Vol. 07, No. 01, pp. 95 –109 (2026)
ISSN: 2708-0757

JOURNAL OF APPLIED SCIENCE AND TECHNOLOGY TRENDS

www.jastt.org

A Novel Architecture and Methodology to Detect Intrusions Against Edge-Based IIoT Using Machine Learning

Sahar L. Qaddoori[1*], Qutaiba I. Ali[2]

[1]*Electronic Department, Electronics Engineering College, Ninevah University, Mosul, Iraq, sahar.qaddoori@uoninevah.edu.iq*
[2] *Computer Engineering Department, Engineering College, University of Mosul, Mosul, Iraq, qut1974@gmail.com*

**Correspondence: sahar.qaddoori@uoninevah.edu.iq*

Abstract
The increasing demand for the Industrial Internet of Things (IIoT), with billions of connected things and the decentralization of data exchange, is gaining momentum, making conventional threat detection and analysis challenging in such distributed environments. In this paper, a security framework for edge nodes, called the Intrusion Detection, Prevention, and Response System (IDPRS), is proposed. It aims to detect MQTT (Message Queuing Telemetry Transport)-based threats using Machine Learning (ML) algorithms. However, ML models cannot be trained on resource-constrained devices; therefore, the approach trains the model on a high-performance platform, which will later serve as the detection engine on an edge node. The edge node can be hosted on low-cost single-board computers (SBCs), such as the Raspberry Pi. The detection model is further monitored and updated using an upgrade algorithm to make it adaptive to emerging threats. The evaluation results demonstrate high detection accuracy and reasonable resource and network overhead.

Keywords: *Intrusion Detection System (IDS), Intrusion Prevention System (IPS), Firewall, Intrusion Response System (IRS), Fog Node, MQTTset dataset.*
Received: August 26th, 2025 / Revised January 18th, 2026 / Accepted: February 05th, 2026 / Online: February 09th, 2026

I. INTRODUCTION

Industrial Internet of Things (IIoT) is the extension of the common principles of IoT to industrial applications and manufacturing processes [1]. The IIoT has evolved over the last 10 years as an emerging technology with the ability to connect and digitize different industries to create significant economic value and enable new industries to grow, thus becoming a major contributor to the world's Gross Domestic Product (GDP) [2-4]. Examples of IIoT applications are connected cars, smart homes, smart cities, smart grids, smart factories, and supply chain systems [1].

The massive amount of data that is produced by the IIoT network's sensors, which are resource-restricted devices with constrained power, connectivity capabilities, and memory [1, 2]. To link sensors and cloud servers, edge devices such as routers, laptops, desktops, cell phones, handheld devices, and microservers are used. The edge devices gather data from appliances and deliver it to nearby servers after carrying out any necessary preprocessing. However, as specific IoT edge-layer devices have gained popularity in industry, several privacy and security concerns have emerged, posing a significant threat to the security and reliability of the IIoT. Intruders may exploit these edge devices [1]. Therefore, this manuscript focuses on the security issues of edge-based IIoT.

One of the most often used solutions for IIoT security challenges is the intrusion detection system (IDS). It is used to detect network traffic, particularly to distinguish between legitimate and malicious traffic, and so helps to eliminate malicious traffic [5, 6]. IDS uses two main techniques: signature-based and anomaly-based detection systems. Signature-based methods detect known threats by scanning network data for defined patterns, whereas anomaly-based systems detect an attack by monitoring the system behavior, traffic, or objects. The system behavior, traffic, or objects are compared against an already well-defined baseline that exhibits normal behavior [7, 8]. Anomaly-based IDS using machine learning techniques has better generalization than signature-based IDS. [8, 9].

One of the main research trends for anomaly-based approaches is the use of ML techniques. These methods learn generalized properties of the traffic for training and use the trained features to classify the input traffic correctly.

doi: 10.38094/jastt71439

Generalized learning from traffic data enables better responses to unknown data using machine learning. Traditional machine learning is well-suited to IIoT edge devices, which typically have relatively low resource consumption and short training times [10, 11].

This study proposes a lightweight fog–edge Intrusion Detection, Prevention, and Response System (IDPRS) for IIoT environments. The main contributions are:

- An integrated IDS/IPS/IRS architecture for edge devices, extending beyond prior IDS-only systems.
- A dynamic upgrading mechanism that monitors edge performance and triggers fog-level retraining, enabling adaptive and up-to-date detection.
- A hybrid feature-selection approach combining Random Forest importance with cross-model consistency to reduce computational overhead.
- A fog-assisted training strategy that builds ML models on fog nodes and deploys them efficiently on constrained edge hardware.
- Coordinated fog–edge response and detection, improving reliability and scalability in distributed IIoT networks.
- Comprehensive performance evaluation covering accuracy, latency, energy consumption, and resource usage to validate real-world applicability.

The manuscript is organized as follows: Section II presents the literature review. In Section III, the general structure of IIoT networks is described. In contrast, Section IV outlines the methodology of the proposed system. The experimental results and discussion are demonstrated in Section V. The comparative study is explained and tabulated in Section VI. Finally, Section VII clarifies the conclusions.

II. Literature Review

With the significant advances in IIoT and the consequent need for network security, intrusion detection is a key area of focus. The last years have seen a plethora of works proposing ML models to address the task of developing IDS solutions in IoT and IIoT networks [12, 13]. The most notable are presented shortly as follows:

In [14], E. Aydogan et al. proposed a genetic programming-based intrusion detection solution for IIoT platforms, presented along with a proof of concept and quantitative evaluation. The study also evaluated the potential of a centralized IDS at the root node of the RPL-based mesh, considering device heterogeneity and resource constraints. Results demonstrated the root node's ability to perform real-time detection of RPL attacks. The approach proposed in [15] also employed machine learning models to secure IoT networks, with a trade-off between performance and computational complexity. An MQTT-based protocol was suggested for NN model updates on constrained devices, with training and optimization performed on robust IoT gateways. In contrast, only the lightweight testing computation is performed on IoT nodes.

In [16], S. Latif et al. proposed a prediction system with high accuracy on IIoT attacks using a lightweight random neural network architecture. The experimental results, obtained using the DS2OS dataset, confirmed the efficiency of the proposed method, with reduced prediction time and validation of the solution under different test conditions. In [17], the authors considered attack detection in fog computing environments using an ensemble of multiple machine learning approaches. Decision Tree, Random Forest, and K-Means were used to process the ensemble results, with all evaluations performed on the KDD Cup '99 dataset. Passban [18], an intelligent IDS that can be directly deployed on low-cost IoT gateways (such as a single-board PC) was also developed. Passban was shown to effectively detect SSH, HTTP brute-force, SYN flood, and port-scanning attacks, and to report a low false-positive rate and high accuracy.

In [19], J. B. Awotunde et al. suggested a framework to detect intrusions in IIoT by deep learning. Rule-based feature engineering was combined with a deep feedforward neural network to detect network intrusions, and the approach was evaluated on the NSL-KDD and UNSW-NB15 datasets, demonstrating robust detection performance. In [20], a two-stage deep learning-based IDS for IIoT networks was proposed. DNNs were used in the first stage to detect attacks. Then, the attacks with a low detection rate were forwarded to a second stage that integrated the Negative Selection Algorithm (NSA) optimized with the Dragonfly Algorithm. Then Dempster-Shafer's rule of combination is used to fuse the outputs of both DNNs and NSA. The suggested model was evaluated using the CICIDS2017, CICIDS2018, and TON IoT datasets, and its effectiveness was confirmed.

In one of the most recent works, in [21], X.-H. Nguyen et al. proposed Realguard, a lightweight DNN-based IDS that can be directly deployed on local gateways to protect IoT devices. Realguard was designed using simple feature extraction combined with a DNN-based detection model and has been shown to accurately identify multiple cyberattacks in real-time, such as botnets, port scans, and FTP-Patator. The CICIDS2017 dataset was used to test the solution, which demonstrated accurate and precise detection with a low processing footprint, making it suitable for resource-constrained devices (e.g., Raspberry Pi).

In this study, a new approach is proposed to develop DL-based IDS for IoT devices. This intelligent system utilizes a four-layer deep Fully Connected (FC) network to detect the traffic that may trigger IoT device attacks. To alleviate such deployment complexities, the proposed system has been built to be independent of the communication protocol. During the experimental performance analysis, the suggested system is observed to be a successful model of simulated as well as actual intrusions [22].

This paper proposes a GA_RF technique for detecting cyber-attacks on the IIoT environment ICSs that employ MQTT protocol. This architecture connects the ICS with edge devices and cloud servers to monitor the data field collected by sensors, using a GA_RF algorithm to identify anomalous results. Typically, the data is processed on-site before being shared with the cloud for storage and processing, and then returned for continuous observation and protection. Also, in a real case, the MQTT-IOT-IDS2020 dataset was used to predict the proposed

GA_RF method, compared with other powerful machine learning and deep learning methods [23].

Zhukabayeva et al. address the growing security vulnerabilities in Industrial Internet of Things (IIoT) Cyber-Physical Systems (CPS) by proposing an integrated framework that leverages edge computing to facilitate real-time traffic analysis and intrusion detection. To mitigate the high latency associated with traditional centralized cloud solutions, the authors developed a hybrid methodology using the NF-ToN-IoT-V2 dataset, combining unsupervised K-means clustering for traffic segmentation with supervised machine learning models—specifically Random Forest, K-Nearest Neighbors, and Logistic Regression—for anomaly classification. The results demonstrated that K-means outperformed DBSCAN in clustering traffic patterns, while the Random Forest algorithm achieved the highest detection accuracy with an F1 score of 0.99, validating the framework's effectiveness in providing scalable, low-latency security for industrial environments [24].

Unlike previous intrusion detection frameworks that focus solely on centralized or cloud-based analysis [Refs. 24, 18, 17, 19], the proposed IDPRS introduces a hybrid fog–edge coordination mechanism that enables distributed learning and real-time response across IIoT nodes. Furthermore, the system integrates adaptive machine learning models that self-tune based on local anomaly patterns, a feature not addressed in earlier work.

However, based on the literature, there remains a need for an innovative, reliable, secure, and resilient IDS for the IIoT edge layer.

III. General Structure of IIoT Networks

In the integration of Industrial Internet of Things (IIoT), a multilevel architecture is commonly used and often structured into three hierarchical levels, as shown in Fig. 1, including edge, fog, and cloud [25]. The edge layer comprises billions of resource-constrained IIoT nodes, including sensors, actuators, vehicles, smart appliances, wearable devices, surveillance cameras, and other intelligent machines. These networks mainly collect large volumes of data in a heterogeneous manner, then transmit the same to the upper levels, where they are handled further [26, 27]. The fog layer lies between the edge and the cloud and typically includes a host of routers, servers, and controllers controlled by Internet Service Providers (ISPs). Mog nodes, compared to edge devices, have enhanced computational and memory capacity and are therefore suitable for workloads that require more memory, energy, and processing units. Moreover, their decentralization in various geographic locations of the ISP infrastructures enables them to be closer to the edge devices, but still allows them to interface with multiple applications and protocols [27]. At the top of the hierarchy is the cloud layer, where powerful servers, supercomputers, and large-scale storage infrastructures are generally located in enterprise data centers. Intensive data management functions, such as large-scale storage, processing, allocation, and general orchestration of IIoT services, are the responsibility of this layer [28].

IV. The Proposed System Methodology

The significant number of IIoT devices installed and used worldwide will naturally generate large amounts of data that must be collected and analyzed in real time to identify potential attacks. Therefore, a centralized approach to intrusion detection is insufficient for IIoT security monitoring, and the proposed methodology must leverage a distributed model architecture. The methodology proposes using a machine-learning-based model that runs on distributed edge nodes, managed and updated by a fog-computing layer. This enables the real-time monitoring of data from various IIoT nodes. The methodology was divided into four main phases: data collection and feature selection, intrusion detection model development and learning, intrusion detection model deployment on edge nodes, and performance monitoring and model updates. The different stages in Fig. 2 are further clarified in the following sections.

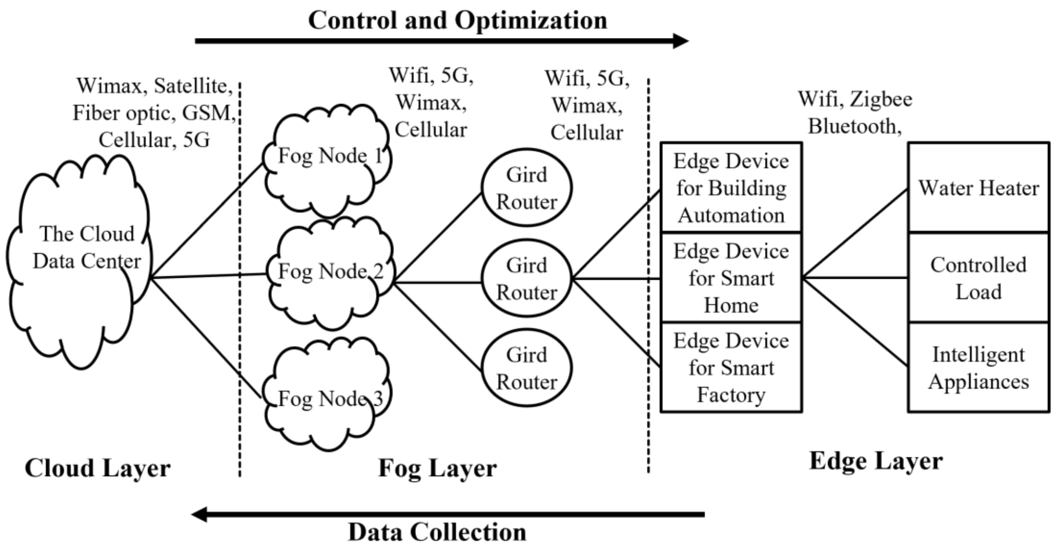

Fig. 1. An example of three hierarchical levels of multi-tier IIoT architecture.

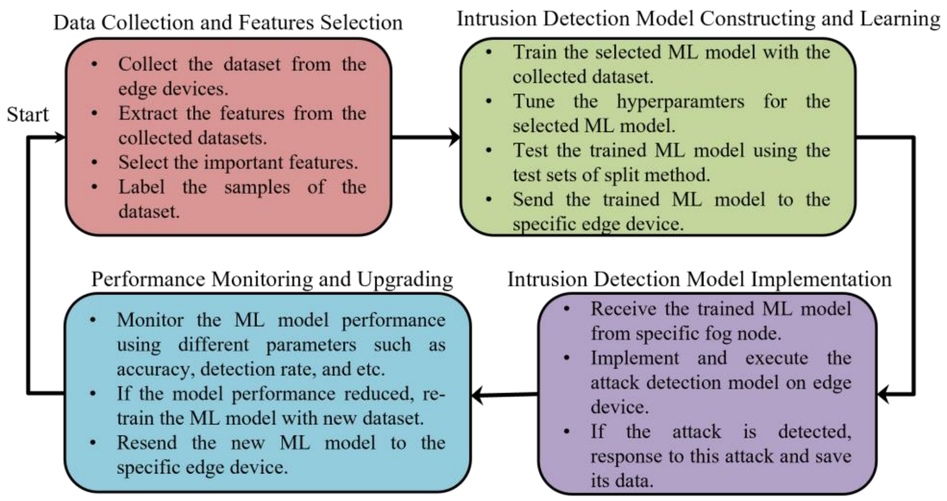
Data Collection and Features Selection
Start
Collect the dataset from the edge devices.
Extract the features from the collected datasets.
Select the important features.
Label the samples of the dataset.
Intrusion Detection Model Constructing and Learning
Train the selected ML model with the collected dataset.
Tune the hyperparamters for the selected ML model.
Test the trained ML model using the test sets of split method.
Send the trained ML model to the specific edge device.
Performance Monitoring and Upgrading
Monitor the ML model performance using different parameters such as accuracy, detection rate, and etc.
If the model performance reduced, re-train the ML model with new dataset.
Resend the new ML model to the specific edge device.
Intrusion Detection Model Implementation
Receive the trained ML model from specific fog node.
Implement and execute the attack detection model on edge device.
If the attack is detected, response to this attack and save its data.

Fig. 2. The suggested intrusion detection, prevention, and response framework in the IIoT system.

A. Data Collection and Feature Selection

In this phase, a system is designed to acquire, store, and analyze IIoT data for edge-node intrusion detection, and the ML model is trained at the nearest fog node. The newly acquired data from edge devices is transmitted to the fog node every month via the intelligent gateway. Since IIoT devices continuously generate a large amount of data. One approach to reduce dataset size is to extract the features of interest from newly acquired data for the ML model and store only those selected features. Therefore, in the proposed model, the fog node receives only the extracted features of interest from edge devices, rather than the entire larger datasets, for analysis through feature reduction. Furthermore, the feature reduction can be represented as a filtering stage, allowing the model to emphasize vital features through learning.

The automated feature selection approach is not in the scope of this manuscript (but it can be used with deep learning algorithms to select the essential and adequate features as future work). Additionally, in future work, an intrusion detection model could be built for this stage, given the absence of a label for the received data.

B. Intrusion Detection Model Constructing and Learning

The second phase involves selecting an appropriate ML model and training it on IIoT data from the fog device to enhance its detection performance against various attacks. For this study, tree-based learning models (namely, Gradient Boosted Trees, Random Forests, and Decision Trees) are adopted due to their widespread use and their ability to offer diverse performance for different feature types. This stage involves several steps, the first of which is training the ML model using the IIoT data collected and prepared in the first phase. This is executed on the fog layer to expedite the training process. Hyperparameter tuning is conducted for each dataset and architecture to improve the ML model's performance further.

The major hyperparameters for the tree-based models used in this study are the number of estimators, the learning rate, and the tree depth. These are preset before training, but can be adjusted during training to improve results. The resulting model is then evaluated using conventional performance measures on data that it has not been trained on. This process may be repeated by further tuning hyperparameters when performance is unsatisfactory, until the best-performing result is obtained. By conducting the computationally most expensive operations offline, the IDS can operate efficiently and smoothly. Fig. 3 illustrates the processing steps for constructing and learning the intrusion detection model.

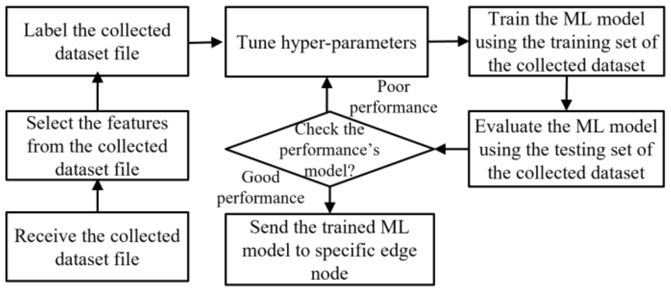
Label the collected dataset file
Tune hyper-parameters
Train the ML model using the training set of the collected dataset
Select the features from the collected dataset file
Poor performance
Check the performance's model?
Evaluate the ML model using the testing set of the collected dataset
Good performance
Receive the collected dataset file
Send the trained ML model to specific edge node

Fig. 3. The processing steps for constructing and learning the intrusion detection model.

C. Intrusion Detection Model Implementation

During the third step of the suggested approach, the intrusion detection model is deployed on edge nodes. Implementations of machine learning applications at this tier have several benefits to IIoT systems, such as the capability of analyzing sensitive data nearer to the devices that produce the data, decreased time between data gathering and processing, lower network bandwidth utilization, and effective execution of tasks at the edge devices or transmitted to fog nodes. Besides, mobility support is achieved through such deployment. The fog layer provides services to manage the detection process, enabling scalability, distribution, and rapid response. The edge node may assume various roles, including operating as a firewall/IPS, an intrusion detection system (IDS), an intrusion response system (IRS), a network traffic broker, or a web server. Fig. 4 illustrates the software architecture of the proposed edge node, with further subsections explaining the role of each component.

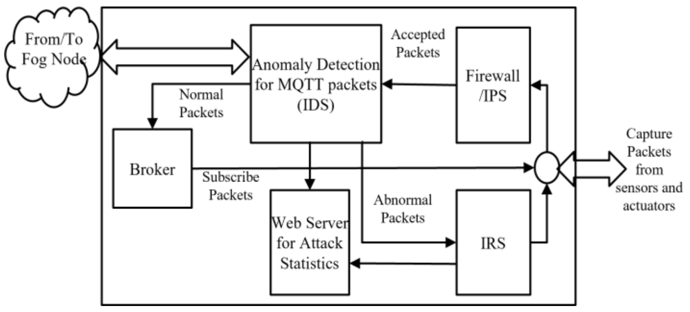
From/To Fog Node
Anomaly Detection for MQTT packets (IDS)
Accepted Packets
Firewall /IPS
Normal Packets
Broker
Subscribe Packets
Web Server for Attack Statistics
Abnormal Packets
IRS
Capture Packets from sensors and actuators

Fig. 4. The details of the software architecture for the proposed edge device

1) Firewall/IPS Module: Firewall/IPS, which is a standard method of preventing network threats, is frequently utilized to block banned connections and separate internal networks from unconfident activities. Because the firewall protects the edge node, the IDS examines all packets that travel through it. Blocklist patterns, such as source and destination addresses in the IP layer and source and destination port numbers in the transport layer, are pre-registered by the blocklist firewall/IPS. Based on this, the packet's robustness and suitability for network access are determined, and access is granted. As a result, the first line of defense against invasions is the firewall/IPS. It identifies current attacks and prevents them by reducing computational costs. The suggested edge node includes a firewall/IPS to construct a trusted barrier between the broker and sensor nodes. The following functions are performed by the firewall/IPS that has been installed:

- Disallow packets whose source IP address is included in the IP address blocklist.
- Block packets whose destination port addresses are included in the port blocklist.
- Block specific IP address prefixes from networks using the blocklist of prefix addresses.
- By defining the packet threshold and time threshold, users may stop excessively rapid requests sent by the same IP, such as ping assaults.

2) IDS Module: Firewalls/IPSs are not always completely reliable and/or efficient enough to protect IIoT networks from all forms of assaults. The packets that pass through the firewall/IPS are forwarded to the IDS under test, which inspects them for threats. The intrusion detection system serves as the second line of defence against attacks. As in the case of several severe attacks targeting vital infrastructure, such as nuclear enrichment facilities, the first line of defence — firewalls/IPSs — might fail. Network features are taken from the packet after it has been approved by the firewall/IPS and preprocessed to match the input of the learned ML model. After a period of time (one week, one month, or another), these characteristics are recorded in a file and regularly sent to the closest fog node to retrain the intrusion detection model. The characteristics are then used to train the multi-class ML model to restrict the packet's normalcy. The packet will be sent to the broker if it is normal. If not, the IRS module will be enabled when the attack's kind is determined. Fig. 5 depicts the processing steps for the proposed intrusion detection module.

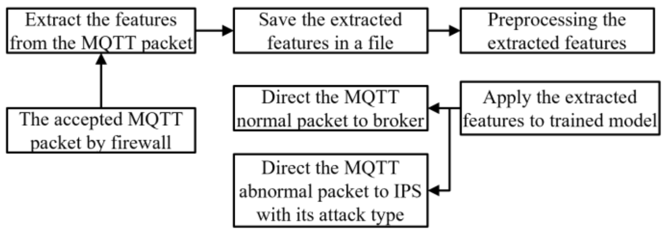
Extract the features from the MQTT packet
Save the extracted features in a file
Preprocessing the extracted features
The accepted MQTT packet by firewall
Direct the MQTT normal packet to broker
Apply the extracted features to trained model
Direct the MQTT abnormal packet to IPS with its attack type

Fig. 5. The processing steps of the proposed Intrusion Detection module

3) IRS Module: Intrusion Response Systems (IRS) can actively block attacks after they have been identified. Malicious or unauthorized behaviors can be effectively addressed by taking the necessary steps to prevent problems from escalating and to return the device to a healthy state. An IRS instantly performs a predetermined set of reactions based on the kind of attack when an IDS module detects an intrusion. An automated method does not necessitate human intervention. As a result, effective countermeasures are required to combat various attack scenarios. The IRS module's actions are as follows:

- Packet Reset Connection: When a publisher or subscriber delivers an anomalous packet, the broker sends a TCP reset packet to cut off the connection.
- Packet Forward: It just forwards the incoming packets without processing them. This is helpful if an attacker bombards the IDS with high-volume traffic to overtax it and start a DoS. Another scenario is that all possible countermeasures increase the burden.
- Packet Drop: It does not send packets to their intended destinations but instead discards packets with datagram or sequence numbers that do not match the protocol's expectations.

By applying the actions outlined in this subsection, the suggested edge node can stop a variety of threats. Other kinds of attacks can be considered for stopping and incorporating into the suggested edge device.

4) Broker Module: The edge device can connect through a wireless connection or a wired link. With the help of IIoT, wireless devices can now be built using inexpensive, small, low-power Wi-Fi-enabled components. The MQTT protocol is utilised in the proposed edge node to transmit and receive messages. MQTT is a lightweight, small-message protocol. The Publish/Subscribe architecture of MQTT is more applicable to IoT applications than the other protocols with a Request/Response model, since it does not require client polling [29]. This results in bandwidth savings and an increase in the device's battery life. The Broker is a crucial component of the MQTT protocol, ensuring that the pub-sub technique operates correctly. To do so, the broker must ensure that customers can accept messages and subscribe to and unsubscribe from devices at any time.

5) The Webserver Module: The web page was designed using a web server module to show the statistics of the collected data for months, weeks, or days as an interactive visualization service. Furthermore, sufficient visibility should be provided to enable the visualization of security threats and to encourage holistic, proactive, and preventive responses. The graphs and statistics exhibited on the created webpage are based on the data file kept at the proposed edge node.

D. Performance Monitoring and Upgrading

The ML model might become less effective at identifying attacks over time. Hence, the intrusion detection module needs to be tested for its performance and updated accordingly. For this purpose, performance metrics like FAR and DR are defined to measure the model's performance. To test effectiveness, new samples are periodically collected from edge nodes and transferred to the fog layer. Because the number of samples may be significant, a representative subset is selected to test the deployed ML model.

For the selected data samples, FAR and DR are calculated and then compared to the same metrics during training (see Stage 2). If the testing FAR is higher than the training FAR or the testing DR is lower than the training DR, it is considered that the model's performance has deteriorated. The ML model is then retrained on the new data and, if necessary, updated hyperparameters to improve its performance. In this regard, two policies are considered for retraining: one uses only new data, and the other merges the latest data with the previous samples used to train the model. After retraining, the model is evaluated, and if it performs well, it is updated to replace the last model deployed in the fog layer.

The intrusion detection module installed at each edge node is either completely updated to a new model or kept up to date with stable performance. In future work, secure edge-fog communications will be established to transfer the collected datasets and trained ML models securely.

In accordance with the provided architecture and implementation, the following section presents the experimental results. It evaluates the efficiency of the proposed IDPRS in detecting and preventing intrusions in an edge-based IIoT environment.

V. EXPERIMENTS AND RESULTS

Based on the above methodology, a series of experiments has been conducted to evaluate detection accuracy, false positive rate, and response time. To implement the proposed edge node, the Raspberry Pi 4B platform can be used because it is a small, inexpensive, and standalone single-board computer.

The suggested fog node server is built on a PC with an Intel Core i7 processor, 16 GB of DDR4 RAM, and an NVIDIA GeForce RTX 2070 with 8 GB of GDDR6 graphics memory. To assess the successful implementation of the system, this section includes three types of assessments that examine the proposed system's performance and demonstrate its suitability for protecting the IIoT network against threats.

A. Performance Analysis of Intrusion Detection Model

The model for detecting attacks combines a variety of techniques. It is used to classify a variety of threats and has been verified. A dataset must be chosen to assess this approach. From Kaggle, a free dataset called MQTTset [30] was downloaded. This dataset was provided by Ivan Vaccari et al. [31]. Six categories of labels in the MQTTset dataset correspond to one type of legal activity (Legitimate) and five types of threats, respectively (DOS, flood, malformed, slowite, and brute force).

In this subsection, the effectiveness of several ML models is evaluated to identify attacks and packet abnormalities precisely. The machine learning algorithms used here include Random Forest (RF), Decision Tree (DT), and Gradient Boosting (GB). The best splitter, the Gini criteria, and the highest depth were applied to the DT until all leaves were genuine, and the RF was then analyzed using two extreme estimators for hyper-parameter tuning. In contrast, the GB is limited to at most 20 estimators. The efficiency of the intrusion detection model built for IIoT networks should be further evaluated through additional experiments to assess the detection method's strengths and weaknesses across various scenarios. Performance indicators, including Precision (P), Accuracy (ACC), Detection Rate (DR), F-score, Recall (R), and Area Under the Curve (AUC), are used to assess the model's performance and identify the most suitable approach for the proposed system.

Where Recall, F1-Score, and Accuracy offer key performance insights, with their equations provided in Equations (1)-(3). Recall evaluates the model’s success in detecting all true positive cases. The F1-Score balances precision and recall, serving as a reliable performance measure for imbalanced datasets, ensuring both accuracy and completeness. Accuracy assesses the overall proportion of correct predictions across all classes, but can be deceptive in imbalanced scenarios [32, 33].

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Number\ of\ Total\ Prediction} \quad (1)$$

$$Recall = Sensitivity = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (2)$$

$$F1score = \frac{2 * Precision * Recall}{Precision + Recall} \quad (3)$$

Also, it can provide a concise mathematical description of the hyperparameter tuning and model update processes applied in the proposed edge–fog IDPRS framework.

For a classifier $f(\cdot \mid h)$with hyperparameters h, the optimal configuration is selected by minimizing the empirical cross-validation loss:

$$h^{*} = \arg \underset{h \in \mathcal{H}}{min} \frac{1}{K} \sum_{k=1}^{K} \mathcal{L}_k(h) \quad (4)$$

Decision Tree and Random Forest models minimize node impurity using Gini impurity or entropy. A split is selected by minimizing the weighted impurity of child nodes. In contrast, Gradient Boosting optimizes a differentiable loss function (e.g., logistic loss for classification), where each new weak learner fits the negative gradient of the loss, and the ensemble is updated using a learning rate. Hyperparameter tuning was performed using grid search with cross-validation.

B. Analysis of Feature Importance

In this section, some of the selected essential features can be leveraged to perform real-time processing and intrusion prediction in a resource-constrained IIoT system. For this purpose, an RF classifier can be used on MQTTset to determine the importance of features in making the correct prediction. The distribution of various attacks across the actual data in the MQTTset dataset during training and testing is shown in Table I. The utilized dataset is split into training and testing sets in a 70-30 ratio. With a split ratio of 0.33, the training set is divided into two subsets: a training set and a validation set, with 67% used for training and 33% for validation.

As shown in Fig. 6, of 33 features, only 18 have high importance and are used by the RF model to make decisions, whereas the remaining features are deemed to have zero importance. The proposed method uses various performance metrics, including precision, accuracy, F1 Score, and recall, across four ML models: top 7, top 10, top 18, and all features, in both training and testing phases. The results presented in Table II indicate that reducing the number of features has no impact on the overall model performance. Further experimentation reveals that the model with the top 10 essential features achieves the same level of accuracy as the model with all features.

TABLE I. THE AVAILABILITY OF DIFFERENT ATTACKS IN THE MQTT SET REAL DATASET AT THE TRAINING AND TESTING STAGES.

Attacks Types	Total	Train 70%	Test 30%
Legitimate	11,915,716	8341001	3574715
DOS	130223	91156	39077
Malformed	14501	10150	4351
Brute Force	10924	7646	3278
Slowite	9202	6441	2761
Flood	613	429	184

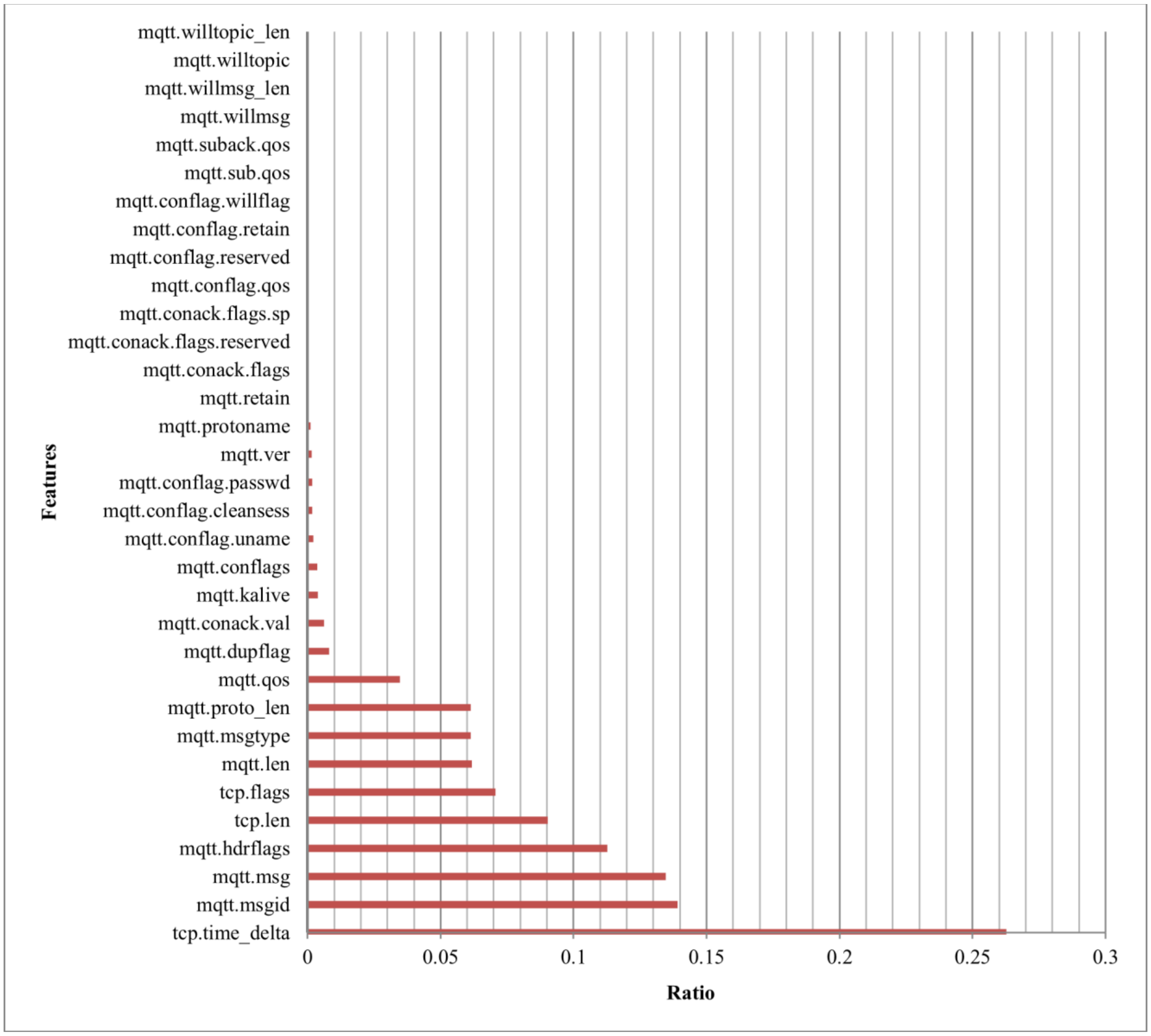

Fig. 6. The extracted features rank ratio at the MQTT set dataset based on a Random Forest model classifier.

TABLE II. COMPARISON OF THE MQTTSET DATASET'S FULL AND REDUCED EXTRACTED FEATURES USING VARIOUS ASSESSMENT MEASURES BASED ON MULTIPLE ML CLASSIFIERS.

ML Classifier	Extracted Features No.	Testing stage				Training stage			
		Precision	F1 Score	Recall	Accuracy	Precision	F1 Score	Recall	Accuracy
RF	7	0.9967	0.99662	0.9968	0.9968	0.9968	0.99668	0.99685	0.9968
	10	0.9967	0.99666	0.9968	0.9968	0.9968	0.99669	0.99687	0.9968
	18	0.9967	0.99666	0.9968	0.9968	0.9968	0.99669	0.99687	0.9968
	33	0.9967	0.99666	0.9968	0.9968	0.9968	0.99669	0.99687	0.9968
GB	33	0.9956	0.9956	0.9959	0.9959	0.9955	0.9954	0.9956	0.9956
	18	0.9955	0.9951	0.9950	0.9950	0.9955	0.9949	0.9948	0.9948
	10	0.9959	0.9956	0.9961	0.9961	0.9957	0.9953	0.9958	0.9958
	7	0.9958	0.9957	0.9959	0.9959	0.9955	0.9954	0.9957	0.9957
DT	7	0.9967	0.9966	0.9968	0.9968	0.9969	0.9967	0.9969	0.9969
	10	0.9967	0.9966	0.9968	0.9968	0.9969	0.9967	0.9969	0.9969
	18	0.9967	0.9966	0.9968	0.9968	0.9969	0.9967	0.9969	0.9969
	33	0.9967	0.9966	0.9968	0.9968	0.9969	0.9967	0.9969	0.9969

Table III also shows that it reduces processing time by less than 30% and achieves better model performance during both training and testing. Top features were selected based on Random Forest importance scores, calculated using the mean decrease in impurity across all trees. Features were ranked, and the top 15 most important were retained for model training. To ensure robustness, it can also be examined whether feature rankings across other models (XGBoost and SVM) are consistent, and features consistently ranked highly across models were prioritized. This combined approach ensures that the selected features are both predictive and generalizable across different classifiers.

C. Intrusion Detection Model Performance based on the Essential Features

The machine learning models for detecting attack packets have been trained and evaluated using the updated feature set (top 10 features). The MQTT set with the new feature set is then submitted to 5-fold cross-validation for each of the three selected classifiers. Figs. 7 and 8 illustrate how the error rate values in the learning and testing stages are converged.

The fivefold cross-validation (see Figs. 7 and 8) indicates that RF and DT achieved the lowest training and testing error rates. Compared to the other techniques, the GB had the fewest fluctuations in the training state. Nevertheless, it performed well across the first four folds of the testing stage, comparable to RF and DT, but poorly in the last fold.

According to Table II, RF and DT achieve higher accuracy, recall, F1 score, and precision than GB. However, compared to GB, RF, and DT are slightly more accurate. Table IV displays the AUC values for RF, GB, and DT.

While the ROC curves of RF, DT, and GB are depicted in Figs. 9, 10, and 11 respectively. According to the area under the ROC curves in Figs. 9, 10, and 11, DT and RF have higher accuracy because they achieve a 99% detection rate in some attacks. All of the areas under the ROC curves for some classes are roughly equal to one. In the case of GB, the area under the ROC curve is nearly 1 for the legitimate class only, and the detection rates for some classes are very low, while other classes have close to 90% detection rates.

TABLE III. THE MQTTSET DATASET'S PERFORMANCE COMPARISON OF FULL AND REDUCED FEATURES FOR VARIOUS ML CLASSIFIERS.

ML Classifier	Extracted Features No.	Training Time(sec)	Testing Time(sec)	Preprocessing time(sec)
RF	7	14.5094	1.04738	47.7604
	10	20.8002	1.2510	48.9029
	18	29.0632	2.1401	65.5551
	33	59.0691	6.1414	96.9495
GB	7	668.616	5.47257	47.7604
	10	721.1012	6.11903	48.9029
	18	991.3863	7.6524	65.5551
	33	1014.20	13.3465	96.9495
DT	7	20.8869	0.64047	47.7604
	10	25.6105	0.81234	48.9029
	18	35.2461	2.0463	65.5551
	33	59.2662	7.7067	96.9495

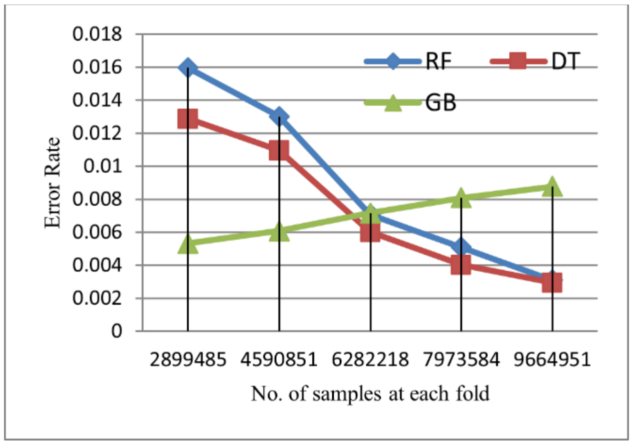

Fig. 7. Training error rate based on different ML classifiers after 5-fold cross-validation by MQTTset

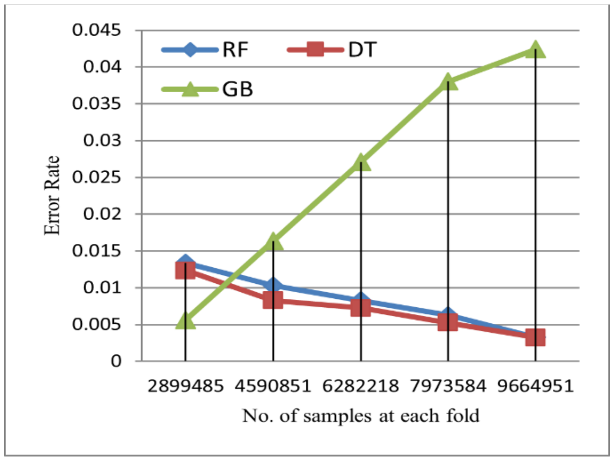

Fig. 8. Testing error rate based on different ML classifiers after 5-fold cross-validation by MQTTset

TABLE IV. THE AUC VALUES FOR DIFFERENT ATTACK TYPES BASED ON VARIOUS ML CLASSIFIERS.

Attacks Type	**RF**	**GB**	**DT**
DOS	0.99	0.45	0.99
Legitimate	0.99	0.98	0.99
Flood	0.96	0.91	0.95
SlowITe	0.97	0.93	0.97
Brute Force	0.94	0.86	0.93
Malformed	0.96	0.58	0.95

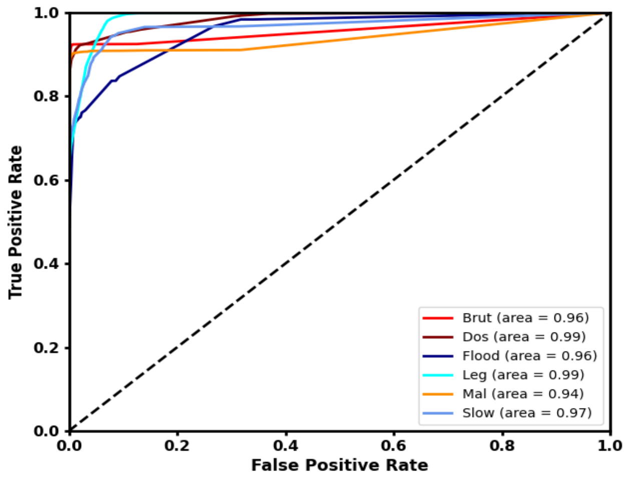

Fig. 9. ROC curve based on real MQTTset dataset for Random Forest

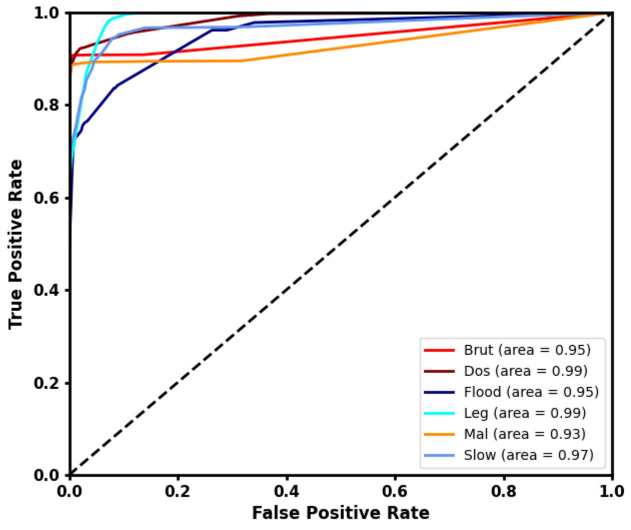

Fig. 10. ROC curve based on real MQTTset dataset for Decision Tree

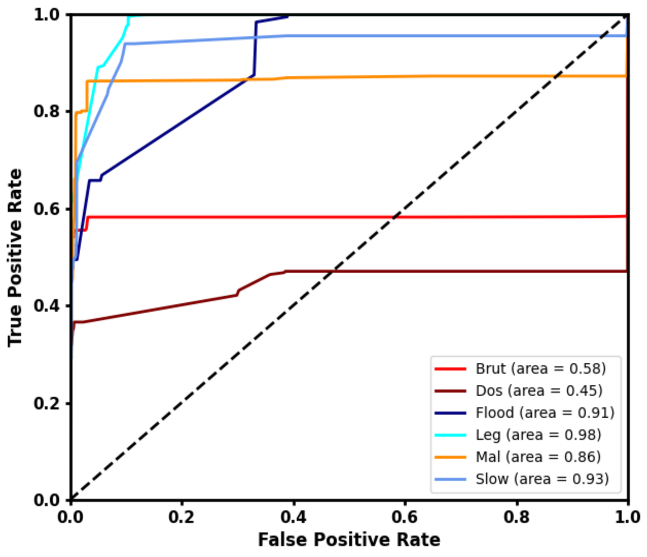

Fig. 11. ROC curve based on real MQTTset dataset for Gradient Boosting

The area under the ROC curve (AUC) values in Table IV indicate that RF and DT have superior accuracy, as they can successfully identify some assaults at a 99% rate. At the same time, some classes have AUC values approximately equal to 1. In the context of GB, the AUC value is almost one for the legitimate class alone, while some classes have extremely low detection rates, while others have rates close to 90%.

Table V shows that RF has the shortest learning time, GB has the longest training time, and DT has the shortest time in the testing state. As the GB's model size is smaller than that of DT and RF, the DT method also runs the fastest on the Raspberry Pi 4. Therefore, the DT method is more suitable for implementation on the Raspberry Pi 4. Finally, it is worth noting that extracting essential features from actual packets for various learned models takes 0.545978 milliseconds on a Raspberry Pi 4.

TABLE V. THE SIGNIFICANT TIMES FOR THE MQTTSET DATASET BASED ON VARIOUS ML CLASSIFIERS.

Classifier	Training Time (sec)	Testing Time (sec)	Prediction Time (msec)	Model Size (MB)
RF	21.47394	2.1002210	1.67393	0.742
GB	972.0119	7.0737800	1.51801	0.137
DT	32.58547	1.6004511	0.66304	0.497

To evaluate the effectiveness of the performance monitoring and upgrading mechanism, key metrics can be measured before and after the fog-level model update as shown in Table VI. For instance, on the MQTTset dataset, an edge node initially achieved **an accuracy of 91.2%, an F1-score of 0.89, a precision of 0.90, and a recall of 0.88**. After detecting performance degradation and triggering the upgrading mechanism, the retrained model improved these metrics to **accuracy = 94.8%, F1-score = 0.93, precision = 0.95, and recall = 0.92**, representing an overall increase of 3–5%. This demonstrates that the upgrading process effectively restores and enhances detection performance, ensuring reliable intrusion detection even under changing traffic conditions.

TABLE VI. THE EFFECTIVENESS OF THE PERFORMANCE MONITORING AND UPGRADING MECHANISM.

Metric	Before Upgrading	After Upgrading	Improvement
Accuracy	91.2%	94.8%	+3.6%
F1-score	0.89	0.93	+0.04
Precision	0.90	0.95	+0.05
Recall	0.88	0.92	+0.04

D. Network Performance

The testbed setup is designed to prove the efficacy of the approach. The Raspberry Pi is considered the proposed edge node for connecting devices to the internet. The MQTT protocol is used to send data between IIoT devices. MQTT is used as the transport protocol for communication in the experiment. The MQTT architecture consists of three nodes: the broker, the publisher, and the subscriber. MQTT broker is a central messaging server that receives data from publishers and subscribers. The publisher is the data sender, and the subscriber is the data receiver [34].

In this experiment, two PCs and a single-board computer (SCB) are used. Two PCs act as both the publishers and subscribers. The SCB is used as the edge device emulator. Mosquitto is installed on the SCB as an open-source MQTT broker for a light-weight MQTT server. Fig. 12 represents the experiment workflow of the testbed.

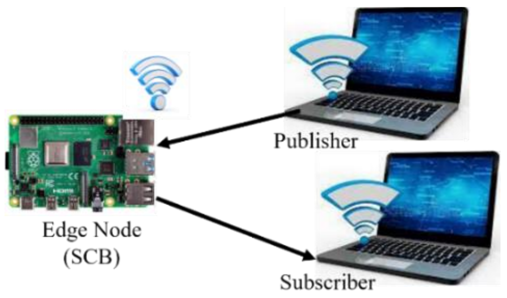

Fig. 12. The workflow of testbed setup for the proposed edge device based on MQTT architecture

The suggested edge node, built on a Raspberry Pi 4, as demonstrated in subsection (IIII.C), will be tested in this subsection to evaluate its performance across various network parameters and to determine whether it is suitable for IIoT networks. The network parameters that are used in this test are as follows:

- Round-trip time (RTT) – it is the duration in which the subscriber receives the ACK for a packet; that is, for every packet sent from a publisher, there is an ACK received (TCP/MQTT communication), which determines the successful delivery of the packet by the subscriber.
- TCP retransmission–which displays all retransmissions in the capture. A few retransmissions are OK; excessive retransmissions are bad. This typically manifests as slow application performance and packet loss for the user.
- Throughput – the number of successfully received packets in a unit time, represented in bps (bits per second).

In the IIoT network, MQTT communication (between the publisher, broker, and subscriber) is tested under two scenarios to determine network parameters (as shown in Figs. 9, 10, 11). In the first scenario, the suggested Intrusion Detection and Response System (IDRS) is deactivated, and in the second, it is triggered. By running a script on a PC that sends messages to the system's topic, the MQTT Broker is tested. The sensor data and the ID of the device that broadcasts the message were included in the script. As a result, the message sent by the sensor node could be received by the broker. The script's job is to send multiple messages simultaneously, each with a unique ID, by taking into account messages from various sensor nodes (publishers) and directing them to different actuator nodes (subscribers).

Fig. 13 demonstrates the round-trip time (RTT) of different numbers of published messages. Fig. 14 illustrates the TCP retransmission of synchronous packets as the number of published messages varies. Fig. 15 represents TCP retransmission packets for different numbers of published messages. Fig. 16 illustrates the throughput of the proposed system as a function of time for various message counts.

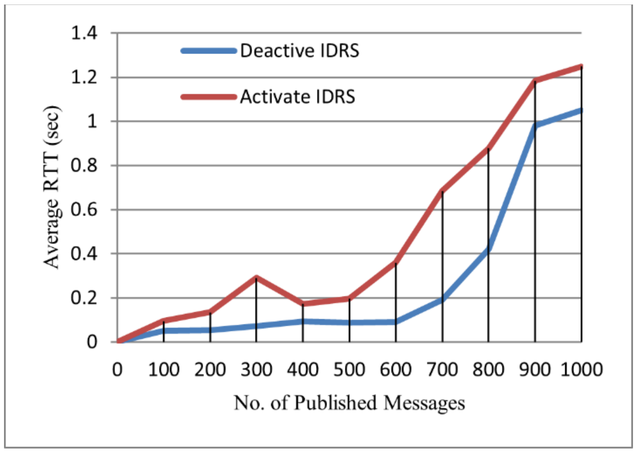

Fig. 13. RTT for varying published message numbers when activating and deactivating the proposed IDRS

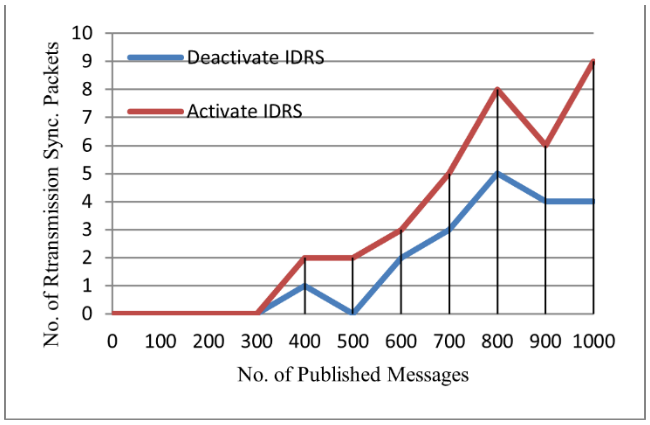

Fig. 14. The TCP retransmission sync. packets for varying published message numbers when activating and deactivating the proposed IDRS

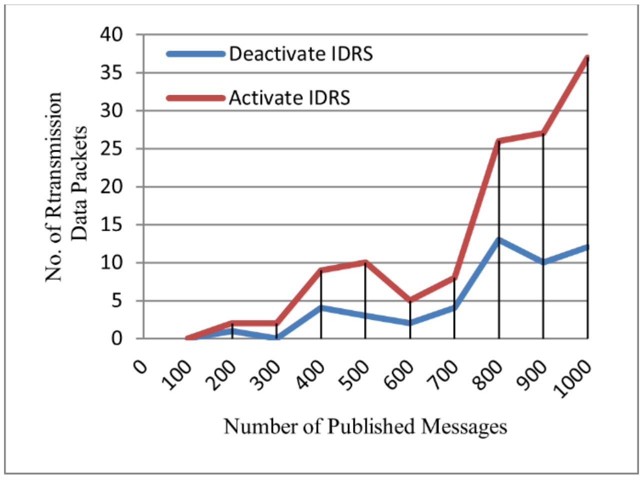

Fig. 15. The TCP retransmission packets for varying published message numbers when activating and deactivating the proposed IDRS

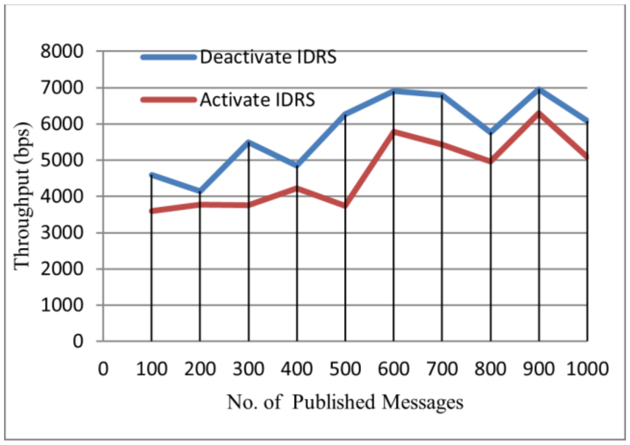

Fig. 16. The suggested system's throughput over time during sending diverse published message numbers when activating and deactivating the proposed IDRS

Fig. 13 shows that the RTT increases as the number of published messages increases, in both scenarios of an activated and a deactivated IDRS. Fig. 14 shows that the retransmitted synchronous packet numbers remain at approximately 5, a low enough number to prevent degradation of network performance, in both scenarios: an activated and deactivated IDRS. Fig. 15 illustrates that the number of retransmitted data packets increases as the number of published messages increases. Fig. 16 shows that the Raspberry Pi 4's throughput is slightly reduced when the IDRS is activated.

Although the implementation was performed on a single Raspberry Pi 4B, the proposed fog–edge IDPRS framework is designed for multi-node deployment. Each edge device can run an independent lightweight IDS instance, while the fog layer synchronizes model updates and detection rules. This distributed setup reduces per-node load and allows parallel processing of network traffic, minimizing detection latency. Based on similar edge-based IDS studies (e.g., RealGuard, Passban), the system is expected to maintain accuracy and responsiveness when scaled, provided communication bandwidth and update frequency are managed appropriately.

E. Performance of Resources Utilization

In IIoT networks, edge nodes are typically battery-operated and have limited processing and storage capabilities. Hence, the power usage, CPU load, and memory occupation of the proposed edge node are measured in the two operational states mentioned earlier. The operating conditions of Raspberry Pi 4 [35] are presented in Fig. 17, where the average consumed current values for different operation modes are displayed. In Low Power Mode, peripherals such as the mouse, keyboard, and screen are switched off, while all peripherals are active in Stable Mode. The regular power usage of the suggested intelligent gateway is obtained by multiplying the average current by the nominal voltage. The nominal voltage of the Raspberry Pi 4 is 5V. The proposed intelligent gateway, with its low complexity and lightweight resource requirements, can be deployed efficiently on a high-performance Raspberry Pi 4, as displayed in Table VII.

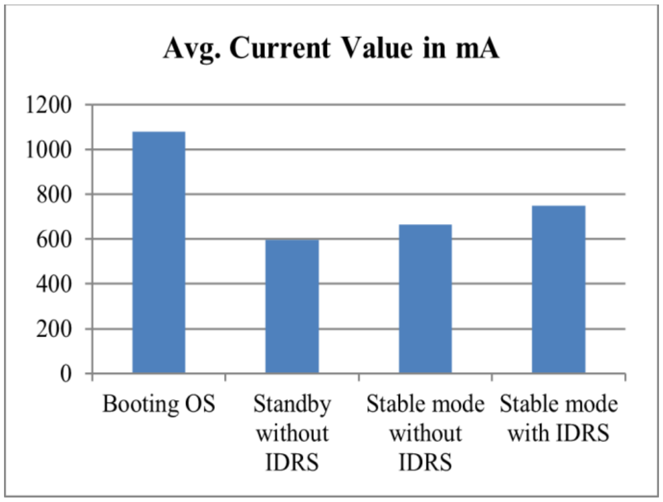

Fig. 17. The average consumed current values for different operation modes on Raspberry Pi 4.

TABLE VII. THE SYSTEM RESOURCES UTILIZATION ON RASBARRY PI4.

System Resources	Ineffective IDRS	Effective IDRS	Overhead Increased
Regular Power usage	3.325 W	3.750 W	12.78%
Typical CPU Process %	5%	16%	11%
Memory usage % (of 8 Gbytes)	1.73% (0.139 Gbyte)	2.52% (0.202 Gbyte)	0.79%

VI. COMPARATIVE STUDY

Despite the remarkable progress reported in the literature, a general, secure, robust, and resilient intrusion detection system for IIoT networks remains an open issue. Table VIII presents the recent works discussed in this section, classifying them by dataset, architectural approach, algorithm, upgrading strategy, validation method, and performance metric.

Although many researchers have proposed operational intrusion detection solutions for IoT/IIoT networks, several issues remain to be addressed. First, many studies validate their approach using a few performance metrics, which do not provide a comprehensive assessment of the method's overall effectiveness. Second, the feasibility of such techniques on resource-constrained devices is not sufficiently demonstrated. Third, to validate the high accuracy of the suggested threat detection solutions, the researchers employed extensive feature engineering (e.g., feature mapping and reduction) to extract optimal characteristics from attack datasets, in conjunction with classification algorithms. Feature engineering involves complex computational procedures and cannot be performed on time-sensitive edge devices with limited processing power. In this context, this paper offers a lightweight intrusion detection, response, and prevention system for edge nodes deployed in IIoT systems. The proposed system is novel in several aspects compared to prior work:

- In industrial IoT settings, IDS, IPS, and IRS functionalities are tightly integrated with a secure architectural design for the edge devices.
- High experimental accuracy is achieved by training machine learning models on high-performance platforms (fog nodes) and running the same on resource-constrained edge devices to lessen the impact of time-intensive feature engineering.
- Performance monitoring and upgrading mechanisms are included to handle emerging attacks.
- A variety of performance evaluation metrics are defined to measure resource utilization, network efficiency, and detection performance.

Although most prior IDS frameworks for IoT and IIoT do not report explicit power-consumption measurements, several studies highlight their computational and resource efficiency, which can be used for qualitative comparison. For example, MEML [15] and Passban IDS [18] report reduced CPU and memory overhead through lightweight machine-learning models optimized for constrained devices. At the same time, Realguard [21] focuses on minimizing processing latency and resource usage on IoT gateways. Similarly, the edge-based frameworks in [17], [23], and [24] emphasize low-complexity detection to support real-time deployment on fog and edge nodes. Compared to the proposed work with these systems, the Raspberry Pi 4B implementation provides measured energy consumption across different workloads, demonstrating that the proposed IDS/IPS/IRS framework achieves comparable lightweight operation while delivering higher detection performance. This comparison highlights that the proposed system not only maintains the low-resource characteristics emphasized in prior work but also explicitly quantifies energy savings, thereby filling a gap in existing research.

Unlike prior edge-based IDS frameworks such as Passban [18], RealGuard [21], and GA_RF [23], which either rely on static models or manual feature selection, the proposed IDPRS framework integrates a real-time performance-monitoring and upgrade mechanism, as shown in Table IX. The system continuously evaluates edge-node detection performance and triggers fog-level retraining when predefined thresholds are exceeded, ensuring adaptive and consistent detection. Additionally, feature selection is automated using Random Forest importance scores, optionally verified across multiple models to retain only the most predictive features. This approach reduces computational load, enhances robustness, and distinguishes the proposed framework from existing edge-based IDS architectures.

TABLE VIII. DETAILS ON RECENT IDS-BASED RESEARCH FOR IOT/IIOT SYSTEMS.

Ref.	**Year**	**Dataset Used**	**Architecture Approach**	**Algorithm Utilized**	**Upgrading Strategy**	**Performance Parameters**	**Prevention and response actions**	**Validation Policy**
[14]	2019	N/A	Centralized	Genetic Programming	×	Detection Efficiency	×	Simulation
[15]	2019	NSL-KDD	Distributed	Neural Network	√	N/A	×	Simulation/Emulation
[16]	2020	DS2OS	Centralized	Random Neural Network	×	Detection Efficiency, Resource Usage	×	Simulation/Emulation
[17]	2020	KDD Cup'99	Centralized	K-Mean	×	Detection Efficiency	×	Simulation
[18]	2020	Private dataset	Distributed	Local Outlier Factor and Isolation Forest	√	Resource Usage, Detection Efficiency	×	Simulation/Emulation
[19]	2021	UNSW-NB15 and NSL-KDD	Centralized	Deep Feedforward Neural Network	×	Detection Efficiency	×	Simulation
[20]	2021	CICIDS 2017, CICIDS 2018, TON IoT	Centralized	Deep Neural Networks	×	Detection Efficiency	×	Simulation
[21]	2022	CICIDS2017	Distributed	Deep Neural Networks	√	Detection Efficiency, Resource Usage	√	Simulation/Emulation
[22]	2023	Generated by author	Distributed	Deep Neural Networks	√	Detection Efficiency	√	Simulation/Emulation
[23]	2024	MQTT-IOT-IDS2020	Distributed	hybrid Genetic Algorithm and Random Forest (GA_RF)	×	Detection Efficiency	×	Simulation/Emulation
Current work	2025	MQTTset	Distributed	Tree-Based Machine Learning	√	Detection Efficiency, Resource Usage, Network Performance	√	Simulation/Emulation

N/A = Not Appropriately Defined

TABLE IX. PERFORMANCE METRICS OF AN EDGE NODE BEFORE AND AFTER APPLYING THE FOG-LEVEL MODEL UPGRADING MECHANISM, DEMONSTRATING IMPROVED DETECTION PERFORMANCE.

Feature / Module	**Passban IDS [18]**	**RealGuard [21]**	**GA_RF [23]**	**Proposed IDPRS Framework**
Edge/IPS/IRS Integration	IDS only	IDS only	IDS only	Full IDS + IPS + IRS integration
Model Upgrading Mechanism	Manual retraining	Static model	Periodic retraining	Real-time monitoring & fog-level automatic updates
Feature Selection	Manual or all features	All available features	Selected based on RF only	Automated RF importance + cross-model consistency check
Adaptive to Edge Performance	No	No	Limited	Yes – triggers upgrades when performance drops
Resource Optimization	Moderate	Moderate	Moderate	Lightweight – optimized for edge deployment
Multi-Node Scalability	Not addressed	Not addressed	Not addressed	Designed for distributed edge–fog nodes

VII. CONCLUSIONS

The ever-increasing number of IIoT devices has led to an emphasis on securing edge devices. In this paper, the authors propose a methodology and architecture for a machine-learning-based intrusion detection mechanism in edge-based IIoT environments. In their design, the IDS, IPS, and IRS functionalities are integrated into a single, compact, and secure edge node. The system aims to detect attacks against the MQTT protocol.

The challenges, such as training the ML model directly on a resource-constrained device like a Raspberry Pi, are overcome by collecting traffic from edge nodes and forwarding it to fog nodes for model training. The trained model is finally installed in the edge device as a threat detection engine. The continuous monitoring of performance and use of an upgrading mechanism to combat emerging attacks are discussed. Experimental results demonstrate that the proposed framework can effectively detect attacks in an MQTT-based IIoT environment while maintaining both resource efficiency and high detection accuracy. In the future, the susceptibility of new kinds of threats to diverse IIoT protocols will be researched. Additionally, if the hardware permits, a higher-complexity model can be used to improve detection performance.

Moreover, to ensure IIoT communication security, protected communication will be established between edge nodes and fog nodes when transferring collected datasets and trained models. This work is limited by its evaluation on a single-device setup, the lack of large-scale traffic testing, and reliance on a single dataset, which restricts claims about generalizability. The upgrading mechanism was conceptually validated but not extensively quantified under real-time load, and energy comparisons with other IDS frameworks were limited. Future work will extend the system to multi-node and federated edge–fog deployments, explore online and continual learning for faster model updates, and assess performance across additional IoT protocols and datasets. Further improvements will include incorporating model compression for lower energy use and investigating adversarial robustness and intelligent, autonomous response strategies.

Although the system is demonstrated on MQTT traffic, the architecture is inherently protocol-agnostic. Only the packet-parsing component is protocol-specific, while the feature extraction, classification pipeline, and response modules remain unchanged. By integrating parsers for other IoT protocols such as CoAP, AMQP, DDS, or HTTP, the same feature vectors can be generated, enabling seamless reuse of the ML models. Additionally, transfer learning or light fine-tuning can be applied to adapt the framework to new datasets. This modular design allows the system to operate across heterogeneous IoT environments and supports future extensions to multi-protocol security monitoring.

References

[1] Z. E. Huma, S. Latif, J. Ahmad, Z. Idrees, A. Ibrar, Z. Zou, *et al.*, "A Hybrid Deep Random Neural Network for Cyberattack Detection in the Industrial Internet of Things," *IEEE Access*, vol. 9, pp. 55595-55605, 2021. https://doi.org/10.1109/2021/3071766.

[2] T. Vaiyapuri, Z. Sbai, H. Alaskar, and N. A. Alaseem, "Deep Learning Approaches for Intrusion Detection in IIoT Networks–Opportunities and Future Directions," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 12, pp. 86-92, 2021. https://doi.org/10.14569/IJACSA.2021.0120411

[3] G. E. I. Selim, E. Hemdan, A. M. Shehata, and N. A. El-Fishawy, "Anomaly events classification and detection system in critical industrial internet of things infrastructure using machine learning algorithms," *Multimedia Tools and Applications*, vol. 80, pp. 12619-12640, 2021. https://doi.org/10.1007/s11042-020-10354-1

[4] Q. Ibrahim and S. Lazim, "An insight review of internet of Things (IoT) protocols, standards, platforms, applications and security issues," *International Journal of Sensors Wireless Communications and Control*, vol. 11, pp. 627-648, 2021. https://doi.org/10.2174/2210327910999201102194157

[5] M. Zolanvari, M. A. Teixeira, and R. Jain, "Effect of imbalanced datasets on security of industrial IoT using machine learning," in *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*, 2018, pp. 112-117. https://doi.org/10.1109/ISI.2018.8587389.

[6] S. Lazim Qaddoori and Q. I. Ali, "An embedded and intelligent anomaly power consumption detection system based on smart metering," *IET Wireless Sensor Systems*, vol. 13, pp. 75-90, 2023. https://doi.org/10.1049/wss2.12054

[7] H. Qiao, J. O. Blech, and H. Chen, "A Machine learning based intrusion detection approach for industrial networks," in *2020 IEEE International Conference on Industrial Technology (ICIT)*, 2020, pp. 265-270. https://doi.org/10.1109/ICIT45562.2020.9067253

[8] H. Alaiz-Moreton, J. Aveleira-Mata, J. Ondicol-Garcia, A. L. Muñoz-Castañeda, I. García, and C. Benavides, "Multiclass classification procedure for detecting attacks on MQTT-IoT protocol," *Complexity*, vol. 2019, pp. 1-11, 2019. https://doi.org/10.1155/2019/6516253

[9] S. Madhawa, P. Balakrishnan, and U. Arumugam, "Roll forward validation based decision tree classification for detecting data integrity attacks in industrial internet of things," *Journal of Intelligent & Fuzzy Systems*, vol. 36, pp. 2355-2366, 2019. https://doi.org/10.3233/JIFS-169946

[10] H. Yao, P. Gao, P. Zhang, J. Wang, C. Jiang, and L. Lu, "Hybrid intrusion detection system for edge-based IIoT relying on machine-learning-aided detection," *IEEE Network*, vol. 33, pp. 75-81, 2019.

[11] S. L. Qaddoori and Q. I. Ali, "An embedded intrusion detection and prevention system for home area networks in advanced metering infrastructure," *IET Information Security*, vol. 17, pp. 315-334, 2023. https://doi.org/10.1049/ise2.12097.

[12] M. A. Khan, M. A. Khan, S. U. Jan, J. Ahmad, S. S. Jamal, A. A. Shah, *et al.*, "A Deep Learning-Based Intrusion Detection System for MQTT Enabled IoT," *Sensors*, vol. 21, pp. 7016-7040, 2021. https://doi.org/10.3390/s21217016

[13] A. Derhab, M. Guerroumi, A. Gumaei, L. Maglaras, M. A. Ferrag, M. Mukherjee, *et al.*, "Blockchain and random subspace learning-based IDS for SDN-enabled industrial IoT security," *Sensors*, vol. 19, pp. 3119-3142, 2019. https://doi.org/10.3390/s19143119.

[14] E. Aydogan, S. Yilmaz, S. Sen, I. Butun, S. Forsström, and M. Gidlund, "A central intrusion detection system for rpl-based industrial internet of things," in *2019 15th IEEE International Workshop on Factory Communication Systems (WFCS)*, 2019, pp. 1-5. https://doi.org/10.1109/WFCS.2019.8758024.

[15] A. Shalaginov, O. Semeniuta, and M. Alazab, "MEML: Resource-aware MQTT-based machine learning for network attacks detection on IoT edge devices," in *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion*, 2019, pp. 123-128. https://doi.org/10.1145/3368235.3368876

[16] S. Latif, Z. Zou, Z. Idrees, and J. Ahmad, "A Novel Attack Detection Scheme for the Industrial Internet of Things Using a Lightweight Random Neural Network," *IEEE Access*, vol. 8, pp. 89337-89350, 2020. https://doi.org/10.1109/2020.2994079.

[17] M. P. Maharani, P. T. Daely, J. M. Lee, and D.-S. Kim, "Attack detection in fog layer for iiot based on machine learning approach," in *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, 2020, pp. 1880-1882. https://doi.org/10.1109/ICTC49870.2020.9289380.

[18] M. Eskandari, Z. H. Janjua, M. Vecchio, and F. Antonelli, "Passban IDS: An intelligent anomaly-based intrusion detection system for IoT edge devices," *IEEE Internet of Things Journal*, vol. 7, pp. 6882-6897, 2020.

[19] J. B. Awotunde, C. Chakraborty, and A. E. Adeniyi, "Intrusion detection in industrial internet of things network-based on deep learning model with rule-based feature selection," *Wireless communications and mobile computing*, vol. 2021, pp. 1-17, 2021. https://doi.org/10.1155/2021/7154587

[20] K. Raja, K. Karthikeyan, B. Abilash, K. Dev, and G. Raja, "Deep Learning Based Attack Detection in IIoT using Two-Level Intrusion Detection System," *Soft computing, Springer, Research Square*, vol. 2021, pp. 1-32, 2021. https://doi.org/10.21203/rs.3.rs-997888/v1.

[21] X.-H. Nguyen, X.-D. Nguyen, H.-H. Huynh, and K.-H. Le, "Realguard: A Lightweight Network Intrusion Detection System for IoT Gateways," *Sensors*, vol. 22, pp. 432-449, 2022. https://doi.org/10.3390/s22020432

[22] A. Awajan, "A novel deep learning-based intrusion detection system for IOT networks," *Computers*, vol. 12, p. 34, 2023.https://doi.org/10.3390/computers12020034

[23] G. T. Francis, A. Souri, and N. İnanç, "A hybrid intrusion detection approach based on message queuing telemetry transport (MQTT) protocol in industrial internet of things," *Transactions on Emerging Telecommunications Technologies*, vol. 35, p. e5030, 2024. https://doi.org/10.1002/ett.5030

[24] T. Zhukabayeva, Z. Ahmad, A. Adamova, N. Karabayev, and A. Abdildayeva, "An Edge-Computing-Based Integrated Framework for Network Traffic Analysis and Intrusion Detection to Enhance Cyber–Physical System Security in Industrial IoT," *Sensors*, vol. 25, p. 2395, 2025. https://doi.org/10.3390/s25082395

[25] L. Zhang, S. Jiang, X. Shen, B. B. Gupta, and Z. Tian, "PWG-IDS: An Intrusion Detection Model for Solving Class Imbalance in IIoT Networks Using Generative Adversarial Networks," *arXiv preprint arXiv:2110.03445*, 2021. https://doi.org/10.48550/arXiv.2110.03445

[26] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications for Centralized and Federated Learning," *IEEE Access*, vol. 10, pp. 40281-40306, 2022. https://doi.org/10.1109/2022/3165809

[27] A. Samy, H. Yu, and H. Zhang, "Fog-based attack detection framework for internet of things using deep learning," *IEEE Access*, vol. 8, pp. 74571-74585, 2020. https://doi.org/10.1109/2020/2988854

[28] I. Butun, M. Almgren, V. Gulisano, and M. Papatriantafilou, "Intrusion Detection in Industrial Networks via Data Streaming," in *Industrial IoT*, ed: Springer, 2020, pp. 213-238.https://doi.org/10.1007/978-3-030-42500-5_6

[29] R. Colelli, S. Panzieri, and F. Pascucci, "Securing connection between IT and OT: the Fog Intrusion Detection System prospective," in *2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4. 0&IoT)*, 2019, pp. 444-448.

[30] MQTTset Dataset [Online]. Available: https: //www.kaggle.com/cnrieiit/mqttset

[31] I. Vaccari, G. Chiola, M. Aiello, M. Mongelli, and E. Cambiaso, "MQTTset, a new dataset for machine learning techniques on MQTT," *Sensors,* vol. 20, pp. 6578-6595, 2020. https://doi.org/10.3390/s20226578

[32] E. Aslan, Y. Ozupak, F. Alpsalaz, and Z. M. Elbarbary, "A Hybrid Machine Learning Approach for Predicting Power Transformer Failures Using Internet of Things Based Monitoring and Explainable Artificial Intelligence," *IEEE Access,* 2025. https://doi.org/ 10.1109/2025/3583773

[33] B. Said, F. I. Bouguenna, Z. Ayyoub, M. Abdullah, Y. Özüpak, R. Bouddou, *et al.*, "Hybrid MPC‐Third‐Order Sliding Mode Control With MRAS for Fault‐Tolerant Speed Regulation of PMSMs Under Sensor Failures," *International Transactions on Electrical Energy Systems,* vol. 2025, p. 5984024, 2025. https://doi.org/10.1155/etep/5984024.

[34] Z. Gao, J. Cao, W. Wang, H. Zhang, and Z. Xu, "Online-Semisupervised Neural Anomaly Detector to Identify MQTT-Based Attacks in Real Time," *Security and Communication Networks,* vol. 2021, pp. 1-11, 2021. https://doi.org/10.1155/2021/4587862

[35] (24/9/2021). *Raspberry Pi Foundation*. Available: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/