitigating spoofing attacks without overloading cloud-based infrastructures.

The system is evaluated under four simulated industrial scenarios, ranging from 100 to 1,000 IoT devices, to assess its scalability and effectiveness against ARP spoofing, IP spoofing, and device identity theft attacks. The results demonstrate that the proposed approach achieves up to 95% detection rates in optimal conditions and 88% in high-density environments. Additionally, detection and response times remain within operational limits, ranging from 150 ms to 220 ms and 300 ms to 450 ms, respectively. The architecture enhances resilience against network congestion and mitigates potential single points of failure in industrial IoT environments.

These findings highlight the system's ability to enhance real-time security in industrial IoT networks, providing a scalable, decentralized, and adaptive solution that outperforms traditional machine learning-based methods. Future work will focus on optimizing computational efficiency and extending the framework to detect additional types of cyber threats, such as distributed denial-of-service (DDoS) attacks and advanced persistent threats (APT).

The remainder of this article is structured as follows. Section 2 presents a literature review, discussing existing solutions for anomaly detection in IoT networks and their limitations. Section 3 details the proposed architecture, including integrating CNN models and the distributed processing framework. Section 4 describes this study's experimental setup, datasets, and evaluation metrics. Section 5 presents and analyzes the results obtained, comparing the proposed approach with existing solutions. Finally, Section 6 discusses the implications of the findings, highlights possible improvements, and outlines future research directions.

2- Literature Review

The security of IoT networks has been a growing research area due to the increasing number of connected devices and their susceptibility to cyberattacks, particularly spoofing attacks [9]. Several approaches have been proposed to detect and mitigate these threats, integrating machine learning and deep learning techniques to enhance network protection. However, existing solutions face persistent scalability, resource consumption, and efficiency challenges in high-density environments, limiting their applicability in industrial IoT settings.

One widely explored method for anomaly detection in IoT networks is using SVMs. Long & Jinsong [10] developed a hybrid model combining entropy-based feature selection with SVM-based detection in software-defined networks (SDN), demonstrating its effectiveness in mitigating DDoS attacks. However, their findings indicate that SVM models are susceptible to hyperparameter configurations, leading to performance degradation under dynamic network conditions. Similarly, Elhag et al. [11] applied SVM-based detection models in Industrial IoT environments, but their results revealed a significant drop in accuracy when exposed to large-scale, high-noise datasets. These findings suggest that while SVMs are effective in structured and controlled environments, they fail to maintain robustness in complex, large-scale IoT networks.

Another widely adopted approach is the Random Forest (RF) algorithm, which has gained popularity due to its capability to handle high-dimensional and heterogeneous IoT data. Maghrabi [12] proposed an RF-based network intrusion detection system for IoT applications, highlighting its ability to improve anomaly classification accuracy.

However, the study also revealed that RF-based models suffer from high computational costs and latency, making them less suitable for real-time security applications. In our work, we observed similar trade-offs—while RF models provided high detection rates, their excessive processing time limited their real-time applicability in large-scale networks.

More recent research has focused on deep learning techniques, particularly CNNs, due to their ability to capture complex temporal and spatial patterns in network traffic. Kanwal et al. [13] demonstrated that CNN-based models outperformed traditional machine-learning approaches in detecting cyber threats in IoT environments. However, their study was focused on wildfire prediction, not network security, leaving a gap in research regarding CNN-based anomaly detection for IoT networks under real-world adversarial conditions. Our work addresses this gap by implementing CNNs optimized explicitly for IoT security, ensuring real-time detection of spoofing attacks with a scalable deployment strategy.

Another critical area of research involves distributed processing for IoT security, which seeks to reduce the computational burden on central servers by leveraging Edge computing. Bukhsh et al. [14] proposed a latency-aware task management method in decentralized Edge environments, demonstrating that distributed processing significantly enhances system scalability and fault tolerance. However, their approach primarily focused on task scheduling and load balancing without integrating advanced security mechanisms for threat detection and mitigation. Our study extends this concept by incorporating CNN-based anomaly detection directly into Edge nodes, enabling a real-time, decentralized security framework that efficiently identifies and mitigates spoofing attacks without relying on a centralized infrastructure.

While existing research has contributed significantly to IoT security, significant challenges remain. SVM-based models lack scalability, RF-based approaches suffer from high computational latency, CNNs have not been widely explored in real-time IoT threat detection, and Edge computing solutions have yet to integrate adaptive deep learning models for security applications. This study addresses these gaps by proposing a CNN-based distributed security architecture that improves detection accuracy, optimizes computational efficiency, and enhances the resilience of IoT networks against spoofing attacks.

3- Material and Methods

3-1- Description of the Experimental Environment

This study designed an experimental environment in an industrial facility to evaluate the effectiveness of an AI-based spoofing detection and mitigation system in IoT networks [15]. The environment includes a variety of IoT devices, a specific network topology, and controlled attack simulation, all implemented in a technical and detailed manner to reflect real-world conditions.

3-1-1- IoT Devices Used

Representative devices from the industrial environment were selected, including sensors, actuators, and communication nodes, with technical characteristics that allow for evaluation of the performance of the proposed system. The sensors used were the DHT22 model for temperature and humidity measurement. These sensors offer an operating range of -40°C to 80°C for temperature and 0% to 100% for humidity, with a precision of ±0.5°C and ±2%, respectively. They operate with a supply voltage of 3.3V to 6V and provide a serial digital output at 1 Hz, facilitating their integration with microcontrollers. In addition, BMP280 pressure sensors were used, capable of measuring in a range of 300 hPa to 1100 hPa with an absolute precision of ±1 hPa and a resolution of 0.16 hPa, operating at 3.3V and communicating via I2C or SPI. The actuators consisted of SSR-25 DA solid-state relays, capable of handling currents up to 25A at 250V AC. A 3-32V DC signal controls these relays and responds less than 10 ms, allowing efficient control of industrial equipment such as electric motors and lighting systems.

ESP32 microcontrollers were used for communication nodes, equipped with a dual-core 240 MHz processor, 520 KB of SRAM, 802.11 b/g/n Wi-Fi, and Bluetooth 4.2 BLE connectivity [16]. These devices feature integrated cryptography, including AES and RSA encryption accelerators and random number generation, which are essential for implementing communication security measures [17].

3-1-2- Network Topology

The topology of the IoT network deployed in the industrial facility was designed to reflect the complexity and security requirements inherent in this environment. Figure 1 presents the layout of key components and their interconnections. In this configuration, sensors and actuators are strategically distributed in various plant sections, connected to ESP32 nodes that act as local gateways. These nodes collect data from the sensors and manage the actuators, performing local processing to minimize latency and network traffic. Communication between the nodes and the central server is established using the MQTT protocol, recognized for its efficiency on resource-constrained devices [18].

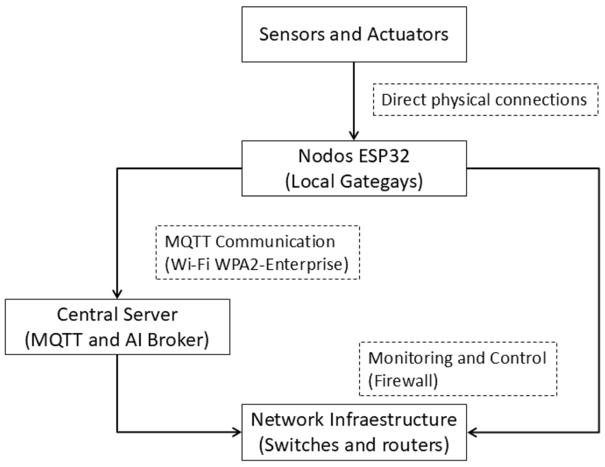

Figure 1. Topology of the IoT network implemented in the industrial facility, highlighting the devices' arrangement and interconnections

The central server, equipped with a 3.4 GHz Intel Core i7 processor and 16 GB of RAM, operates under the Ubuntu 20.04 LTS system. This server hosts an MQTT broker to manage communications and houses the detection algorithms based on Artificial Intelligence (AI). Communication between the nodes and the server is carried out through a wireless network protected with WPA2-Enterprise, using authentication based on X.509 digital certificates to guarantee the devices' identities.

The network infrastructure is logically segmented by VLANs configured on Cisco Catalyst 2960 manageable switches, allowing IoT traffic to be isolated from the rest of the corporate network. In addition, firewall rules are implemented on Cisco ISR 4321 routers to control access and monitor traffic, allowing only the connections necessary for the system's operation. This architecture ensures efficient and secure communication between IoT devices and the central server, maintaining the integrity and confidentiality of data in a complex industrial environment.

3-1-3- Spoofing Attack Simulation

To evaluate the system's resilience to spoofing attacks, test scenarios were designed where different types of attacks were emulated in a controlled manner. In the ARP Spoofing attacks, the Ettercap tool [19] was used on an attacking machine with a Kali Linux operating system [20]. In addition, the network nodes' ARP caches are poisoned, which allows the interception and modification of traffic between the devices and the central server. This simulates an attacker who positions himself in the middle of the communication to capture sensitive data or inject malicious information.

For IP Spoofing attacks, the Scapy library in Python is used, which allows the generation of packets with falsified IP addresses. The attacker sends messages to the central server pretending to be a legitimate node, trying to access restricted resources or send unauthorized commands to the actuators. Packet sequences were configured to imitate the typical behavior of the devices but with slight variations to evaluate the detection capacity of the system.

In the Device Spoofing attacks, an ESP32 node was compromised to extract its credentials and certificates. Using this information, a malicious device was configured that attempted to establish communication with the central server using the stolen credentials [21]. The goal is to assess whether the system can detect behavioral discrepancies, such as differences in the device's physical characteristics (e.g., MAC addresses) or atypical communication patterns.

During all tests, measures were implemented to ensure the security and integrity of the systems. The attacks were carried out at scheduled times and under supervision, isolating the sections of the network involved to avoid unwanted effects on the industrial facility's operations. The data generated, including traffic logs, event logs, and system responses, were collected for further analysis.

The information obtained during the attacks' simulation trains and validated the AI-based detection models. Supervised and unsupervised machine learning techniques were applied to identify abnormal patterns and suspicious behaviors, tuning the algorithms to improve precision and reduce the false positive rate.

3-2- Data Collection and Preprocessing

Data collection and preprocessing are key to developing intrusion detection systems in industrial IoT networks. These phases ensure the information used is accurate, consistent, and suitable for training AI models. To ensure diversity and representativeness, the data set used in this study integrates real and synthetic network traffic from multiple sources, including controlled laboratory environments and publicly available datasets specialized in IoT network anomalies.

3-2-1- Network Traffic Capture

Data traffic is captured on the industrial facility's IoT network to analyze network behavior and detect potential anomalies. This real-world traffic collection is complemented by publicly available datasets, namely IoT-23, CICIoT2023, and RT-IoT2022, which provide a diverse representation of IoT threats.

- IoT-23 Dataset: A widely used dataset that includes benign and malicious traffic from various IoT devices, covering real-world attack scenarios such as botnet intrusions and spoofing.
- CICIoT2023 Dataset: Developed by the Canadian Institute for Cybersecurity, this dataset incorporates different types of attacks, including DDoS, reconnaissance, and spoofing, while simulating interactions from over 100 IoT devices.
- RT-IoT2022 Dataset: Provides attack scenarios applied to IoT networks, including SSH brute force, DDoS, and port scanning.

These datasets ensure a balance between the real world and simulated traffic, allowing the model to generalize effectively to different network conditions.

To further enhance data diversity, network traffic was captured over 30 days using Wireshark, an open-source tool renowned for inspecting packets in-depth and providing detailed visibility into network traffic. The capture was performed at strategic points in the network, covering both the ESP32 nodes and the central server. The collection parameters were configured to record all incoming and outgoing packets, including headers and payloads, allowing for a detailed analysis of communication protocols and patterns.

3-2-2- Extracted Features

Several features are extracted from the captured network traffic that act as key indicators to identify possible spoofing attacks [22]. These features are grouped into several categories, each providing specific information about the network's behavior and the devices involved. The frequencies and sequences of messages exchanged between devices were analyzed for communication patterns. Detecting anomalies in these patterns may indicate malicious activities, such as spoofing attempts. An unexpected increase in message frequency from a device could suggest that it has been compromised [23]. For response times, the time intervals between requests and responses on the network were measured. Significant deviations in these times may signal interference or spoofing attempts, as an attacker could introduce additional latencies by intercepting and retransmitting messages.

For digital signatures, cryptographic signatures present in packets are verified to ensure data authenticity. The absence or alteration of these signatures may indicate unauthorized manipulation, suggesting a possible spoofing attack [24]. Additionally, source and destination IP and MAC addresses of each packet were recorded and analyzed. Inconsistencies, such as duplications or unexpected changes in these addresses, can indicate spoofing attacks, where an attacker attempts to impersonate another legitimate device on the network.

The ports and protocols used in communications were also identified. Unusual or unauthorized ports or protocols may indicate malicious activity, as attackers often employ non-standard techniques to avoid detection. To ensure feature diversity across different network environments, future work will explore dynamically adapting feature selection to various IoT architectures and communication protocols. Table 1 summarizes the main features extracted and their relevance in detecting spoofing attacks.

Table 1. Features Analyzed for Identity Theft Detection

Category	Features Analyzed	Indicators of Attack
Communication Patterns	Frequency and sequence of messages between devices	Atypical behaviors that suggest malicious activity
Response Times	Time intervals between requests and responses	Significant deviations that indicate interference or impersonation attempts
Digital Signatures	Verification of cryptographic signatures in packets	Alterations or absences that suggest unauthorized tampering
IP and MAC Addresses	Recording of source and destination addresses	Duplications or inconsistencies that indicate possible impersonation
Ports and Protocols	Identification of ports and protocols used	Unusual or unauthorized uses that may indicate malicious activity

Analysis of these features allows for identifying atypical behaviors that could indicate malicious activity, such as significant deviations in the time intervals between requests and responses or source and destination addresses that suggest spoofing. Integrating these features into detection models improves the precision and effectiveness of identifying spoofing attacks in industrial IoT network environments.

3-2-3- Data Cleaning and Normalization

Before feeding the AI models, data cleaning and normalization techniques are applied to ensure quality and consistency.

- Missing data treatment is performed by imputing missing values with the arithmetic means, preserving dataset integrity.
- Noise and outlier removal is conducted using standard deviation analysis, excluding values beyond three standard deviations from the mean, reducing the impact of anomalous readings [25].
- Feature normalization is applied using the Min-Max technique, scaling features to [0,1], facilitating convergence and improving model performance.

While these techniques improve dataset quality, real-world industrial environments often introduce more unpredictable noise sources, which current datasets may not fully capture. To further bridge the gap between synthetic and real-world data, we plan to incorporate transfer learning approaches, where models pre-trained on controlled datasets are fine-tuned with real-time IoT traffic captured from industrial networks.

3-3- Implementation of Artificial Intelligence Algorithms

AI algorithm implementation focuses on accurate anomaly detection in industrial IoT networks by integrating advanced machines and deep learning models. Each stage of the implementation, from model selection to performance evaluation, is designed to maximize efficiency and minimize false positives and negatives.

3-3-1- Model Selection

Algorithm selection considers both theoretical performance and problem-specific characteristics. SVM, RF, and CNN are used due to their complementary capabilities.

SVMs, which search for an optimal hyperplane to separate classes, are ideal for high-dimensional spaces. Mathematically, the model is optimized by solving:

$$\min_{w,b,\xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi_i \tag{1}$$

Subject to:

$$Y_i \, (W \cdot X_I + b) \geq 1 - \xi_i, \;\; \xi_i \geq 0 \tag{2}$$

where Y_i is the class label for sample i, and X_i is its feature vector, W and b are the model parameters, ξ_i are the slack variables to handle misclassified points, and C is the hyperparameter that regulates the compromise between maximizing margin and minimizing errors.

Random Forest, an ensemble method based on multiple decision trees, combines predictions from individual trees by majority voting. This reduces the risk of overfitting and improves robustness. The construction of each tree involves random sampling of features and records, which is mathematically defined as:

$$\hat{Y} = \frac{1}{M} \sum_{m=1}^{M} h_m(x) \tag{3}$$

where $h_m(x)$ represents the prediction of the mm-th tree, and MM is the total number of trees.

CNNs were selected to capture spatial and temporal patterns in network traffic. Each convolutional layer performs an operation defined by:

$$y = f(W * x + b) \tag{4}$$

where W are the learned filters, * denotes the convolution operation, b is the bias, and ff is a nonlinear activation function such as ReLU.

3-3-2- Model Training

Training started with splitting the dataset into 70% for training and 30% for testing, ensuring a balanced representation of regular and anomaly classes. K-fold cross-validation (k=5) was applied to enhance model robustness and prevent overfitting, allowing the evaluation of generalization across different dataset partitions. In this process, the dataset is split into k partitions, where each partition is used as a test set once, while the remaining k-1 partitions are used for training. The final performance metric was calculated as the average of the individual metrics of each partition:

$$Average\ Metric = \frac{1}{k}\sum_{1=1}^{k} Metric_i \tag{5}$$

Metric$_i$ represents the metric evaluated on the *i-th* partition of the validation set, such as precision, recall, or AUC-ROC.

Hyperparameter tuning was performed using Grid Search, exploring combinations of parameters such as kernel and C in SVM, maximum depth and number of trees in Random Forest, and filter size and learning rate in CNN—the optimization process aimed to maximize detection precision while minimizing classification errors.

Federated Learning (FL) was integrated into the model training process to enhance adaptability and data privacy. It allows distributed training across multiple Edge nodes while preserving local data confidentiality. Unlike conventional centralized learning, FL enables each Edge node to train a regional model using its network traffic data without transmitting raw data to the central server. Instead, only model weight updates are periodically shared and aggregated through a Federated Averaging (FedAvg) process, ensuring global model improvement without compromising data privacy.

Each Edge node trained its own CNN model locally using network traffic data specific to its environment. The trained model weights were periodically sent to an aggregation server, where an updated global model was computed by averaging the contributions from multiple Edge nodes. This new model was then redistributed back to the nodes, allowing them to refine their detection capabilities while keeping the raw data localized.

The integration of FL in this system provided multiple advantages. By preventing raw data transmission, FL enhanced privacy, reducing the risk of sensitive network information being exposed. Training models on local data improved adaptability to different IoT environments, as each node learned from its own traffic patterns and attack scenarios. Additionally, minimizing the amount of data exchanged with the central server reduced network bandwidth consumption, optimizing overall communication efficiency.

Despite these advantages, FL introduced new computational and latency challenges. The need for periodic model synchronization between nodes introduced additional delays, impacting real-time detection capabilities. Edge nodes with limited resources required optimizations to perform frequent model updates, so implementing model compression techniques and lightweight CNN architectures is necessary. Evaluating FL's impact on detection precision and response times was essential, particularly in large-scale IoT deployments with thousands of interconnected devices.

3-3-3- Performance Evaluation

The evaluation of the models included key metrics to measure their effectiveness. Precision, defined as the proportion of correct predictions:

$$Precision = \frac{TP}{Tp + FP} \tag{5}$$

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

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

While specificity measured the ability to identify regular instances correctly:

$$Specificy = \frac{TN}{TN + FP} \tag{7}$$

Meanwhile, the area under the ROC curve (AUC-ROC) provides a comprehensive measure of the model's ability to distinguish between classes. An AUC-ROC closer to 1 indicates optimal performance.

3-4- Integration of the Detection System into the IoT Network

Integrating the detection system into the industrial IoT network was carefully designed to ensure seamless interaction between devices, network infrastructure, and AI modules. This system combines advanced analytics capabilities with effective incident management, balancing safety and performance.

3-4-1- Detection System Architecture

The detection system is based on a distributed architecture that combines local processing on Edge nodes with advanced analysis on a central server. IoT devices, such as sensors and actuators, generate network traffic initially sent to Edge nodes. These nodes, equipped with basic processing capabilities, perform preliminary analysis tasks to identify suspicious patterns and reduce the load on the leading network.

The central server hosts advanced AI models that perform deep anomaly detection and event classification. This server interacts with an MQTT broker that facilitates efficient and secure communication between the distributed devices and the server [26]. The architecture is complemented by a centralized database, which stores historical records to train and update AI models, and an incident response module responsible for executing automated or manual actions in case of attack detection. Figure 2 presents this architecture, highlighting the interaction between the different components.

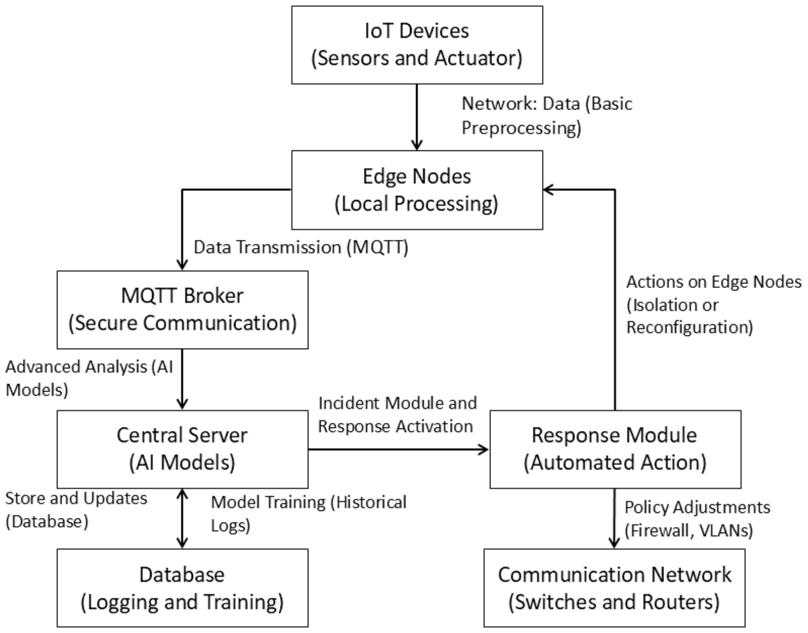
IoT Devices (Sensors and Actuator)
Network: Data (Basic Preprocessing)
Edge Nodes (Local Processing)
Data Transmission (MQTT)
MQTT Broker (Secure Communication)
Actions on Edge Nodes (Isolation or Reconfiguration)
Advanced Analysis (AI Models)
Incident Module and Response Activation
Central Server (AI Models)
Response Module (Automated Action)
Store and Updates (Database)
Model Training (Historical Logs)
Policy Adjustments (Firewall, VLANs)
Database (Logging and Training)
Communication Network (Switches and Routers)

Figure 2. The architecture of the Integrated Detection System in the IoT Network.

3-4-2- Incident Response Procedures

The Incident Response module operates as the core of mitigation actions. When an attack, such as a phishing attempt, is detected, the module triggers predefined procedures, including isolating the compromised device, reconfiguring policies on the network infrastructure, and notifying security personnel.

Adaptive policy-based rules are used to automate these responses. For example, in the event of an increase in the frequency of suspicious messages from a specific device, the module can temporarily disconnect the associated Edge node to prevent the attack's spread. This process is complemented by the generation of real-time alerts, which include technical details of the detected event, such as the IP address of the attacker and the nature of the malicious traffic.

3-4-3- Performance Impact Assessment

Implementing the sensing system involves a thorough assessment of its impact on the overall performance of the IoT network. Key metrics such as latency, resource consumption, and scalability were analyzed to ensure enhanced security does not compromise critical operations.

On the latency side, Edge nodes significantly reduce the time needed to process data locally before sending it to the central server. This approach distributes the workload, keeping response times within acceptable limits for industrial applications [27]. Furthermore, resource consumption at the Edge nodes and the central server was optimized through efficient data processing and compression techniques. The system's scalability is achieved through the modular design of the architecture, which allows the integration of new devices and the expansion of sensing capabilities without affecting existing performance. This ensures that the system can adapt to the growing demands of an ever-evolving IoT network.

3-5- Evaluation of the Effectiveness of the Detection System

The detection system's evaluation was structured in two stages: stress and scalability tests to analyze its performance under different scenarios and a methodological comparative analysis to validate its effectiveness against other existing detection solutions. These stages were designed to ensure the system meets the operational and security requirements expected in industrial IoT networks.

3-5-1- Stress Testing and Scalability

Stress testing aims to assess the system's ability to handle increasing traffic volumes and an increasing number of connected devices. A controlled environment was implemented using traffic simulators such as Apache JMeter and synthetic data generators based on statistical distributions replicating authentic IoT traffic patterns. Devices are configured to generate events that mimic normal activities and anomalies, such as spoofing patterns.

In each iteration of the tests, the number of devices gradually increased from 50 to 1000 while keeping the data generation intervals and traffic characteristics constant. The evaluation included average system latency, packet processing time at the Edge nodes and the central server, and resource consumption (CPU, memory, and bandwidth). These metrics were recorded using monitoring tools, such as Prometheus and Grafana, integrated into the system [28, 29]. To ensure scalability, additional tests were performed simulating the dynamic addition of new devices to the network. This procedure assessed the system's ability to adapt its detection processes without interruptions or significant performance degradation. In addition, aspects such as load redistribution across Edge nodes and the central server's ability to handle concurrent requests were analyzed.

3-5-2- Comparative Analysis

The comparative analysis was designed to validate the system's effectiveness in terms of precision, efficiency, and resource consumption by comparing it with alternative solutions. This procedure involved implementing representative models based on recurrent neural networks (RNN) and traditional rule-based methods run under the same experimental conditions.

The methodological approach included preparing the dataset for the three systems, ensuring an identical partition into training and test sets. Standardized metrics, such as precision, sensitivity, and specificity, were implemented to evaluate each system's performance in classifying events. These metrics were calculated based on the predictions generated by the models and the actual values noted in the test data.

Additionally, parameters related to the response time of each system were analyzed using automated tools that recorded the intervals from the detection of an event to the execution of corresponding actions. This methodological analysis identified the differences in latency inherent to the approaches used. As a final element, an evaluation framework was designed to measure resource consumption in each solution. This procedure considered CPU and memory measurement on Edge devices and the central server, employing specific profiling tools, such as *htop* and *iotop*, which provided detailed data on resource usage at the process level [30].

3-6- Ethical and Privacy Considerations

Implementing a detection system in industrial IoT networks involves handling large volumes of data that may contain sensitive information related to an organization's critical devices or processes, although not directly linked to humans. Therefore, it is essential to ensure that the measures adopted comply with ethical standards and privacy regulations.

3-6-1- Sensitive Data Management

The detection system was designed to adhere to best practices in data handling, ensuring data privacy and integrity at all process stages. Although the collected data does not include information directly related to humans, it may expose details about the operational infrastructure or communication patterns in the IoT network, which could be considered sensitive information. Advanced anonymization techniques were implemented to protect this data. Each captured data packet goes through a pseudonymization process that replaces the IP and MAC addresses of the devices with unique identifiers generated using secure hash functions, such as symmetric encryption (SHA-256) [31]. This ensures that the information cannot be directly traced to a specific device without access to the corresponding decryption keys.

Additionally, the data stored in the central database is encrypted using AES-256 algorithms. This approach ensures that the data is inaccessible in the event of unauthorized access to physical or logical storage. During transmission, the TLS 1.3 protocol protects data in transit, preventing man-in-the-middle attacks [32].

Implementing restricted access policies achieved compliance with international regulations, such as the General Data Protection Regulation (GDPR). Only authorized administrators can access the processed data, and all access is logged by an audit system that documents every action performed.

3-6-2- Informed Consent

Although this project does not directly involve human users, ethical practices are required to ensure transparency and consent from stakeholders such as industrial IoT network administrators and those responsible for the technological infrastructure. To this end, a documented consent protocol was developed that clearly defines the system's objectives, data collection types, and measures implemented to protect privacy.

Before implementing the system, those responsible for the industrial environment received a detailed briefing specifying how the data would be handled and the tools used for anomaly detection. This document also included clauses on the secure deletion of collected data after meeting the study's objectives or periodically to prevent unnecessary accumulation.

Transparency in the implementation process was also ensured through briefings with administrators, where queries were resolved, and ethical measures were presented in detail. These procedures ensured that the organizations fully understood the system's scope and implications, promoting trust in its operation.

4- Results

4-1- Evaluation of the Performance of the Detection System

The detection system evaluation was carried out using a previously preprocessed dataset that includes IoT network traffic from recognized sources in the scientific community, thus reinforcing the validity of the results. The IoT-23 Dataset stands out among the selected datasets, as it captures benign, and malware traffic executed on IoT devices in a controlled environment. This set was complemented by the CICIoT2023 Dataset, developed by the Canadian Institute for Cybersecurity, which includes multiple types of attacks, such as DDoS, surveillance, and spoofing, along with diverse traffic generated by over 100 IoT devices.

These data were combined with samples from the RT-IoT2022 Dataset, which incorporates real-world attack scenarios on IoT devices, such as SSH brute force, DDoS, and port scanning with Nmap, reflecting common patterns in industrial IoT networks. Approximately 100,000 samples were worked with, evenly distributed between normal and anomalous events, ensuring a balanced representation for training and evaluating the models.

The data was subjected to 5-fold cross-validation (k=5) to ensure that performance metrics were statistically robust and representative across different dataset partitions. Additionally, controlled noise levels ranging from 0% to 50% were introduced in the test samples to analyze the impact of varying levels of disturbances on model performance. This step was critical in determining each model’s robustness under real-world industrial conditions, where network noise is a common challenge. These techniques and the inclusion of data from recognized sources provide a solid foundation for evaluating the detection system in industrial IoT networks.

Table 2 presents the performance metrics of the SVM, RF, and CNN models under two qualitative noise levels: low *("Low")* and high *("High")*. These noise categories reflect realistic conditions in industrial IoT networks, from relatively stable environments to those with more significant disturbances due to interference, faulty devices, or malicious traffic. Each row in the table summarizes the average performance metrics of the models based on multiple iterations, ensuring that the reported values capture overall trends rather than isolated results.

Table 2. Model Performance Metrics Under Different Noise Levels

Model	Noise Level	Precision (%)	Sensitivity (%)	Specificity (%)	AUC-ROC
SVM	Low	90	88	91	0.89
SVM	High	85	83	86	0.85
Random Forest	Low	92	90	93	0.91
Random Forest	High	88	85	89	0.87
CNN	Low	95	93	94	0.96
CNN	High	91	89	92	0.92

The results indicate that CNN consistently outperforms SVM and Random Forest across different noise levels. Notably, under high noise conditions, the CNN model maintains an AUC-ROC of 0.92, a precision of 91%, and a sensitivity of 89%, demonstrating its superior capability to handle adverse conditions. In contrast, SVM and RF exhibit a more pronounced drop in sensitivity and specificity under high-noise scenarios, highlighting their vulnerability to noisy data. These findings suggest that deep learning-based approaches, particularly CNNs, offer more stable performance in challenging industrial IoT environments with frequent network disturbances.

Figure 3 includes two complementary graphs that allow a more visual understanding of the results. Graph (A) shows the variation in sensitivity and specificity of the models in a continuous range of noise levels, from 0% to 50%. On the other hand, graph (B) illustrates the discriminative capacity of the models, highlighting how noise affects their performance.

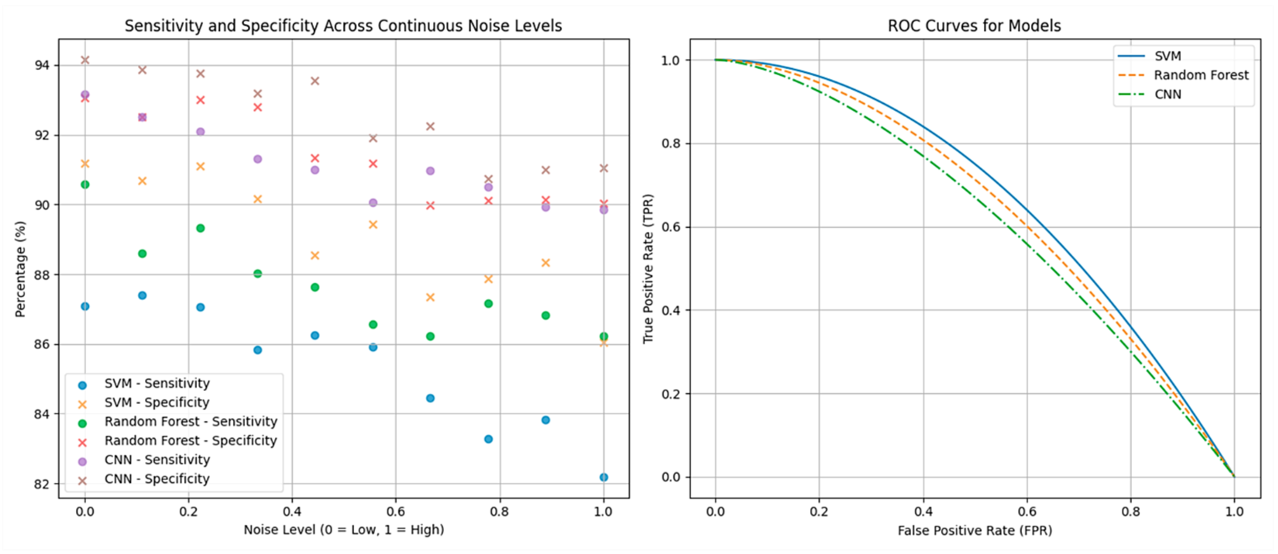

Figure 3. Sensitivity and Specificity Analysis with Continuous Noise Levels and ROC Curves; Graph A: Sensitivity and Specificity according to Noise Levels; Graph B: Comparative ROC Curves for SVM, Random Forest and CNN Models

The trends observed in Graph A confirm that CNNs maintain a more controlled performance degradation compared to SVM and RF models. While CNN exhibits a moderate decrease in sensitivity and specificity as noise increases, both SVM and RF show a more pronounced decline, particularly when noise surpasses 30%. This highlights CNNs' higher adaptability in handling real-world IoT network conditions where unpredictable disturbances may be present.

Similarly, Graph B reinforces CNN's superior performance, with its ROC curve maintaining a higher AUC-ROC value (close to 0.96 in low-noise conditions and 0.92 in high-noise environments). This demonstrates its ability to distinguish between normal and anomalous traffic with high reliability, even under adverse conditions. On the other hand, the ROC curves for SVM and RF display a more significant drop in performance, confirming their sensitivity to variations in data quality.

Combining datasets from IoT-23, CICIoT2023, and RT-IoT2022 and introducing realistic noise conditions, this study provides a robust and comprehensive evaluation of anomaly detection performance in industrial IoT networks. These findings highlight the advantages of CNN-based approaches for maintaining detection accuracy in real-world deployments where network disturbances and adversarial conditions are inevitable.

4-2-Scalability Analysis

The scalability analysis evaluated how the proposed system responds to increasingly connected devices and growing network traffic, simulating realistic conditions in an industrial IoT environment. Key performance metrics, including average latency, CPU and memory utilization, and processing capacity, were analyzed to determine the system's efficiency and robustness under different configurations.

The system was tested under five configurations, ranging from 50 to 1,000 connected devices. Each configuration generated heterogeneous traffic patterns with variable transmission rates, reflecting diverse real-world IoT deployments. To ensure statistical reliability, each configuration was subjected to 5,000 iterations, capturing fluctuations in system behavior and minimizing anomalies. Latency, CPU usage, memory consumption, and processing capacity were monitored in real-time using specialized tracking tools, allowing precise analysis of system behavior under increasing loads.

Table 3 summarizes the performance metrics, illustrating the system's response to load levels. As connected devices increase, latency rises from 25 ms (50 devices) to 150 ms (1,000 devices), following a nearly linear trend. CPU and memory usage also increase proportionally, reaching 85% CPU utilization and 1,280 MB of memory in the highest-load scenario. Additionally, processing capacity (throughput) decreases from 200 to 100 requests per second, indicating the impact of higher loads on system responsiveness.

Table 3. System Performance Metrics Under Different Device Loads

Number of Devices	Average Latency (ms)	CPU Usage (%)	Memory Usage (MB)	Processing Capacity (Requests/s)
50	25	30	256	200
200	45	40	512	180
500	75	60	768	150
800	110	75	1024	120
1000	150	85	1280	100

Multiple optimization strategies have been incorporated to ensure that Edge nodes with limited computing power can effectively handle CNN-based detection models. First, model compression techniques such as quantization and pruning have been applied to reduce the computational complexity of CNN inference, minimizing memory usage and processing time. Quantization reduces model precision to lower-bit representations, significantly decreasing memory requirements while maintaining detection accuracy. Conversely, Pruning removes redundant neurons and connections, optimizing computational efficiency without degrading model performance.

Second, inference acceleration methods have been employed, including edge-specific optimizations such as TensorFlow Lite and ONNX Runtime. These frameworks allow lightweight CNN deployments on constrained hardware by leveraging hardware acceleration mechanisms in modern Edge devices, such as ARM-based Neural Processing Units (NPUs). This ensures that CNN inference can be executed efficiently without introducing excessive latency or resource overhead.

Third, an adaptive workload distribution mechanism has been implemented to dynamically allocate processing tasks between Edge nodes and the central server based on real-time resource availability. Under low-load conditions, Edge nodes handle most detection tasks independently, reducing reliance on centralized computing. However, as computational demand increases, processing tasks are offloaded to the central server to prevent individual nodes from becoming bottlenecks. This hybrid processing strategy balances resource utilization and ensures continuous real-time operation even in high-density IoT environments.

Figure 4 presents two graphs that visually depict the impact of device load on system performance. These graphs provide deeper insights into how latency, resource usage, and processing capacity evolve as the number of devices increases.

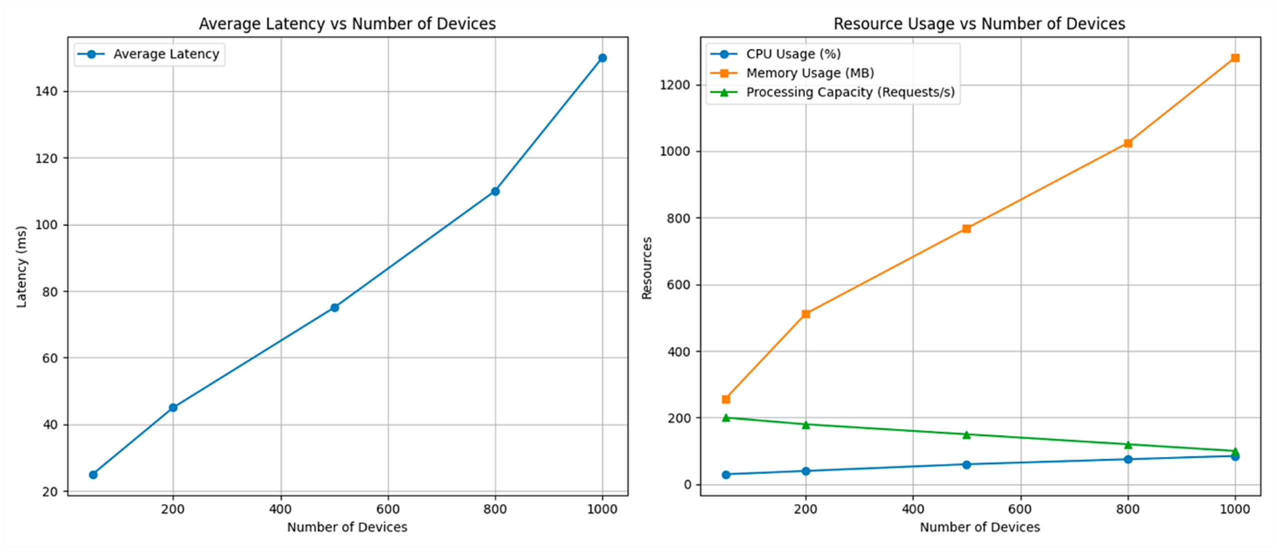

Figure 4. System Scalability Analysis, Graph A: Average Latency Variation Based on the Number of Connected Devices, Graph B: Resource Usage and Processing Capacity Based on Device Load

Graph A shows that average latency increases progressively as the number of devices grows, from 25 ms to 150 ms. This trend is expected due to the additional computational workload introduced by an increasing number of devices and higher network traffic. However, the relationship remains linear, indicating that the system efficiently distributes requests across nodes and the central server. Unlike non-optimized architectures where latency grows exponentially, the controlled growth observed here suggests that the proposed system successfully mitigates the impact of high loads through effective request handling and resource distribution.

Graph B examines CPU and memory usage alongside processing capacity under varying load conditions. CPU utilization starts at 30% for 50 devices and reaches 85% for 1,000 devices, showing a proportional increase as expected in resource-intensive IoT environments. However, the system maintains stable operation even at maximum load, suggesting that computational efficiency remains within acceptable margins. Memory consumption scales from 256 MB to 1,280 MB as connected devices increase, maintaining a controlled and predictable growth pattern that aligns with the system's dynamic resource allocation strategies. The lack of abrupt increases suggests that memory usage is efficiently managed, preventing unnecessary bottlenecks. Processing capacity exhibits a predictable decline, decreasing from 200 to 100 requests per second as the number of devices increases. This drop highlights the expected impact of handling larger workloads but remains within an operationally viable range, ensuring the system does not experience failure or excessive performance degradation.

The results indicate that the system can handle increasing device loads without experiencing exponential latency growth, confirming its scalability. The nearly linear increase in latency reflects a well-balanced resource allocation strategy that prevents network congestion and avoids performance degradation. The increase in CPU and memory usage remains proportional to the number of devices, ensuring that resources are utilized efficiently. Even under the highest load configuration, the system remains operational and responsive. The decline in processing capacity as the number of devices grows highlights a natural limitation in handling concurrent requests but does not compromise the system's overall stability. These findings confirm that the proposed system is scalable, robust, and capable of adapting to increasing network demands without excessive resource exhaustion or critical failures.

The scalability analysis demonstrated that the proposed system efficiently manages increasing IoT network loads of up to 1,000 connected devices, maintaining controlled latency and resource consumption growth. However, the study does not include experimental validation for larger-scale deployments with tens of thousands of devices, which would introduce additional challenges regarding computational load and network traffic distribution. Scaling beyond this range would require additional optimizations, such as hierarchical Edge processing architectures, model partitioning, or adaptive resource allocation strategies to balance detection performance and real-time processing requirements. Future work will evaluate the system under extended scalability conditions, considering high-density IoT deployments where thousands of devices communicate simultaneously. This will involve further testing with distributed Edge architectures and dynamic workload management to minimize latency spikes while preserving detection accuracy.

4-3- Comparison of AI Models

The SVM, RF, and CNN models were compared using three key metrics: precision, response time, and resource consumption (CPU and memory). These metrics were evaluated under low and high noise conditions to simulate realistic IoT network environments and assess each model's robustness in handling data variability. Table 4 presents the quantitative results of the key metrics for each model, calculated as an average of 5,000 iterations per model, ensuring statistical reliability and reducing variability in performance measurements.

Table 4. Comparison of AI Models Under Low and High Noise Conditions

Model	Condition	Precision (%)	Response Time (ms)	CPU Usage (%)	Memory Usage (MB)
SVM	Low Noise	90	30	40	280
SVM	High Noise	85	40	50	320
Random Forest	Low Noise	92	35	45	330
Random Forest	High Noise	88	45	55	380
CNN	Low Noise	95	20	55	430
CNN	High Noise	91	30	65	470

The CNN model demonstrated superior precision in low-noise conditions, achieving 95%, while Random Forest and SVM obtained 92% and 90%, respectively. Under high-noise conditions, all models experienced a decline in precision, but CNN maintained the highest accuracy at 91%, outperforming Random Forest at 88% and SVM at 85%. This highlights CNN's ability to manage complex and noisy data more effectively than traditional machine learning models, making it particularly suited for industrial IoT environments where network traffic often includes high levels of anomalies and noise.

Regarding response time, CNN consistently exhibited lower value than the other models. Even under high noise, CNN maintained an average response time of 30 ms, compared to 40 ms for SVM and 45 ms for Random Forest. This faster processing speed enhances its suitability for real-time applications, where rapid anomaly detection and response are critical. However, the improved performance of CNN comes at the cost of higher resource consumption. Under high noise conditions, CNN required 65% CPU utilization and 470 MB of memory, whereas SVM and RF consumed 50%, 320 MB, and 55% and 380 MB, respectively. This trade-off between computational efficiency and accuracy must be considered when deploying AI-driven security solutions in IoT networks with resource-constrained devices.

Figure 5 presents three graphs visually comparing the models' performance across different noise levels. Graph A illustrates model precision under low and high noise conditions, revealing that while all models experience a decline in precision due to noise, CNN consistently maintains a higher accuracy, reinforcing its superior generalization capabilities.

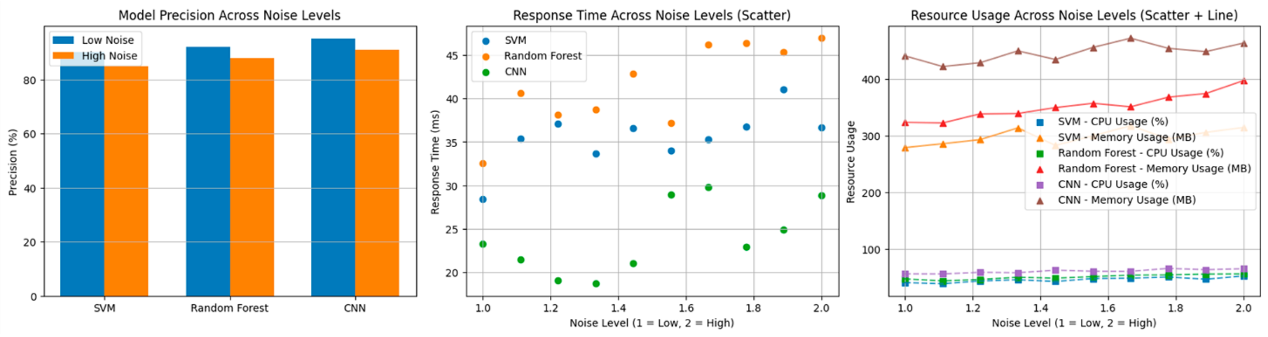

Figure 5. Comparative Performance of AI Models in IoT Networks, Graph A: Precision of the Models According to Noise Levels., Graph B: Variability in Response Time Under Different Noise Levels., Graph C: Resource Consumption Under Different Noise Levels.

Graph B examines response time variations across different noise levels. The results indicate that CNN maintains the most stable response time across low and high noise conditions. In contrast, SVM and Random Forest exhibit more significant fluctuations, particularly in high-noise environments. This stability in response time is crucial for real-time IoT applications, where rapid decision-making is required to mitigate security threats.

Graph C analyzes CPU and memory consumption for each model under low and high noise conditions. The results confirm that CNN, while consuming more resources, provides a proportional increase in performance, with higher precision and lower response time. In contrast, SVM and RF exhibit lower resource usage but fail to match CNN's effectiveness under challenging conditions. These findings suggest that CNN is the most robust option for anomaly detection in IoT networks, provided that the available infrastructure can support its computational demands.

The comparative analysis confirms that CNN is the most effective model for IoT security applications, offering the highest precision and fastest response times across varying noise conditions. However, its increased computational cost may necessitate optimization strategies when deployed in resource-constrained environments. The balance between accuracy, response time, and resource consumption is crucial in selecting the appropriate AI model for a given industrial IoT scenario.

4-4- Impact on IoT Network Performance

The assessment of IoT network performance impact focused on analyzing how the system deployment affects key metrics, including latency, resource usage, and traffic distribution between Edge nodes and the central server. These results provide a comprehensive view of system behavior in high-load industrial environments where network efficiency and scalability are crucial.

Table 5 presents the latency, resource usage, and traffic distribution values before and after system deployment, incorporating a percentage analysis of the observed changes to quantify the system’s impact. The results indicate that the deployment introduces a 66.67% average latency increase from 30 ms to 50 ms. This increase is attributable to the additional real-time detection and analysis processes, which introduce computational overhead to the standard data flow. However, the increase remains within acceptable margins for most industrial applications, ensuring system functionality is not compromised despite the additional processing requirements.

Table 5. System Impact on IoT Network Performance

Metric	Before System	After System	Change (%)
Latency (ms)	30	50	+66.67
CPU Usage (%)	10	25	+150.00
Memory Usage (MB)	100	300	+200.00
Edge Processing Traffic (%)	30	70	+133.33
Central Server Traffic (%)	70	30	-57.14

The system deployment results in a 150% increase in CPU usage, from 10% to 25%, and a 200% increase in memory consumption, from 100 MB to 300 MB. These increments reflect the computational cost of executing detection algorithms, particularly on Edge nodes with limited processing capabilities. While these increases are significant, redistributing processing loads away from the central server mitigates potential network congestion and improves overall scalability.

The most notable improvement is observed in traffic distribution. The proportion of locally processed traffic increases from 30% to 70%, significantly reducing the burden on the central server, which now handles only 30% of the traffic instead of the previous 70%. This redistribution optimizes network efficiency and scalability, allowing the system to handle larger device loads without overwhelming central processing resources.

Figure 6 presents three graphs that provide a detailed visualization of the system's impact. Graph A illustrates the increase in average latency after system deployment, showing a linear growth pattern from 30 ms to 50 ms as the number of devices increases. This behavior confirms the predictable nature of the additional computational overhead introduced by real-time processing, ensuring the system remains stable as the load increases. The controlled increase suggests that the architecture is well-designed to handle scalability without significant degradation in performance.

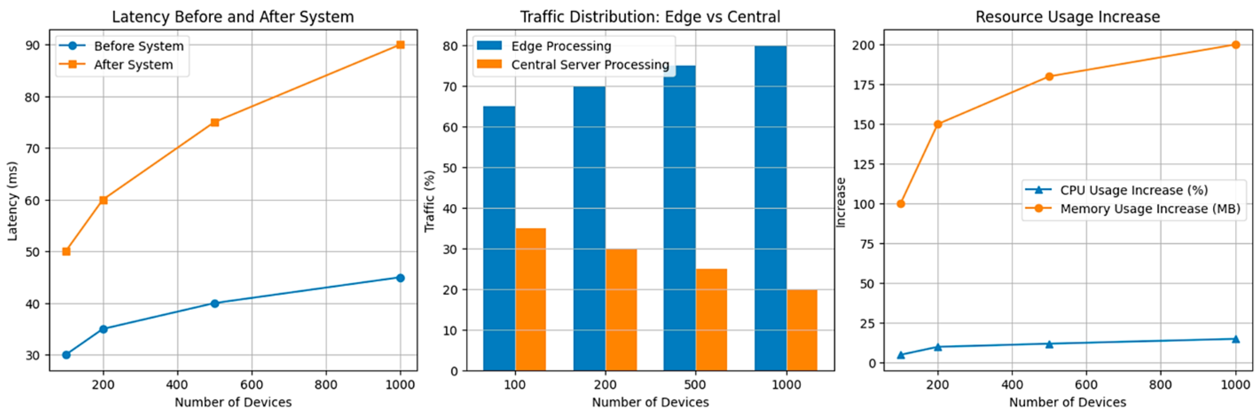

Figure 6. Analysis of the Impact on the Performance of the IoT Network, Graph A: Latency Before and After System Implementation; Chart B: Distribution of Processed Traffic on Edge Nodes and the Central Server; Graph C: Increase in Resource Use.

Graph B highlights the shift in traffic distribution between Edge nodes and the central server, showing a progressive increase in locally processed traffic. As the number of connected devices increases, the proportion of traffic handled by the Edge nodes grows consistently, reaching 80% with 1,000 devices. This result confirms that the decentralized processing approach effectively offloads the central infrastructure, enhancing the overall resilience and efficiency of the network.

Graph C presents an increase in CPU and memory usage as the number of devices grows. CPU usage remains within a controlled range, suggesting that processing optimizations help prevent excessive computational strain, even under high device loads. Memory usage increases significantly, which aligns with the system's design and leverages Edge computing resources to support real-time detection without excessive reliance on the central server.

The analysis confirms that while system integration introduces measurable impacts on latency and resource consumption, these effects are effectively counterbalanced by improvements in network efficiency. The shift in traffic distribution towards Edge processing significantly reduces the load on central infrastructure, ensuring the network remains scalable and adaptable to growing IoT demands.

4-5- Incident Response Validation

To validate the proposed system's performance, four simulated scenarios were designed to represent different levels of network complexity and load. These scenarios assessed the system's capability to detect and mitigate spoofing attempts, ensuring the evaluation reflected realistic and controlled conditions. Each scenario introduced variations in the number of connected devices, the intensity of the attacks, and the noise level in the captured data, allowing for a comprehensive performance assessment under diverse operational conditions.

In the first scenario, the network consisted of 100 IoT devices and simulated low intensity spoofing attempts, targeting only a single node. This scenario established baseline performance under optimal conditions. The results indicated a high detection rate of 95%, with an average detection time of 150 ms and a response time of 300 ms. Additionally, 97% of the compromised devices were successfully isolated, with a false positive rate of 3%, demonstrating the system's accuracy in low-complexity environments.

The second scenario doubled the number of devices to 200, introducing multiple simultaneous attempts to increase attack complexity. Under these conditions, the detection rate slightly decreased to 92%, while detection and response times increased to 180 ms and 350 ms, respectively. Despite this, the system maintained a high effectiveness rate in isolating compromised devices, reaching 94%, with a false positive rate of 4%, confirming its resilience against increasing attack intensity.

The third scenario expanded the network to 500 IoT nodes, with spoofing attacks targeting multiple nodes in different network segments. The detection rate remained stable at 90%, with detection and response times rising to 200 ms and 400 ms, respectively. Although the percentage of successfully isolated devices slightly declined to 92%, the system exhibited strong adaptability to more complex attack patterns, with a false positive rate increasing slightly to 5%.

In the fourth scenario, the system was subjected to extreme conditions, with 1,000 IoT nodes and high-intensity attacks co-occurring at multiple points in the network. The detection rate decreased to 88%, while detection and response times rose to 220 ms and 450 ms, respectively. The percentage of correctly isolated devices was 89%, with a false positive rate of 6%. These results highlight the system's operational limits in high-density environments, emphasizing the need for further optimization in resource allocation and distributed processing to maintain high detection rates under extreme conditions.

Table 6 presents the results obtained in these scenarios, providing a detailed comparison of key metrics, including detection rate, detection and response time, percentage of correctly isolated devices, and false positive rate.

Table 6. Validation Results in Simulated Scenarios.

Scenario	Detection Rate (%)	Correctly Isolated Devices (%)	False Positive Rate (%)
Scenario 1	95	97	3
Scenario 2	92	94	4
Scenario 3	90	92	5
Scenario 4	88	89	6

Figure 7 visually represents detection and response times across different scenarios, illustrating how these key metrics vary as network complexity increases.

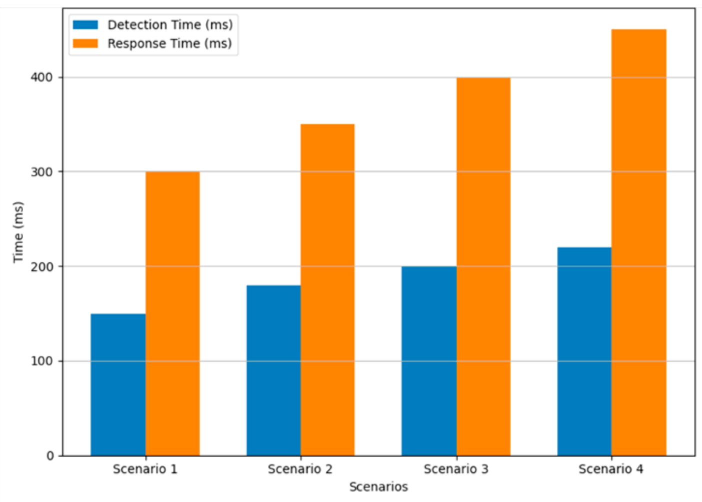

Figure 7. Comparison of Detection and Response Times by Scenario

The results confirm that the system remains highly effective in detecting and mitigating spoofing attacks across different levels of network complexity. Graph A in Figure 7 illustrates the variations in detection and response times, revealing a consistent increase in latency as attack complexity rises. While this trend is expected due to the additional processing required for analyzing and responding to threats, the overall response time remains within acceptable thresholds for industrial IoT applications. The system's ability to maintain detection rates above 88% even under extreme conditions demonstrates its robustness in handling real-world attack scenarios. However, the increase in response time in high-density networks suggests potential areas for improvement in computational efficiency and distributed processing strategies.

The effectiveness of compromised device isolation remains high across all scenarios, exceeding 89% even in the most complex environments, reinforcing the system's ability to contain security threats. The slight rise in false positive rates under more intense attack conditions suggests that further fine-tuning of the detection model could help balance sensitivity and specificity, optimizing performance under high network loads.

The overall findings confirm that the proposed system provides a reliable and scalable solution for IoT network security, capable of detecting and mitigating spoofing attacks in various conditions. Detection accuracy remains consistently high, and response times remain within acceptable operational limits. However, as network size and attack intensity increase, latency and computational overhead also grow, suggesting the need for optimization in distributed processing strategies.

5- Discussion

The results obtained in this study align with and expand upon trends reported in existing literature, particularly regarding the limitations of traditional machine learning approaches and the advantages of deep learning-based solutions for IoT security. Prior works, such as those by Alaghbari et al. [33] and Altulaihan et al. [34], demonstrated the effectiveness of SVMs and RFs in detecting anomalies in IoT networks, particularly in environments with stable traffic patterns and moderate loads. However, these studies also highlighted significant challenges related to scalability, response time, and computational resource consumption under high-load conditions, which were similarly observed in this study. In high-complexity scenarios, SVM and RF models exhibited a decline in detection rates, an increase in response time, and higher computational overhead, confirming their limitations in large-scale industrial IoT networks [35].

In contrast, the CNN-based approach implemented in this study effectively overcame these challenges, achieving consistently high detection rates, even in scenarios involving multiple simultaneous attacks. The results align with Alamatsaz et al. [8], who demonstrated that CNNs can capture complex patterns in network traffic, improving detection precision while reducing false positives. The findings further reinforce that deep learning architectures can offer superior anomaly detection capabilities in dynamic and noisy environments where traditional machine learning models struggle.

A key innovation introduced in this study is the distributed processing architecture leveraging Edge nodes as primary detection points, representing a significant advancement over traditional centralized approaches. This architecture aligns with the findings of Maghrabi [12], who emphasized the importance of decentralized computing for improving scalability and reducing latency in IoT networks. The proposed system effectively delegates detection and mitigation tasks to Edge nodes, reducing the computational burden on central servers and enhancing scalability. The results demonstrate that this approach enables real-time threat detection while maintaining system efficiency in networks with up to 1,000 connected devices.

Beyond scalability, the results indicate that Edge processing also mitigates the impact of increased attack complexity, a challenge widely acknowledged in industrial IoT environments—existing studies, such as Kumar et al. [9] and Bukhsh et al. [14], have explored the role of Edge computing in distributed threat detection; however, their implementations primarily focused on load balancing and general anomaly detection rather than targeted spoofing attack mitigation. This study's findings extend this knowledge by demonstrating how, when integrated into an Edge computing framework, CNN-based detection models can efficiently detect and isolate malicious activity while minimizing false positives.

Although the proposed system has demonstrated robust detection performance, one key challenge in real-world deployment is ensuring that the dataset used for model training accurately represents actual IoT network conditions. This study used publicly available datasets, including IoT-23, CICIoT2023, and RT-IoT2022, to train and evaluate detection models. These datasets incorporate a wide range of attack scenarios and IoT traffic patterns, contributing to the representativeness of the experiments. However, industrial IoT environments exhibit even more significant variability regarding device heterogeneity, traffic behaviors, and attack sophistication.

One limitation of the data sets is that they do not fully capture complex, large-scale industrial networks with diverse real-world device interactions and security threats. While they provide a strong foundation, future work will focus on reducing the gap between simulated and real traffic by collaborating with IoT device manufacturers and industrial partners to collect live traffic data from production environments. This will enable the model to generalize more effectively to unseen attack patterns and network anomalies.

Additionally, future research will explore integrating transfer learning techniques, where models pre-trained on controlled datasets are fine-tuned using real-time IoT traffic captured from operational networks. This approach would allow the system to adapt to evolving threats without requiring complete retraining, ensuring long-term reliability in practical deployments. Another enhancement explored in this study is the integration of Federated Learning (FL) to decentralize model training and improve system adaptability. Unlike centralized training approaches that require direct access to raw network data, FL allows each Edge node to train its own CNN model locally while only transmitting model weight updates to a central aggregation server. This method significantly enhances data privacy, preventing sensitive network traffic from being exposed beyond the local environment. Moreover, FL reduces network bandwidth consumption by limiting the data exchanged with the central server.

However, FL introduces new computational and latency challenges that require careful consideration. While it enhances privacy and local adaptability, the need for periodic synchronization between Edge nodes and the aggregation server can introduce additional delays, potentially affecting real-time detection capabilities. Additionally, Edge nodes with limited processing power may struggle to handle frequent model updates, necessitating optimization techniques such as model compression or lightweight CNN architecture. In preliminary evaluations, the integration of FL demonstrated comparable detection precision to traditional centralized training approaches while offering improved scalability for large-scale IoT deployments. However, response times exhibited slight increases due to model synchronization overhead, suggesting that further optimizations are needed to balance detection accuracy and real-time performance in federated settings. Future research will focus on refining FL strategies, such as adaptive model aggregation and asynchronous update mechanisms, to mitigate these latency issues while maintaining high detection rates.

Regarding the system's overall effectiveness, although detection rates remained high across all scenarios, the observed increase in false positive rates in more complex cases (up to 6%) suggests further improvements in model optimization and training data diversity. Implementing advanced hyperparameter tuning techniques and incorporating a broader dataset with an increased representation of real-world IoT network variations could help mitigate this issue. Additionally, while response times remain within acceptable industrial thresholds, further load distribution optimization among Edge nodes could enhance system responsiveness under extreme conditions.

Despite the improvements offered by this approach, certain limitations must be considered when interpreting the results. The dataset used for training and evaluation combines simulated and real-world data captured in a controlled environment. While this ensures diversity and representativeness, real-world IoT deployments exhibit significantly greater traffic patterns, device configurations, and attack sophistication variability. Consequently, generalizing these results to all IoT networks requires additional validation in diverse operational settings.

Moreover, the simulated scenarios assume a homogeneous distribution of IoT devices and attack vectors, which may not fully replicate the complexity of real-world deployments. In practical settings, IoT devices vary widely in processing power, network connectivity, and security protocols, which could impact the system's detection accuracy and efficiency. Additionally, the attack simulations in this study focused exclusively on spoofing attempts, while other prevalent threats, such as distributed DDoS attacks and APT, remain unaddressed [36]. Future studies should consider expanding the scope of attack scenarios to provide a more comprehensive evaluation of the system's robustness.

Another aspect to consider is the consumption of computational resources, particularly at Edge nodes. Although the results showed that the system maintains acceptable performance in terms of latency and memory usage, implementing CNN models on capacity-limited devices might not be feasible in extremely constrained IoT networks [37]. This limitation aligns with findings from Alamatsaz et al. [8] and Bukhsh et al. [14], who emphasized that computationally intensive deep learning models may not always be feasible for lightweight IoT deployments. Future research could explore model compression techniques, such as pruning and quantization, to reduce computational overhead without compromising detection accuracy.

Despite these limitations, the findings of this study provide valuable insights into the design and implementation of AI-based threat detection systems for IoT networks. The integration of CNNs with an Edge computing framework represents a significant step forward in addressing key challenges related to IoT security, particularly in terms of response time, scalability, and computational efficiency [38]. Additionally, implementing resource-efficient inference techniques, such as CNN model compression (quantization and pruning) and adaptive workload distribution, has proven effective in optimizing the computational overhead at Edge nodes. These strategies allow real-time inference to be executed without overwhelming resource-constrained devices, ensuring that detection accuracy is maintained while preventing system bottlenecks. The findings suggest that the proposed optimizations contribute to a balanced trade-off between computational efficiency and detection precision, making the approach viable for large-scale industrial IoT networks. These results establish a solid foundation for future advancements in IoT security, highlighting the potential of AI-driven distributed architectures to enhance the resilience of large-scale industrial networks.

While the results confirm the system’s ability to maintain stable operation in networks of up to 1,000 connected IoT devices, further evaluation is required to determine its behavior under significantly larger-scale deployments. In practical industrial environments, IoT networks can reach tens of thousands of devices, posing new challenges in traffic management, computational efficiency, and threat detection under extreme conditions.

Future optimizations will explore multi-tier Edge architectures, hierarchical model aggregation, and adaptive load-balancing techniques to improve performance in large-scale settings. Additionally, federated learning and distributed inference strategies could minimize communication overhead and enhance system scalability. While the current study provides a strong foundation for detecting and mitigating spoofing attacks in IoT networks, extending to more extensive infrastructures will require further empirical validation and system refinement to ensure real-world applicability on a global scale.

6- Conclusions

This study presents a solution for detecting and mitigating spoofing attacks in IoT networks by integrating CNN with a distributed architecture based on Edge nodes. The results demonstrate the proposed system's effectiveness, highlighting its ability to operate in complex and high-density environments and balance precision, response time, and scalability. These features position it as a viable solution for industrial applications and other critical environments where IoT network security is essential.

One of the main contributions of this work is validating the system in simulated scenarios that replicate realistic conditions. Regarding detection rate, the system reached 95% in low-complexity scenarios and 88% in high-load conditions and multiple simultaneous attacks. These results highlight the system's robustness, even in the face of the increasing complexity of the threats evaluated. Using CNN allows for capturing complex patterns in traffic data, improving anomaly detection precision compared to traditional methods such as SVMs and Random Forests.

The distributed design, which delegates processing tasks to Edge nodes, proved to be fundamental for the scalability and efficiency of the system. This architecture significantly reduced the load on the central server, improving the distribution of processed traffic while maintaining acceptable response times. In scenarios with up to 1,000 connected devices, detection and response times gradually increased from 150 to 220 ms and 300 to 450 ms, respectively, demonstrating a robust response to high device density and simultaneous attacks. These metrics underline the importance of distributed processing in addressing scalability challenges in industrial IoT networks.

Another relevant aspect is the low false positive rate observed in all scenarios, ranging between 3% and 6%. This metric is crucial to ensure that the alerts generated are relevant and actionable, minimizing the impact on the network's regular operation. Furthermore, the system successfully isolated between 97% and 89% of compromised devices, highlighting its effectiveness in containing threats once detected. These capabilities are essential to ensure the operational continuity of IoT networks in critical industrial environments.

Despite these achievements, the study also identified limitations that must be addressed in future research. One of the main constraints was the dataset used, which included both simulated data and data captured in controlled environments. Although this combination provided diversity in traffic patterns, it does not fully represent the variability of IoT networks in real deployments. The lack of heterogeneity in IoT device capabilities and attack configurations also simplified some aspects of the experimental design, which could have influenced the results.

Additionally, this study recognizes that IoT devices vary widely in computational capacity, which can impact the deployment of CNN models on resource-constrained nodes. Future research should explore optimization techniques such as model compression, quantization, and lightweight architectures to ensure efficient inference without compromising detection accuracy. Further enhancements could involve integrating federated learning, allowing distributed model training across Edge nodes while preserving data privacy. This would improve adaptability and detection precision while minimizing communication overhead.

Another area of interest is scalability beyond the tested 1,000-device configuration. While the system maintained strong performance at this scale, larger industrial deployments could introduce additional challenges in traffic management, computational load, and detection reliability. Future work should investigate hierarchical Edge architectures and adaptive resource allocation mechanisms to optimize performance in extensive IoT networks.

Extending the system to detect other attack types, such as DDoS attacks and advanced persistent threats, would enhance its applicability. Dynamic prioritization techniques and real-time resource allocation could improve responsiveness under high-load conditions. These directions for future research will help refine and expand the system, ensuring its effectiveness in increasingly complex industrial IoT environments.

7- Declarations

7-1- Author Contributions

Conceptualization, W.V.-Ch. and J.G.; methodology, W.V.-Ch. and I.O.-G.; software, J.G.; validation, W.V.-Ch. and J.G.; formal analysis, I.O.-G.; investigation, W.V.-Ch.; resources, W.V.-Ch.; data curation, J.G.; writing—original draft preparation, I.O.-G.; writing—review and editing, W.V.-Ch.; visualization, J.G.; supervision, I.O.-G; project administration, W.V.-Ch.; funding acquisition, W.V.-Ch. All authors have read and agreed to the published version of the manuscript.

7-2- Data Availability Statement

The data presented in this study are available on request from the corresponding author.

7-3- Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

7-4- Institutional Review Board Statement

Not applicable.

7-5- Informed Consent Statement

Not applicable.

7-6- Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this manuscript. In addition, the ethical issues, including plagiarism, informed consent, misconduct, data fabrication and/or falsification, double publication and/or submission, and redundancies have been completely observed by the authors.

8- References

[1] Kathole, A. B., Vhatkar, K. N., & Patil, S. D. (2024). IoT-Enabled Pest Identification and Classification with New Meta-Heuristic-Based Deep Learning Framework. Cybernetics and Systems, 55(2), 380–408. doi:10.1080/01969722.2022.2122001.

[2] Irgat, E., Çinar, E., Ünsal, A., & Yazıcı, A. (2023). An IoT-Based Monitoring System for Induction Motor Faults Utilizing Deep Learning Models. Journal of Vibration Engineering and Technologies, 11(7), 3579–3589. doi:10.1007/s42417-022-00769-5.

[3] Olawale, O. P., & Ebadinezhad, S. (2023). The Detection of Abnormal Behavior in Healthcare IoT Using IDS, CNN, and SVM. Lecture Notes on Data Engineering and Communications Technologies, 166, 375–394. doi:10.1007/978-981-99-0835-6_27.

[4] Prathapchandran, K., & Janani, T. (2021). A trust aware security mechanism to detect sinkhole attack in RPL-based IoT environment using random forest – RFTRUST. Computer Networks, 198. doi:10.1016/j.comnet.2021.108413.

[5] Shukla, S., Thakur, S., & Breslin, J. G. (2022). Anomaly Detection in Smart Grid Network Using FC-Based Blockchain Model and Linear SVM. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Vol. 13163 LNCS, 157–171. doi:10.1007/978-3-030-95467-3_13.

[6] Abraham, O. A., Ochiai, H., Hossain, M. D., Taenaka, Y., & Kadobayashi, Y. (2024). Electricity Theft Detection for Smart Homes: Harnessing the Power of Machine Learning with Real and Synthetic Attacks. IEEE Access, 12, 26023–26045. doi:10.1109/ACCESS.2024.3366493.

[7] Kim, H. (2024). Sub-optimal Internet of Thing devices deployment using branch and bound method. IET Networks, 1-9. doi:10.1049/ntw2.12119.

[8] Alamatsaz, N., Tabatabaei, L., Yazdchi, M., Payan, H., Alamatsaz, N., & Nasimi, F. (2024). A lightweight hybrid CNN-LSTM explainable model for ECG-based arrhythmia detection. Biomedical Signal Processing and Control, 90. doi:10.1016/j.bspc.2023.105884.

[9] Kumar, P., Bagga, H., Netam, B. S., & Uduthalapally, V. (2022). SAD-IoT: Security Analysis of DDoS Attacks in IoT Networks. Wireless Personal Communications, 122(1), 87–108. doi:10.1007/s11277-021-08890-6.

[10] Long, Z., & Jinsong, W. (2022). A hybrid method of entropy and SSAE-SVM based DDoS detection and mitigation mechanism in SDN. Computers and Security, 115. doi:10.1016/j.cose.2022.102604.

[11] Elhag, S., Alghamdi, A. M., & Al-Shomrani, N. A. (2023). Toward an Improved Security Performance of Industrial Internet of Things Systems. SN Computer Science, 4(2), 131. doi:10.1007/s42979-022-01566-3.

[12] Maghrabi, L. A. (2024). Automated Network Intrusion Detection for Internet of Things: Security Enhancements. IEEE Access, 12, 30839–30851. doi:10.1109/ACCESS.2024.3369237.

[13] Kanwal, R., Rafaqat, W., Iqbal, M., & Weiguo, S. (2023). Data-Driven Approaches for Wildfire Mapping and Prediction Assessment Using a Convolutional Neural Network (CNN). Remote Sensing, 15(21), 5099. doi:10.3390/rs15215099.

[14] Bukhsh, M., Abdullah, S., & Bajwa, I. S. (2021). A Decentralized Edge Computing Latency-Aware Task Management Method with High Availability for IoT Applications. IEEE Access, 9, 138994–139008. doi:10.1109/ACCESS.2021.3116717.

[15] Al-Kasassbeh, M., Almseidin, M., Alrfou, K., & Kovacs, S. (2020). Detection of IoT-botnet attacks using fuzzy rule interpolation. Journal of Intelligent and Fuzzy Systems, 39(1), 421–431. doi:10.3233/JIFS-191432.

[16] Elhattab, K., & Abouelmehdi, K. (2024). Intelligent agriculture system using low energy and based on the use of the internet of things. Bulletin of Electrical Engineering and Informatics, 13(2), 1286–1297. doi:10.11591/eei.v13i2.6346.

[17] Panika, L., Maharana, V., Rao, J. K., Chandrakar, H., & Biswas, S. D. (2024). Securing the Sensitive Data Transfer in Cloud using AES and RSA Algorithms. International Journal of Research Publication and Reviews, 5(1), 867–871. doi:10.55248/gengpi.5.0124.0127.

[18] Paris, I. L. B. M., Habaebi, M. H., & Zyoud, A. M. (2023). Implementation of SSL/TLS Security with MQTT Protocol in IoT Environment. Wireless Personal Communications, 132(1), 163–182. doi:10.1007/s11277-023-10605-y.

[19] Chavoshi, H. R., Salasi, A. H., Payam, O., & Khaloozadeh, H. (2023). Man-in-the-Middle Attack Against a Network Control System: Practical Implementation and Detection. 2023 IEEE 64th International Scientific Conference on Information Technology and Management Science of Riga Technical University, ITMS 2023 – Proceedings, 10317671. doi:10.1109/ITMS59786.2023.10317671.

[20] Mvah, F., Kengne Tchendji, V., Tayou Djamegni, C., Anwar, A. H., Tosh, D. K., & Kamhoua, C. (2024). Countering ARP spoofing attacks in software-defined networks using a game-theoretic approach. Computers and Security, 139. doi:10.1016/j.cose.2023.103696.

[21] Mvah, F., Kengne Tchendji, V., Tayou Djamegni, C., Anwar, A. H., Tosh, D. K., & Kamhoua, C. (2024). GaTeBaSep: game theory-based security protocol against ARP spoofing attacks in software-defined networks. International Journal of Information Security, 23(1), 373–387. doi:10.1007/s10207-023-00749-0.

[22] Shombot, E. S., Dusserre, G., Bestak, R., & Ahmed, N. B. (2024). An application for predicting phishing attacks: A case of implementing a support vector machine learning model. Cyber Security and Applications, 2. doi:10.1016/j.csa.2024.100036.

[23] Shukla, S., Misra, M., & Varshney, G. (2024). HTTP header based phishing attack detection using machine learning. Transactions on Emerging Telecommunications Technologies, 35(1), e4872. doi:10.1002/ett.4872.

[24] Ghantous, W., Pintore, F., & Veroni, M. (2024). Efficiency of SIDH-based signatures (yes, SIDH). Journal of Mathematical Cryptology, 18(1), 20230023. doi:10.1515/jmc-2023-0023.

[25] Sewwandi, M. A. N. D., Li, Y., & Zhang, J. (2024). K-Outlier Removal Based on Contextual Label Information and Cluster Purity for Continuous Data Classification. Expert Systems with Applications, 237. doi:10.1016/j.eswa.2023.121347.

[26] Kamoun, K., Hmissi, F., Ouni, S., & Ouni, S. (2024). Improvement of MQTT semantic to minimize data flow in IoT platforms based on distributed brokers. Transactions on Emerging Telecommunications Technologies, 35(2), 4945. doi:10.1002/ett.4945.

[27] Yan, S., Zhang, P., Huang, S., Wang, J., Sun, H., Zhang, Y., & Tolba, A. (2023). Node Selection Algorithm for Federated Learning Based on Deep Reinforcement Learning for Edge Computing in IoT. Electronics (Switzerland), 12(11), 2478. doi:10.3390/electronics12112478.

[28] Ciric, V., Milosevic, M., Sokolovic, D., & Milentijevic, I. (2024). Modular deep learning-based network intrusion detection architecture for real-world cyber-attack simulation. Simulation Modelling Practice and Theory, 133. doi:10.1016/j.simpat.2024.102916.

[29] Malhotra, R., Bansal, A., & Kessentini, M. (2024). Deployment and performance monitoring of docker based federated learning framework for software defect prediction. Cluster Computing, 27(5), 6039–6057. doi:10.1007/s10586-024-04266-0.

[30] Tassi, A., Warren, D., Wang, Y., Bhamare, D., & Behravesh, R. (2024). On Optimization of Next-Generation Microservice-Based Core Networks. IEEE Transactions on Vehicular Technology, 73(6), 9199–9204. doi:10.1109/TVT.2024.3355335.

[31] Nakamura, K., Hori, K., & Hirose, S. (2021). Algebraic fault analysis of sha-256 compression function and its application. Information (Switzerland), 12(10). doi:10.3390/info12100433.

[32] Michelena, Á., Aveleira-Mata, J., Jove, E., Bayón-Gutiérrez, M., Novais, P., Romero, O. F., Calvo-Rolle, J. L., & Aláiz-Moretón, H. (2024). A novel intelligent approach for man-in-the-middle attacks detection over internet of things environments based on message queuing telemetry transport. Expert Systems, 41(2), e13263. doi:10.1111/exsy.13263.

[33] Alaghbari, K. A., Lim, H. S., Saad, M. H. M., & Yong, Y. S. (2023). Deep Autoencoder-Based Integrated Model for Anomaly Detection and Efficient Feature Extraction in IoT Networks. Internet of Things, 4(3), 345–365. doi:10.3390/iot4030016.

[34] Altulaihan, E., Almaiah, M. A., & Aljughaiman, A. (2024). Anomaly Detection IDS for Detecting DoS Attacks in IoT Networks Based on Machine Learning Algorithms. Sensors, 24(2), 713. doi:10.3390/s24020713.

[35] Shieh, C. S., Nguyen, T. T., Chen, C. Y., & Horng, M. F. (2023). Detection of Unknown DDoS Attack Using Reconstruct Error and One-Class SVM Featuring Stochastic Gradient Descent. Mathematics, 11(1), 108. doi:10.3390/math11010108.

[36] Elamparithi, P., Kalaivani, S., Vijayalakshmi, S., Keerthika, E., Koteswari, S., & Raaj, R. S. (2024). A Machine Learning Approach for Detecting DDOS Attack in IoT Network Using Random Forest Classifier. International Journal of Intelligent Systems and Applications in Engineering, 12(2s), 495–502.

[37] Canizo, M., Triguero, I., Conde, A., & Onieva, E. (2019). Multi-head CNN–RNN for multi-time series anomaly detection: An industrial case study. Neurocomputing, 363, 246–260. doi:10.1016/j.neucom.2019.07.034.

[38] Priyanga, S., Krithivasan, K., Pravinraj, S., & Shankar Sriram, V. S. (2020). Detection of Cyberattacks in Industrial Control Systems Using Enhanced Principal Component Analysis and Hypergraph-Based Convolution Neural Network (EPCA-HG-CNN). IEEE Transactions on Industry Applications, 56(4), 4394–4404. doi:10.1109/TIA.2020.2977872.