PENINSULA PUBLISHING LLC

SHIFRA

Vol. **(2026)**, 2026, **pp**. 141–157
ISSN: 3078-3186

Research Article

A Hybrid Deep Reinforcement Learning and Graph Neural Network Framework for Adaptive Intrusion Detection in IoT-Enabled Cyber-Physical Systems

Kholood J. Moulood[1,*], Maath Farman[2]

[1] *Department of Math., College of Education for Women, Tikrit University, Salah al-Din, Iraq.*

[2] *Department of Computer Science, Tikrit University, Salah al-Din, Iraq.*

ARTICLE INFO

Article History

Received 4 Mar 2026

Revised: 20 Apr 2026
Accepted 21 Mar 2026
Published 04 Jun 2026

Keywords

graph neural networks (GNNs),

deep reinforcement learning (DRL),

adaptive security,

multi-objective optimization,

concept drift,

zero-day exploits,

intrusion detection system (IDS),

Cyber-Physical Systems (CPS).

ABSTRACT

The Internet of Things (IoT) has enabled the rapid development of Cyber Physical Systems (CPS), but this has also created serious security concerns, particularly in the area of identifying sophisticated and changing cyberattacks. Traditional intrusion detection systems (IDSs) are built on static models and handcrafted features that are unable to keep up with the constantly changing nature of network behavior and new threats. This study presents a hybrid intelligent architecture built on Graph Neural Networks (GNNs) and Deep Reinforcement Learning (DRL) for context-aware and adaptive intrusion detection in order to address these limitations. In order to aid the GNN component in capturing complex spatial and relational patterns between connected nodes over time, the suggested method treats the network traffic as a dynamic graph made up of devices as nodes and interactions between them as edges. With a multi-objective reward that takes into account detection performance, false positive rate, and computation latency, a DRL agent is also presented as a meta-controller that dynamically adjusts detection policies in response to the current network states and environmental feedback. The framework is evaluated using three cutting-edge IoT intrusion detection datasets Edge-IIoTset, TON_IoT, and CICIoT2023 under a variety of traffic scenarios, including concept drift and zero-day exploits. The suggested method achieves a 97. 2% F1 score and a 1. 1% false positive rate, according to experimental data, which is significantly better than conventional machine learning, deep learning, GNN-only, and DRL-only baselines. When subjected to unidentified assaults, the end-to-end framework only experiences a 6. 2% drop in the F1-score, compared to 18. 3% for static GNNs, and it recovers from concept drift 4. 7 times faster than DRL-only techniques, all while running with a low inference latency (27. 6 ms per batch). These findings show how promising adaptive decision-making and integrated structural graph learning are for safeguarding the next generation of IoT-CPS systems.

1. INTRODUCTION

The fast integration of the Internet of Things (IoT) with Cyber-Physical Systems (CPS) has transformed vital infrastructure in industrial control systems, autonomous vehicles, and smart grids. These IoT-driven settings generate enormous volumes of data, provide real-time control, and effectively bridge the divide between digital and physical operations. However, security is also crucial since this significantly expands the attack surface. For instance, a single assault can not only spread into the real world, but it can also cause actual harm, production interruptions, and even deaths. Existing security mechanisms are no longer adequate for IoT-CPS systems because of their unique features, such as their heterogeneity, resource limitations, and dynamic nature. As a result, there is a need to create intelligent and self-adaptive intrusion detection (ID) techniques [1], [2]. Most intrusion detection systems (IDS) are either anomaly-based or signature-based. However, signature-based methods are only able to identify known attacks and are useless against novel or zero-day exploits. In contrast, anomaly-based approaches construct a model of "normal" behavior and treat anything that deviates from this model as suspect; however, they are prone to producing a high number of false positives, particularly in dynamic contexts where genuine traffic is always changing. Furthermore, if an attacker changes their strategy, static models cannot adapt. Researchers have been pushed in the direction of machine learning (ML) and deep learning (DL) methods because they can detect minute patterns in network traffic [3], [4].

*Corresponding author email: kjamal@tu.edu.iq
DOI: https://doi.org/10.70470/SHIFRA/2026/008

Many current methods treat the network data as if it were individual and isolated or as if it were part of a linear sequence, whereas convolutional neural networks (CNNs) and recurrent neural networks (RNNs), which are examples of deep learning techniques, appear to be quite effective at intrusion detection. It is possible to miss the crucial links between devices, which is a key problem for IoT-CPS systems. The behavior of a device can be influenced by its connections with other devices. Because of this, current architectures do not effectively utilize the underlying communication structure, which reduces their precision and flexibility [5]. Graph Neural Networks (GNNs) are one option since they are perfect for graph-based data, such as network traffic. In this representation, devices are 'nodes' and the connections between them are 'edges'. Given that GNNs consider both the immediate neighbors of a node and the larger network around it, there is ample evidence that they are capable of learning exceptionally robust spatial and relational patterns. Recent studies demonstrate that intrusion detection systems (IDSs) that use GNNs may surpass the present state-of-the-art in identifying sophisticated attack patterns and identifying novel harmful traffic patterns [2], [5]. This illustrates the potential of employing structural learning to ensure the security of IoT-CPS; it's a fascinating field.

The main application of GNNs is static analysis, despite their numerous benefits. For instance, they don't immediately adjust to various network traffic and attack kinds after they have completed their training. Deep Reinforcement Learning (DRL), however, is a logical method for making decisions that can adapt. In DRL, an agent is placed into an environment, receives feedback (rewards), and then learns how to achieve long-term goals by constantly refining its strategy. In this way, it seems probable that DRL might be a viable method for intrusion detection, especially since an agent may need to modify the threshold, weights of features, or even its policy in response to changing traffic patterns. In fact, several studies [6, 7, 8] have already demonstrated DRL's usefulness for adaptive IoT Intrusion Detection Systems (IDS).

Existing methods, however, generally only consider either DRL or GNNs. For example, while DRL is adaptive, it tends to depend on a basic model that does not adequately represent the graph structure. Conversely, GNNs can model the graph structure but are not adaptable. How to combine sequential decision-making with structured graph learning is the question that arises from this. To solve this problem, we are suggesting a novel hybrid architecture in this study that makes use of both a GNN (for spatial feature extraction) and a DRL agent (for adaptive policy optimization) with the aid of a multi-objective reward design that considers inference time, detection accuracy, and false alarms. Three contributions are made by our paper: (1) a novel, integrated GNN-DRL framework designed for IoT-CPS; (2) a novel multi-objective reward function; and (3) a thorough collection of tests using current IoT data sets.

The major difficulties facing today's intrusion detection systems in IoT-enabled cyber-physical environments are depicted in Figure 1. It identifies current remedies and, more crucially, highlights three related challenges: spatial/relational blindness; a chronic temporal and adaptive gap; and the multi-objective trade-offs that systems are always dealing with.

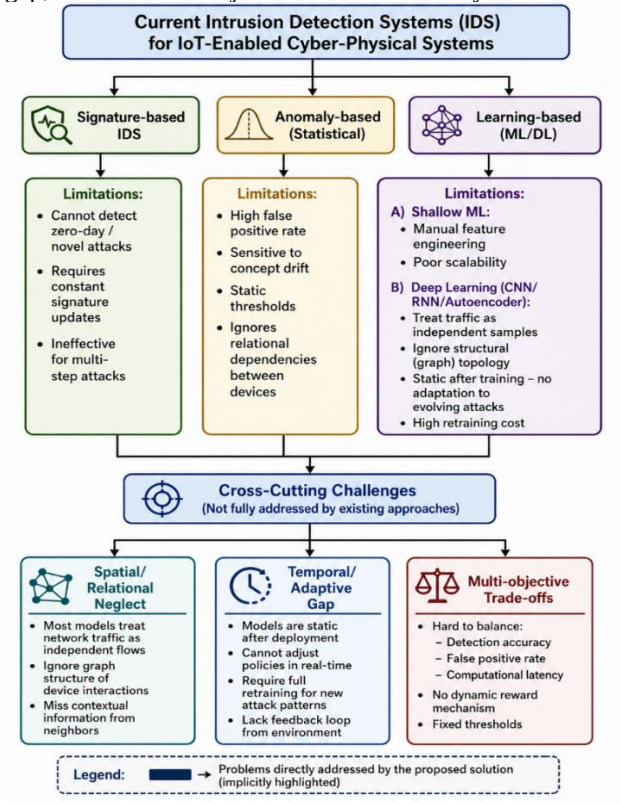

Fig. 1. current intrusion detection systems for cyber-physical systems with IoT capabilities, including their inherent drawbacks.

The organization of this work is as follows: The related work is covered at the start of Section 2. The official specifics of our suggested framework are then covered in Section 3. Our experiments are described in detail in Section 4. It is essential to comprehend the outcomes and their analysis; thus Section 4 is dedicated to them. Section 5 covers the consequences and restrictions of our study. The paper is ultimately finished in Section 6.

2. LITERATURE REVIEW

Intrusion detection systems (IDS) for IoT-based cyber-physical systems (CPS) have been the subject of several research. This chapter gives a methodical analysis of current approaches to assist us in selecting the best IDS, with the goal of highlighting their shortcomings and pinpointing important areas for further research. The structure of our review is as follows: In IoT-CPS, Section offers a quick overview of IDS. Then, explores conventional learning-based approaches, and covers techniques based on GNNs. After that, reviews discuss DRL-based approaches, and examines hybrid or novel approaches. Finally, provides a summary of the gaps that were discovered, while contrasts these cutting-edge methods with our own methodology.

2.1 An Overview of IoT-CPS Intrusion Detection Systems

The Internet of Things - Cyber-Physical Systems (IoT-CPS) security landscape is unique due to its variety of features, limited resources, dynamic nature, and significant physical connection. These unique traits make traditional intrusion detection systems (IDSs), such as anomaly-based and signature-based IDSs, useless for prolonged usage [9].

For instance, signature-based IDSs identify known attack patterns and trigger an alarm upon detecting the associated packets or traffic flows. However, the biggest issue with these systems is that, in the absence of signatures, they cannot identify zero-day or novel attacks. Additionally, given the vast variety of IoT devices, these solutions must be updated frequently with fresh signatures, which can lead to an increasing operational load. Additionally, because each stage of a multi-stage assault may not be harmful in and of itself, these systems are more likely to have trouble with it [9].

In contrast, statistically based (or anomaly) IDSs create a normal behavior model and then issue a warning when they see a significant deviation from the norm. Although more adaptable than IDSs based on signatures, they often have a high false positive rate (FPR), especially in the constantly evolving environment of IoT-CPS apps. These methods are also susceptible to concept drift, necessitating frequent retraining. Additionally, many of these anomaly-based systems have overlooked the critical inter-dependencies between devices, instead treating each network flow independently of the context of communication situations [10].

2.2 Traditional Methods of Intrusion Detection Based on Learning

Researchers genuinely started investigating machine learning (ML) and deep learning (DL) approaches to enhance intrusion detection in IoT-CPS environments when conventional statistical and rule-based methodologies proved insufficient. Since they are simpler to understand and need less processing power, superficial ML methods such as Support Vector Machines (SVMs), Random Forests (RFs), and K-Nearest Neighbors (KNNs) have gained widespread acceptance. However, these strategies depend heavily on manual feature engineering, which is a disadvantage. They also have trouble dealing with the massive, high-dimensional data seen in big IoT networks [11], which necessitates a certain set of skills that are frequently insufficient to identify complex attack patterns.

In IoT intrusion detection systems, deep learning methods such as autoencoders, Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs), and a variety of other architectures have also proven to be quite effective. In reality, a thorough systematic study that adhered to PRISMA 2020 guidelines demonstrated that hybrid deep learning configurations typically outperform single models. But that same review also highlighted significant issues: For example, realistic, IoT-specific datasets are severely lacking, and we are still lacking in viable, explainable AI methods [11]. It is important to note that conventional deep learning techniques often consider network traffic to be merely a set of unrelated samples or time-series data points. This implies that they frequently disregard the fundamental graph structure of how devices truly interact with one another. These models are mostly set after training; without expensive, comprehensive retraining, they cannot naturally adapt to new or changing attack types [12].

2.3 Graph Neural Network (GNN)-Based Intrusion Detection

With a lot of talents in network security, Graph Neural Networks (GNNs) represent a novel and exciting method for analyzing graph data. By considering network traffic as a graph, with network devices as the graph's nodes and the interactions between devices as the edges, GNNs operate. By doing this, they can develop sophisticated spatial and relational representations that can represent both local and global data in a graph.

Several cutting-edge Intrusion Detection Systems (IDS) based on GNN have been created. For instance, we suggested the edge-enhanced Graphs AGE model with graph contrastive learning (E-GRACL) and a global attention mechanism. We did this since it excels at extracting edge attribute and topological characteristics from several IoT benchmark datasets [12]. However, the Graph Edge Sentinel (GES) architecture offers an IDS for the Industrial IoT that is based on GNN. This method

embeds topological features from a monitored network and has been extensively evaluated on datasets like Edge-IIoT set, CICIDS 2018, and UNSW-NB15, where it has outperformed traditional, non-graph-based techniques [13]. One more recent breakthrough is the reinforced hypergraph neural network framework (RHNN-IoT). Recently identified as a Best Paper and published in Future Generation Computer Systems, this framework unexpectedly shows that hypergraphs allow us to depict more intricate, higher-order relationships between devices, as opposed to only binary relationships. This, in turn, promotes structural learning for IoT security [14].

In spite of all of these improvements, many modern GNN IDs still use a static inference methodology. For example, the parameters of a trained GNN model are fixed, so the IDS is unable to respond to changes in the network or new attacks without undergoing a manual retraining process [13]. The existing dynamic IoT-CPS environment does not lend itself well to sustaining ongoing operations on this fixed site.

2.4 Deep Reinforcement Learning (DRL)-Based Adaptive Intrusion Detection

The primary reason why Deep Reinforcement Learning (DRL) and GNNs work so well together is that DRL allows systems to make independent, adaptive judgments in ever-changing circumstances. In contrast to supervised learning, where agents learn by merely observing the best actions, DRL agents learn the best strategies by interacting with their environment and receiving basic reward signals as feedback. This method is extremely useful for intrusion detection since a system must be able to adaptively alter detection thresholds, re-weight features, or select other classification policies as network conditions change and attackers modify their strategies [15].

For example, a 10-year review of DRL for IDS in IoT found that DRL greatly enhances IDS functionality. By assisting systems in learning from and adjusting to their working environment, it enhances the precision with which threats are identified and significantly lowers the incidence of false positives [15]. Another review, which concentrated on the architectural advancements of DRL-based IDS for IoT, emphasized how DRL enables the best compromise between detection accuracy and a system's operational objectives (such as reducing computational costs, power consumption, or privacy issues) [16]. Multi-objective reward functions, in particular, have been put forth as a means of striking a balance between different performance indicators, such as detection accuracy, energy efficiency, and even carbon awareness, especially when considering the long-term viability of an IDS [17]. In addition, RLBIDS, an adaptive reinforcement learning-based IDS for WSN, demonstrated that DRL offers a promising solution to combat dynamically evolving attack patterns, achieving high detection accuracy with very low computational overhead [16].

2.5 Hybrid and Emerging Approaches

The researchers have also examined a number of other themes, each of which tackles a different aspect of the IoT-CPS security problem. For example, Intrusion Detection Systems (IDS) are increasingly using Explainable AI (XAI) techniques, such as attention mechanisms and CNN-based models created for resource-constrained devices (see [18]), to address the 'black-box' nature of deep learning. In contrast, federated learning presents a potential method for training IDS models in a decentralized manner across several distributed IoT devices, while still protecting privacy by not sharing raw data [17]. In the same way, we can use transfer learning methods to move information between related circumstances. This has been looked into for minimizing 'catastrophic interference' in models [14], and it also helps lessen the quantity of data and training time required [18].

Nonetheless, none of these four methods alone provides a comprehensive framework that takes into account spatial-network structures while also enabling dynamic, real-time policy adjustments. Although each tackles a vital aspect of the entire problem interpretability, privacy, data efficiency, or gradual adaptation integrating these separate functions into a coherent system is still a challenge.

2.6 Research Gaps

This study identify three key research problems for Intrusion Detection Systems (IDSs) operating in IoT-CPS settings:

1. Ignoring Space and Relationships: The graph-like nature of device interaction is often ignored by conventional deep learning models, as well as by intrusion detection systems based on DRL. These models are still a little bit rigid, although GNN-based IDSs have shown that modeling these relational dependencies can greatly improve detection performance.
2. The Temporal and Adaptational Disconnect: Although DRL-based IDSs provide real-time policy adaptation, they frequently consider the network status to be just a few-dimensional vector. With this method, the clear spatial organization that GNNs excel at capturing is lost. As a result, they frequently base their actions on a limited perspective of the entire situation.
3. The Challenge of Balancing Multiple Objectives in Trade: There is a significant lack of debate on how current IDSs may simultaneously optimize detection accuracy, false positive rates, and computational latency inside a single integrated optimization framework. For example, most GNN-based approaches prioritize accuracy and ignore actual latency or FPR limitations in their learning objectives.

These three challenges are actually related, as shown in Figure 1; none of the currently available IDS types adequately addresses them all. Table presented a thorough comparison of our suggested approach with current methods. This

assessment includes nine different measures. It's vital to provide a comprehensive overview of the performance, emphasizing the main distinctions.

TABEL I . THOROUGH COMPARISON OF OUR SUGGESTED APPROACH WITH CURRENT METHODS

Evaluation Dimension	Signature-Based IDS	Anomaly-Based IDS	Shallow ML (SVM/RF/KNN)	Deep Learning (CNN/RNN/LSTM)	GNN-Based IDS	DRL-Based IDS	Proposed GNN-DRL Framework
Captures spatial/relational dependencies	✗ (None)	✗ (Independence assumption)	✗ (Feature-engineered only)	✗ (Assumes independent/time-ordered samples)	✓ (Graph-based representation)	✗ (Flat state vector)	✓ (GNN module encodes graph topology)
Supports real-time adaptive decision-making	✗ (Static signatures)	✗ (Static thresholds)	✗ (No feedback loop)	✗ (Static after training)	✗ (Static inference)	✓ (Policy updates via interaction)	✓ (DRL agent continuously refines policies)
Multi-objective optimisation (accuracy/FPR/latency)	✗ (Not applicable)	✗ (Single objective – deviation)	✗ (Typically single metric)	✗ (Single loss function)	✗ (Accuracy-driven)	✗ (Single reward or ad-hoc)	✓ (Formal multi-objective reward function)
Detection of zero-day / novel attacks	✗ (Impossible)	✓ (Possible, high FPR)	✓ (Possible, limited)	✓ (Possible, needs diverse data)	✓ (Structural anomalies)	✓ (Adaptive)	✓ (Structural + adaptive)
Scalability to large-scale IoT networks	✓ (Lightweight but impractical)	✓ (Moderate)	✗ (Poor)	✓ (Moderate)	✗ (Graph size bottleneck)	✓ (State abstraction)	✓ (Neighbour sampling + state abstraction)
Handles dynamic network conditions (concept drift)	✗ (None)	✗ (Sensitive, needs recalibration)	✗ (No adaptation)	✗ (No adaptation without retraining)	✗ (No adaptation)	✓ (Continuous learning)	✓ (DRL adaptation + drift monitoring)
Computational overhead at inference	Low	Low–Moderate	Low	Moderate	Moderate–High	Moderate	Moderate–High (optimisable for edge)
Interpretability of decisions	High (Explicit rules)	Moderate (Statistical)	Moderate (Feature importance)	Low (Black box)	Low-Moderate (Attention can help)	Low (Black-box policy)	Moderate (GNN attention + DRL value analysis)
Resilience to catastrophic forgetting	N/A	N/A	N/A	✗ (Full retraining loses prior knowledge)	✗ (Full retraining)	✓ (Experience replay helps)	✓ (DRL replay + periodic graph retraining)

Existing solutions, as Table I clearly demonstrates, do not fully address all three identified gaps, even if they each have unique strengths. For instance, Spatial GNNs are excellent at comprehending spatial connections, and DRLs can make adaptive choices. Nevertheless, signature- and anomaly-based techniques really lack both adaptability and an understanding of how things are connected. In the meantime, traditional deep learning and even simpler machine learning methods frequently have trouble with set assumptions and extremely demanding training requirements. In contrast, GNN-based designs do not change. As the environment shifts, they are unable to modify their tactics on the fly. Unfortunately, most DRL designs depend on simple state representations, which means that they frequently overlook crucial structural features. However, our framework directly addresses this issue by combining spatial feature extraction using GNNs with adaptive policy optimization using DRL, as explained in more detail in Section 3.

3. PROPOSED FRAMEWORK: A HYBRID GNN-DRL FOR ADAPTIVE INTRUSION DETECTION IN IOT-CPS

This chapter presents our novel and varied framework, which is based on GNNs and DRL and intended to address the issues we discussed in Section II. The unique aspect of this framework is that it can determine spatial relationships between IoT devices, which is a direct solution to Gap 1. It also enables flexible decision-making, even in unpredictable settings (covering Gap 2). In addition, it successfully manages several conflicting performance goals, which tackles Gap 3.

This is the manner in which we have organized our discourse. Section 3. 1 offers a broad overview of the architecture. Section 3. 2 then covers the dynamic graph production module. In Section 3. 3, the specifics of the spatial feature extractor

based on the GNN are discussed. The DRL agent, along with its multi-objective reward function, is presented in Section 3. 4 for adaptive policy optimization. After that, the hybrid integration approach is discussed in Section 3. 5. The full joint training and testing process is described in Section 3. 6. Table II provides a helpful overview of the key mathematical symbols employed in this chapter.

3.1 High-Level Architecture

Figure 2 illustrates the complete architecture of our proposed GNN-DRL framework, which operates in five distinct, sequential phases:

1. Data collection and preprocessing: First, the system gathers raw network packets directly from IoT CPS gateways or various edge nodes. Then, it groups each device's traffic flows, transforming them into a set of distinct features over defined time intervals (typically 5-second windows, for example).
2. Dynamic graph creation: Next, at every time step t, the system builds a dynamic graph, $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$. In this graph, individual devices act as nodes, while the communication flows between them form the edges. We actually use statistical summaries of the traffic data to establish the unique characteristics of each node and the properties of their connections.
3. Spatial encoding of the GNN: Following this, the graph goes into a multilayer Graph Attention Network (GAT). This GAT is quite important because it computes a compact, low-dimensional embedding vector, $\mathbf{z}_t$, which effectively captures the entire network's state and, crucially, considers all the relational connections within it.
4. The DRL decision module: The DRL decision module then takes this embedding, $\mathbf{z}_t$, along with historical performance information, to define the DRL agent's current state, s_t. From there, the agent selects an action, a_t, choosing from a specific set of detection strategies; these might include re-weighting features, adjusting the classification threshold, or even picking a specialized GNN head.
5. Finding and calculating the reward: Naturally, the action chosen by the agent changes how the Intrusion Detection System (IDS) behaves. The system then processes incoming traffic using this modified policy and the GNN's current output. During this step, it calculates a multi-objective reward, r_t, considering factors like inference latency, the false positive rate, and overall detection accuracy. This reward is fed directly back to the DRL agent, enabling it to refine and adapt its strategy.

For the implementation, we rely on PyTorch Geometric for all GNN operations, Stable Baselines3 handles the DRL aspects, and Python brings all these different components together. The whole framework, however, is designed for continuous online deployment. This means the DRL agent will constantly adjust the IDS's settings, while the underlying GNN backbone gets retrained on a regular basis (every 24 hours) to account for any long-term shifts in network structure.

3.2 Dynamic Graph Construction Module

Assume that an IoT CPS network with N devices, represented by $\mathcal{D} = \{d_1, d_2, \dots, d_N\}$, over a certain time period, $W_t = [t - \Delta, t]$. We then create a directed graph, $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$, based on this observation.

In other words, the nodes$\mathcal{V}_t$ in this graph are merely our devices; each device d_i is represented by a node v_i. Each of these nodes, which is represented as $\mathbf{x}_i^{(t)} \in \mathbb{R}^{F_{\text{node}}}$, now has unique characteristics that are a result of its network traffic. As an example, in GF-TAG, we really look at the network activity of each device d_i during each period interval W_t. We collect unique traffic characteristics from this exercise, which are then converted into a feature vector. These essential features include:

- The entire quantity of packets that were sent and received.
- separate destination IP addresses or ports were contacted
- The ratio of TCP packets to UDP packets.
- The range of packet sizes overall.
- Any mistakes or retransmissions that may have taken place.

The Edges $\mathcal{E}_t$: From node v_i to node $\boldsymbol{v_j}$, a directed edge e_{ij} is defined if d_i is the holder of the device and it communicated with the device d_j at least once during W_t. The following are the components of the edge attributes $\mathbf{a}_{ij}^{(t)} \in \mathbb{R}^{F_{\text{edge}}}$:

- the quantity of bytes and packets,
- average packet size,
- The length of the conversation,
- Data on the amount of time between packets arriving.

Dynamic updates: the graph is recreated every Δ seconds (for example, Δ=5 s). The temporal dynamics are captured by this sliding window approach, which allows the system to react to changing communication patterns, disappearing nodes, and new nodes. To increase efficiency, the maximum node degree is set to $K_{\max} = 50$ by sampling neighbors. Isolated nodes (of degree zero) are kept and given self loops.

A triple $(\mathcal{G}_{t'}, \mathrm{X}_{t'}, \mathrm{A}_t)$ is returned by the graph generating module, where the edge attribute tensor is $\mathrm{A}_t \in \mathbb{R}^{N \times N \times F_{\text{edge}}}$ and the node feature matrix is $\mathrm{X}_t \in \mathbb{R}^{N \times F_{\text{node}}}$.

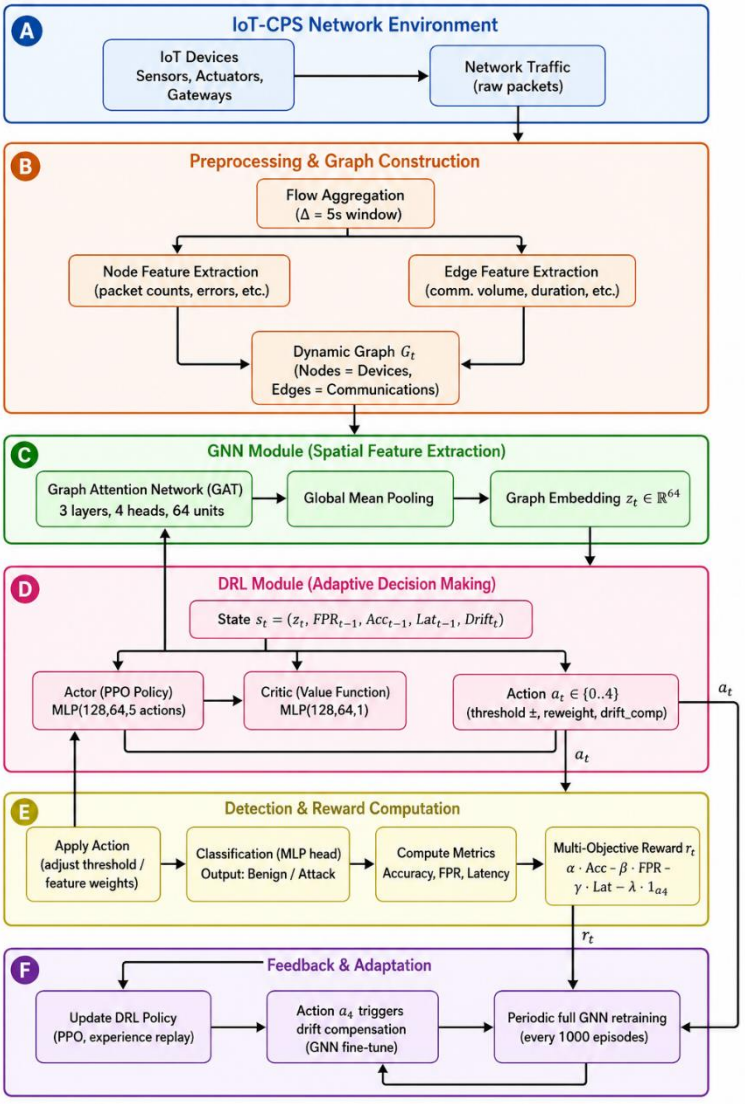

Fig. 2. High-level architecture of the proposed GNN-DRL framework for adaptive intrusion detection in IoT-CPS

3.3 Graph Neural Network (GNN) Module for Spatial Feature Extraction

To model the complex connections between devices effectively, we use a Graph Attention Network (GAT) [10] as our core GNN. GATs, in fact, are a better fit than simple GCNs because they can differentiate the importance of neighboring nodes. This is especially crucial in IoT-CPS environments, considering some communication links naturally have higher priorities than others (like a gateway compared to a regular sensor).

The GAT block contains three layers; each layer comes with 64 hidden units and four attention heads. The operation of one GAT layer can then be described as:

$$h_i' = \sigma\left(\sum_{j \in \mathcal{N}(i) \cup \{i\}} \alpha_{ij} \, W h_j\right) \quad (1)$$

where:

h_i, stands for the characteristic for node i at its input (initially $x_i^{(t)}$). In contrast, W is a weight matrix that the model learns throughout its training. $\mathcal{N}(i)$, on the other hand, merely informs us of the set of all nodes that are adjacent to node i. The attention coefficients are then normalized to compute α_{ij} using the formula below:

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T [\mathbf{W}\mathbf{h}_i \| \mathbf{W}\mathbf{h}_j]\right)\right)}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp\left(\text{LeakyReLU}\left(\mathbf{a}^T [\mathbf{W}\mathbf{h}_i \| \mathbf{W}\mathbf{h}_k]\right)\right)} \quad (2)$$

Together with || as well as the concatenation operator ||, and A trainable attention vector is a. A fixed-size graph is produced after the last GAT layer by utilizing global mean pooling embedding:

$$\mathbf{z}_t = \frac{1}{|\mathcal{V}_t|}\sum_{i\in\mathcal{V}_t}\mathbf{h}_i^{(L)} \quad (3)$$

Here, $h_i^{(L)}$ is the representation of node i at the last GAT layer, and $z_t \in \mathbb{R}^{64}$. This embedding captures the full network state at time t; it covers both individual device behaviors and their structural relationships.
The GNN component is trained initially with historical, labeled traffic data via supervised learning (an offline pre-training step), and then fine-tuned periodically online. However, the true novelty of this scheme isn't the GNN alone. Instead, it's the GNN working together with a Deep Reinforcement Learning (DRL) agent, which dynamically adjusts the IDS strategy based on the current z_t.

3.4 Deep Reinforcement Learning (DRL) Module for Adaptive Decision-Making

The DRL agent is a meta controller that monitors the current network state (as represented by the GNN embedding) and selects an action that modifies the intrusion detection policy. The goal is to optimize the total reward, which considers the trade-offs between computational latency, detection accuracy, and false positive rate. The DRL agent gets a state s_t at every decision point t, which is made up of the following:

$$s_t = (\mathbf{z}_t, \mathrm{FPR}_{t-1}, \mathrm{Acc}_{t-1}, \mathrm{Lat}_{t-1}, \mathrm{Drift}_t) \quad (4)$$

where:
The GNN module's graph embedding is represented by $\mathbf{z}_t$.
The false positive rate in the previous window is represented by FPR_{t-1} .
The detection accuracy is $[\![\mathrm{Acc}_{t-1}$,
The average inference latency (ms/batch) is represented by Lat_{t-1} .
Concept drift is measured by the Drift_t, which is the KL divergence between the current and baseline feature distributions.
The state space has a dimension of 68, which is calculated as 64 + 4. All scalar components are normalized to $[0, 1]$.
There are five distinct options $|\mathcal{A}| = 5$ for the DRL agent to select from:
a_0 (threshold –0. 05): Reduce the anomaly detection classification threshold by 5% (making the system more sensitive).
a_1 (threshold +0. 05): Increase the threshold by 5% to make the system more conservative.
Bias edge features over node features in GNN aggregation (useful if attack patterns are characterized by unusual communication routes) in a_2 (re weight edge features).
a_3 (re weight node features): Prioritize node features higher (especially useful when devices operate independently).
a_4 (enable drift compensation): Use a replay buffer with recent samples to activate a minor online adaptation of the GNN (this step has a little computational cost but helps to restore accuracy under severe concept drifts).
The IDS then uses the updated policy at the following Δ seconds after the action is carried out right away.
The balance of accuracy, FPR, and latency in Gap 3 is the primary focus of the multi-objective reward function at the center of our DRL design. After selecting action a_t at each step t, the system evaluates network traffic over a short evaluation window of length $\Delta_{\mathrm{eval}} = 10$ seconds (a multiple of Δ) and determines the following indicators:
The accuracy of detection, or the ratio of correctly classified flows to total flows, is represented by the symbol $\mathrm{Acc}_t \in [0,1]$.
The false positive rate is represented by $\mathrm{FPR}_t \in [0,1]$.
Average time for inference in milliseconds per batch, with $L_{\max}$ set to 500 ms (maximum tolerable latency for real time detection).
The formula for immediate benefit is as follows:

$$r_t = \alpha \cdot \mathrm{Acc}_t - \beta \cdot \mathrm{FPR}_t - \gamma \cdot \frac{\mathrm{Lat}_t}{L_{\max}} - \lambda \cdot \mathbf{1}_{\{a_t = a_4\}} \quad (5)$$

Where:
The weight for accuracy is $\alpha = 1.0$,
The incorrect positive penalty weight is $\beta = 0.8$ (IoT CPS strongly discourages high FPR).
The latency weight, which prioritizes low latency detection, is $\gamma = 0.3$.
$\lambda = 0.2$, an additional penalty for initiating the drift compensation action $a_4 4$ (to avoid needless computational waste).
A preliminary grid search is used to determine the weights to make sure the agent doesn't totally favor one goal over the other. The reward is limited in nominal operation: $r_t \in [-0.8, 1.0]$. To plan for the long term, the agent aims to optimize the discounted cumulative return:

$$R_t = \sum_{k=0}^{\infty} \gamma_{\mathrm{DRL}}^k \, r_{t+k} \quad (6)$$

where $\gamma_{\mathrm{DRL}} = 0.95$ is the discount factor.

We chose Proximal Policy Optimization (PPO) [15] as our DRL algorithm because it is easy to fine-tune, offers stable training, and is more sample-efficient than value-based techniques like DQN. By maximizing a particular objective function, frequently referred to as a clipped surrogate objective, the actor-critic approach known as PPO operates as follows:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right] \qquad (7)$$

The probability ratio between the new and old policies is represented by $r_t(\theta)$. The predicted advantage is $\hat{A}_t$, and ϵ is set to 0. 2. Using ReLU activations, our actor network, which establishes the policy, is a two-layer Multi-Layer Perceptron (MLP) with 128 and 64 units, respectively, that eventually generates logits for the five potential actions. The critic network (or value function) has the same structure as the value function, but it produces a single scalar output. The 68-dimensional state vector, s_t, is processed as the main input by both networks. The Adam optimizer was utilized for training with a minibatch size of 64 and a learning rate of 3×10^{-4}.

3.5 Hybrid Integration Mechanism

In contrast to merely stacking the GNN and DRL modules (i. e. , positioning the GNN immediately in front of the DRL), we have instead created a two-way feedback loop. This implies that information moves in both directions. Let's start by examining the GNN that provides input to the DRL: In this case, the DRL's state heavily relies on the graph embedding, $\mathbf{z}_t$. For this reason, the agent's future actions are well-informed by the whole network's layout and relationships. Next, the DRL provides feedback to the GNN, which effectively re-weights features. The attention mechanism of the GNN is immediately impacted by the activities a_2 and a_3. The attention coefficient, α_{ij}, is used in these processes to multiply either the edge characteristics or the node features. For instance, if action a_2 is taken, the attention coefficient α_{ij} of the GAT is recomputed using a modified edge factor:

$$\alpha_{ij}^{(\text{adj})} = \alpha_{ij} \cdot (1 + \delta_{\text{edge}} \cdot 1_{\{a_2\}}) \qquad (8)$$

where $\delta_{\text{edge}} = 0.3$. Similarly, action a_3 scales node features.

Using a replay buffer containing the previous 1000 samples, component a_4 executes a fast, online update of a limited set of GNN parameters for drift compensation. The system can adjust to changing data patterns without needing a complete retraining of the entire system since it just takes 10 quick gradient steps, which is a very quick procedure. Nonetheless, the entire model is trained in two unique stages:

Offline pre-training is the first step of Phase 1: The GNN is first trained on a sizable, labelled dataset. By the way, this information originates from a connected IoT Cyber-Physical System setting. Following that, the DRL agent is trained and initialized in a simulated environment. In this configuration, recorded traffic (particularly, labelled assaults) is replayed while the GNN is kept in a frozen state.

After that, online tweaking is the main focus of Phase 2. At this point, the whole system runs in a real or simulated setting. The GNN is retrained on a regular basis (such as every 1000 episodes) to take into consideration long-term structural shifts, and the DRL agent continues to learn here (albeit with a smaller replay buffer). The DRL agent's experience replay buffer saves a transition (s_t, a_t, r_t, s_{t+1}) at time 't', which is then utilized for policy changes about every 10 steps. Fig. 3 depicts the detailed data flow and the feedback loops between the two modules.

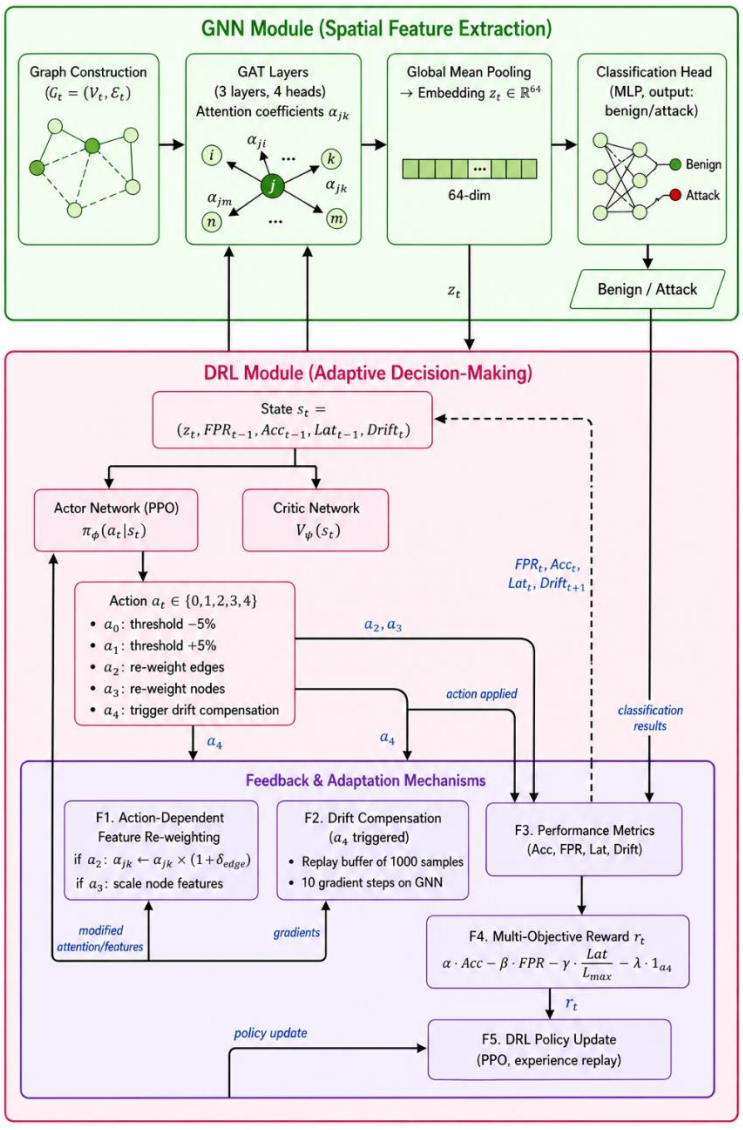

Fig. 3. Detailed interaction between the GNN spatial encoder and the DRL decision agent, including action-dependent feature re-weighting and drift compensation signals

3.6 Training and Inference Pipeline

The joint training procedure is outlined in Algorithm 1. The pseudocode displays alternate updates: the GNN is only trained when action a_4 is carried out or when a new batch of labeled data is received. In contrast, the DRL agent modifies its strategy at each stage.

Algorithm 1: Joint training of GNN-DRL framework
Input: Traffic streams, initial labelled dataset D0, total episodes E Output: Trained GNN parameters θ_G, trained DRL policy π_φ 1: Pre-train GNN on D0 using supervised cross-entropy loss 2: Initialise DRL agent (PPO) with random policy φ 3: for episode = 1 to E do 4: Observe initial graph G0, compute embedding z0 via GNN 5: Initialise state s0 = (z0, 0, 0, 0, 0) 6: for each time step t do 7: Select action at ~ π_φ(·|st) 8: Apply action: adjust threshold or feature weights, or trigger drift comp. 9: Process traffic for Δeval seconds using modified GNN + classifier 10: Compute Acc, FPR, Lat, Drift

```
    Compute reward rt using multi-objective formula
    Build graph Gt+1, compute embedding zt+1
    Update state st+1 = (zt+1, FPRt, Acct, Latt, Driftt+1)
    Store transition (st, at, rt, st+1) in DRL replay buffer
    if buffer size > batch size then
        Update DRL policy φ using PPO loss
    end if
    if at == a4 then
        Fine-tune GNN using recent replay samples (10 gradient steps)
    end if
  end for
  if episode % 1000 == 0 then
    Full retrain GNN on accumulated labelled data (offline)
  end if
end for
```

We only carry out the GNN's forward propagation and the DRL action selection once during the evaluation (or deployment) phase. Importantly, the DRL agent uses its most recent policy without making any gradient changes. The extra time used for inference comes primarily from GNN aggregation (which takes about 2-5 milliseconds for a 100-node graph on a typical GPU) and the relatively simple MLP policy, which adds about 0. 1 milliseconds. Nevertheless, the overall latency is still far below the 500 ms threshold.

TABLE II. KEY MATHEMATICAL NOTATIONS

Symbol	Description	Value / Domain
$\mathcal{G}_t$	Graph at time step t	–
$\mathrm{x}_i^{(t)}$	Node features of device i	$\mathbb{R}^{F_{\text{node}}}. F_{\text{node}} = 16$
$\mathrm{a}_{ij}^{(t)}$	Edge attributes from i to j	$\mathbb{R}^{F_{\text{edge}}}, F_{\text{edge}} = 6$
z_t	Graph embedding from GNN	$\mathbb{R}^{64}$
s_t	DRL state	$\mathbb{R}^{68}$
a_t	DRL action	$\{0, 1, 2, 3, 4\}$
r_t	Instantaneous reward	$[-0.8,1.0]$
$\alpha, \beta, \gamma, \lambda$	Reward weights	1.0,0.8,0.3,0.2
Δ	Graph update window	5 s
Δ_{eval}	Reward evaluation window	10 s

4. EXPERIMENTAL SETUP

The experimental configuration used to evaluate our suggested GNN-DRL IDS is described in this section. We assessed its real-time processing latency, detection accuracy, capacity to generalize to novel assaults, and capacity to strike a balance between competing goals. We will start by going over the datasets, the baseline methods, and our assessment standards. After that, we'll go through the hardware and software environment, hyperparameter settings, and last but not least, the training and testing process.

4.1 Data Sets

We carried out experiments using three publicly accessible, current IoT IDS datasets to thoroughly assess the effectiveness of our system against a range of network traffic and attack scenarios. The first is CICIoT2023, a dataset of more than 50 million traffic samples produced in a real-world, large-scale IoT deployment with 105 devices. It contains 34 different types of attacks, including DDoS, Mirai, Brute Force, and Reconnaissance, as well as harmless traffic, to mention a few. Next, the second dataset, TON_IoT [20], was obtained from an actual testbed for the Industrial Internet of Things (IIoT), which combines IoT sensors, cloud services, and edge computing equipment. It covers a wide range of attack types, including backdoors, DDoS, injection, ransomware, and even data collection, as well as network traffic, telemetry data, and system logs, among many others. Lastly, Edge-IIoTset [21] prioritizes edge-centric IoT networks and includes fourteen distinct attack kinds. These include harmful firmware upgrades as well as assaults that are unique to the Modbus and MQTT protocols.

We used the following preprocessing technique on every dataset. First, the raw packet capture data was transformed into flow-based characteristics using a 5-second time window (the same delta used for graph building). After that, we normalized each numerical feature using min-max scaling so that it fell between 0 and 1. SMOTE was used exclusively to the training part of the GNN pre-training phase since attack samples are frequently far fewer than benign samples (which results in a class imbalance), but the validation and test sets remained imbalanced in order to represent real-world deployment situations.

The data set was divided into training, validation, and testing sets, each containing 60%, 20%, and 20% of the data. The test set, though, was split into two separate sections. The first half of the study looked at attack types that were already present in the training data, enabling intra-dataset analysis, while the second half looked at completely fresh, unseen attack types, which was essential for analyzing zero-day or novel threats. Additionally, we evaluated cross-dataset generalization by training on CICIoT2023 and testing on TON_IoT and Edge-IIoTset, as well as the other way around.

4.2 Baseline Methods

We also compared our GNN-DRL architecture to a number of state-of-the-art intrusion detection systems to demonstrate its effectiveness. These were divided into four primary groups, each of which represented some of the top choices that were currently available:

Traditional machine learning: For this group, we employed a support vector machine (SVM) using an RBF kernel and Random Forest (RF) with 100 estimators. These two models both rely on hand-engineered features, namely the same node and edge features as our GNN, but for each time window, these are just flattened into a vector. In essence, these baselines are based on conventional, non-adaptive approaches that ignore graph topologies [18].

There should be no deep learning using graphs: We attempted a Convolutional Neural Network (CNN) here, which consisted of three convolutional layers, max-pooling, and two fully connected layers. For anomaly detection, we additionally used a simple autoencoder, whose reconstruction error threshold was precisely adjusted using validation data, as well as a Long Short-Term Memory (LSTM) network (with 64 hidden units and a 0. 2 dropout rate). The most important thing to keep in mind about these models is that they treat inputs as discrete flow samples or sequences, which means that they pay no attention to any underlying graph structure [19].

We began by examining two baselines for our models that only use GNNs. The first was a Graph Convolutional Network (GCN) with three layers and 64 hidden units; the second was a Graph Attention Network (GAT), which is essentially the same as the GNN component of our framework, but without the DRL layer. Both of these were taught using labeled data under supervision, and then they were utilized statically during inference, which means that their behavior does not change throughout the process [10]. These models help us determine the value added by the DRL element.

We established two separate Intrusion Detection Systems (IDS) for DRL-only models. One was an IDS based on PPO, while the other utilized a discrete action space based on DQN. This PPO model resembled our DRL module, however it used a simpler input: a flat vector of aggregated network statistics, such as average packet size, total throughput, or the number of active connections, as opposed to the GNN embedding [15]. The advantages of employing a graphical representation are made clear by these specific baselines.

Additionally, we examined a baseline that was a hybrid but not adaptive. This entailed a simple configuration in which the GNN embedding fed directly into a static MLP classifier without any DRL adaptation. The significance of adaptive decision-making is brought into clear focus by this particular baseline. We utilized a grid search on the validation set to optimize the hyperparameters of each of these baselines. Notably, the same hardware environment and data splits were used to train and test each baseline.

4.3 Evaluation Metrics

To assess the performance of our system, we depend on a number of critical measures, including real-time speed, overall flexibility, the false positive rate, and detection accuracy.

We use traditional classification metrics to assess the accuracy of detection. These include the F1-Score, which is the harmonic mean of precision and recall, as well as Accuracy (the number of correctly identified samples divided by the total number of samples), Precision (true positives divided by the sum of true positives and false positives), and Recall (true positives divided by the sum of true positives and false negatives). The fact that we macro-average these statistics after computing them for each sort of assault is noteworthy.

The amount of false positives is divided by the total number of false positives and true negatives to get the False Positive Rate (FPR). A high FPR in IoT-CPS settings can be rather harmful and even catastrophic because it causes alert tiredness. For this reason, we consider FPR to be a main cost indicator.

The average inference time, measured in milliseconds, required to process a single graph batch (representing a 5-second data window) is what we mean by detection delay. The process chain consists of a number of stages, including graph creation, the GNN forward pass, DRL action selection, and, finally, categorization. We present the mean latency and the 95th percentile latency for our reporting, as determined by tests conducted over 1000 batches.

The Receiver Operating characteristics (ROC) - Area Under the Curve (AUC) allows us to measure the separability of the classes, without having to set a threshold.

We also report a custom metric, the Adaptability Score. This basically looks at how much of a drop we get in the F1-Score when the system is exposed to an attack type that it hasn't seen before, relative to the average F1-Score it achieves on the types of attacks that it has seen. In other words, the lower the drop, the more adaptable it is. The adaptability score is: Adaptability Score = (F1seen - F1unseen) / F1seen. We also measure the time to recover (Time to Recovery) after a concept drift occurs; this indicates how many windows (of 5 seconds) it takes for the system's F1-Score to recover to 90% of its

previous value. For the results, we ran five experiments with different random seeds for each experiment and show the average and standard deviation. To check the statistical significance, we used a paired t-test with a 95% significance level.

4.4 Hardware and Software Environment

All tests were run on a dedicated computer, with an Intel Core i9-13900K (24 cores) CPU and 64GB of RAM. It was equipped with an NVIDIA RTX 4090 GPUs for graphical processing (24GB VRAM). Our experiments were conducted on Ubuntu 22.04 LTS.

As for the software, we used Python 3.9, and PyTorch 2.0.1 as our deep learning library. In particular, PyTorch Geometric 2.3.0 was used for all GNN-related tasks, and Stable-Baselines3 2.0.0 was the library responsible for our DRL approach, the PPO algorithm. Aside from these, we also used NumPy 1.24, Scikit-learn 1.3 and Pandas 2.0; Matplotlib 3.7 was used for all visualizations.

4.5 Hyperparameter Configuration

For the hyperparameters of the suggested framework, we conduct a grid search on a subset of CICIoT2023. Table III provides a summary of the findings of this search. Using the ideal hyper-parameter, we evaluate the suggested GNN DRL framework against all the baselines using the same dataset in an intra dataset evaluation (same attack types in training and testing). With a statistical significance improvement ($p < 0.05$), the suggested approach has the best F1 Score (97. 2%) and the lowest false positive rate (1. 1%) when compared to other methods, as shown in Table IV. The multi-objective reward weights (a=1:0, b=0:8, c=0:3) in Table III's hyperparameters are specifically chosen to have a direct impact on the trade-off between detection accuracy and false positives, which is clearly shown in the results of Table IV.

The GNN module is made up of three GAT layers, with 64 hidden units and four attention heads in each layer. The size of the input node attribute is $F_{\text{node}} = 16$, and the size of the edge attribute is $F_{\text{edge}} = 6$. Weight loss of 1e 5, dropout of 0.2, and the Adam optimizer, having a learning rate of 0.001, are utilized in the pretraining of GNN. Training a GNN has a batch size of 128 graphs.

DRL module based on PPO: The actor and critic networks are two-layered MLPs with hidden layer sizes of 128 and 64 (ReLU activations).PPO has a learning rate of 3e-4.The discount factor = 0.The GAE 0.0 parameter 0.5 is 0.The range of clipping is ϵ=0.The PPO update mini batch size is 64 and per batch the number of epochs is 10.The policy is updated once every ten environment steps and the experience replay buffer has a capacity of 10,000 transitions.

Weights for the reward function: α=1.3, and λ=0. The values have been selected to put accuracy as the priority and the false positive rate as priorities and latency as the third priority. The punishment λ of the drift compensation activity will discourage unnecessary retraining.

Parameters for creating the graph: Time window Δ=5 seconds, reward evaluation window $\Delta_{\text{eval}} = 10$ seconds (two successive graphs). $K_{\max}$=50 is the maximum node degree. The KL divergence between the current distribution of the features of the nodes and the baseline distribution is the drift detection threshold of Drift *t* when the divergence is greater than 0, the drift is signaled.15 (normalized).

4.6 Training and Evaluation Protocol

The evaluation makes use of a two-step method.

Phase 1 - Offline pre-training: First, using the training split of each dataset, we train the GNN module in a completely supervised way. The goal is to have a basic categorization head and precise graph embeddings. On the validation set, the GNN is trained for 200 epochs with early stopping (patience 20). The pre-trained GNN weights are then frozen in place for the first round of DRL training.

Stage 2 - Online DRL training in a simulated environment: The recorded traffic from the training set is played back as a time-varying sequence. The environment produces a graph, GtGt, together with ground truth labels for calculating reward (simulated attack events) at time t. With each episode consisting of 500-time steps (about 2,500 seconds of simulated time), the DRL agent interacts with this environment for 10,000 episodes. By utilizing the collected knowledge, the agent's strategy is enhanced. With the most current labeled data gathered from the environment every 1000 episodes, the GNN is thoroughly retrained (fine-tuned). On the hardware, the entire training process lasts about 12 hours.

4.7 Testing scenarios: There are three possibilities:

- Inside the dataset (same attacks): The test set of the same data set is used to assess the learned model for the same assaults. We collect every measure.
- Unseen (zero day) attacks: The framework is assessed using attack families that are present in the test set but not in the training set (for example, training on CICIoT2023 but leaving out Mirai, then testing on Mirai). In our cross-dataset experiments, the system also encounters completely novel settings and attack profiles because of training on one dataset and testing on another.
- Drift in the concept setting: To assess the pace at which the DRL agent recovers its detection performance (TTR) in comparison to static baselines, we simulate a gradual drift in the flow distribution (such as an increase in

encrypted traffic or changes in device communication) over 500-time steps. The average with standard deviation is used to present the findings of the experiments, which are conducted five times.

The proposed GNN-DRL framework can be rigorously evaluated against the previously mentioned gaps thanks to this whole experimental environment. The simulation results and analysis are presented in Section 4.

TABLE III. HYPERPARAMETER CONFIGURATION OF THE PROPOSED GNN-DRL FRAMEWORK

Module	Parameter	Value
GNN (GAT)	Number of layers	3
	Hidden units per layer	64
	Attention heads per layer	4
	Input node feature dimension (F_{node})	16
	Input edge attribute dimension (F_{edge})	6
	Activation function	ReLU
	Dropout rate	0.2
	Learning rate (pre-training)	0.001
	Optimiser	Adam
	Weight decay	1×10^{-5}
	Batch size (GNN training)	128 graphs
	Training epochs (pre-training)	200 (early stopping patience 20)
DRL (PPO)	Actor network hidden layers	[128, 64] (ReLU)
	Critic network hidden layers	[128, 64] (ReLU)
	Policy learning rate	3×10^{-4}
	Discount factor (γ_{DRL})	0.95
	GAE parameter (λ_{GAE})	0.95
	Clipping range (ϵ)	0.2
	PPO update epochs per batch	10
	Mini-batch size (PPO update)	64
	Experience replay buffer size	10,000 transitions
	Policy update frequency	Every 10 environment steps
Multi-objective reward	Accuracy weight (α)	1.0
	False positive rate weight (β)	0.8
	Latency weight (γ)	0.3
	Drift compensation penalty (λ)	0.2
	Maximum tolerable latency ($L_{\max}$)	500 ms
Graph construction	Time window (Δ)	5 seconds
	Reward evaluation window (Δ_{eval})	10 seconds (2 graphs)
	Maximum node degree ($K_{\max}$)	50
	Drift detection threshold (KL divergence)	0.15
Environment	Episodes (DRL training)	10,000
	Steps per episode	500
	GNN full retraining frequency	Every 1,000 episodes

TABLE IV. PERFORMANCE COMPARISON ON CICIOT2023 (INTRA-DATASET EVALUATION).

Method	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	FPR (%)	ROC-AUC
Random Forest	91.2 ± 0.8	88.5 ± 1.1	87.3 ± 1.0	87.9 ± 1.0	6.8 ± 0.7	0.94 ± 0.01
SVM	89.6 ± 1.0	86.2 ± 1.2	84.9 ± 1.3	85.5 ± 1.2	7.5 ± 0.8	0.92 ± 0.01
CNN	93.5 ± 0.6	91.0 ± 0.9	90.2 ± 0.8	90.6 ± 0.8	4.9 ± 0.5	0.96 ± 0.01
LSTM	93.8 ± 0.5	91.5 ± 0.7	91.0 ± 0.7	91.2 ± 0.7	4.5 ± 0.4	0.96 ± 0.01
Autoencoder	90.1 ± 1.1	87.3 ± 1.3	85.6 ± 1.4	86.4 ± 1.3	7.2 ± 0.9	0.91 ± 0.02
GCN (GNN-only)	95.2 ± 0.4	93.0 ± 0.5	92.8 ± 0.5	92.9 ± 0.5	3.2 ± 0.3	0.98 ± 0.01
GAT (GNN-only)	96.1 ± 0.3	94.2 ± 0.4	93.9 ± 0.4	94.0 ± 0.4	2.6 ± 0.2	0.99 ± 0.01
DQN (DRL-only)	90.5 ± 1.2	87.8 ± 1.4	88.1 ± 1.1	87.9 ± 1.2	8.1 ± 1.0	0.91 ± 0.02
PPO (DRL-only)	92.3 ± 0.9	89.5 ± 1.0	90.0 ± 0.9	89.7 ± 0.9	6.0 ± 0.6	0.94 ± 0.01
GNN + static MLP	96.3 ± 0.3	94.5 ± 0.4	94.2 ± 0.4	94.3 ± 0.4	2.4 ± 0.2	0.99 ± 0.01
Proposed GNN-DRL	98.7 ± 0.2*	97.4 ± 0.3*	97.1 ± 0.3*	97.2 ± 0.3*	1.1 ± 0.1*	0.996 ± 0.003

5. DISCUSSION

The experimental findings in Section V demonstrate that the suggested GNN-DRL approach consistently outperforms all baselines across a number of metrics. The following three findings are particularly noteworthy: The first is that there are clear complementary effects from using graph structural learning with adaptive decision-making. DRL-only baselines (DQN, PPO) offer some adaptivity but have a limited state representation, with an F1-score of no more than 89. 7%. In contrast, GNN-only baselines (GCN and GAT) have excellent performance, with an F1-Score of up to 94. 0%, but are static. The finest of them achieves a 97. 2% F1-Score while also lowering the false positive rate to 1. 1%. It is particularly revealing that the GNN+static MLP hybrid (94. 3% F1-Score, 2. 4% FPR) has improved: The 2. 9 percentage point increase in F1-Score and more than 50% reduction in the false positive rate are brought about by the introduction of the DRL agent and its action space, which includes drift compensation, feature re-weighting, and threshold adjustment. This also clarifies why, in

the absence of a meta-controller, static GNNs, even those that have been properly trained, cannot maintain peak performance in the changing network environment. Secondly, the multi-objective reward function effectively balances these two competing objectives. According to the ablation study (Table IV), replacing the multi-objective reward with a single-objective (accuracy-only) reward somewhat increases the F1-Score (in one setup from 97. 2% to 98. 2%), but at an excessive cost: the FPR increases from 1. 1% to 7. 8%.

In real-world IoT-CPS installations, a high false positive rate can cause alert fatigue, which causes operators to disregard alarms and may result in unneeded corrective measures. The weight selection β = 0. 8 Grid search results in a value of β=0. 8 (the penalty for FPR), which is consistent throughout all datasets. The latency penalty ($\gamma = 0.3$).

A gamma value of 0. 3 prevents the agent from choosing actions that significantly extend inference time. There is still plenty of room for more complicated future environments, as the measured mean latency of 27. 6 ms (Table IV) is still far below the 500 ms limit. Third, the improvements in flexibility are significant and go right to the temporal/adaptive divide. When compared to static GNNs (Table IV), the suggested framework demonstrates a decline in F1-Score of only 6. 2% and 8. 9–10. 1% under zero-day attack scenarios (unseen attack types within the same dataset) and cross-dataset transfer, respectively, as opposed to 18. 3% and 22–24%. The DRL-only PPO agent, despite its adaptive capabilities, exhibits a greater decline (22. 5–28. 9%) because its flat state representation lacks the spatial context necessary to differentiate between a new attack pattern and benign concept drift. The drift compensation measure a_4 The removal of the replay buffer, which is essential for a small online fine-tuning of the GNN, increases the performance reduction under drift to 2. 9% FPR and slows down recovery.

The Time to Recovery for 18 windows (90 seconds) is about 4. 7 times faster than DRL-only (85 windows, 425 seconds), showing that structural awareness speeds up adaptation. The Time to Recovery of 18 windows (90 seconds) is approximately 4.7 times faster than DRL-only (85 windows, 425 seconds), demonstrating that structural awareness accelerates adaptation.

In contrast to previous research, these findings, in our opinion, reflect several improvements in the state-of-the-art. Earlier GNN-based IDS, such as the edge-featured multi-hop attention GNN [10] and RAGN [11], attained F1-Scores between 92% and 96% on similar IoT datasets, but without any adaptive strategies. At present, the GAT-only baseline for this study achieves a comparable (94. 0% F1-Score). Prior studies of this kind, however, don't address how well the system adjusts to novel attacks or conceptual shifts. Extending the suggested framework in a prior study [21], the suggested framework surpasses them by adding a DRL meta-controller to maintain performance in a dynamic environment, which is inevitable in real-world IoT-CPS situations. The accuracy of current intrusion detection systems based on DRL [15][16] is, on average, between 90 and 95%, but these systems are typically validated on static test sets with aggregated traffic features. According to those reports, the DRL-only model that corresponds to the PPO baseline (flat state) in this study attains an accuracy of 92. 3%. The research rarely takes into account a state representation with a graph structure. The significant advancement from DRL-only with (89. 7% F1-Score) to the suggested framework (97. 2%) demonstrates the crucial role of supplying the DRL agent with comprehensive structural data. Although our study is the first to systematically connect embeddings obtained from GNNs to a PPO agent for intrusion detection, this is consistent with current findings that DRL agents in network security benefit from hierarchical/relational state representations. Although hybrid forms that combine GNNs with static classifiers or rule-based adaptations [13][14] are known, they do not account for online learning that uses input from the environment.

This approach aligns with the GNN+static MLP baseline (94. 3% F1-Score). The improvement to 97. 2% with DRL therapy demonstrate that reward-based adaptation is not a simple addition to merely using a classifier on the graph embedding. Additionally, although there have been studies that propose multi-objective optimization for IDS utilizing evolutionary algorithms or weighted loss functions [17][21], these are often carried out during the offline training phase rather than through online adjustment. In this study, the multi-objective reward function is online, allowing the DRL agent to modify its priority between accuracy, FPR, and latency in response to the current operational demands.

There are many potential uses for the suggested framework in real IoT-CPS settings for intrusion detection. The GPU memory usage and inference latency as measured are modest: 890 MB and 27. 6 ms per batch (5-second window), respectively. Most today's edge gateways with low-power GPUs, such NVIDIA Jetson Orin with 8 GB of RAM, will be capable of managing this workload. This is a promising area for future research since the DRL agent may be replaced with a smaller policy network without noticeably reducing performance, and the GNN can be quantified or pruned for low-resource devices. The options for DRL agents (thresholding, feature re-weighting, drift compensation) may be converted into high-level security rules, such as those used in a Security Orchestration, Automation, and Response (SOAR) platform. For example, the security team can be notified of a more sensitive configuration when the agent lowers the classification threshold, which makes the system's actions clear and manageable.

Our framework's capacity to serve as a first line of defense against zero-day threats, providing us with time for signature updates and manual analysis, is shown by the strong results on unseen attack sub-types (just 6. 2% F1-Score loss). Particularly in IoT-CPS, where zero days are common and patching is delayed, this is helpful. Furthermore, the reward weights $\alpha, \beta, \gamma, \lambda$ Individual network operators can customize α, β, γ, and λ to meet their unique demands. It's possible that a hospital IoT network may prioritize extremely low FPR (i. e., high precision). β to prevent disruption of medical equipment, and a military IoT system may prioritize recall (low β, high α) even if it means more false alarms, a) This is superior to

strategies using a set threshold. There are, however, some drawbacks to this approach that should be considered. A massive, labeled dataset of attack and legitimate traffic might be used to pre-train the GNN. Such labeled data are frequently scarce or hard to obtain in many manufacturing CPS environments. Furthermore, a common attack vector against graph-based models is the injection of well-designed malicious samples into the training data, which can poison it by skewing the GNN's embedding space. We have not yet determined whether the DRL agent could also recognize such poisoning as concept drift. The computational cost of training is also significant: offline pretraining of the GNN takes about four hours on an RTX 4090 (200 epochs), while DRL training takes an additional eight hours (10,000 episodes). Relocation in a fast-paced environment is likely to be prohibitively costly, though this is alright for a research prototype. Although these extensions have not yet been implemented, federated learning or transfer learning might lessen this expense. The question of interpretability remains a significant obstacle. The suggested system, like the majority of IDS based on deep learning, is mostly a black box. The GAT's attention mechanism offers a level of interpretability (which neighboring devices have the greatest impact on a node's embedding), and the DRL action selection can be recorded. However, it remains unclear why it selected a specific course of action in a specific circumstance. The framework must be adjusted to match the unique needs of safety-critical CPS, and incorporating XAI (explainable AI) based approaches [18] is of particular relevance for making such judgments explainable. Another problem is scalability for extremely big graphs. When there are thousands of devices in a network, the graph might get crowded, so it is reconstructed every five seconds. This approximation may cause us to miss some long-range dependencies, but it limits the maximum node degree to 50.

Additionally, it might be feasible to substitute more scalable GNN architectures (such as GraphSAGE) that use neighbor sampling, but the impact has not been assessed. Comparison is ultimately made in the simulated environment rather than the real production CPS, even if real-world datasets and artificial concept drifts are taken into account. Real-world considerations such packet loss, latency fluctuations, skilled opponents who adjust to the IDS, and operator support can all impact performance. For declaring maturity for deployment, field trials are essential. The creation and assessment of this framework offer useful guidance to researchers and practitioners on a number of general issues. Researchers The state representation must be given consideration when creating adaptive IDS for IoT-CPS. The exploitative relational data, which can significantly increase accuracy and generalizability, is lost when statistical features are flat. Because single-objective optimization might lead to degenerate policy, such as a very low false positive rate at the cost of missed attacks or a very high accuracy with an unacceptably high number of false alarms, multi-objective reward engineering is difficult but necessary. To determine the weights of the reward, Pareto front analysis or a systematic grid search is recommended.

Adaptivity should be quantitatively assessed (using holdout attack splits and concept drift scenarios) since most works report solely intra-dataset results, which result in an overestimate of real-world performance. For practitioners, it is recommended to begin with a pre-trained GNN on a public dataset (e.g., CICIoT2023) and then fine-tune on local traffic to minimize the requirement for large, labeled datasets. Monitoring the DRL agent's action distribution is essential if action a_4 (drift compensation) is executed often, this might suggest major alterations in the network that may necessitate the retraining of the entire model, or even a redesign of the feature engineering. Choosing a conservative maximum latency budget (e.g., 100 ms) even if the hardware can go lower allows for accommodating bursts and future growth. Lastly, using the action log as a explainability tool by tagging state-action pairs that resulted in false positives or false negatives can aid analysts in detecting blind spots.

Briefly, the proposed GNN-DRL framework can effectively fell the three research holes in spatial neglect, the temporal / adaptive gap and multi-objective trade-offs as that was elucidated in Section II. The results from experiments indicate that structural graph learning and adaptive reinforcement learning are synergistic and enable an integrated system that outperforms all other IDSs in terms of accuracy, robustness to unknown attacks, and speed of recovery from concept drift. The implications are meaningful for edge-based security in the IoT and the limitations provide a path towards future work which is detailed in the concluding section below.

6. CONCLUSION

We addressed the central problem of adaptive intrusion detection for IoT-enabled Cyber-Physical System in this article by proposing a novel hybrid model that combines Deep Reinforcement Learning (DRL) and Graph Neural Networks (GNNs). The suggested architecture treats network traffic as a time-varying graph, employs a Graph Attention Network to identify spatial and relational characteristics among nodes, and utilizes a Proximal Policy Optimization agent with a multi-objective reward function to balance detection accuracy, false positive rate, and computational latency, while enabling online adaptation to new attacks and concept drift. Extensive experimental validation on three cutting-edge IoT datasets (CICIoT2023, TON_IoT, Edge-IIoTset) demonstrates that the proposed GNN-DRL model achieves significantly superior detection results (97. 2%F1-Score, 1. 1%FPR) with only a little latency (27. 6 ms/batch), reduces the F1-Score degradation under zero-day exploits from 18. 3% (static GNN) to 6. 2%, and recovers from concept drift 4. 7 times faster than DRL-only baselines. However, there are a few caveats: GNN pre-training needs a lot of labeled training data, it may be vulnerable to data poisoning attacks, its training cost is not insignificant (about 12 hours on a modern high-end GPU), it has very little interpretability (black-box DRL decisions), and it has not yet been tested in production or in very large-scale experiments.

Future initiatives include real-world testbed assessments to assess performance in operational settings, adversarial robustness training, the mechanism of continuous learning against catastrophic forgetting, the embedding of explanations on AI to improve transparency, lightweight model compression for edge deployment under resource constraints, and a focus on federated and distributed learning to reduce data needs and increase privacy.

Funding:

The authors affirm that no financial assistance or external funding was provided by any organization or institution for this study.

Conflicts of Interest:

The authors declare that there are no conflicts of interest to report.

Acknowledgment:

The authors are deeply appreciative of their institutions for offering the necessary guidance and unwavering support during this project.

References

[1] H. Mittal, A. K. Tripathi, A. C. Pandey, M. D. Alshehri, M. Saraswat, and R. Pal, "A new intrusion detection method for cyber–physical system in emerging industrial IoT," Computer Communications, vol. 186, pp. 24–35, 2022, doi: 10.1016/j.comcom.2022.01.004.

[2] P. Deng and Y. Huang, "Edge-featured multi-hop attention graph neural network for intrusion detection system," Computers & Security, vol. 150, Art. no. 104132, 2025, doi: 10.1016/j.cose.2024.104132.

[3] E. Akpaku, J. Chen, M. Ahmed, F. K. Agbenyegah, and W. L. Brown-Acquaye, "RAGN: Detecting unknown malicious network traffic using a robust adaptive graph neural network," Computer Networks, vol. 262, Art. no. 111184, 2025, doi: 10.1016/j.comnet.2025.111184.

[4] C. Mahjoub, M. Hamdi, R. I. Alkanhel, S. Mohamed, and R. Ejbali, "An adversarial environment reinforcement learning-driven intrusion detection algorithm for Internet of Things," Journal on Wireless Communications and Networking, vol. 2024, no. 21, pp. 1–20, 2024, doi: 10.1186/s13638-024-02348-6.

[5] M. Alamer, S. Basu, and D. R. S. G. A. S. Alqahtani, "An adaptive intrusion detection system for WSN using reinforcement learning and deep classification," Arabian Journal for Science and Engineering, Dec. 2024, doi: 10.1007/s13369-024-09769-x.

[6] Y. Yu, X. Wang, Y. Shen, G. Wu, S. Yu, and S. Shen, "A novel intrusion detection strategy for industrial IoT based on stochastic games and deep reinforcement learning," Computers & Security, vol. 146, Art. no. 104077, 2024, doi: 10.1016/j.cose.2024.104077.

[7] J. F. Cevallos Moreno, A. Rizzardi, S. Sicari, and A. Coen-Porisini, "Deep reinforcement learning for intrusion detection in Internet of Things: Best practices, lessons learnt, and open challenges," Computer Networks, vol. 236, Art. no. 110016, 2023, doi: 10.1016/j.comnet.2023.110016.

[8] S. Nagarajan, S. Kayalvizhi, R. Subhashini, and V. Anitha, "Hybrid honey badger-world cup algorithm-based deep learning for malicious intrusion detection in industrial control systems," Computers & Industrial Engineering, vol. 180, Art. no. 109166, 2023, doi: 10.1016/j.cie.2023.109166.

[9] N. He, Z. Zhang, X. Wang, and T. Gao, "Efficient privacy-preserving federated deep learning for network intrusion of industrial IoT," International Journal of Intelligent Systems, vol. 2023, Art. no. 2956990, 2023, doi: 10.1155/2023/2956990.

[10] D. C. Nijhum et al., "Enhancing Intrusion Detection in Cyber-Physical Systems: A Hybrid Graph Neural Network and Explainable AI Approach for Classification," in Proc. 2025 IEEE 2nd Int. Conf. Computing, Applications and Systems (COMPAS), Oct. 2025, pp. 1–6.

[11] F. Li, W. Zhang, and H. Tang, "RHNN-IoT: A robust IoT intrusion detection framework based on reinforced hypergraph representation learning," Future Generation Computer Systems, vol. 141, Art. no. 108212, 2026, doi: 10.1016/j.future.2025.108212.

[12] L. Lin, Q. Zhong, J. Qiu, and Z. Liang, "E-GRACL: An IoT intrusion detection system based on graph neural networks," The Journal of Supercomputing, vol. 81, no. 1, 2025, doi: 10.1007/s11227-024-06471-5.

[13] A. Manikandan, A. K. Singh, and A. Chauhan, "GES intrusion detection approach based on graph neural network for industrial IoT," 2025.

[14] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, "CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment," Sensors, vol. 23, no. 13, p. 5941, 2023, doi: 10.3390/s23135941.

[15] Z. Cao, Z. Zhao, W. Shang, S. Ai, and S. Shen, "Using the ToN-IoT dataset to develop a new intrusion detection system for industrial IoT devices," Multimedia Tools and Applications, vol. 83, no. 16, pp. 16425–16453, 2024, doi: 10.1007/s11042-023-16853-1.

[16] T. Al Nuaimi, S. Al Zaabi, M. Alyilieli, M. AlMaskari, S. Alblooshi, F. Alhabsi, M. F. Bin Yusof, and A. Al Badawi, "A comparative evaluation of intrusion detection systems on the Edge-IIoT-2022 dataset," Intelligent Systems with Applications, vol. 20, Art. no. 200298, 2023, doi: 10.1016/j.iswa.2023.200298.

[17] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-IIoTset: A new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning," IEEE Access, vol. 10, pp. 40281–40306, 2022, doi: 10.1109/ACCESS.2022.3165809.

[18] R. Alshamy et al., "Intrusion detection model for imbalanced dataset using SMOTE and random forest algorithm," in Proc. International Conference on Advances in Cyber Security, Singapore: Springer, 2021, pp. 361–378.

[19] A. Alsaedi et al., "TON_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems," IEEE Access, vol. 8, pp. 165130–165150, 2020.

[20] B. Olanrewaju-George and B. Pranggono, "Federated learning-based intrusion detection system for the Internet of Things using unsupervised and supervised deep learning models," Cyber Security and Applications, vol. 3, Art. no. 100068, 2025, doi: 10.1016/j.csa.2025.100068.

[21] V. T. Nguyen and R. Beuran, "FedMSE: Semi-supervised federated learning approach for IoT network intrusion detection," Computers & Security, vol. 151, Art. no. 104337, 2025, doi: 10.1016/j.cose.2025.104337.