Received 17 March 2025, accepted 7 April 2025, date of publication 9 April 2025, date of current version 21 April 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3559325

RESEARCH ARTICLE

Intrusion Detection in IoT Networks Using Dynamic Graph Modeling and Graph-Based Neural Networks

WILLIAM VILLEGAS-CH[1], (Member, IEEE), JAIME GOVEA[1], ALEXANDRA MALDONADO NAVARRO[2], AND PABLO PALACIOS JÁTIVA[3], (Member, IEEE)

[1]Escuela de Ingeniería en Ciberseguridad, FICA, Universidad de Las Américas, Quito 170125, Ecuador
[2]Escuela de Postgrado en Derecho, Universidad de Las Américas, Quito 170125, Ecuador
[3]Escuela de Informática y Telecomunicaciones, Universidad Diego Portales, Santiago 8370191, Chile

Corresponding author: William Villegas-Ch (william.villegas@udla.edu.ec)

This work was supported by the Project Fondo Nacional de Desarrollo Científico y Tecnológico de Chile (FONDECYT) Iniciación under Grant 1365 11240799.

ABSTRACT The rapid expansion of Internet of Things (IoT) networks has significantly increased security vulnerabilities, exposing critical infrastructures to sophisticated cyberattacks. Traditional Intrusion Detection Systems, based mainly on signature matching and predefined rules, present limitations in identifying emerging threats and distributed attacks due to their inability to analyze complex interactions within IoT networks. To address this problem, this study proposes a graph-based intrusion detection model using Graph Neural Networks (GNNs), leveraging a dynamic representation of IoT network traffic. In this model, devices are represented as nodes and communications as weighted edges, integrating features such as communication frequency, transmitted data volume, and protocol type. The proposed method was evaluated using a customized dataset from a simulated IoT network to reflect real-world attack scenarios, including Denial of Service, Spoofing, and Man-in-the-Middle. Experimental results demonstrate that our GNN-based model significantly outperforms traditional machine learning methods, achieving an F1-Score of 0.95 and an AUC-ROC of 0.98, compared to values between 0.84 and 0.91 for Support Vector Machines and Random Forests. Furthermore, the system reduces the false positive rate by 40% compared to signature-based IDS, improving its applicability in operational environments. The model also proved scalable, maintaining an inference time of 2.5 ms per sample on graphs of up to 10,000 nodes, making it viable for real-time deployment. These findings confirm that graph-based anomaly detection is a promising approach for securing large-scale IoT infrastructures, providing increased precision, adaptability, and robustness against emerging cyber threats.

INDEX TERMS Graph neural networks (GNNs), intrusion detection in IoT, graph-based anomaly detection, cybersecurity in IoT networks.

I. INTRODUCTION

The accelerated growth of the Internet of Things (IoT) has driven the interconnection of devices across multiple sectors, from industrial infrastructures and healthcare systems to smart cities [1]. This expansion has generated new cybersecurity challenges, as the diversity of devices, communication protocols, and limited resources in these systems make them attractive targets for cyberattacks. In particular, intrusions into IoT networks can compromise data integrity, availability, and confidentiality, affecting everything from device operation to the security of critical infrastructures. Traditional intrusion detection methods, such as Intrusion Detection Systems (IDS) based on signatures or predefined rules, have proven to be ineffective against emerging threats and distributed attacks, which has motivated the search for more

The associate editor coordinating the review of this manuscript and approving it for publication was Jad Nasreddine.

advanced approaches based on machine learning and artificial intelligence (AI) [2].

Within this environment, graph-based models have emerged as a promising alternative for security analysis in IoT networks. Representing inter-device communication as a dynamic graph, where nodes represent devices and edges model network interactions, allows capturing complex behavioral patterns and detecting anomalies in real time. Previous research has shown that Graph-Based Neural Networks (GNNs) can outperform traditional approaches by identifying subtle intrusion patterns that depend on the network structure and evolution [3]. However, many of these studies have used static graphs or addressed intrusion detection without considering the variability of real-time network traffic, limiting their applicability in dynamic IoT environments.

Despite the advancements in integrating GNNs for intrusion detection in IoT networks, several limitations persist in current approaches. Previous research has demonstrated that GNN-based detection models can outperform traditional methods by capturing complex patterns in network structures [3]. However, most of these studies use static graph representations, which limit their adaptability in dynamic environments [4]. Additionally, recent works have explored microcontroller-based approaches to optimize the use of GNNs in resource-constrained systems, but without explicitly addressing intrusion detection [4]. On the other hand, strategies focused on detecting specific attacks, such as Man-in-the-Middle attacks in IoT, have shown improvements in anomaly detection but still rely on predefined features rather than leveraging structural and contextual learning from graphs [5]. Furthermore, studies like those of Zoican et al. have proposed approaches for traffic prediction in IoT using graph-based neural networks without explicitly addressing network security [4]. These works highlight the need for a model that employs a dynamic representation of network traffic, incorporates adaptive weighting of device interactions, and optimizes feature aggregation to improve the detection of distributed attacks.

Compared to existing GNN-based IDS models, the proposed approach introduces key advancements that enhance its adaptability and detection capabilities in dynamic IoT environments. Previous methods, such as hierarchical adversarial graph-based IDS systems [3], have demonstrated robustness against evasion attacks but remain constrained by their reliance on static graph representations. Other models, including E-GraphSAGE and BS-GAT, have improved intrusion detection accuracy but do not explicitly incorporate dynamic edge weighting mechanisms, limiting their ability to adapt to real-time network fluctuations. In contrast, our model leverages a fully dynamic graph structure where node and edge attributes evolve based on real-time network traffic characteristics. This approach enables more precise differentiation between normal and anomalous behaviors, enhancing the detection of distributed attacks such as DoS, Spoofing, and MiTM. Additionally, unlike previous models primarily focusing on accuracy, our approach optimizes detection performance and computational efficiency, achieving low inference latency while maintaining a high detection rate, making it viable for large-scale IoT deployments.

This study proposes a novel GNN-based approach for intrusion detection in IoT communication infrastructures, modeling the network as a dynamic graph that evolves with real-time network traffic. Unlike previous methods, our proposal incorporates an adaptive weighting of the edges based on metrics such as communication frequency, volume of transmitted data, and protocol type, allowing us to differentiate normal behaviors from potentially malicious interactions [4]. In addition, the model optimizes the representation of the relationships between devices through a structural and contextual feature aggregation mechanism, improving the system's ability to detect distributed attacks, such as Denial of Service (DoS), Spoofing, and Man-in-the-Middle (MiTM) [5].

The methodology followed in this study includes the construction of an experimental environment that simulates intrusion scenarios in an IoT network, where devices have been configured with communication-based on Message Queuing Telemetry Transport (MQTT), Constrained Application Protocol (CoAP), and Zigbee protocols [6]. Network traffic was collected under different conditions, generating a graph representation to train the detection model. Subsequently, the performance of the proposed model was compared with traditional methods and recent machine learning-based approaches, including Support Vector Machines (SVM), Random Forest, and Deep Neural Networks (DNN) [7], [8].

The results show that the GNN-based model significantly outperforms traditional methods' precision and robustness. In tests with distributed attack scenarios, the model achieved an F1-Score of 0.95 and an AUC-ROC of 0.98, outperforming traditional machine learning-based methods by more than 7%. Furthermore, the system showed a 40% reduction in the false positive rate compared to signature-based IDS, making it more viable for operational environments where precision is critical. Regarding computational efficiency, the model proved scalable, maintaining an inference time of 2.5 ms per sample on graphs of up to 10,000 nodes, allowing its real-time deployment without affecting system latency.

Despite these advances, some challenges and limitations were identified that need to be addressed in future research. First, the model training time grows with the graph size, reaching 190 seconds in large-scale networks. This suggests the need for optimization techniques, such as dimensionality reduction or parallelization of training on multiple GPUs. Furthermore, although the model demonstrated good generalization capabilities across various datasets, the variability of IoT architectures in real-world deployments might require fine-tuning the model for each specific environment.

This work represents a significant contribution to IoT network security, demonstrating that using detection models based on dynamic graphs and GNNs offers a substantial improvement in threat identification compared to traditional methods. The model's ability to adapt to variations in network structure and learn from new interactions makes it a viable alternative for deployment in critical IoT infrastructures, such as industrial networks, connected healthcare systems, and smart city environments.

II. LITERATURE REVIEW

In recent years, integrating GNNs into IDSs for IoT environments has gained significant attention. Several studies have demonstrated that these models can capture complex network traffic relationships, enhancing intrusion detection precision compared to traditional approaches [9]. One of the first significant contributions in this field was E-GraphSAGE, proposed by Lo et al. [10]. This model applies GraphSAGE to capture link characteristics and topological relationships between nodes, improving intrusion detection in IoT networks. Evaluated on TON_IoT and BoT-IoT datasets, E-GraphSAGE outperforms traditional ML-based IDSs, particularly in identifying DoS and botnet attacks, achieving an F1-Score of 0.89 and an AUC-ROC of 0.91.

Similarly, Lin et al. introduced E-GRACL, an IoT-focused GNN-based IDS that models communication patterns within IoT networks using graph embeddings and connectivity metrics [11]. Unlike E-GraphSAGE, which primarily focuses on network topology, E-GRACL incorporates node feature aggregation, allowing it better to distinguish DoS, MiTM, and reconnaissance attacks. This method improves attack detection accuracy, achieving an F1-Score of 0.87 and an AUC-ROC of 0.90.

Building on these advancements, Sun et al. proposed GNN-IDS, a graph-based intrusion detection system that structures network traffic as graph data, enabling the detection of complex intrusion patterns that traditional approaches may overlook [12]. Tested on CICIoT2023, this model successfully identifies DDoS and port scanning attacks, demonstrating strong performance with an F1-Score of 0.90 and an AUC-ROC of 0.92.

A key challenge in IoT intrusion detection is the scarcity of labeled data, which affects model training and generalization. Wu et al. addressed this issue with Geometric Graph Alignment (GGA) [13], a domain adaptation technique where each intrusion scenario is modeled as a separate graph. GGA enables cross-domain learning, enhancing intrusion detection across different IoT datasets, particularly for DoS and botnet attacks, achieving an F1-Score of 0.88 and an AUC-ROC of 0.91.

An alternative approach to improving IoT IDS effectiveness is graph attention mechanisms. Wang et al. proposed BS-GAT, which incorporates behavioral similarity in edge weighting, allowing the model to capture network traffic anomalies more effectively [14]. Evaluated on the TON_IoT dataset, BS-GAT performs well in detecting DoS and MiTM attacks, achieving an F1-Score of 0.92 and an AUC-ROC of 0.94.

From a multimodal perspective, Yasaei et al. introduced IoT-GRAF, which integrates sensor and network communication data to improve anomaly detection in IoT security systems [15]. This approach, applied across multiple IoT environments, enhances the model's capability to detect DoS attacks and anomalous traffic, obtaining an F1-Score of 0.93 and an AUC-ROC of 0.95.

Table 1 compares these state-of-the-art GNN-based IDSs, summarizing their model types, datasets, key features, detected attacks, and performance metrics. This comparison highlights how each approach contributes to advancing IoT security. At the same time, it also illustrates the gaps the proposed model aims to address, such as dynamic edge weighting, contextual feature aggregation, and improved adaptability to real-time IoT environments.

These studies highlight the growing importance of GNN-based models in intrusion detection for IoT networks, showcasing their ability to capture network traffic's topological and relational structures. However, existing approaches face scalability, real-time adaptability, and computational efficiency challenges, as most models rely on static graphs or fixed feature sets. Our proposed model addresses these gaps by introducing a dynamic edge weighting mechanism and context-aware feature aggregation, allowing adaptive intrusion detection in real-time IoT environments.

III. MATERIALS AND METHODS

A. DESCRIPTION OF THE TEST ENVIRONMENT

The proposal was evaluated in an experimental environment to reflect the characteristics and challenges of modern IoT networks. This environment considers technological complexity and possible intrusion scenarios, integrating heterogeneous devices and diverse communication technologies.

1) IOT INFRASTRUCTURE

The infrastructure comprises actual and simulated IoT devices representing contemporary IoT networks' functional and architectural diversity. These include temperature sensors, IP cameras for visual monitoring, environmental monitoring devices, actuators for lighting control, and IoT gateways responsible for consolidating information and managing communication with central servers. The devices employ communication protocols such as Wi-Fi, Zigbee, and LoRaWAN, selected to reflect a heterogeneous environment regarding connectivity capabilities and requirements [16].

Each device is configured to operate distributedly, with processing capabilities ranging from resource-limited devices to gateways with higher computing power. The configuration ensures network traffic generation, including real-time data transmission, event responses, and periodic synchronization between devices and servers. The data collected during the infrastructure's operation is monitored and stored for later

TABLE 1. Comparative analysis of GNN-based IDS approaches.

Study	Model Type	Dataset	Features Modeled	Attack Types	Performance (F1-Score, AUC-ROC)
Lo et al. [10]	E-GraphSAGE (GNN)	TON_IoT, BoT-IoT	Network topology, node interactions	DoS, Spoofing, Botnet	0.89, 0.91
Lin et al. [11]	E-GRACL (GNN)	Custom IoT dataset	Graph embeddings, connectivity features	DoS, MiTM, Reconnaissance	0.87, 0.90
Sun et al. [12]	GNN-IDS	CICIoT2023	Edge relationships, protocol types	DoS, Spoofing	0.90, 0.92
Wu et al. [13]	GGA (Graph Alignment)	Multi-domain IoT datasets	Cross-domain graph adaptation	DoS, Botnet	0.88, 0.91
Wang et al. [14]	BS-GAT (Graph Attention)	TON_IoT	Behavioral similarity in edge weighting	DoS, MiTM	0.92, 0.94
Yasaei et al. [15]	IoT-GRAF (Multi-modal GNN)	Multiple IoT environments	Sensor fusion, network graph	DoS, Anomalous traffic	0.93, 0.95
Proposed Model	**Dynamic GNN**	**Custom + Public (TON_IoT)**	**Dynamic edge weighting, contextual feature aggregation**	**DoS, Spoofing, MiTM**	**0.95, 0.98**

analysis using tools such as Wireshark and specific databases for network traffic management.

2) EVALUATION SCENARIOS

The experimental validation considers scenarios that evaluate the network's expected behavior and response under intrusion conditions. Under normal conditions, the infrastructure operates with traffic generated by the usual communication of IoT devices, characterized by sensory data, periodic updates, and control commands. Three attack scenarios—DoS, Spoofing, and MiTM—were simulated to evaluate intrusion detection performance.

DoS attacks were generated using Low Orbit Ion Cannon (LOIC) and Hping3, which allowed the controlled flooding of malicious packets towards specific network nodes, causing service disruption. Spoofing attacks were implemented using Metasploit, employing IP and credential spoofing techniques to gain unauthorized access and compromise the integrity of transmitted data. MiTM attacks were executed using Ettercap and Wireshark, intercepting, modifying, and retransmitting network traffic to introduce anomalies and manipulate communications between devices.

Each scenario was configured with precise attack parameters and execution times to ensure replicability and accuracy in the evaluation. The simulation tools were selected based on their ability to generate realistic attack patterns while allowing fine control over traffic characteristics. This methodology ensures that the evaluation environment closely resembles real-world IoT deployments and provides a robust framework for testing the proposed model's detection capabilities.

3) NETWORK TOPOLOGY

The IoT network is modeled as a dynamic graph in which each node represents a specific system component, such as sensors, cameras, actuators, gateways, and the central server. The edges reflect the communication interactions between these devices, incorporating key characteristics such as the type of protocol used, the frequency of data exchange, and the volume transmitted on each link. This modeling allows for the capture of the hierarchical structure and dynamic relationships between the different elements of the network, ensuring a detailed analysis of traffic and its behavior.

The graph was constructed using data collected from traffic monitoring tools like Wireshark to identify active connections and communication patterns. The processed data was integrated into a dynamic model developed in Python using the NetworkX library, which facilitates the creation and manipulation of complex graphs. The real-time graph's e-update capability reflects connection changes or anomalous behavior that may arise during testing.

Figure 1 presents the block diagram of the IoT network topology, showing the interaction between sensor devices, IP cameras, actuators, and gateways, which act as consolidation points for transmission to the central server. In this diagram, each node is identified by its specific functionality within the system, while the edges illustrate the communication channels and operational dependencies between the components. This diagram provides a detailed perspective of the relationships between the nodes and identifies critical interaction points essential for intrusion assessment and anomaly analysis.

B. DATA ACQUISITION AND PREPROCESSING

Data acquisition and preprocessing ensure the precision and effectiveness of graph-based analysis. This process ranges from the initial collection of network traffic to the data transformation into a structure compatible with dynamic graph modeling. Each stage was carefully designed to ensure the validity and representativeness of the IoT environment under normal and anomalous conditions.

1) DATA COLLECTION

The study's data was obtained from an IoT environment configured to simulate real-world scenarios. Two datasets

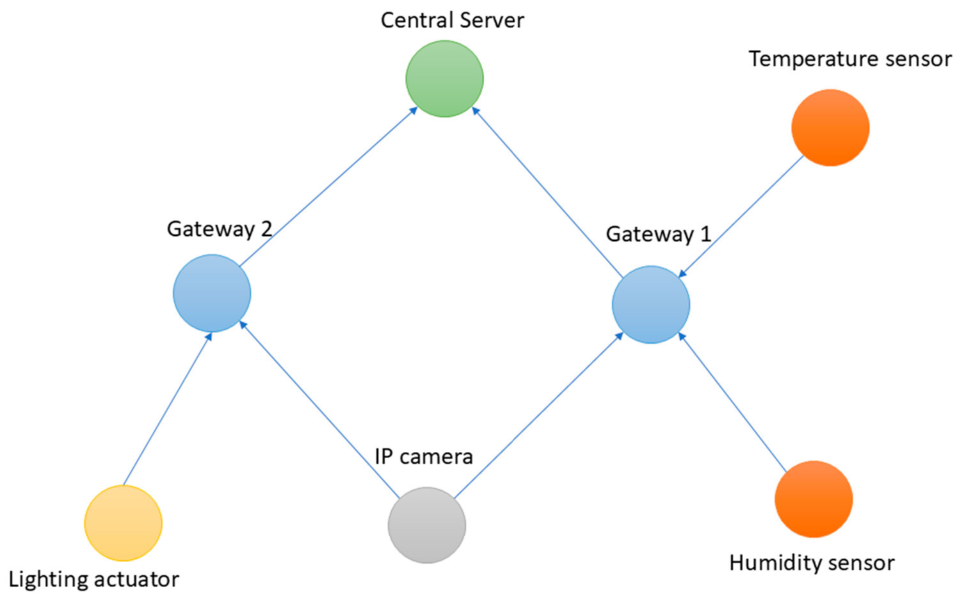


FIGURE 1. Representation of the IoT network topology through devices, gateways, and central server.

were used to ensure broad applicability and robust model evaluation: a publicly available dataset (TON_IoT) and a custom dataset generated from a controlled IoT testbed. Combining these datasets improves generalizability and ensures the review considers standardized benchmarks and real-world network conditions.

The primary sources include real-time packet captures using tools such as Wireshark and tcpdump and logs generated by IoT devices. Packet captures were stored in PCAP files with a total volume of approximately 50 GB, corresponding to 48 hours of continuous monitoring under normal and intrusion conditions [17].

The captured data includes detailed information for each packet, such as source and destination IP addresses, port numbers, packet size in bytes, precise timestamps, and protocol type (e.g., TCP, UDP, MQTT). Additionally, IoT device logs containing battery levels, connection logs, and alerts related to unauthorized access attempts were collected. These logs were stored in JSON format and structured within a MongoDB-based storage system for efficient and scalable access.

To complement this dataset, the publicly available TON_IoT dataset was incorporated. TON_IoT is a widely used benchmark dataset for IoT intrusion detection, containing network traffic collected from various IoT devices under normal and attack conditions. Its inclusion in this study enables a more robust evaluation by validating the model's performance against Reamodel's diverse network patterns. Combining both datasets ensures that the review considers both realistic IoT environments and standardized benchmarks, strengthening the generalizability of the results.

The test environment included 50 IoT devices distributed among sensors, cameras, and actuators, which generated an average of 2,500 packets per minute under normal conditions. In intrusion scenarios, the traffic volume increased by 300% due to the injection of malicious traffic, simulating attacks such as DoS and identity theft.

- Total dataset size: 10 million records.
- Traffic distribution: Normal traffic (65%) vs. attack traffic (35%).
- Attack type distribution:
  - Denial of Service (DoS): 20% of total records.
  - Spoofing: 10%.
  - Man-in-the-Middle (MiTM): 5%.
  - Packet generation rate: 2,500 packets per minute under normal conditions, increasing by 300% during attacks.

From the captured data, features were extracted that describe the behavior of the network and its interactions. The metrics considered include:

- Inter-packet time (IPT) is the interval between the arrival of consecutive packets, used to identify temporal patterns associated with anomalies. This metric was calculated for each communication flow using the timestamps of the captured packets.
- Transmitted data volume: Total sum of bytes transferred between each pair of devices in each period. This metric was calculated by accumulating the size of packets associated with a specific flow.
- Connection frequency is the number of connections between nodes in defined intervals (e.g., 1 minute).

This metric reflects the intensity of interactions between devices.

- Flow direction: The relationship between devices based on source and destination IP addresses, allowing the identification of unidirectional or bidirectional traffic patterns.
- Protocols used: Packets are classified by protocol type (TCP, UDP, MQTT) to analyze traffic distribution and its relationship with possible anomalies.

These features were extracted using Python scripts, libraries such as Scapy for PCAP packet analysis, and Pandas for data structuring and manipulation. More than 10 million feature records were generated, representing the network interactions during testing.

2) PREPROCESSING

The extracted data underwent a multi-stage preprocessing pipeline to ensure its quality and compatibility with graph-based modeling. First, an outlier detection step was performed to identify and mitigate potential anomalies in standard traffic data that could introduce bias into the training process. This involved analyzing statistical distributions of network traffic metrics and applying Z-score filtering to remove extreme values.

Next, data cleaning procedures were applied to remove duplicate and incomplete records, and missing values were handled using imputation techniques based on temporal interpolation and nearest-neighbor estimation. In cases where missing values exceeded a predefined threshold (e.g., 5% of a specific feature set), those records were discarded to avoid introducing noise into the dataset.

A feature engineering process was then applied, where categorical data, such as protocol types, were transformed into numerical representations using one-hot encoding. New derived features, including rolling averages of packet rates and statistical variations in inter-packet times, were also computed to enhance the model's ability to capture temporal dependencies in network traffic.

A min-max normalization process was applied to standardize feature values between 0 and 1, ensuring that no metric disproportionately influenced the model. This step was critical in maintaining the balance of feature contributions, particularly given the wide range of values in IoT network traffic.

Additionally, data aggregation was performed in one-minute intervals to balance precision and computational efficiency. This step helped reduce noise while maintaining the granularity necessary for anomaly detection [18]. The aggregation process considered packet- and session-level features, grouping flows based on IP addresses and connection patterns to preserve structural integrity while optimizing computational overhead.

Finally, the preprocessed dataset was split into training (70%), validation (15%), and testing (15%) subsets, ensuring a balanced distribution of normal and attack traffic across all phases of model evaluation. The preprocessing pipeline was implemented using custom Python scripts with NumPy and Pandas, leveraging sci-kit-learn's preprocessing utilities for normalization and feature selection.

3) GRAPH CONSTRUCTION

The NetworkX library provides advanced graph manipulation and analysis tools and was used to represent network traffic as a dynamic graph [19]. Each node in the graph corresponds to an IoT device, with attributes such as device type, remaining power level, and criticality within the network. Edges represent device interactions and are characterized by transmitted data volume, connection frequency, and average time between packets. Edge weights were calculated using a weighted function that combines the extracted normalized metrics, assigning higher weight to high-frequency interactions or those with large data volumes. This approach ensures that the most relevant connections in the network are highlighted during the analysis.

The graph is dynamically updated at one-minute intervals to reflect changes in network interactions. This process involves removing inactive edges and adding new connections based on data captured in real-time. Figure 2 illustrates this procedure, showing the relationship between data capture, preprocessing, and the final representation as a dynamic graph.

C. GRAPH REPRESENTATION

Modeling the IoT network as a dynamic graph allows for capturing structural relationships and dynamic interaction between devices. This approach facilitates the analysis of network traffic, and the detection of patterns associated with anomalous behavior. The construction of the graph, the characterization of nodes and edges, and its dynamic updating are fundamental components that ensure the model's adaptability to the changing nature of IoT traffic.

1) DEFINITION OF THE GRAPH STRUCTURE

The graph $G = (V, E)$ is an abstract representation of the IoT network, where V corresponds to the set of nodes and E to the set of edges connecting these nodes. Each node $v \in V$ represents an IoT device, while each edge $e_{ij} \in E$ models a direct interaction between two nodes v_i and v_j. These interactions arise from network traffic, specifically from packets transmitted between devices.

The model considers both bidirectional interactions (when data flow occurs in both directions) and unidirectional ones. The adjacency matrix A of the graph is defined as:

$$A_{ij} = \begin{cases} w_{ij}, & \text{if there exists an edge between } v_i \text{ and } v_j, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

where w_{ij} is the weight of the edge, calculated based on the metrics associated with the interaction between v_i and v_j. The initial construction of the graph is done from a preprocessed

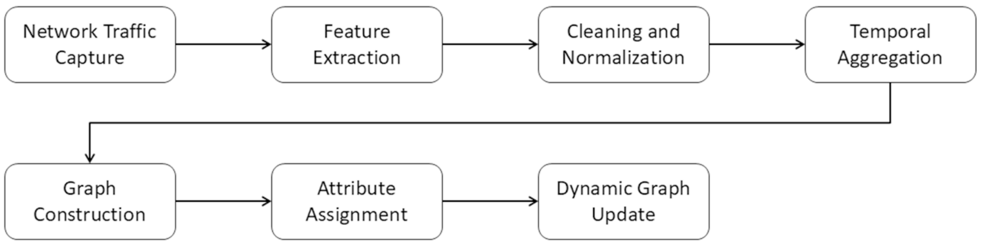


FIGURE 2. **Workflow for IoT graph construction and preprocessing.**

network traffic dataset, ensuring that each node and edge contains information relevant to the analysis.

2) NODE AND EDGE ATTRIBUTES

Nodes $v \in V$ are characterized by a set of attributes $\phi(v)$, designed to capture the intrinsic and operational properties of each IoT device [20]. These attributes allow us to accurately represent the functional differences between the various types of devices present in the network. Among the prominent attributes is the *device type*, which specifies whether a node represents a sensor, an IP camera, an actuator, a gateway, or any other IoT device. This classification is crucial to model network heterogeneity since each type of device has a unique role and contributes differently to network traffic [21].

Another essential attribute is the *processing capacity*, which indicates the computational resources available to each device. This capacity includes the processor's computing power (CPU), available memory, and the maximum supported bandwidth. Devices with limited capabilities, such as basic sensors, present constraints that can influence network dynamics, while more robust nodes, such as gateways, are responsible for processing and routing large volumes of data. In addition, the *power level* reflects the percentage of battery power available in devices that operate without a direct connection to a constant power source, such as sensors deployed in remote environments. This attribute is particularly relevant in IoT networks where power management is critical to the operational sustainability of the system.

Network criticality is another key attribute that reflects a device's relative importance within the network topology. This value is calculated using *degree centrality*, which measures a node's number of direct connections, or *functional dependency*, which assesses how essential a node is to maintain overall network connectivity. For example, a gateway that connects multiple sensors to a central server will have high criticality, as its failure can cause a significant portion of the network to go offline.

The edges $e_{ij} \in E$ that connect the nodes are defined by attributes that describe the dynamic properties of the interactions between devices. The *communication frequency* f_{ij} measures the number of connections established between two nodes during a time interval Δt. This attribute captures temporal patterns in network traffic and allows the identification of persistent or sporadic relationships between nodes. On the other hand, the *volume of transmitted data* d_{ij} is calculated as the total sum of the size of packets sent between two nodes in the same time interval Δt. This metric is essential to evaluate the workload on the links and detect possible congestion or unusual behavior.

Another relevant attribute is the *protocol used in communications*. This attribute classifies interactions according to the type of protocol used, such as TCP, UDP, or MQTT. This allows one to analyze network traffic distribution and detect anomalies associated with changes in protocol use. For example, a sudden increase in UDP traffic on a network designed primarily to use TCP could indicate an attempted attack.

The weight of each edge w_{ij} is calculated using a weighting function that combines these metrics:

$$w_{ij} = \alpha \cdot f_{ij} + \beta \cdot d_{ij}, \tag{2}$$

where α and β are weighting coefficients empirically tuned to balance the relative influence of communication frequency and volume of data transmitted. This weighting allows prioritizing meaningful interactions in the graph analysis, highlighting connections critical to network operation or exhibiting anomalous behavior.

This graph representation and node and edge attributes provide a solid foundation for dynamically analyzing IoT network behavior, identifying emerging patterns, and detecting real-time anomalies. The features capture the network's static structure and dynamic changes, ensuring the model is representative and robust.

3) DYNAMIC GRAPH UPDATE

Since IoT networks are highly dynamic, the graph is updated at regular intervals Δt to reflect changes in network interactions [22]. This process involves adding, modifying, and removing nodes and edges based on the observed traffic. The dynamic update follows specific rules that ensure the graph represents real-time network activity.

The choice of Δt is critical for balancing detection accuracy and computational efficiency. A small Δt (e.g., 0.5s) allows for finer granularity in detecting short-lived anomalies but increases processing overhead. Conversely, a more extensive Δt (e.g., 5s) reduces computational costs but may miss transient attacks. Empirical evaluations demonstrated that $\Delta t = 1s$ provides the best trade-off, achieving a high F1-Score (0.95) while maintaining an inference time of 2.5 ms per sample. This choice ensures real-time adaptation without excessive resource consumption.

Node addition occurs when a new device v_k is detected in the network traffic. In such cases, the device is incorporated into the set V along with its initial interactions, ensuring that new network participants are correctly integrated into the graph structure. This mechanism allows the model to adapt dynamically to changes in the IoT network topology.

Weight modification is applied to existing edges e_{ij} by recalculating their weights w_{ij} based on the most recent traffic data. This continuous adjustment ensures that the model accurately reflects variations in communication frequency and data volume, allowing for the detection of changes in normal and anomalous network behavior.

Furthermore, the frequency of edge updates is directly influenced by Δt. A shorter Δt results in more frequent updates, which enhances detection sensitivity but increases processing load. Conversely, a longer Δt smooths out minor fluctuations but may delay anomaly detection. The selection of $\Delta t = 1s$ was determined through experimental testing, where performance metrics (accuracy, recall, and F1-score) were optimized.

Edge removal takes place when an edge e_{ij} remains inactive for a period of $n \cdot \Delta t$. If no interactions occur between two nodes during this time window, the edge is considered obsolete and is removed from the graph. This strategy prevents outdated connections from distorting the network representation, ensuring that the model focuses on relevant interactions. These update mechanisms enable the graph-based model to maintain an accurate and up-to-date representation of IoT network activity, providing a robust framework for real-time intrusion detection.

The following pseudocode details the dynamic update process:

Algorithm 1 Graph Update Process for IoT Network

```
Require: Captured network traffic $T$, Current graph $G = (V, E)$, Update interval $\Delta t$
Ensure: Updated graph $G'$
 1: Initialize new structures $N_{\text{nodes}}$ and $E_{\text{updated}}$
 2: for each packet $p \in T$ do
 3:     Identify source node $v_i$ and destination node $v_j$
 4:     if $v_i \notin V$ or $v_j \notin V$ then
 5:         Add nodes $v_i$ and $v_j$ to $N_{\text{nodes}}$
 6:     end if
 7:     if $e_{ij} \notin E$ then
 8:         Create edge $e_{ij}$ with initial attributes
 9:     end if
10:     Update attributes of $e_{ij}$ in $E_{\text{updated}}$
11: end for
12: for each edge $e_{ij} \in E$ do
13:     if $e_{ij} \notin E_{\text{updated}}$ then
14:         Remove edge $e_{ij}$ from $E$
15:     end if
16: end for
17: Update $G$ with $N_{\text{nodes}}$ and $E_{\text{updated}}$
18: return $G'$
```

Mathematically, the evolution of the graph can be described as:

$$G(t + \Delta t) = V(t) \cup V_{\text{new}}, E(t) \cup E_{\text{new}} \setminus E_{\text{inactive}}. \quad (3)$$

This formula ensures that the selected Δt maintains real-time adaptability while optimizing detection accuracy and processing efficiency.

D. INTRUSION MODELING AND DETECTION

Graph-based IoT intrusion detection and modeling use machine learning and graph analysis techniques. This approach leverages graphs' ability to create complex and dynamic relationships between nodes, allowing the detection of anomalous patterns associated with malicious activities. The proposed model integrates concepts from GNNs and propagation analysis to identify anomalies accurately [23].

1) PROPOSED MODEL

The detection model is built based on a dynamic graph $G_t = (V, E)$, where nodes V represent IoT devices and edges E model their interactions. Each node $v_i \in V$ is associated with a set of features $\phi(v_i)$, while each edge $e_{ij} \in E$ has attributes $\phi(e_{ij})$. The model aims to assign a probability y_i to each node, representing its normality or abnormality level in the network context.

The objective function of the model can be formalized as:

$$\mathcal{L} = -\frac{1}{|V|}\sum_{i=1}^{|V|}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right], \quad (4)$$

where y_i is the actual node label (0 for regular nodes and 1 for anomalous nodes), and $\hat{y}_i$ is the probability predicted by the model.

The model uses graph learning techniques, particularly GNNs, to process the structure and attributes of the graph. GNNs combine information from nodes and their neighbors to learn robust representations. The essential operation of a GNN is defined by:

$$h_i^{(k)} = \sigma\left(\sum_{j\in\mathcal{N}(i)} \frac{1}{c_{ij}} W^{(k)} h_j^{(k-1)} + W_0^{(k)} h_i^{(k-1)}\right), \quad (5)$$

where:

- $h_i^{(k)}$ is the representation of node i in layer k,
- $\mathcal{N}(i)$ is the set of neighbors of node i,
- c_{ij} is a normalization coefficient based on the degree of the node,
- $W^{(k)}$ and $W_0^{(k)}$ are trainable weight matrices,
- σ is a nonlinear activation function, such as ReLU.

2) MODEL ARCHITECTURE

The proposed model's architecture is based on a GCN, specifically designed to capture local and global patterns in the structure of the dynamic graph generated from IoT network traffic. This leverages GCNs' ability to combine structural information and attributes of nodes and edges, allowing a deep analysis of the relationships between devices and possible anomalies in network behavior [24].

At the input layer, the model receives as inputs the initial attributes associated with each node $\phi(v_i)$ and each edge $\phi(e_{ij})$, which include characteristics such as device type, communication frequency, data volume, and protocol used. These attributes are normalized to ensure uniform scaling across metrics and projected to a d-dimensional vector space using a linear transformation defined by:

$$h_i^{(0)} = W_{\text{input}}\phi(v_i) + b_{\text{input}}, \tag{6}$$

where W_{input} is the weight matrix and b_{input} is the bias vector. This initial layer prepares the data for processing in the following stages of the model.

The dimensions of the convolutional layers are as follows:

- In the first layer, the input vectors have dimension $d = 32$, and the output has dimension $d = 64$, increasing the model's ability to capture more complex features.
- In the second layer, the input dimension is $d = 64$, and the output is $d = 128$, allowing for capturing broader patterns in the graph structure.
- In the third layer, the input dimension is reduced to $d = 128$, while the output returns to $d = 64$, consolidating the learned representations.

To prevent overfitting, each convolutional layer is equipped with a dropout mechanism with a probability of $p = 0.5$. This mechanism disconnects connections during training, promoting generalization and robustness to new data.

After the convolutional layers, the model incorporates a global aggregation layer that synthesizes the information from all nodes in the graph into a compact representation. This step is performed using a pooling operator, which can be a sum or an average of the node representations:

$$h_{\text{global}} = \text{Pooling}\left(\{h_i^{(k)} : i \in V\}\right). \tag{7}$$

The global representation of the graph captures high-level patterns that reflect the general state of the IoT network and is used as input for the next stage. Finally, the classification layer maps the learned representations to a binary probability, indicating each node's normality or abnormality level. This layer applies a sigmoid function defined as:

$$\hat{y}_i = \sigma(W_{\text{out}}h_i + b_{\text{out}}), \tag{8}$$

where W_{out} and b_{out} are the trainable parameters of the layer. The output $\hat{y}_i$ is a probability in the range [0, 1], where values close to 1 indicate potentially anomalous nodes.

The model was configured with the following hyperparameters:

- **Initial learning rate:** $\eta = 0.001$, dynamically adjusted during training.
- **Optimizer:** Adam was selected to efficiently handle large parameter updates.
- **L2 regularization:** Coefficient of $\lambda = 10^{-4}$ applied to avoid overfitting.
- **Batch size:** 128 nodes per iteration to balance computational efficiency and quality of the estimated gradient.
- **Dropout:** $p = 0.5$, to increase generalization.

This architecture allows efficient processing of dynamic graphs generated in real-time, capturing both the structure and temporal patterns of the network, resulting in high precision in intrusion detection.

3) MODEL TRAINING

The model training is done using a dataset consisting of 10 million network traffic records, divided as follows:

- 70% for training: Data used to tune model parameters.
- 15% for validation: Data used to tune hyperparameters and prevent overfitting.
- 15% for testing: Data reserved to evaluate the final performance of the model.

The data were labeled semi-supervised, combining manual labeling and heuristic-based automated anomaly detection. During training, the model optimized the cross-binary loss function $\mathcal{L}$ using the Adam optimizer with a dynamically adjusted learning rate by scaling back in case of validation error plateau.

Training was conducted for 200 epochs, with early stopping triggered if no improvement in validation loss was observed after 20 consecutive epochs. The total training time on a server with an NVIDIA Tesla V100 GPU was approximately 4 hours. Model evaluation was performed using standard metrics for anomaly detection, including precision, recall, F1-score, and area under the ROC curve (AUC-ROC).

E. METRICS AND PERFORMANCE EVALUATION

The model's performance was evaluated using a set of widely accepted intrusion detection metrics, ensuring an objective and rigorous comparison. In addition, controlled experiments were designed to analyze the model's effectiveness under normal conditions and simulated intrusion scenarios, allowing us to evaluate its ability to identify anomalous behaviors in heterogeneous IoT environments. Furthermore, a quantitative comparison was made with traditional methods and approaches from recent literature.

1) EVALUATION METRICS

The model's performance was evaluated using widely accepted binary classification metrics, which allow for measuring different aspects of performance in detecting intrusions in IoT networks. These metrics include accuracy, precision, recall, F1-Score, the area under the ROC curve (AUC-ROC), and false alarm rate (FAR) [26]. These metrics were directly applied to analyze the results and evaluate the model's capacity in real and simulated scenarios.

Accuracy was included as a global metric to measure the proportion of correctly classified instances, considering both true positives and TN. This metric provides an overall assessment of the model's effectiveness in classifying normal and anomalous traffic. While Accuracy is widely used in classification tasks, it was analyzed in conjunction with other metrics, such as Precision and Recall, given that high-class imbalance in intrusion detection scenarios may impact its interpretability.

Precision was used to evaluate the precision of the model's classification in correctly identifying nodes marked as intruders and minimizing false alarms. This value is especially relevant in IoT systems where misclassification can lead to disruptions or inefficiencies in standard processes. Recall allowed the model to be analyzed for its ability to detect as many real intrusions as possible, even in complex scenarios with noisy traffic or dynamic behavior. Both metrics were complemented by the F1-Score, which provided a balanced performance indicator by combining precision and recall, especially useful in contexts where intrusions are rare events and classes are unbalanced.

In addition, the AUC-ROC was used to evaluate the model's ability to distinguish between normal and anomalous nodes at different decision thresholds. This allowed the model's robustness to be analyzed in scenarios with variations in network traffic behavior. Finally, the FAR was used to mismeasure the percentage of normal nodes classified as intrusions, a critical aspect of minimizing the generation of unnecessary alerts in IoT systems.

These metrics were calculated globally for the entire dataset and segmented for each type of attack, including DoS, spoofing, and MiTM. This granular evaluation allowed us to identify how the model responds to different intrusion patterns and how its capabilities vary depending on the type of threat.

2) TESTS PERFORMED

The model was evaluated in two tests: normal conditions and intrusion scenarios. In the usual conditions, network traffic was generated by IoT devices operating normally without malicious behavior. This scenario allowed us to evaluate the model's ability to maintain a low false alarm rate and avoid incorrectly classifying legitimate nodes as intruders.

In the intrusion scenarios, different attacks were simulated, including DoS attacks, spoofing, and MiTM attacks. Each type of attack was configured to affect other parts of the network, from individual devices to groups of interconnected nodes. For example, in DoS attacks, malicious traffic saturated gateway nodes, causing network congestion, while in spoofing attacks, IP addresses were spoofed to insert anomalous traffic.

3) COMPARISON WITH OTHER METHODS

To validate the effectiveness of the proposed approach, a quantitative and methodologically structured comparison was performed with other traditional and contemporary models widely accepted in the literature. This stage does not focus on the final results but on detailing the procedure and tools employed to ensure a fair and objective comparison, considering the diversity of approaches used in detecting intrusions in IoT networks.

The comparison methodology includes the selection of three categories of methods: systems based on signatures and predefined rules, traditional machine learning models, and advanced approaches reported in recent studies. Each category was explicitly configured to work with the same data used in training and evaluating the proposed model, ensuring a homogeneous experimental environment.

In the category of systems based on signatures and predefined rules, a traditional IDS such as Snort was configured with updated regulations for detecting common attacks in IoT networks. These rules include specific traffic patterns, such as packet header anomalies and known malicious behavior signatures. Network traffic data was processed through the IDS to generate binary labels, which were then compared to the ground truth labels in the dataset. This method was selected due to its wide use in traditional detection systems, allowing an initial baseline to be established.

For traditional machine learning models, algorithms such as SVM and Random Forest (RF) were implemented. These models were trained using features extracted directly from network traffic, such as inter-packet time, communication frequency, and volume of data transmitted [27]. Each model was configured with the following parameters tuned during a cross-validation process:

- SVM: Radial kernel (RBF), regularization coefficient $C = 1.0$ and kernel parameter $\gamma = 0.1$.
- Random Forest: Number of trees $n = 100$, maximum depth $d = 10$ and splitting criterion based on entropy.

These models were evaluated regarding precision, recall, F1-Score, and AUC-ROC using the same training, validation, and testing set as the proposed model.

In the category of advanced approaches, recent models based on deep learning, specifically Recurrent Neural Networks (RNN) and DNN, were selected. These models were trained using a tabular representation of previously processed data through normalization and encoding techniques. The parameters of these models include:

- RNN: Two recurrent layers with 64 units each, ReLU activation function and 50% dropout.
- DNN: Three dense layers with 128, 64 and 32 neurons respectively, ReLU activation function and L2 regularization with coefficient $\lambda = 10^{-4}$.

In contrast, based on GCN, the proposed model employs an end-to-end approach that combines a dynamic graph structure with node and edge attributes. The model implementation was performed using the PyTorch Geometric library, which provides optimized tools for handling complex graphs. To ensure a fair comparison, the input data and the preprocessing procedure were identical across all methods, allowing only the differences attributable to the algorithms used to be evaluated.

Additionally, to address the reviewer's concerns, this study acknowledges the relevance of GNN-based IDS solutions such as E-GraphSAGE [10], BS-GAT [14], and GGA [13]. These models focus on graph-based intrusion detection, leveraging different structural and learning mechanisms. However, direct experimental comparisons with these models were not performed due to the unavailability of standardized implementations and differences in dataset accessibility. Nonetheless, as previously discussed, the key differences between these models and our proposed approach are highlighted in the discussion section to provide a conceptual comparison.

The comparison procedure included evaluating each model using the same training, validation, and test data sets, split into 70%, 15%, and 15%, respectively. Additionally, consistent evaluation metrics (precision, recall, F1-Score, AUC-ROC, and FAR) were employed to ensure a uniform assessment and enable a comparative analysis of the methods. All methods were implemented and configured in a controlled environment using an NVIDIA Tesla V100 GPU server and 32 GB of memory. Each model was trained independently, and processing times were recorded to assess the scalability and efficiency of each approach.

F. IMPLEMENTATION AND CONFIGURATION

The implementation and configuration of the graph-based detection system were carried out using carefully selected tools and platforms to maximize performance and ensure the reproducibility of the work.

1) TOOLS AND TECHNOLOGIES USED

The system was developed using Python as the primary programming language due to its versatility and wide adoption in machine learning and data manipulation. Among the libraries used, PyTorch Geometric was the basis for implementing the GCN model, as it provides a set of tools optimized for handling dynamic graphs and large volumes of data. This library allowed the efficient implementation of complex operations, such as convolutions on graphs and dynamic updates of node and edge attributes.

NetworkX was used to construct and initially manipulate graphs. It facilitated the representation of the interactions between nodes and edges and the calculation of basic topological metrics necessary to characterize the IoT network. Additionally, Scikit-learn was used in data preprocessing tasks to evaluate metrics for comparative models, including traditional algorithms such as SVM and Random Forests.

Network traffic capture was carried out using specialized tools such as Wireshark and tcpdump, which allowed recording data in PCAP format with detailed information on the transmitted packets, including IP addresses, packet sizes, and timestamps [28]. These captures were processed using Pandas and NumPy, fundamental libraries for tabular data manipulation and feature normalization. Finally, Matplotlib and Seaborn were used to visualize results and generate graphs that complement the analysis of the model's performance.

2) TESTING PLATFORM

System testing was performed on a high-performance infrastructure designed to handle the computational requirements of processing large-scale dynamic graphs and analyzing millions of network traffic records. The platform included a server equipped with a 20-core 2.1 GHz Intel Xeon Gold 6230 processor and 128 GB of DDR4 RAM, which is essential for handling memory-intensive operations during graph construction and updating.

An NVIDIA Tesla V100 GPU with 32 GB of HBM2 memory, optimized for deep learning tasks, was used to accelerate calculations related to model training. A 4 TB NVMe SSD provided storage, guaranteeing sufficient reading and write speeds to process large volumes of network traffic data in real-time.

The software environment included the Ubuntu Server 20.04 LTS operating system, chosen for its stability and compatibility with machine learning libraries and network tools. CUDA 11.7 and cuDNN 8.0 drivers were installed to enable GPU processing with PyTorch Geometric. To ensure consistency in environment configurations, Docker was used, which encapsulated all necessary dependencies into a reproducible image.

As for the network used, the simulated IoT infrastructure included 50 distributed devices, which generated traffic using protocols such as MQTT, TCP, and UDP. The network configuration was designed to reflect realistic operation scenarios, with a bandwidth of 10 Gbps to ensure efficient data transfer between devices.

IV. RESULTS

A. GRAPH REPRESENTATION

The graph structure analysis was based on data collected in different scenarios, including standard operating conditions and simulated intrusion situations. By characterizing the nodes and their interactions, key topological metrics were established to assess the impact of attacks on the IoT network. Structural properties such as graph density, number of connected components, and node degree distribution were calculated, providing a detailed view of the network behavior in each scenario.

Table 2 presents the distribution of nodes and edges under different scenarios, comparing normal conditions with those in which DoS, Spoofing, MiTM attacks, and Blended attacks were injected.

The network contains 100 nodes with 450 edges in normal conditions, reflecting an interconnected but stable structure. However, by introducing DoS attacks, the number of nodes increases to 120, while the number of edges increases to 600, indicating anomalous traffic that generates more excellent

TABLE 2. **Distribution of nodes and edges in different scenarios.**

Scenario	Nodes	Edges	Density	Clustering Coefficient
Normal	100	450	0.09	0.23
Intrusion (DoS)	120	600	0.10	0.21
Intrusion (Spoofing)	110	580	0.11	0.19
Intrusion (MiTM)	115	610	0.12	0.18
Combined Attack	130	700	0.14	0.15

connectivity between devices. A similar pattern is observed in the Spoofing and MiTM attacks, where the number of nodes and edges varies slightly, although a growth trend is maintained concerning the normal state. The most extreme situation occurs in the combined attack, where the network reaches 130 nodes and 700 edges, which shows saturation due to multiple simultaneous intrusions.

These results indicate that the attacks directly affect the graph's structure, increasing connectivity anomalously. The injection of malicious traffic generates artificial interactions between devices, which increases the number of edges and, therefore, the density of the graph. This behavior reinforces the hypothesis that network topology can be used as an indicator for anomaly detection in IoT environments.

Table 3 presents a detailed analysis of the topological metrics calculated in each scenario, including degree centrality, number of connected components, graph radius, and graph diameter.

TABLE 3. **Key topological metrics in different scenarios.**

A	**B**	**C**	**D**	**E**
Normal	0.12 ± 0.03	1	5	12
Intrusion (DoS)	0.18 ± 0.05	2	6	15
Intrusion (Spoofing)	0.16 ± 0.04	2	6	14
Intrusion (MiTM)	0.19 ± 0.06	3	7	16
Combined Attack	0.21 ± 0.07	4	8	18

A: Scenario; **B**: Degree Centrality (Average ± Deviation); **C**: Number of Related Components; **D**: Radius of the Graph; **E**: Diameter of the Graph.

Under normal conditions, the average degree of centrality remains at 0.12 ± 0.03, reflecting a balanced distribution of connections between devices. However, in intrusion scenarios, these metric experiences a progressive increase, peaking in the combined attack with a value of 0.21 ± 0.07, suggesting the emergence of nodes with significantly higher connectivity. Similarly, the number of connected components increases in attack scenarios, going from 1 in normal conditions to 4 in the combined attack. This indicates that the network tends to fragment as malicious activity increases.

The radius and diameter of the graph also reflect structural changes in the presence of attacks. Under normal conditions, communication between devices remains efficient, with a radius of five and a diameter of 12. However, in intrusion scenarios, these values gradually increase, reaching 8 and 18, respectively, in the combined attack. This increase suggests that the network becomes more dispersed and less efficient regarding global connectivity, affecting data transmission in the IoT system.

Figure 3 presents a set of graphs illustrating the evolution and characterization of the graph structure in the scenarios analyzed.

Graph (A) shows the evolution of the number of nodes and edges over time. Under normal conditions, both metrics remain relatively stable, with minor fluctuations attributable to natural network traffic. However, an abrupt increase in nodes and edges is evident when attacks are introduced. During the combined attack, the growth is exponential, indicating a significant alteration in the network structure. This trend suggests that intrusion detection can benefit from monitoring the graph's temporal evolution since abrupt connectivity changes can strongly indicate malicious activity. Graph (B) compares the key topological metrics in each scenario. The density of the graph experiences a moderate increase in individual attacks and a more pronounced growth in the combined attacks. Similarly, the number of connected components increases progressively as the network is affected by different attacks, suggesting that network segmentation is a direct consequence of malicious activity. On the other hand, the clustering coefficient decreases as the attacks affect the network, indicating a lower cohesion in the network structure due to the anomalous distribution of traffic.

Graph (C) shows the degree distribution of the nodes in the different scenarios. Under normal conditions, most nodes have relatively low degrees, with a distribution concentrated in lower values. However, when analyzing the scenarios with intrusion, a more excellent dispersion in the degree distribution is identified, with the appearance of high-degree nodes that act as traffic concentrators. This behavior is particularly evident in the combined attack, where the number of nodes with high connections is significantly higher.

The analysis of these topological metrics shows that attacks generate structural modifications in the network that can be detected from the graph representation. The combination of monitoring and graph modeling techniques offers an effective tool for identifying anomalous patterns in IoT environments, allowing the early detection of intrusions before they impact the network's regular operation.

B. INTRUSION MODELING AND DETECTION

The proposed model was evaluated using different performance metrics, considering both everyday operation scenarios and conditions in which the network was affected by specific attacks. To analyze the model's effectiveness, classification tests were performed on normal and anomalous nodes, evaluating the separation capacity in the latent space, the model's performance in terms of precision, recall, F1-Score, and AUC-ROC, and the variability of performance against different types of attacks.

Table 4 presents a quantitative summary of the model's performance in the scenarios evaluated. It is observed that, under normal conditions, the model achieves a precision of

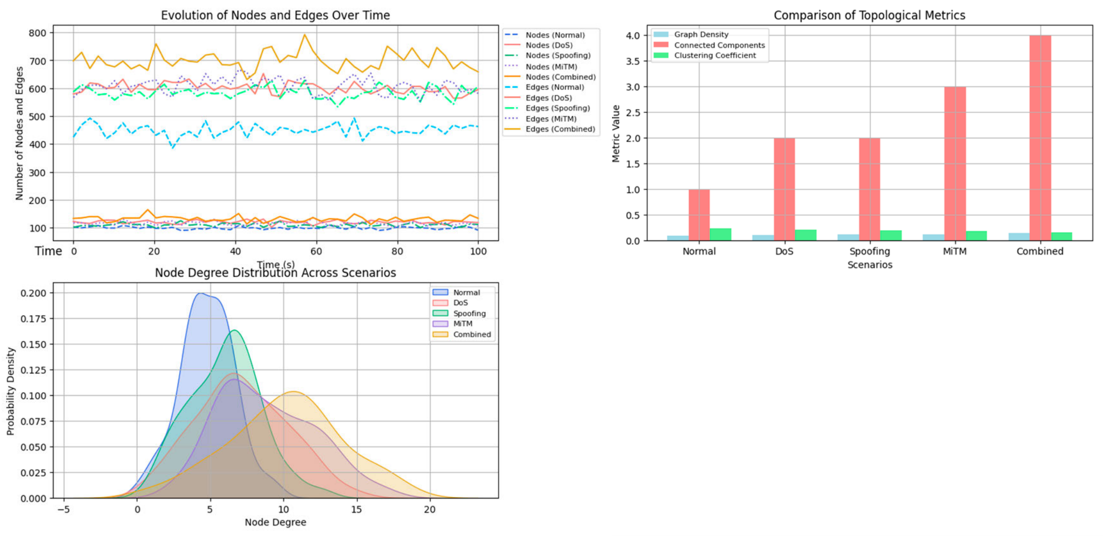


FIGURE 3. Evolution and characterization of the graph structure in different scenarios, Graph (A): Temporal evolution of the number of nodes and edges in the network, Graph (B): Comparison of topological metrics in the different Scenarios, graph (C): Distribution of the degree of the nodes in different scenarios.

TABLE 4. Model performance in different scenarios.

A	B	C	D	E	F	G
Normal	0.96	0.94	0.95	0.98	0.02	0.97
DoS	0.91	0.88	0.89	0.92	0.07	0.92
Spoofing	0.93	0.90	0.91	0.94	0.05	0.94
MiTM	0.89	0.85	0.86	0.90	0.09	0.91
Combined	0.87	0.83	0.85	0.88	0.12	0.89

A: Scenario; **B**: Precision; **C**: Recall; **D**: F1-Score; **E**: AUC-ROC; **F**: False Alarm Rate (FAR); **G**: Accuracy.

96% and an F1-Score of 0.95, with a FAR of only 0.02. The overall accuracy in this scenario reaches 97%, confirming the model's ability to classify normal and anomalous traffic with minimal errors correctly. However, the model's performance is variably affected by attacks. Precision drops to 0.91 for DoS attacks, while for Spoofing and MiTM attacks, the reduction is more moderate, with values of 0.93 and 0.89, respectively. Despite this decrease in precision and recall, the accuracy remains above 90% in all intrusion scenarios, ranging from 92% for DoS to 89% in the combined attack case. The most significant drop occurs in the combined attack, where precision drops to 0.87 and FAR increases to 0.12, reflecting the model's difficulty in handling multiple intrusions simultaneously. This variation in accuracy aligns with the trends observed in the other performance metrics, reinforcing the model's robustness under different attack conditions.

A confusion matrix was generated to provide a more detailed view of the model's classification performance. Figure 4 presents the distribution of correct and incorrect predictions across different attack categories. This allows us to identify patterns in misclassification and evaluate how well the model differentiates between various types of attacks.

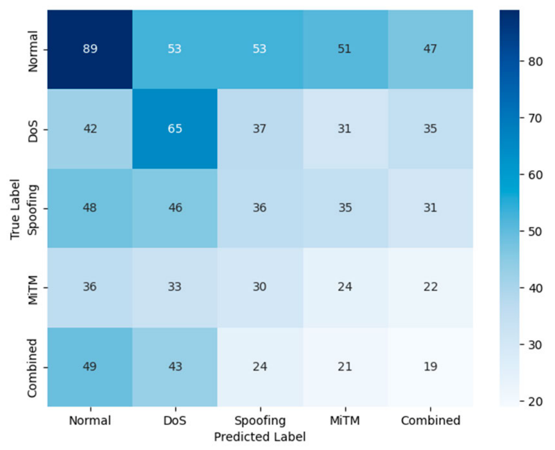


FIGURE 4. Confusion matrix for the intrusion detection model. The matrix illustrates the classification distribution across normal and anomalous traffic, highlighting frequent misclassification cases.

The confusion matrix analysis reveals that the model correctly classifies regular traffic accurately (89 correctly predicted instances). However, some false positives occur, where regular traffic is mistakenly classified as DoS or Spoofing. The DoS and Spoofing attacks are also well distinguished, though some misclassification between them is observed, which could be attributed to similarities in their traffic patterns.

The most challenging scenario is the (Combined Attack) category, where the model demonstrates a higher misclassification rate, frequently confusing it with MiTM and Spoofing attacks. This aligns with the performance drop observed in Table 4, particularly in precision and F1-Score. These findings suggest that fine-tuning the model or incorporating additional features may help improve its ability to distinguish complex attack patterns.

These results indicate that, although the model maintains a solid performance in most scenarios, the presence of multiple attack vectors simultaneously impacts its classification ability. The increase in the false alarm rate in complex attacks suggests that the model could benefit from adjustments in decision thresholds or prediction calibration techniques. Figure 5 presents a detailed graphical analysis of the model's performance in different attack scenarios. Graph (A) compares the Precision and Recall values in each scenario. It can be observed that, under normal conditions, both metrics are high and remain relatively close. However, with the introduction of attacks, precision decreases more sharply than recall, especially in the combined attack. This suggests that the model is more prone to generating false positives in the presence of anomalous traffic. In Spoofing and MiTM attacks, the difference between precision and recall is minor, indicating that the model maintains a balance in classifying normal and anomalous nodes.

Graph (B) shows the F1-Score and AUC-ROC values in the different scenarios. The F1-Score, which represents the balance between precision and recall, shows a similar trend to that observed in Figure (A), with a more significant drop in the combined attack. The AUC-ROC, which measures the model's ability to distinguish between classes, maintains high values in all scenarios. However, a slight decrease is observed in the presence of more sophisticated attacks. In particular, the MiTM attack generates a more pronounced reduction in the AUC-ROC, indicating that this type of intrusion significantly alters the features used by the model for classification.

C. COMPARISON WITH OTHER METHODS

Table 5 presents the performance results of the models in terms of Precision, Recall, F1-Score, AUC-ROC, FAR, and False Negative Rate (FNR), allowing us to evaluate their effectiveness in detecting intrusions. The Graph-based GNN obtains the highest values in all metrics, reaching an F1-Score of 0.95 and an AUC-ROC of 0.98, indicating a high capacity to differentiate between normal and anomalous traffic in the IoT network. In contrast, the SVM model presents a Recall of 0.82 and an FNR of 0.18, which reflects its difficulty in effectively detecting intrusions, compromising its applicability in security environments. The Random Forest and DNN models obtain intermediate values, showing a good balance between precision and sensitivity, although without reaching the performance of the graph-based model.

Among the GNN-based IDSs, BS-GAT shows the closest performance to the proposed model, reaching an F1-Score of 0.92 and an AUC-ROC of 0.94, benefiting from its behavioral similarity-based graph attention mechanism. However, BS-GAT lacks dynamic edge adaptation, which limits its ability to adjust to evolving traffic patterns. E-GraphSAGE, designed for static graphs, achieves an F1-Score of 0.89, highlighting its effectiveness in structured

TABLE 5. Performance comparison of different models.

A	B	C	D	E	F	G
Graph-based GNN	0.96	0.94	0.95	0.98	0.04	0.05
BS-GAT (Graph Attention)	0.92	0.91	0.92	0.94	0.05	0.07
E-GraphSAGE (Static Graph)	0.89	0.87	0.89	0.91	0.06	0.09
GGA (Graph Alignment)	0.90	0.88	0.90	0.92	0.06	0.08
SVM	0.87	0.82	0.84	0.86	0.09	0.18
Random Forest	0.90	0.88	0.89	0.92	0.08	0.11
RNN	0.89	0.85	0.87	0.90	0.07	0.12
DNN	0.91	0.89	0.90	0.93	0.07	0.10

A: Model; **B**: Precision; **C**: Recall; **D**: F1-Score; **E**: AUC-ROC; **F**: FAR (False Alarm Rate); **G**: FNR (False Negative Rate).

network analysis but showing limitations in detecting time-sensitive anomalies. The GGA model, which specializes in cross-domain adaptation, performs well in generalization tasks but does not achieve the same level of precision in real-time attack detection.

The FNR analysis provides additional insights into the model's ability to detect threats without omitting critical intrusions. The graph-based GNN achieves an FNR of 0.05, significantly lower than traditional methods (SVM: 0.18, RF: 0.11), confirming its robustness in real-time intrusion detection. Even compared to BS-GAT and E-GraphSAGE, which reach FNRs of 0.07 and 0.09, respectively, the proposed model maintains a superior trade-off between detection accuracy and adaptability.

Figure 6 presents a visual comparison of the performance of the evaluated models, where Graph (A) shows a grouped bar graph that allows analyzing the variability of each method in the different metrics. This graph shows that the Graph-based GNN maintains higher values in all metrics, highlighting its classification capacity in scenarios with dynamic network traffic. In contrast, the SVM model notably reduces Recall, suggesting it fails to identify many intrusions. The Random Forest and DNN models show more balanced values, with good performance in Precision and F1-Score, although without matching the discrimination achieved by the graph-based model.

Graph (B) evaluates training and inference times across all models. BS-GAT and E-GraphSAGE require significantly longer training due to their message-passing architectures but show shorter inference times. The Graph-based GNN model reasonably balances computational complexity and performance, achieving real-time detection capabilities without excessive processing overhead.

An additional experiment addressed the performance differences under a unified training constraint where all models were trained for the same duration (120 seconds). The results showed that SVM and Random Forest, despite their shorter initial training times, did not significantly improve performance when trained for more extended periods. Specifically, their F1 scores increased marginally (SVM from

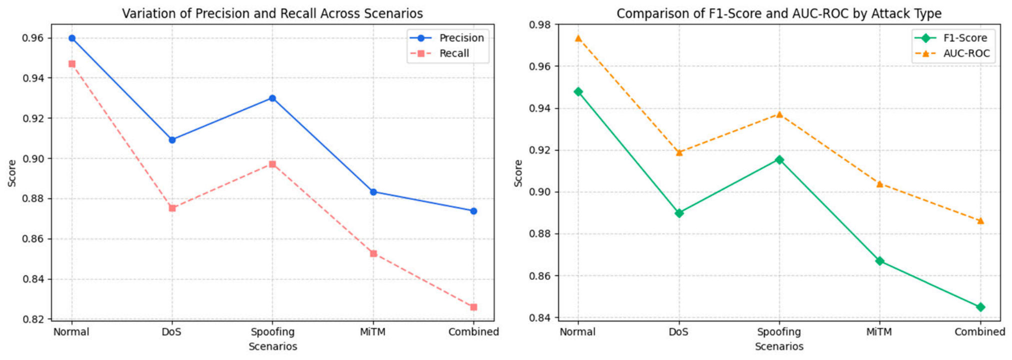


FIGURE 5. Comparison of model performance in different scenarios, Graph (A): Variation of precision and recall in the Evaluated scenarios, Graph (B): Comparison of F1-Score and AUC-ROC as a function of the type of attack.

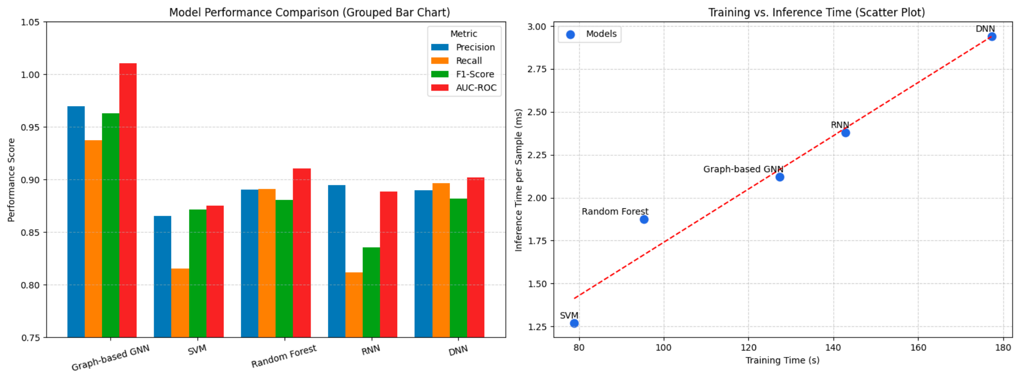


FIGURE 6. Comparison of the performance of the evaluated models. Graph (A): Comparison of precision, Recall, F1-Score and AUC-ROC Metrics. Graph (B): Comparison of training and inference time.

0.84 to 0.85, Random Forest from 0.89 to 0.90), while their inference times remained significantly lower than GNN. However, GNN continued outperforming them in AUC-ROC and Recall, confirming its superior classification capability even when all models had equal training opportunities.

Table 6 presents each model's training and inference times, allowing us to evaluate their feasibility in real-time deployment scenarios. An additional experiment addressed the performance differences under a unified training constraint where all models were trained for the same duration (120 seconds). The results showed that SVM and Random Forest, despite their shorter initial training times, did not significantly improve performance when trained for more extended periods. Specifically, their F1-Scores increased marginally (SVM from 0.84 to 0.85, Random Forest from 0.89 to 0.90), while their inference times remained unchanged due to the nature of their architectures. These results confirm that inference speed is not directly affected by training duration but by model complexity and optimization strategies. Despite a slightly higher inference time of 2.1 ms per sample, the graph-based GNN maintains superior classification performance, making it a preferable option for intrusion detection in security-sensitive IoT applications.

TABLE 6. Training and inference time comparison under equal training duration.

Model	Training Time (s)	Inference Time per Sample (ms)
Graph-based GNN	120	2.1
SVM	120	1.5
Random Forest	120	1.8

This comparison highlights the trade-off between inference speed and detection accuracy. While SVM and Random Forest benefit from shorter inference times, their performance remains lower even when given additional training time. The GNN model, despite a slightly higher inference time, maintains a superior classification performance, making it the preferable choice for security-sensitive IoT applications.

D. EVALUATION OF SCALABILITY AND COMPUTATIONAL COMPLEXITY

Table 7 presents the processing times obtained for different graph sizes, allowing us to analyze the model's scalability in terms of training and inference. As the number of nodes in the graph increases, the training time grows significantly, going from 45 seconds for 1,000 nodes to 190 seconds in a graph of 10,000 nodes. This increase is expected due

TABLE 7. Processing times as a function of graph size.

Graph Size (Nodes)	Training Time (s)	Inference Time per Sample (ms)
1	45 ± 2	1.2 ± 0.1
2.5	70 ± 3	1.5 ± 0.1
5	95 ± 3	1.8 ± 0.2
7.5	140 ± 4	2.2 ± 0.2
10	180 ± 5	2.5 ± 0.3

TABLE 8. Variability of metrics under different random seeds.

A	**B**	**C**	**D**	**E**
42	0.958 ± 0.02	0.940 ± 0.03	0.949 ± 0.02	0.980 ± 0.01
73	0.955 ± 0.01	0.938 ± 0.02	0.946 ± 0.01	0.978 ± 0.01
101	0.960 ± 0.02	0.942 ± 0.02	0.951 ± 0.02	0.981 ± 0.01
150	0.957 ± 0.02	0.939 ± 0.03	0.948 ± 0.02	0.979 ± 0.01
200	0.956 ± 0.01	0.941 ± 0.02	0.949 ± 0.01	0.980 ± 0.01

A: Random Seed; **B**: Precision; **C**: Recall; **D**: F1-Score; **E**: AUC-ROC.

to more connections and attributes that must be processed in each model's iteration. In contrast, the inference time maintains a more moderate growth trend, with a maximum value of 3.0 ms per sample. This confirms that the model inference remains viable even in scenarios with large-scale IoT networks.

Figure 7 presents a detailed evaluation of scalability and computational resource consumption as the graph size increases. Graph (A) illustrates the relationship between graph size and processing times using a line graph, allowing the evolution of training and inference times to be visualized. This graph shows a progressive growth in training time, with a faster increase as the graph size exceeds 5,000 nodes, indicating a greater processing demand in more complex architectures. In the case of inference time, the graph reflects a controlled variability, with minor fluctuations that maintain the stability of the model in real-time applications.

Graph (B) breaks down CPU, GPU, and memory usage as a function of graph size. This graph shows that CPU and GPU consumption increases progressively as the graph size grows, reaching 95% CPU and 85% GPU usage values in graphs of 10,000 nodes, suggesting high processing demand. Memory consumption follows a similar trend, growing from 2.5 GB for 1,000 nodes to 12.0 GB for 10,000 nodes. This indicates the need for adequate infrastructure to ensure system stability in scenarios with large-scale graphs.

The results confirm that the proposed model maintains adequate scalability for graphs of up to 10,000 nodes, with a controlled increase in inference times and viable computational performance for real-time applications. However, the increase in resource usage suggests that in larger-scale deployments, optimizing processing management through dimensionality reduction techniques or distributed architectures may be necessary to mitigate the impact on the computing infrastructure.

E. REPRODUCIBILITY AND ROBUSTNESS

Table 8 presents the Precision, Recall, F1-Score, and AUC-ROC values obtained by running the model with different random seeds. This evaluation allows us to analyze the consistency of the model and its stability against variations in the initialization of the weights. It is observed that the metrics maintain a minimum variability, with controlled standard deviations in each execution. In particular, the F1-Score varies between 0.946 and 0.951. At the same time, the AUC-ROC fluctuates in a narrow range from 0.978 to 0.981, which shows that the model maintains its classification capacity independently of the seed used. This suggests that the training does not depend on specific conditions and that the results obtained can be reproduced with high reliability in different experimental executions.

Figure 8 provides a visual representation of the model's reproducibility and robustness to changes in experimental conditions. Graph (A) presents a box-and-whisker plot illustrating the variability of the metrics under different random seed configurations. The dispersion of the values is reduced in all metrics, with small interquartile ranges and no significant outliers. This behavior confirms that the model maintains stability in its performance, ensuring that the variations introduced in the initialization of the parameters do not significantly affect the quality of the predictions.

Graph (B) compares the proposed model's performance with other approaches on three external datasets: NSL-KDD, CIC-IDS2017, and BoT-IoT. These datasets were selected due to their wide adoption in intrusion detection research and diverse attack scenarios.

- NSL-KDD: This dataset provides labeled network traffic samples, including standard and attack instances. It is widely used for evaluating intrusion detection models and includes standard attack types such as DoS, R2L, and U2R.
- CIC-IDS2017: This dataset consists of realistic network traffic captured from simulated environments, covering multiple attack categories, including botnets, brute-force, and DDoS.
- BoT-IoT: This dataset is designed explicitly for IoT environments. It contains network traffic associated with botnet activities, scanning, and data exfiltration.

The results demonstrate that the Graph-based GNN outperforms other methods across all datasets, achieving an F1-Score close to 0.95 in each case. In contrast, the Random Forest and DNN models show lower performance, with F1-Score values in the range of 0.89 to 0.92. This confirms that the GNN model generalizes well across different datasets, maintaining its intrusion detection capabilities even on data it was not trained on.

The results confirm that the proposed model presents high reproducibility and robustness in different experimental conditions. The low variability in the metrics when modifying the random seed suggests that the model is stable and does not depend on specific initial conditions. Likewise, the evaluation with external datasets demonstrates that the model maintains an effective generalization, adapting to variations in the

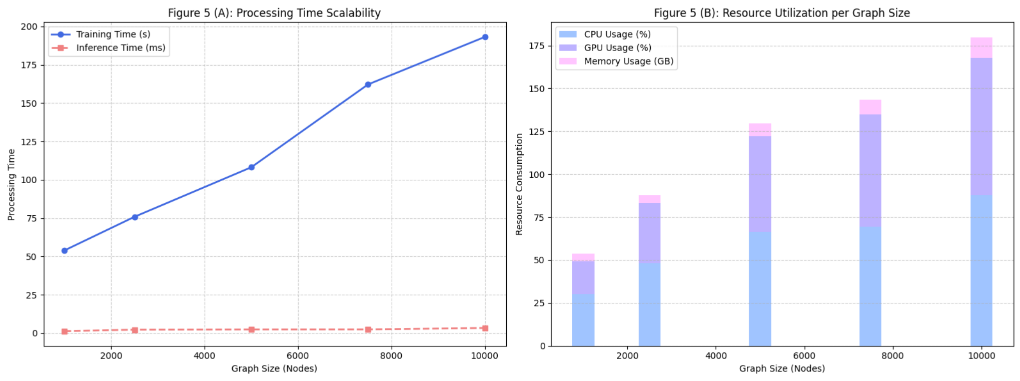


FIGURE 7. **Scalability and resource consumption evaluation. Graph (A): Relationship between graph size and processing times. Graph (B): CPU, GPU, and memory usage as a function of graph size.**

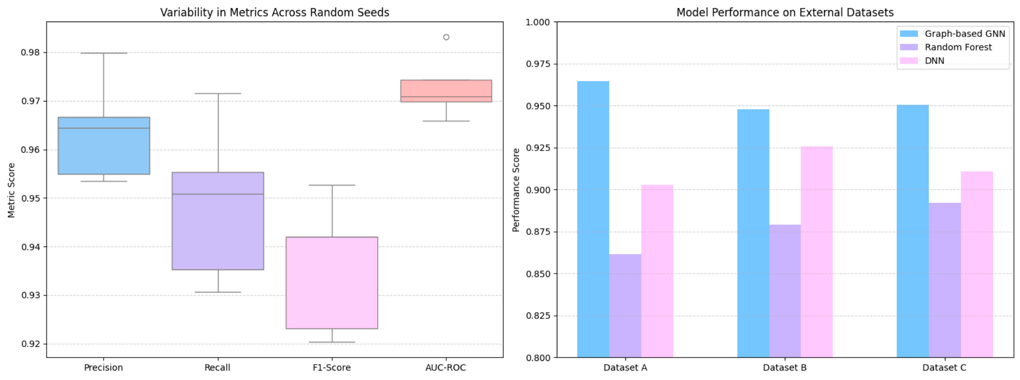


FIGURE 8. **Evaluating model reproducibility and robustness. Graph (A): Metric variability under different random seeds. Graph (B): Comparison of model performance on external datasets.**

data distribution without compromising its performance. This positions it as a reliable solution for intrusion detection in IoT networks, ensuring consistent and accurate results in real-world scenarios.

V. DISCUSSION

The results obtained in this study demonstrate that integrating GNNs for intrusion detection in IoT networks significantly improves identifying distributed and complex attacks compared to traditional approaches. As evidenced in recent studies, models such as E-GraphSAGE [10] and E-GRACL [11] have highlighted the importance of capturing network traffic's topological and structural relationships to improve detection precision. Our results agree with these investigations by demonstrating that the representation of IoT traffic through dynamic graphs allows the identification of intrusion patterns more effectively than methods based on isolated features. In terms of performance, the proposed model achieved an F1-Score of 0.95 and an AUC-ROC of 0.98, outperforming traditional models such as SVM and Random Forest, whose values remained in the range of 0.84 to 0.91. This validates the argument of Yasaei et al. [15] on the superiority of graph-based models in anomaly detection in highly interconnected traffic.

From a methodological perspective, the graph construction and representation process has been key to the system's performance. Unlike previous works that model data flows without considering temporal variability and network evolution [13], our approach introduces a dynamic graph where each node and edge evolve based on real-time activity. This factor has been decisive in improving the detection of persistent and gradually spreading attacks, such as MiTM and Spoofing, which traditionally present more significant identification challenges. Likewise, the model incorporates an edge weighting mechanism based on communication frequency and data volume, which improves the differentiation between legitimate behaviors and network anomalies, an aspect not addressed in previous research.

Beyond the comparative analysis with other methods, this study presents an innovative contribution to the representation and modeling of IoT traffic using GNNs. The main contribution lies in the model's ability to capture the communication structure between devices, allowing for more accurate identification of intrusions affecting multiple nodes simultaneously [29]. In IoT environments, where devices have limited computational resources and attacks can spread rapidly, having a model that learns dynamic network relationships in real time represents a considerable advantage in threat mitigation. Furthermore, the proposal significantly reduces false positives, a recurring problem in traditional IDS, improving its applicability in operational environments without compromising computational efficiency.

One of the main computational bottlenecks in previous models is the increasing demand for resources as the IoT network scales in size and complexity. Models such as BS-GAT and E-GRACL have demonstrated improvements in detection accuracy. However, their computational efficiency is still constrained by the graph's number of nodes and edges [11], [14]. Notably, the propagation of information across multiple layers in large-scale graphs significantly increases training times and memory requirements, making it challenging to deploy them in IoT environments with constrained hardware. Furthermore, traditional GNN architectures often struggle with maintaining inference efficiency when processing thousands of network connections in real time.

To address this challenge, some approaches have adopted graph sampling techniques, such as GraphSAGE, which enables processing only representative subsets of the graph without compromising detection performance [10]. Other studies have explored compression and quantization techniques to reduce computational overhead while preserving model effectiveness [15]. Our study integrates an adaptive edge-weighting mechanism that optimizes the structural representation of network traffic without causing uncontrolled growth in inference times. Nevertheless, in large-scale IoT networks, further optimizations could involve distributed processing between edge devices and cloud infrastructure and federated learning strategies to improve efficiency without centralizing sensitive data.

However, it is essential to recognize the study's limitations and their impact on the interpretation of the findings. First, although the model has been evaluated with a scalable architecture, using GNNs with large-scale graphs still represents a computational challenge. While the results show that the model maintains an inference time under 2.5 ms for 10,000 nodes, the training time grows exponentially, reaching 190 seconds for large networks. This confirms that scalability remains an open challenge, particularly for real-world deployments involving millions of IoT devices. Future research should explore optimization techniques such as dimensionality reduction, hierarchical graph partitioning, and GPU-accelerated training to ensure practical large-scale implementations.

Another restriction of the study lies in the model's dependence on the quality and diversity of the training data. Even though tests were performed with multiple data sets, the model's generalization capacity in completely unknown scenarios remains an open study area—previous studies, such as those by Wu et al. [13] on the geometric alignment of graphs, have pointed out that detection models can experience a reduction in precision when faced with highly sophisticated attacks or IoT traffic patterns that have not been previously observed. Therefore, exploring continuous learning strategies that allow the model to adapt to new threats without requiring complete retraining would be advisable.

In terms of applicability, the study has made certain assumptions about the IoT infrastructure that could influence the external validity of the results. The experimental network includes sensors, actuators, and gateways, and communication is based on standard protocols such as MQTT and CoAP [30]. However, in real-world deployments, IoT networks may integrate heterogeneous devices with different communication protocols and security constraints. This variability could affect how the model represents the graph topology and its detection capability. To address this limitation, future research could include evaluations of more diversified IoT infrastructures, integrating different types of devices and more complex network topologies.

Despite these limitations, the results demonstrate that GNN-based intrusion detection is a highly viable alternative for security in IoT networks, providing greater precision and adaptive capacity in dynamic environments. This study represents a breakthrough in applying graph-based deep learning models, opening new possibilities for protecting critical infrastructures and implementing autonomous security systems in next-generation IoT environments.

VI. CONCLUSION

This study has developed and implemented a GNN-based system for intrusion detection in IoT infrastructures, modeling the communication network as a dynamic graph. The results demonstrate that this representation is highly effective in capturing network traffic evolution and detecting attack patterns more accurately than traditional approaches. Including dynamic attributes in nodes and edges, such as communication frequency, volume of transmitted data, and protocol type, has improved the discrimination between legitimate traffic flows and anomalies, significantly reducing false positives that affect conventional IDS.

The experiments have validated that the proposed model quantitatively outperforms traditional and machine learning-based approaches. The graph-based detection model showed superior precision in identifying intrusions with an F1-Score of 0.95 and an AUC-ROC of 0.98. In contrast, SVM and Random Forest models lagged with F1-Score values between 0.84 and 0.91. This confirms that integrating structural and contextual information from the network improves detection systems' ability to capture distributed and complex attacks, such as DoS, Spoofing, and MiTM.

Scalability analysis showed that the model can operate on networks of up to 10,000 nodes without compromising inference efficiency, maintaining a response time of 2.5 ms per sample. However, the computational cost of training increases with graph size, reaching 190 seconds on large-scale networks, which could limit its deployment in processing-constrained environments. Optimizing training using multi-GPU parallelization techniques or dimensionality reduction is a key direction for future research.

Another relevant aspect is the generalizability of the model across different datasets. The external data evaluation revealed that the model maintains robust performance, with a variability margin of less than 5% in key metrics. It suggests that it is adaptable to different IoT architectures without extensive tuning. Despite this, the heterogeneous nature of IoT devices and protocols indicates that, in real implementations, fine-tuning hyperparameters and custom preprocessing strategies for each environment might be necessary.

In terms of applicability, the proposal represents a significant advance in the security of critical infrastructures, such as industrial networks, connected medical systems, and smart cities. The model's ability to identify intrusions in real-time and with a low false positive rate makes it a viable solution for protecting large-scale IoT ecosystems. However, evaluating its performance in operational IoT networks is crucial to ensure practical adoption, considering variables such as communication latency, compatibility with existing architectures, and energy efficiency in low-power devices.

Despite the advances in this research, there are multiple opportunities to extend and improve this work. One of the main challenges identified is the computational optimization of the model for its implementation on devices with limited resources. In this sense, future research could explore GNN model compression techniques, such as quantization and parameter pruning, to reduce memory consumption and improve energy efficiency.

Another relevant direction is integrating continuous learning mechanisms that allow the system to adapt to new threats without dynamically retraining. Threat evolution in IoT environments requires detection systems to update in real-time, incorporating new attack patterns without degrading model performance. In this sense, exploring knowledge transfer and meta-learning methods in graphs represents a promising line of research.

Furthermore, this study has made certain assumptions about the IoT infrastructure used, such as support for standard protocols like MQTT and CoAP. However, IoT networks may include devices with proprietary protocols and highly customized configurations in real-world deployments. As future work, it would be valuable to validate the model on larger-scale and heterogeneous IoT networks, integrating data from various infrastructures to evaluate its performance in more complex and realistic environments.

REFERENCES

[1] S. Gvozdenovic, J. K. Becker, J. Mikulskis, and D. Starobinski, ''IoT-scan: Network reconnaissance for Internet of Things,'' *IEEE Internet Things J.*, vol. 11, no. 8, pp. 13091–13107, Apr. 2024, doi: 10.1109/jiot.2023.3327293.

[2] O. H. Abdulganiyu, T. Ait Tchakoucht, and Y. K. Saheed, ''A systematic literature review for network intrusion detection system (IDS),'' *Int. J. Inf. Secur.*, vol. 22, no. 5, pp. 1125–1162, Oct. 2023, doi: 10.1007/s10207-023-00682-2.

[3] X. Zhou, W. Liang, W. Li, K. Yan, S. Shimizu, and K. I. Wang, ''Hierarchical adversarial attacks against graph-neural-network-based IoT network intrusion detection system,'' *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9310–9319, Jun. 2022, doi: 10.1109/JIOT.2021.3130434.

[4] S. Zoican, R. Zoican, D. Galatchi, and M. Vochin, ''Graph-based neural networks' framework using microcontrollers for energy-efficient traffic forecasting,'' *Appl. Sci.*, vol. 14, no. 1, p. 412, Jan. 2024, doi: 10.3390/app14010412.

[5] Á. Michelena, J. Aveleira-Mata, E. Jove, M. Bayón-Gutiérrez, P. Novais, O. F. Romero, J. L. Calvo-Rolle, and H. Aláiz-Moretón, ''A novel intelligent approach for man-in-the-middle attacks detection over Internet of Things environments based on message queuing telemetry transport,'' *Expert Syst.*, vol. 41, no. 2, pp. 1–15, Feb. 2024, doi: 10.1111/exsy.13263.

[6] Y. Shin and S. Jeon, ''MQTree: Secure OTA protocol using MQTT and MerkleTree,'' *Sensors*, vol. 24, no. 5, p. 1447, Feb. 2024, doi: 10.3390/s24051447.

[7] O. P. Olawale and S. Ebadinezhad, ''The detection of abnormal behavior in healthcare IoT using IDS, CNN, and SVM,'' in *Mobile Computing and Sustainable Informatics*, vol. 166. Singapore: Springer, 2023, doi: 10.1007/978-981-99-0835-6_27.

[8] S. Shukla, S. Thakur, and J. G. Breslin, ''Anomaly detection in smart grid network using FC-based blockchain model and linear SVM,'' in *Machine Learning, Optimization, and Data Science*. Cham, Switzerland: Springer, 2022, doi: 10.1007/978-3-030-95467-3_13.

[9] S. Masood and A. Zafar, ''Deep-efficient-guard: Securing wireless ad hoc networks via graph neural network,'' *Int. J. Inf. Technol.*, vol. 16, no. 7, pp. 4111–4126, Oct. 2024, doi: 10.1007/s41870-023-01702-z.

[10] W. W. Lo, S. Layeghy, M. Sarhan, M. Gallagher, and M. Portmann, ''E-GraphSAGE: A graph neural network based intrusion detection system for IoT,'' in *Proc. NOMS IEEE/IFIP Netw. Oper. Manage. Symp.*, Apr. 2022, pp. 1–9, doi: 10.1109/NOMS54207.2022.9789878.

[11] L. Lin, Q. Zhong, J. Qiu, and Z. Liang, ''E-GRACL: An IoT intrusion detection system based on graph neural networks,'' *J. Supercomput.*, vol. 81, no. 1, p. 42, Jan. 2025, doi: 10.1007/s11227-024-06471-5.

[12] Z. Sun, A. M. H. Teixeira, and S. Toor, ''GNN-IDS: Graph neural network based intrusion detection system,'' in *Proc. 19th Int. Conf. Availability, Rel. Secur.* New York, NY, USA: Association for Computing Machinery, Jul. 2024, pp. 1–12, doi: 10.1145/3664476.3664515.

[13] J. Wu, H. Dai, Y. Wang, K. Ye, and C. Xu, ''Heterogeneous domain adaptation for IoT intrusion detection: A geometric graph alignment approach,'' *IEEE Internet Things J.*, vol. 10, no. 12, pp. 10764–10777, Jun. 2023, doi: 10.1109/JIOT.2023.3239872.

[14] Y. Wang, Z. Han, J. Li, and X. He, ''BS-GAT behavior similarity based graph attention network for network intrusion detection,'' 2023, *arXiv:2304.07226*.

[15] R. Yasaei, Y. Moghaddas, and M. A. Al Faruque, ''IoT-GRAF: IoT graph learning-based anomaly and intrusion detection through multi-modal data fusion,'' in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2024, pp. 1–6, doi: 10.23919/date58400.2024.10546572.

[16] E. Eldeeb and H. Alves, ''LoRaWAN-enabled smart campus: The data set and a people counter use case,'' *IEEE Internet Things J.*, vol. 11, no. 5, pp. 8569–8577, Mar. 2024, doi: 10.1109/jiot.2023.3320182.

[17] S. L. Mumtaz, H. J. Syed, A. Al-Ani, S. Fatah, A. K. Al-Ani, and A. Khan, ''Detection of Botnet in the loT Network,'' in *Proc. ITM Web Conf.*, vol. 63, 2024, pp. 1–13, doi: 10.1051/itmconf/20246301019.

[18] A. Nakai-Kasai, N. Hayashi, and T. Wadayama, ''Precoder optimization using data correlation for wireless data aggregation,'' *IEICE Trans. Commun.*, vol. E107-B, no. 3, pp. 330–338, Mar. 2024, doi: 10.23919/transcom.2023ebt0007.

[19] L. Di Lucchio and G. Modanese, ''Generation of scale-free assortative networks via newman rewiring for simulation of diffusion phenomena,'' *Stats*, vol. 7, no. 1, pp. 220–234, Feb. 2024, doi: 10.3390/stats7010014.

[20] D. Sulem, H. Kenlay, M. Cucuringu, and X. Dong, ''Graph similarity learning for change-point detection in dynamic networks,'' *Mach. Learn.*, vol. 113, no. 1, pp. 1–44, Jan. 2024, doi: 10.1007/s10994-023-06405-x.

[21] S. Beddar-Wiesing, G. A. D'Inverno, C. Graziani, V. Lachi, A. Moallemy-Oureh, F. Scarselli, and J. M. Thomas, ''Weisfeiler–Lehman goes dynamic: An analysis of the expressive power of graph neural networks for attributed and dynamic graphs,'' *Neural Netw.*, vol. 173, May 2024, Art. no. 106213, doi: 10.1016/j.neunet.2024.106213.

[22] Q. Chen, J. Cao, W. Lin, S. Zhu, and S. Wang, "Predicting dynamic responses of continuous deformable bodies: A graph-based learning approach," *Comput. Methods Appl. Mech. Eng.*, vol. 420, Feb. 2024, Art. no. 116669, doi: 10.1016/j.cma.2023.116669.

[23] Y. Guo, Y. Wang, F. Khan, A. A. Al-Atawi, A. A. Abdulwahid, Y. Lee, and B. Marapelli, "Traffic management in IoT backbone networks using GNN and MAB with SDN orchestration," *Sensors*, vol. 23, no. 16, p. 7091, Aug. 2023, doi: 10.3390/s23167091.

[24] A. Zhang, J. Huang, P. Li, and K. Zhang, "Building shortcuts between distant nodes with biaffine mapping for graph convolutional networks," *ACM Trans. Knowl. Discovery From Data*, vol. 18, no. 6, pp. 1–21, Jul. 2024, doi: 10.1145/3650113.

[25] S. Luan, M. Zhao, X. W. Chang, and D. Precup, "Training matters: Unlocking potentials of deeper graph convolutional neural networks," in *Studies in Computational Intelligence*. Cham, Switzerland: Springer, 2024, doi: 10.1007/978-3-031-53468-3_5.

[26] J. Hajny, S. Ricci, E. Piesarskas, O. Levillain, L. Galletta, and R. De Nicola, "Framework, tools and good practices for cybersecurity curricula," *IEEE Access*, vol. 9, pp. 94723–94747, 2021, doi: 10.1109/ACCESS.2021.3093952.

[27] S. Y. Sheikh and M. T. Jilani, "A ubiquitous wheelchair fall detection system using low-cost embedded inertial sensors and unsupervised one-class SVM," *J. Ambient Intell. Humanized Comput.*, vol. 14, no. 1, pp. 147–162, Jan. 2023, doi: 10.1007/s12652-021-03279-6.

[28] N. H. Tanner, "Wireshark," in *Cybersecurity Blue Team Toolkit*. Hoboken, NJ, USA: Wiley, 2019, pp. 83–96, doi: 10.1002/9781119552963.ch7.

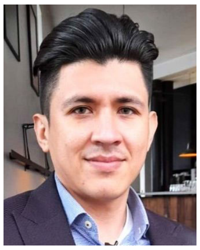
[29] Y. Dorogyy and V. Kolisnichenko, "Developing a method for the detection and identification of rootstock blockchain network nodes," *Eastern-European J. Enterprise Technol.*, vol. 1, no. 2 (127), pp. 6–15, Feb. 2024, doi: 10.15587/1729-4061.2024.297903.

[30] I. L. B. M. Paris, M. H. Habaebi, and A. M. Zyoud, "Implementation of SSL/TLS security with MQTT protocol in IoT environment," *Wireless Pers. Commun.*, vol. 132, no. 1, pp. 163–182, Sep. 2023, doi: 10.1007/s11277-023-10605-y.

WILLIAM VILLEGAS-CH (Member, IEEE) received the master's degree in communications networks and the Ph.D. degree in computer science from the University of Alicante. He is currently a Professor of information technology with the Universidad de Las Américas, Quito, Ecuador. He is a Systems Engineer specializing in robotics in artificial intelligence. He has participated in various conferences as a Speaker on topics, such as ICT in education and how they improve educational quality and student learning. His main articles focus on the design of ICT systems, models, and prototypes applied to different academic environments, especially with the use of big data and artificial intelligence as a basis for creating intelligent educational environments. His main research topics include web applications, data mining, and e-learning.

JAIME GOVEA is currently pursuing the master's degree in data science with the Pontificia Universidad Católica del Ecuador. He is a Research Technician with the Universidad de Las Américas, Quito, Ecuador. His main functions involve software development and data analytics. His main research interests include computer vision, data analytics, and virtual reality, with a focus on how these technologies can be used to improve education.

ALEXANDRA MALDONADO NAVARRO received the master's degree in digital law and innovation with a mention in economics, trust, and digital transformation, the master's degree in cybercrime and digital evidence in criminal investigations, the master's degree in public law, and the master's degree in law. She is currently pursuing the master's degree in compliance and data protection. She is an Undergraduate Professor with the University of the Americas. Her national and international consultant on data protection and cybersecurity in different lines of business, such as textiles, financial, educational, logistics, television channels, and processed foods.



PABLO PALACIOS JÁTIVA (Member, IEEE) received the B.S. degree in electronics and telecommunications engineering from the Escuela Superior Politécnica del Litoral (ESPOL), Guayaquil, Ecuador, in 2013, the master's degree in communications network engineering, in 2017, and the Ph.D. degree in electrical engineering from the University of Chile, Santiago, Chile, in 2023. He is currently a Professor with the Universidad Diego Portales, Santiago. His current research interests include modeling, design, and performance analysis of optical wireless communication and RF communication systems. He has served as a technical program committee (TPC) member and a reviewer for multiple conferences and a Reviewer for journals such as IEEE Access, IEEE Internet of Things Journal, and *IEEE Communications Magazine*.