TopoGNN: A topology-aware GNN framework for uncovering IoT attack signatures

Ki On Chan, Yoohwan Kim, Shaikh Arifuzzaman

***Abstract*—The rapid proliferation of Internet of Things (IoT) devices in critical infrastructure increases cybersecurity risks, necessitating advanced intrusion detection for heterogeneous networks. This study bridges graph topology analysis with Graph Neural Networks (GNNs) through four key contributions: (a) a comprehensive graph analysis of the ToN-IoT dataset and IoT attack surfaces, quantifying triangular subgraphs, unweighted-weighted diameter contrast, community segmentation, and degree distributions; (b) experimental evaluation of Graph Convolutional Network (GCN), Graph SAmple and aggreGatE (GraphSAGE), and Graph Attention Network (GAT) models. These models are enhanced by a dual imbalance mitigation strategy combining class weighting and focal loss to detect rare threats, yielding detailed performance reports; (c) topology-driven diagnostics revealing ransomware's tri-modal connectivity, exclusive scanning from hubs, confusion between Password/Injection/XSS attacks induced by triangular subgraphs, and model-specific vulnerabilities like GAT's bridge-edge precision gaps; and (d) derivation of a topology-aware framework. This framework translates insights from (a) and diagnostics from (c) into operational rules, including bridge-edge validation, degree stratification, triangle monitoring, and threat-specific thresholds. These insights collectively inform the proposed framework, advancing IoT cybersecurity through topology-aware detection. Future work will focus on operational validation in critical infrastructure environments such as industrial control systems (ICS), and adapting to evolving network topologies.**

***Index Terms*—Graph Neural Networks, Intrusion Detection, IoT Security, Network Topology Analysis, Class Imbalance Mitigation**

I. Introduction

The escalating frequency and sophistication of cyberattacks targeting the Internet of Things (IoT), particularly IoT-enabled critical infrastructure such as smart grids, transportation networks, and healthcare systems, underscores serious security challenges in these interconnected ecosystems [1]. While IoT integration improves efficiency, it also expands the attack surface through heterogeneous protocols, resource-constrained edge devices, and dynamic communication patterns. Adversaries exploit these weaknesses using methods including zero-day exploits and multi-stage lateral movement campaigns across networked devices. This method of linking successful attacks across multiple systems is seen in cyberattacks like Black Basta's 2022–2024 ransomware campaigns, which exploited Active Directory (AD) for lateral movement within enterprises [2]. While IoT environments lack centralized directories like AD, they face structurally similar lateral movement threats, where attackers pivot between devices such as sensors, controllers, and servers by leveraging protocol weaknesses or device vulnerabilities.

Conventional Intrusion Detection Systems (IDS) struggle with these threats due to limited contextual analysis. This limitation is compounded by many legacy systems lacking defenses against coordinated attacks, and rule-based approaches struggling with IoT-specific challenges like protocol diversity. Furthermore, most machine learning methods process events in isolation, ignoring contextual relationships such as attack chains. This limitation is critical in IoT environments where threats emerge through dispersed interaction patterns. Fig. 1 illustrates such an attack chain that challenge conventional IDS that miss contextual relationships [3] [4].

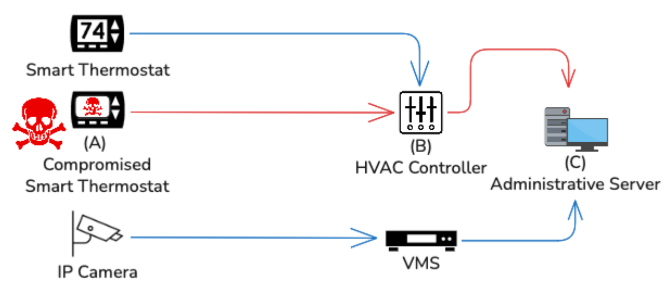


Fig. 1. Multi-stage lateral movement campaign within an IoT network, contrasting with normal operational traffic. Normal connections in blue lines include an uncompromised Smart Thermostat communicating with the HVAC Controller (B), and an IP Camera sending data to the Video Management System (VMS). Both the HVAC Controller (B) and VMS connect to the Administrative Server (C) for central management. The attack chain in red lines illustrates an adversary's progression from a compromised Smart Thermostat, pivoting through the HVAC Controller (B), to reach the Administrative Server (C). This exemplifies how multi-stage lateral movement creates contextual threats that evade conventional IDSs by mimicking legitimate interactions.

Graph Neural Networks (GNNs), particularly Graph Attention Networks (GATs), offer a transformative approach by modeling IoT ecosystems as graphs. Nodes represent entities such as devices and users. Edges encode interactions such as network flows. GATs are particularly effective because they can propagate contextual information across these connections. This helps identify attack patterns like those shown in Fig. 1. By analyzing node features including device type and traffic patterns along with graph topology, GATs compute attention scores indicating relationship importance. Based on this learned representation, the model computes anomaly scores that identify deviations from normal patterns. These scores flag high-risk attack paths when scores exceed predetermined thresholds. Such analysis detects both localized anomalies and global attack patterns such as botnet activity [5].

Despite this significant potential for detecting contextual threats in IoT, recent GNN research often prioritizes architectural novelty over practical deployment. This can create significant operational gaps for IoT security [4]. Many models use limited datasets, lack class-level performance analysis, and overlook the crucial role of topological properties in anomaly detection. These limitations hinder model improvement and effective real-world application in dynamic IoT environments.

To address some of these limitations, our work delivers four key advances:

a) **Graph Analysis**: Comprehensive study of the ToN-IoT dataset [6]–[13], quantifying triangular subgraphs, diametric contrast between unweighted and weighted diameters, community segmentation, and degree distributions.

b) **GNN Enhanced Framework**: Augmentation of Graph Convolutional Network (GCN) [14], Graph SAmple and aggreGatE (GraphSAGE) [15], and Graph Attention Network (GAT) [16] with dual imbalance mitigation strategy, including normalized inverse-square-root class weighting to address global data skew and focal loss to emphasize challenging samples. Model performance was then assessed using classification metrics, confusion matrices and classification reports.

c) **Topology-Driven Insights**: Identification of tri-modal ransomware behavioral patterns, hub-centric scanning at high degree, and triangular subgraph-induced confusion between Password/Injection/XSS classes. Additionally, model-specific vulnerabilities including GAT's MITM precision gaps due to bridge-edge artifacts, GraphSAGE's topological mislabeling, and GCN's baseline utility were revealed.

d) **Operational Framework**: Conversion of structural insights into operational rules, including bridge-edge validation via Louvain community checks, degree stratification separating peripheral and hub, triangle density monitoring to preempt misclassification zones, and threat-specific model assignments.

The remainder of this paper is organized as follows: Section II reviews foundational GNN architectures, adapted frameworks, and their limitations. Section III details our methodology, covering graph analysis, preprocessing, graph construction, data imbalance mitigation, and model architecture. Section IV presents evaluation metrics and experimental results across GCN, GraphSAGE, and GAT variants. Section V discusses topological performance drivers, novel attack signatures, our operational framework alongside its future research directions. Section VI concludes the work.

II. Related Work

In response to the limitations of conventional IDS, GNNs have emerged as a promising approach for modeling complex network interactions. Their ability to capture both structural and feature-based relationships in graph data shows potential in cybersecurity applications, including intrusion detection.

Fig. 2 illustrates the architecture of a typical message-passing GNN. The model processes an input graph defined by adjacency matrix $A \in \{0,1\}^{N\times N}$ encoding node connections where N denotes the number of nodes in the graph, and node features $X \in \mathbb{R}^{N\times D}$ representing attributes where D indicates the dimensionality of each node's feature vector. This data passes through stacked GNN layers where each layer refines node embeddings by aggregating neighborhood information. Through iterative message-passing, nodes progressively incorporate information from wider graph regions. The final layer's embeddings H^l feed into a task-specific prediction head that generates outputs including node classifications, graph classifications, or link predictions.

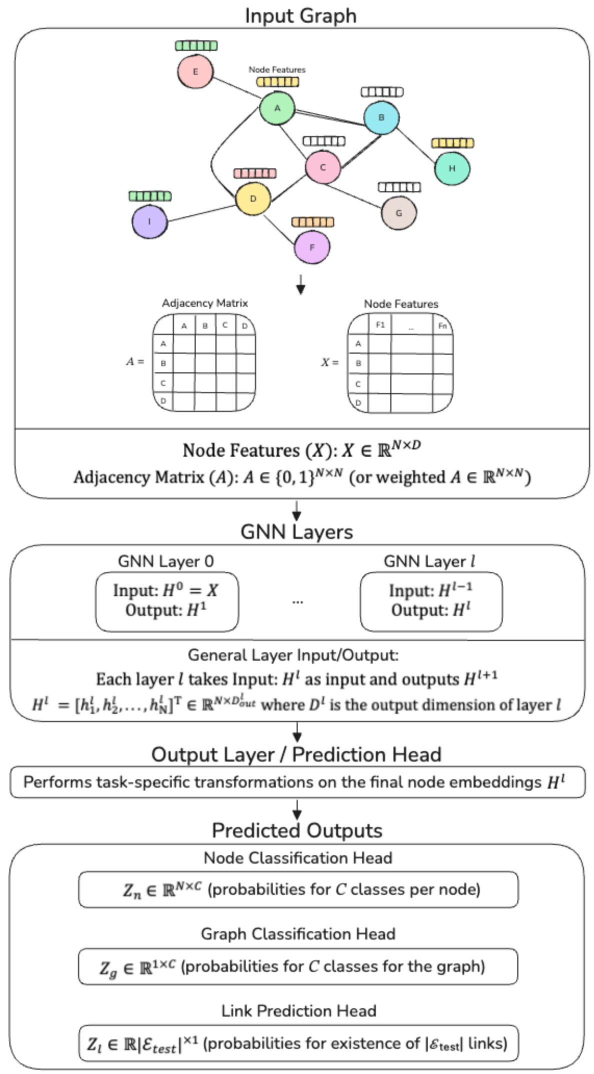


Fig. 2. Architecture of a message-passing GNN showing the complete processing pipeline from input graph to final predictions. The input graph comprises an adjacency matrix encoding node connections and node feature vectors representing attributes. Nodes labeled A through I serve as abstract identifiers representing heterogeneous devices, with visual color variations indicating diverse node features. Node ordering reflects preprocessing such as sorted IP addresses rather than inherent sequence. Stacked message-passing layers iteratively refine node embeddings through neighborhood aggregation, culminating in task-specific output predictions including node classifications, graph classifications, or link predictions.

Early foundational work established the building blocks of modern GNNs. Table I gives a compact comparison of three widely used models, GCN, GraphSAGE and GAT. The table summarizes each model's typical depth, core algorithm, aggregation style, per-layer computational and memory costs under the assumptions stated below, inductive capability and relative scalability.

TABLE I
COMPARISON OF REPRESENTATIVE GNN ARCHITECTURES

Feature	GCN (Kipf & Welling, 2017)	GraphSAGE (Hamilton et al., 2017)	GAT (Veličković et al., 2018)
Typical Depth	1–3	1–5	1–3
Core Algorithm	Message passing using localized spectral and spatial convolution	Sampling neighbors with learnable aggregation functions	Attention-based neighborhood aggregation using learned edge weights
Aggregation	Fixed degree-normalized sum of neighbor features	Learnable aggregation (mean, LSTM, or max-pooling) over sampled neighbors	Multi-head attention with learned coefficients for neighbors
Time Complexity (Per-Layer)	$O\big(\lvert\mathcal{E}\rvert\, d_h\big) + O\big(\lvert\mathcal{V}\rvert\, d_i\, d_o\big)$ for full-batch aggregation and feature projection	$O\big(B \cdot K^L \cdot d_h\big)$ for sampled mini-batch forward pass, approximately $O(\lvert\mathcal{E}\rvert\, d_h)$ for full-graph	$O\big(H \cdot \lvert\mathcal{E}\rvert \cdot d_h\big)$ for full-batch attention-weighted aggregation plus $O\big(\lvert\mathcal{V}\rvert\, d_i\, d_o\big)$ for feature projection
Memory (Peak)	$O\big(\lvert\mathcal{V}\rvert\, d_h + \lvert\mathcal{E}\rvert\big)$ for node activations and adjacency matrix	$O\big(B \cdot K^L \cdot d_h\big)$ during forward pass plus $O\big(\lvert\mathcal{E}\rvert\big)$ for storing adjacency	$O\big(\lvert\mathcal{V}\rvert\, d_h + H \cdot \lvert\mathcal{E}\rvert\big)$ for node activations and attention coefficients across all heads
Inductive	No, but can be used inductively on new nodes by applying their learned weights to local features and neighborhoods	Yes, designed to generalize to unseen nodes and graphs	Yes, designed to generalize using shared weights and local attention mechanisms
Scalability	Medium, limited by full-batch training on large graphs	High, due to neighbor sampling decoupling runtime from all nodes	Low, attention cost grows with the number of edges and attention heads

We use the following notation. $|\mathcal{V}|$ is the number of nodes, $|\mathcal{E}|$ is the number of edges, d_i is the input feature dimension, d_h is the hidden or projected feature dimension used during aggregation, and d_o is the output dimension. For GraphSAGE, K denotes the number of neighbors sampled at each layer and L denotes the number of layers. B denotes the number of target nodes in a sampled mini-batch. For GAT, H denotes the number of attention heads.

All complexity expressions refer to per-layer cost. For GCN, assuming a full-batch sparse implementation, the aggregation cost is $O(|\mathcal{E}|\, d_h)$ and the feature projection cost, corresponding to the linear transformation of node features, is $O(|\mathcal{V}|\, d_i\, d_o)$. GraphSAGE with neighbor sampling has a mini-batch forward cost of $O(B \cdot K^L \cdot d_h)$, which is an improvement over the full-graph cost of $O(|\mathcal{E}|\, d_h)$ without sampling. In GAT, the aggregation cost scales with $O(H \cdot |\mathcal{E}| \cdot d_h)$ due to H attention heads, plus the same projection cost as GCN.

Memory estimates reflect peak usage during the forward pass, dominated by node activations and graph structure. GCN requires $O(|\mathcal{V}|\, d_h + |\mathcal{E}|)$ for node activations and adjacency storage. GraphSAGE's memory cost includes $O(B \cdot K^L \cdot d_h)$ for the expanded neighborhood in a sampled mini-batch, plus $O(|\mathcal{E}|)$ if the full adjacency is stored. For GAT, memory scales as $O(|\mathcal{V}|\, d_h + H \cdot |\mathcal{E}|)$, accounting for activations and attention coefficients per edge for each attention head.

A. Foundational GNN Architectures

1) Graph Convolutional Networks (GCNs): Introduced by Kipf and Welling in 2017, GCNs advanced graph representation learning by simplifying spectral graph convolutions for semi-supervised node classification. GCNs operate by approximating localized first-order spectral filters, which enables aggregation of feature information from direct neighbors within a graph structure [14]. As illustrated in Fig. 3, the core propagation for a single GCN layer l involves:

a) **Self-Loop Augmentation:** The adjacency matrix A is modified with self-connections, represented by the identity matrix I_N, resulting in $\hat{A} = A + I_N$. This ensures a node incorporates its own features during the update.
b) **Degree Matrix Calculation:** The diagonal degree matrix $\hat{D}$ is computed from $\hat{A}$, where each entry $\hat{D}_{ii} = \sum_j \hat{A}_{ij}$ represents node i's degree including its self-loop. The index j iterates over the columns. Off-diagonals are zero. This enables fair aggregation in later steps.
c) **Symmetric Normalization:** The matrix $\hat{A}$ is symmetrically normalized using $\hat{D}$ to produce $\tilde{A} = \hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}$. This normalization facilitates stable average aggregation of neighbor features, implicitly weighting contributions based on node degrees.
d) **Graph Convolution:** The normalized adjacency matrix $\tilde{A}$ acts on the input node feature matrix $H^l \in \mathbb{R}^{N \times D_{in}^l}$. This operation aggregates neighbor features. A learnable linear transformation defined by the weight matrix $W^l \in \mathbb{R}^{D_{in}^l \times D_{out}^l}$ is then applied to the features. Finally, a nonlinear activation function σ is applied.

This entire step is encapsulated by the propagation rule:

$$H^{l+1} = \sigma\left(\tilde{A}H^l W^l\right) = \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^l W^l\right) \quad (1)$$

The output H^{l+1} represents transformed node features after layer l [14]. Despite computational efficiency, GCNs' transductive nature requires full graph visibility during training [15], limiting applicability to dynamic or unseen nodes. This poses significant challenges in evolving IoT environments, where device interactions are fluid and new nodes frequently emerge.

2) Graph SAmple and aggreGatE (GraphSAGE): Hamilton et al. addressed the transductive limitation of GCNs with GraphSAGE in 2017. GraphSAGE is an inductive framework designed to generalize to unseen nodes or entire graphs. GraphSAGE achieves this by learning node embedding functions through local neighborhood aggregation, decoupling

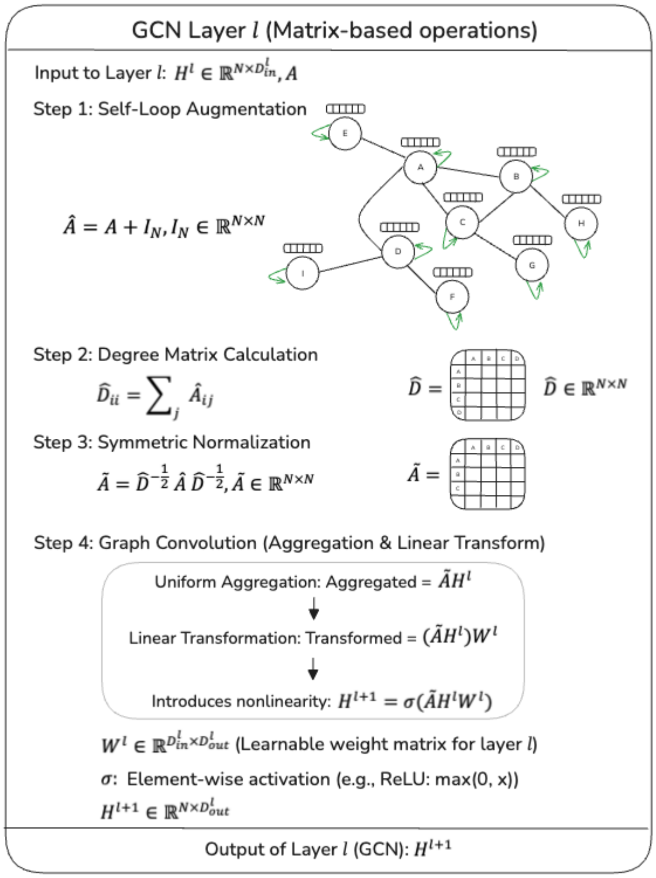


Fig. 3. Core operations of a matrix-based GCN layer at depth l including self-loop augmentation, degree matrix computation, symmetric normalization, and feature aggregation combined with learnable linear transformation, producing the output layer H^{l+1}.

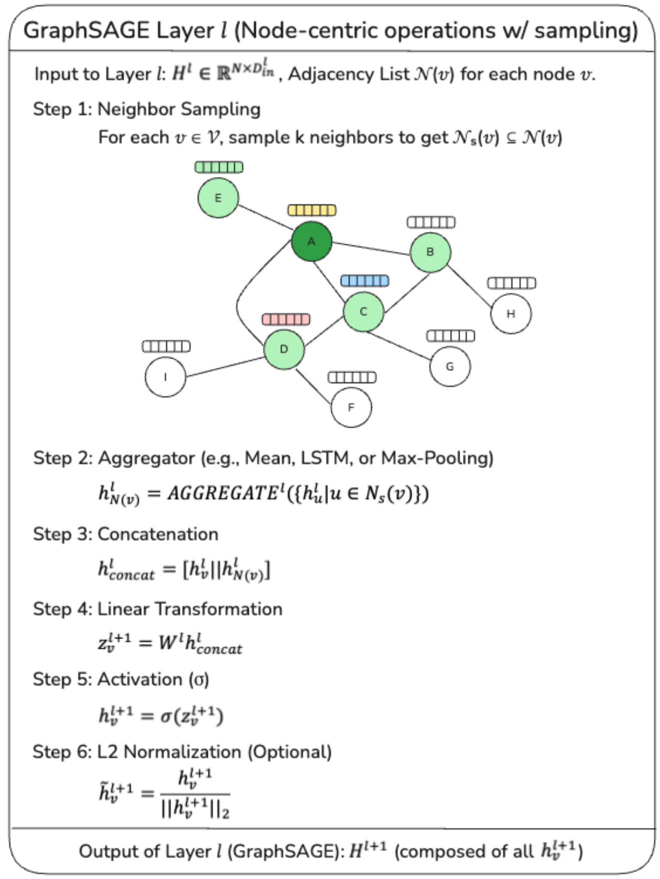


Fig. 4. Core operations of a node-centric GraphSAGE layer at depth l including neighbor sampling, learnable aggregation functions (mean, LSTM, or max-pooling), feature concatenation, linear transformation, activation, and optional L2 normalization, producing the output layer H^{l+1} for inductive node embeddings.

training from global graph structure [15]. As illustrated in Fig. 4, the propagation for node v at layer l involves:

a) **Neighbor Sampling:** From the graph's adjacency matrix A, the set of direct neighbors $\mathcal{N}(v)$ of node v is identified. A fixed-size uniform sample $\mathcal{N}_s(v)$ of k neighbors is drawn from $\mathcal{N}(v)$. This enables scalability and inductive capability by avoiding full neighborhood traversal.

b) **Aggregator** A set of features $\{h_u^l \mid u \in \mathcal{N}_s(v)\}$ are combined via a learnable aggregator function to produce $h_{\mathcal{N}(v)}^l = \text{AGGREGATE}^l\left(\{h_u^l\}\right)$, where common aggregators include MEAN, LSTM, or Max-Pooling.

c) **Concatenation:** The node's current embedding h_v^l is concatenated with the aggregated neighborhood vector $\left[h_v^l \| h_{\mathcal{N}(v)}^l\right]$. This preserves the node's intrinsic features while integrating neighborhood context.

d) **Linear Transformation:** The concatenated vector undergoes linear projection: $W^l \cdot \left[h_v^l \| h_{\mathcal{N}(v)}^l\right]$ where W^l is a learnable weight matrix. This step transforms features, adjusts dimensions by either reducing or expanding them, and merges the node's own features with its neighborhood features into a richer representation.

e) **Activation:** A non-linear function σ is applied to the transformed features. This enables the model to learn non-linear relationships between self and neighbor representations, maintain gradient stability during training, and support hierarchical pattern recognition when stacked across layers.

f) **Normalization (Optional):** For unsupervised learning, the final output embedding is often L2-normalized:

$$\tilde{h}_v^{l+1} = \frac{h_v^{l+1}}{\|h_v^{l+1}\|_2} \tag{2}$$

The output h_v^{l+1} represents the updated embedding for node v. The output layer $H^{l+1} \in \mathbb{R}^{V\times D_{out}^l}$ is formed by stacking all node embeddings. By using neighbor sampling and local aggregation, GraphSAGE is able to handles dynamic networks with new nodes [15].

3) Graph Attention Networks (GATs): Introduced by Veličković et al. in 2018, GATs use masked self-attention to dynamically weight neighbor contributions during aggregation [16]. This overcomes the limitations of fixed-weight aggregation in GCNs and GraphSAGE. As illustrated in Fig. 5, the propagation for node v at layer l involves:

a) **Feature Transformation:** Given input features $H^l \in \mathbb{R}^{N\times D_{\text{in}}^l}$ and neighborhood $\mathcal{N}(v) = \{u \mid (v,u) \in \mathcal{E}\} \cup$

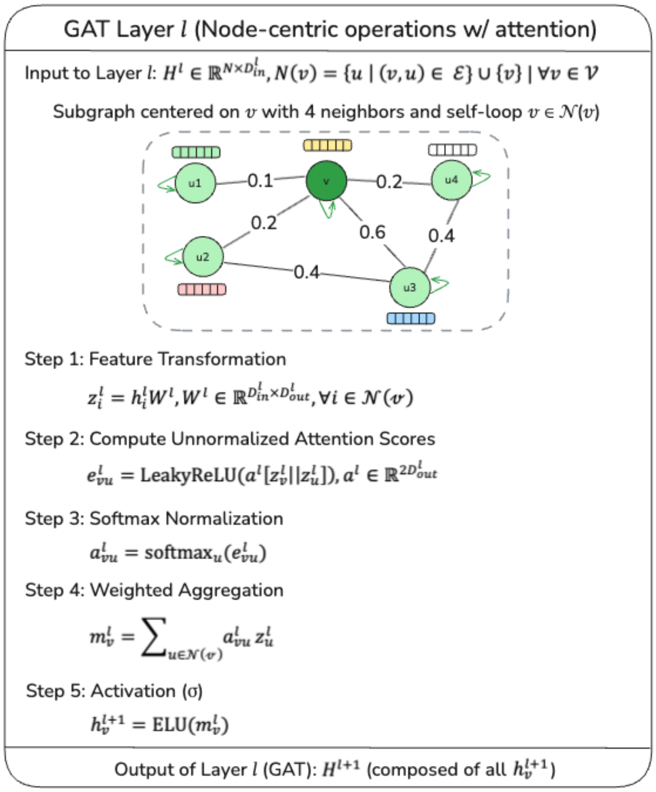


Fig. 5. Core operations within a GAT layer at depth l including feature transformation, computation of unnormalized attention scores, softmax normalization, masked self-attention weighting for dynamic neighbor aggregation, and ELU activation, producing the output layer H^{l+1}.

$\{v\}$ where $\mathcal{E}$ is the edge set. For example, the subgraph in Fig. 5 shows v with 4 neighbors and its self-loop. Then apply a shared linear projection for each node i in $\mathcal{N}(v)$: $z_i^l = h_i^l W^l$ where $W^l \in \mathbb{R}^{D^l_{\text{in}} \times D^l_{\text{out}}}$ is a learnable weight matrix. This enables dimension adjustment and prepares features for attention-based interaction.

b) **Attention Scores:** Compute unnormalized attention between v and $u \in \mathcal{N}(v)$:

$$e_{vu}^l = \text{LeakyReLU}\left(a^l \cdot \left[z_v^l \parallel z_u^l\right]\right) \tag{3}$$

where $a^l \in \mathbb{R}^{2D^l_{\text{out}}}$ is a learnable vector. This captures asymmetric pairwise relationships through directional relevance scoring.

c) **Softmax Normalization:** Apply softmax over neighbors $u \in \mathcal{N}(v)$: $a_{vu}^l = \text{softmax}_u\left(e_{vu}^l\right)$. This converts scores into interpretable probabilities, ensuring $\sum_u a_{vu}^l = 1$.

d) **Weighted Aggregation:** Compute context vector: $m_v^l = \sum_{u \in \mathcal{N}(v)} a_{vu}^l z_u^l$. This dynamically emphasizes important neighbors while suppressing noise.

e) **Activation:** Apply nonlinearity to finalize the output:

$$h_v^{l+1} = \text{ELU}\left(m_v^l\right) \tag{4}$$

where Exponential Linear Unit (ELU) handles negative values better than ReLU.

Multi-Head Attention uses K parallel heads. Each head operates as an independent attention mechanism with unique learned parameters, enabling the model to focus on different types of relationships simultaneously. For example, sensor-to-gateway communication and global traffic flows. For intermediate layers, outputs are concatenated: $h_v^{l+1} = \|_{k=1}^{K}$ ELU $\left(m_v^{l,k}\right)$. For the final layer, aggregated outputs are averaged before activation: $h_v^{l+1} = \text{ELU}\left(\frac{1}{K}\sum_{k=1}^{K} m_v^{l,k}\right)$. This stabilizes learning by capturing different relational aspects across heads.

The output layer $H^{l+1} \in \mathbb{R}^{N \times D^l_{\text{out}}}$ is formed by stacking all h_v^{l+1} [16]. For IoT security, GATs' attention mechanism is particularly valuable. It can dynamically identifies and weights suspicious device interactions while suppressing benign neighbors, enabling real-time detection of malicious nodes in evolving networks where attack patterns shift continuously.

B. Adapted GNN Frameworks for Network Security

Building on these foundational architectures, recent work adapts GNNs to address network security challenges through modified graph structures and learning objectives. Two recent innovations include frameworks that enhance temporal modeling for intrusion detection and graph transformations that improve generalization in encrypted environments. We examine these below.

1) TCG-IDS: This framework advances network intrusion detection by integrating temporal graph learning with self-supervised contrastive learning. It addresses limitations of traditional IDSs in handling evolving threats, high false positives, and dependency on labeled data. Performance evaluated on NF-CSE-CIC-IDS2018-V2 and NF-UNSW-NB15-v2 [17]. TCG-IDS achieved 99.48% and 91.48% balanced accuracy with batch processing times of 0.37s and 0.51s, demonstrating real-time suitability [18]. Core innovations are three contrastive learning strategies illustrated in Fig. 6:

a) **Temporal Contrasting:** Learn temporally consistent embeddings by contrasting the same node at different time steps. It pulls together embeddings from nearby time steps and push apart embeddings from distant time steps. This preserves the ability to flag behavioral shifts.

b) **Asymmetric Contrasting:** Learns a directional mapping from each node embedding to the embeddings of its neighbors by using a small network that takes a node's vector and produces predictions for its neighbors. Any mismatch between those predicted vectors and the actual neighbor embeddings is penalized during training, which shapes the representations to capture both the local graph structure and asymmetric relationships between nodes.

c) **Masked Contrasting:** Enhances robustness to incomplete data by splitting node embeddings into multiple factors. It generates masked versions by replacing individual factors with a mask token. Contrastive learning matches each factor-masked embedding to its original full embedding. It also distinguishes it from masked embeddings of other nodes. This captures multi-facet features.

Collectively, these contrastive strategies enable TCG-IDS to dynamically model evolving threats, capture directional

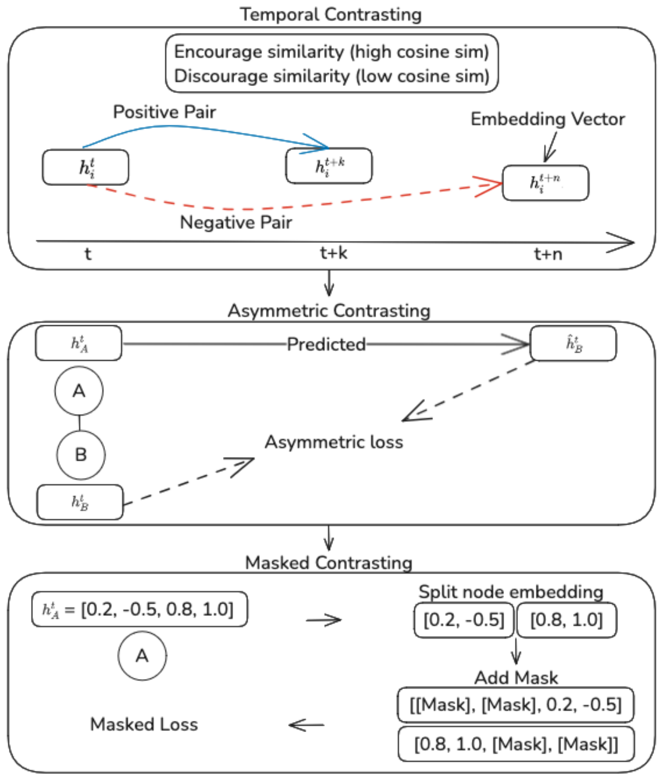


Fig. 6. Three contrastive learning strategies in TCG-IDS framework. Temporal Contrasting aligns embeddings over time to capture behavioral shifts. Asymmetric Contrasting models directional neighbor relationships. Masked Contrasting improves robustness by learning multi-facet features from incomplete data.

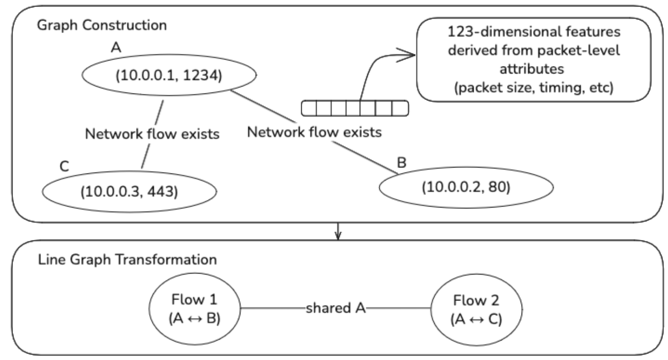


Fig. 7. Graph construction and line graph transformation in L-GraphSAGE. Each (IP, Port) pair is a node in the base graph, with edges representing flows. The line graph reinterprets flows as nodes and draws connections between them based on shared endpoints.

network relationships, and maintain robustness to incomplete data [18].

2) LineGraph-GraphSAGE (L-GraphSAGE): This framework enhances encrypted traffic identification in Internet of Vehicles (IoV) environments. It transforms traditional communication graphs into line graphs to improve model generalization across dynamic network conditions like route switching [19]. As illustrated in Fig. 7, it construct a graph where nodes represent (IP, Port) pairs and edges represent communication flows between them. These edges are enriched with 123-dimensional statistical features derived from packet-level attributes such as size, inter-arrival time, and protocol. This graph is then transformed into a line graph, where each node represents a flow and edges are drawn between flows that share an endpoint, capturing inter-flow relationships.

L-GraphSAGE was evaluated on the public UNSW Sydney dataset and on a custom five-class real-world IoV traffic dataset. It achieved an accuracy of 94.23%, representing an improvement of 0.23% over prior methods. The model also demonstrated generalization across heterogeneous network conditions. Training on α and evaluating on β produced an F1 increase of 0.20% with a relative gain of 96.92%, whereas training on β and evaluating on α yielded an F1 increase of 0.60% with a relative gain of 75.00% [19].

C. Limitations in Existing Class Imbalance Handling

While foundational and adapted GNNs advance network threat detection, their performance remains critically vulnerable to class imbalance in datasets. Current mitigation techniques primarily operate in isolated domains:

a) **Algorithm-level adjustments**: Cost-sensitive learning and modified graph convolutions
b) **Data-level interventions**: Resampling and synthetic generation
c) **Architectural strategies**: Ensemble methods

However, these approaches face limitations when applied to graphs data [20]. This limitation motivates our dual imbalance mitigation strategy introduced in Section III-D.

III. Methodology

The ToN-IoT dataset [6]–[13], released in 2021 by the UNSW and the Australian Defence Force Academy, serves as a benchmark for evaluating cybersecurity models in IoT and IIoT environments. Unlike earlier datasets like NSL-KDD [21] [22] and UNSW-NB15 [23], ToN-IoT captures realistic, multi-modal data from diverse environments. The dataset includes network traffic, operating system logs from multiple platforms, and telemetry data from over 10 IoT sensors.

The full dataset contains 22 million records. For this study, we focus on the Train_Test_Network subset, which provides labeled network flows reflecting bidirectional communications between hosts. The subset sampled 461,043 records (2% of the original), spanning both categorical and continuous domains. Each flow is annotated with:

- **Traffic-level features**: Duration, protocol types
- **Semantic attributes**: HTTP methods, DNS query responses, SSL cipher suites
- **Label**: 1 benign class called “normal” and 9 attack categories such as DDoS, MITM, and ransomware

A. Graph Analysis

Table II summarizes the principal structural characteristics of the undirected, weighted communication graph constructed from the Train_Test_Network dataset. Each node represents a unique IP host, and each undirected edge (i, j) is weighted by the total number of time intervals $t \in T$ during which i and j communicated:

$$w_{ij} = \sum_{t \in T} \mathbb{1}\big[(i, j) \in E_t\big], \tag{5}$$

where E_t is the set of active edges at time t and $\mathbb{1}$ is the indicator function.

TABLE II
UNDIRECTED WEIGHTED GRAPH STRUCTURAL CHARACTERISTICS

Metric	Value
Total Triangles	153
Unweighted Diameter (Largest Component)	5
Weighted Diameter (Largest Component)	1685
Number of Communities (Weighted Louvain)	8
Largest Community Size	327
Median Community Size	16.5
Modularity (Q)	0.437
Weighted Average Clustering Coefficient	0.0001

1) Triangle Analysis: A total of 153 triangles were identified. A triangle $\triangle_{ABC}$ is a three-node cycle such that edges exist between all pairs $\{A, B\}, \{B, C\}$, and $\{C, A\}$:

$$\triangle_{ABC} = A \leftrightarrow B \leftrightarrow C \leftrightarrow A \tag{6}$$

Triangles often signify localized clustering and strong mutual connectivity among nodes. In this undirected graph, the presence of 153 triangles reflects frequent three-way communication patterns among subsets of hosts. These triangular structures typically indicate stable teamwork between devices, such as tightly connected IoT gadgets within a subnetwork. Similarly, closed triplets can represent functional clusters. For example, sensors communicating with both an edge node and a gateway, or coordinated activity flows in operational settings.

From a cybersecurity perspective, a sudden rise in triangles could signal coordinated lateral movement during cyberattacks. For instance, peer-to-peer malware spreading can create densely connected triads over time. Consequently, monitoring triangle density and evolution over time offers a lightweight structural indicator for emerging threats.

Furthermore, these triangle structures hold implications for graph-based machine learning performance. Models like GCNs rely on the local neighborhood structure during message passing, where tightly connected components can facilitate information aggregation within the component. Similarly, neighbor-sampling methods like GraphSAGE and edge-attention mechanisms like GATs leverage these structures by adapting to edge strength and connectivity patterns. The observed number of triangles suggests the graph contains groups of high local structure surrounded by fewer links spanning the entire network as shown in Fig. 8. This topology well-suited to hybrid or hierarchical GNN architectures, which leverage both localized learning and broader network dynamics.

2) Diameter and Connectivity: To characterize the global communication structure of the IoT network, we calculated both unweighted and weighted diameters on its largest connected component.

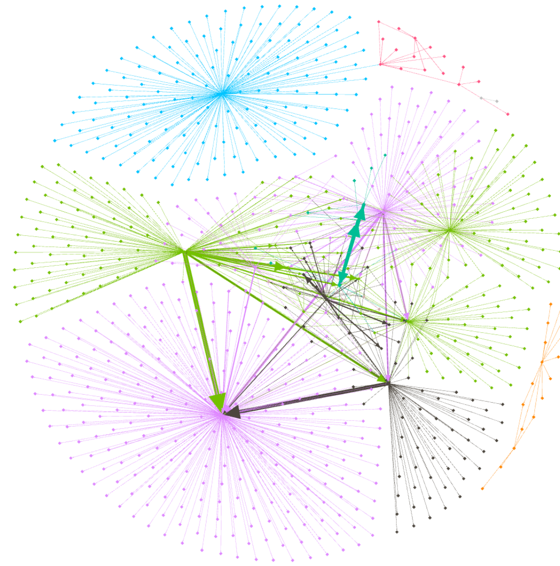


Fig. 8. ToN-IoT network topology showing groups of high local structure surrounded by fewer links spanning the entire network, and the number of communities.

- **The unweighted diameter is 5 hops**: This measures the longest shortest path in terms of hop count between any two reachable nodes, calculated using breadth-first search (BFS). A diameter of 5 indicates a relatively compact core network, suggesting a densely connected main network likely formed by gateway nodes.
- **The weighted diameter is 1685**: This represents the highest total cost of the least-cost path between any two nodes using our frequency-based weighting scheme. This was computed using Dijkstra's algorithm [24] in NetworkX [25].

In this scheme, each edge is assigned a weight equal to its raw communication frequency. High-frequency links incur substantial traversal cost, modeling an attacker's avoidance of well-monitored channels. Conversely, low-frequency links accrue minimal cost, highlighting stealth pathways preferred by malware. As a result, routes composed primarily of busy links appear long, whereas those that rely on seldom-used connections appear short. This weighting is specifically designed to detect low-and-slow lateral movement, C2 beaconing, and data exfiltration over obscure protocols that exploit infrequently monitored links.

The large gap between the unweighted diameter (5 hops) and the weighted diameter (1685) highlights fundamental network heterogeneity. Although every node lies within a few hops topologically, nodes linked by infrequent communications appear far apart when communication frequency is factored in. This distinction is crucial when designing GNNs. Conventional GCNs assume nearby nodes share similar features, an assumption that becomes problematic for long-range connections between structurally distant components, a pattern characteristic of stealth communication channels. In contrast, GATs learn explicit edge importance via attention mechanisms. This capability enables GATs to identify security-critical routes by emphasizing anomalous interactions across distant structural positions.

From a cybersecurity perspective, the small unweighted diameter suggests potential for rapid threat propagation across the network core within just 5 hops. Simultaneously, the large weighted diameter indicates that malicious activity over infrequently used connections may evade conventional detection. This dual behavior necessitates weight-aware learning approaches. For comprehensive threat coverage, an inverse-frequency weighting scheme provides complementary value. In this scheme, each link's weight equals the inverse of its communication frequency (1/frequency), interpreting frequent connections as low-cost preferred paths. This approach detects attacks exploiting high-frequency channels such as DDoS amplification and credential stuffing, reduces the effective distance for legitimate traffic patterns, and exposes anomalies within busy communication pathways.

Together, these perspectives reveal the network's core tension between topological efficiency and communication irregularity. Nodes maintain close proximity in hop count while exhibiting significant variance in effective distance under frequency-based weighting. These dual characteristics are valuable for developing integrated detection systems. By combining both weighting approaches, such systems can identify stealthy low-frequency threats through security weighting (high weight = low frequency), while also monitoring critical infrastructure attacks via operational weighting (high weight = high frequency). This dual strategy provides comprehensive coverage against evolving network threats.

3) Community Detection via Louvain: To identify modular structures in the IoT communication network, we applied the Louvain algorithm for community detection. This efficient method optimizes modularity in weighted graphs through iterative node movement and community aggregation [26]. In our context, a community represents a device group with denser internal connections than external links, typically corresponding to clusters of functionally similar sensors.

Louvain maximizes modularity Q, defined as:

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j) \tag{7}$$

where A_{ij} is the edge weight between nodes i and j, $k_i = \sum_j A_{ij}$ is the weighted degree of node i, $m = \frac{1}{2} \sum_{i,j} A_{ij}$ is total network edge weight, and $\delta(c_i, c_j)$ is 1 when nodes i and j belong to the same community and 0 otherwise [26]. This quantifies how much intra-community connectivity exceeds expected values in a random network with identical node degrees.

When applied to our undirected graph, where edge weights represent communication frequencies, Louvain identified 8 communities with modularity score of $Q = 0.437$. Since values exceeding 0.3 are commonly taken as a sign of significant community structure in complex networks, this confirms distinct modular organization in IoT interactions. The largest community contains 327 nodes, with a median community size of 16.5, reflecting a heterogeneous mix of large hubs and small sensor clusters typical in IoT deployments.

This segmentation has functional and security implications. Communities reflect subnet interaction patterns with frequent intra-group communication. Understanding these communities yields dual benefits.

a) **Modeling Efficiency**: Inductive GNNs like GraphSAGE can exploit this modular structure to generalize representations across similar subgraphs. GATs can utilize community membership to prioritize intra-community edges through attention mechanisms.
b) **Security**: Inter-community bridging edges may indicate anomalous lateral movements, motivating hybrid anomaly detection systems that combines community-aware graph structures with edge-level behavioral features.

Overall, Louvain-based detection quantifies IoT modularity and uncovers latent patterns that can guide GNN architecture design and enhance security monitoring.

4) Degree Distributions and Structural Diversity: Fig. 9 presents four complementary node-degree distributions on log-log axes: (a) undirected unweighted, (b) undirected weighted, (c) directed unweighted, and (d) directed weighted. These views collectively distinguish between connection count and data volume, and between symmetric and asymmetric communication. The distributions illustrate how various node roles emerge. Some devices exhibit sparse connectivity or low traffic, whereas others act as communication hubs or traffic aggregators. This structural variation is characteristic of heterogeneous IoT environments. Security-relevant anomalies arise when nodes appear in unexpected regions of these distributions. For instance, a device with few expected peers but unusually high traffic output may signal a compromise or misconfiguration. Conversely, low-activity devices could be benign or may indicate dormant threats. Such complementary views help contextualize network behaviors and detect suspicious patterns more effectively.

The undirected distributions in Fig. 9(a,b) provide a baseline view of overall network structure. The unweighted version highlights how many peers each device connects to, while the weighted version captures the total volume of communication. In contrast, the directed distributions in Fig. 9(c,d) offer crucial insight into asymmetric behaviors. The directed unweighted plot shows distinct in-degree and out-degree profiles. The steeper initial slope of the in-degree curve indicates most devices receive connections from few partners, while the shallower out-degree slope suggests scanning activity is distributed across more nodes. Both distributions develop straight right tails, highlighting critical infrastructure nodes handling disproportionate incoming connections and persistent scanners initiating exceptional outgoing connections. This separation reveals directional imbalances that may indicate scanning activity, especially when combined with suspicious traffic patterns. The weighted directed plots further contextualize these behaviors by revealing differences in traffic volume, helping to distinguish opportunistic scanning from more severe or targeted threats.

The use of logarithmic scaling across all subplots informs GNN design decisions. The heavy-tailed distributions introduce extreme variance and skewness, motivating logarithmic transformations to node features such as degree or weighted degree, helping to normalize their range. GCNs, for instance, incorporate degree normalization to mitigate the dominance of high-degree nodes during message aggregation. In GATs, attention coefficients are learned functions of node and edge features. When communication volume is encoded as an edge feature, GATs can learn to assign higher attention to high-volume links. Moreover, neighborhood sampling strategies must be carefully designed to account for distribution skewness, ensuring that both peripheral low-activity devices and central infrastructure nodes are adequately represented during training.

Tracking the temporal evolution of these degree distributions can enables early detection of abnormal network behavior. For instance, gradual rightward shifts in out-degree distributions in Fig. 9c signal lateral movement as compromised nodes initiate new connections. Sudden spikes in weighted degree in Fig. 9(b,d) may reflect bursts of data exfiltration or other high-volume activities. Statistical outlier detection techniques can highlight nodes that deviate from expected patterns.

This analysis illustrates the interplay between baseline connectivity and dynamic threat activity within a unified visualization framework. As shown in Fig. 9, unweighted metrics help identify abnormal communication patterns, while weighted measures capture the intensity or severity of potential threats. The prevalence of heavy-tailed distributions across all variants underscores the need for tailored security strategies, where logarithmic feature transformation and distribution-aware outlier detection are essential for isolating structural and traffic anomalies.

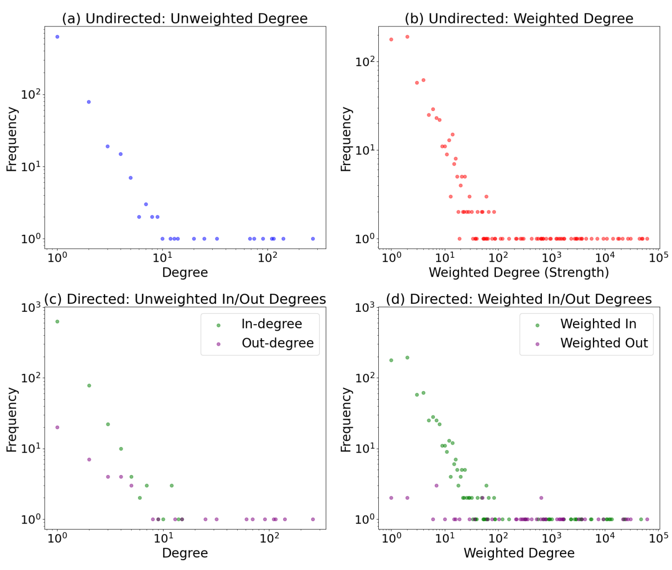


Fig. 9. Composite degree distributions on log-log scales: (a) undirected unweighted, (b) undirected weighted, (c) directed unweighted in/out degrees, and (d) directed weighted in/out degrees

In summary, this analysis revealed key structural characteristics of the IoT communication network. The identification of 153 triangles indicates localized clustering that support stable teamwork but could enable coordinated threats. A diametric contrast exists between the compact unweighted diameter of 5 hops and the large weighted diameter of 1685. This highlights the network's core tension between efficient topology and heterogeneous communication patterns. Consequently, dual threat detection strategies are essential: security weighting detects low-frequency stealth attacks while inverse-frequency weighting exposes high-frequency abuses. Louvain-based community detection uncovered 8 significant functional modules with modularity $Q = 0.437$. The largest community contains 327 nodes and median size of 16.5. This segmentation both aids GNN generalization and focuses security monitoring on inter-community bridges where anomalous lateral movement may occur. Degree distributions consistently exhibited heavy-tailed properties across all variants. These establish critical connectivity baselines and expose asymmetric threat patterns, such as scanning via elevated out-degree. Given the extremely low weighted clustering coefficient of 0.0001 confirming global sparsity, logarithmic feature handling in GNNs becomes essential to balance representation of peripheral devices and critical hubs. Collectively, these metrics provide a multifaceted foundation for detecting stealth threats, guiding GNN design, and developing topology-aware security solutions.

B. Preprocessing Pipeline

Fig. 10 provides an overview of the workflow. The preprocessing pipeline prepares network traffic data for graph-based modeling through sequential stages. First, the classification target is extracted and any redundant label columns are removed. Source and destination IP addresses are preserved exclusively for graph construction. These structural identifiers are excluded from all subsequent transformations to prevent them from influencing learned feature representations.

Categorical variables are identified using data-type inspection and subject-matter knowledge. Examples include protocol types and connection flags. These categorical features undergo screening based on cardinality. Features that have more than 100 distinct values are discarded to mitigate sparsity. The remaining categorical fields are label-encoded to ensure compatibility with graph-learning frameworks.

For numerical features, we address skewness by applying a logarithmic transformation to make their distributions more symmetric. Any missing values within these numerical fields are imputed using the median, reducing sensitivity to outliers. All processed features, both transformed numerical and encoded categorical, are then normalized using z-score normalization. This yields a coherent set of numerical descriptors suitable as edge and node-level attributes. These processed features are then used as input for GCN, GraphSAGE, and GAT architectures.

C. Graph Construction

The communication graph is built by representing each unique IP address as a node and each network flow as a directed edge. A unified index of all source and destination IPs is created and each IP is assigned a distinct integer identifier to enable efficient graph operations. Fig. 11 illustrates an example subgraph of IP-to-IP flows in the ToN-IoT network.

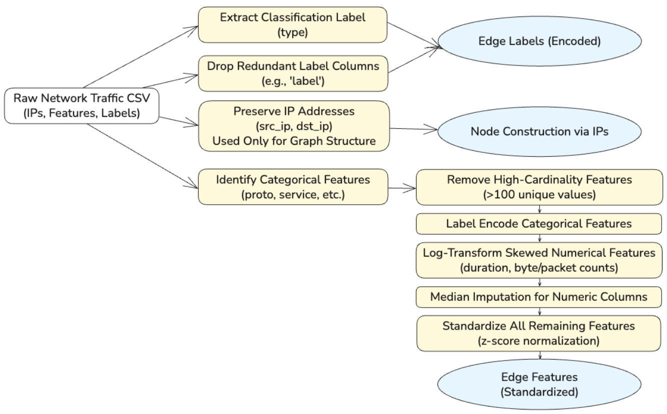


Fig. 10. Preprocessing pipeline for network traffic data

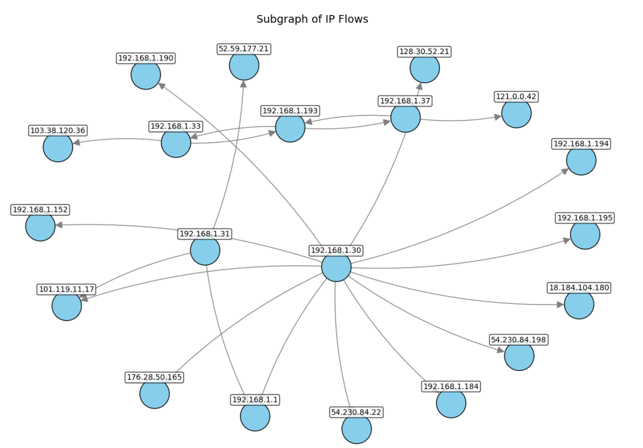


Fig. 11. Example subgraph of IP-to-IP flows in the ToN-IoT network. Each unique IP address represents a node, and each network flow is a directed edge.

Basic statistics for the full graph appear in Table III. There are 776 unique IP nodes and 211,043 directed edges corresponding to individual flows. Each node has an average degree of 271.96, the maximum out-degree is 61,633 and the maximum in-degree is 47,795.

TABLE III
BASIC GRAPH STATISTICS

Metric	Value
Unique IPs (nodes)	776
Flows (directed edges)	211,043
Average node degree	271.96
Maximum out-degree	61,633
Maximum in-degree	47,795

Node features combine structural and behavioral characteristics. The structural component contains the total occurrences of each IP as a source and as a destination. For behavioral characteristics, we aggregate flow-level statistics per IP. This includes the mean and standard deviation of source byte counts, and the average flow duration. These five features are combined into a unified per-node matrix, which undergoes z-score normalization before conversion to tensor format.

Edges are created by mapping source and destination IPs from each flow record to their integer indices, establishing directed connections. Edge features incorporate all flow attributes (excluding IP addresses and labels) in their z-score normalized tensor format. For supervised learning, edge labels are generated by integer encoding the original classification targets (e.g.,"backdoor", "normal"). Table IV shows this encoding scheme.

TABLE IV
EDGE LABEL ENCODING

Source IP	Destination IP	Original Label	Encoded Label
192.168.1.33	192.168.1.193	backdoor	0
192.168.1.152	192.168.1.190	normal	1
192.168.1.31	176.28.50.165	injection	2

The final graph object contains four key components for GNN training and evaluation:

a) A node feature tensor capturing IP communication behavior
b) An edge index tensor defining the directed topology
c) An edge feature tensor containing flow attributes
d) An edge label tensor for classification tasks

D. Dual Imbalance Mitigation Strategy

The labeled flow data exhibit significant class imbalance, as shown in the top panel of Fig. 12. This imbalance risks biasing the model toward majority classes during training. To address this, we compute class weights w_c inversely proportional to the square root of class frequencies f_c, then normalize them to maintain consistent gradient scaling:

$$w_c = \frac{1}{\sqrt{f_c}}, \quad \widetilde{w}_c = \frac{C\, w_c}{\sum_{k=1}^{C} w_k}, \tag{8}$$

where C is the number of classes. This normalization ensures the average weight across classes is 1, preventing abrupt changes in parameter updates that could destabilize training. The resulting weights, visualized in the bottom panel of Fig. 12, prioritize rare classes while bounding the maximum penalty to avoid excessive gradient noise.

These normalized class weights $\widetilde{w}_c$ are applied within a weighted cross-entropy (CE) loss, where y_i denotes the true class y of sample i:

$$\mathcal{L}_{\text{wCE}} = -\sum_{i=1}^{N} \widetilde{w}_{y_i} \, \log p_{i,y_i} \tag{9}$$

Fig. 13 illustrates how class weighting scales the loss. For a sample with true class y, the weighted penalty $\ell = -\widetilde{w}_y \log p$ is amplified for rare classes. The rarest class with $\widetilde{w}_c = 0.0310$ incurs 6.9× higher penalties than the most frequent class with $\widetilde{w}_c = 0.0045$ at $p = 0.1$, addressing class imbalance during training.

To further focus learning on hard examples, we incorporate focal loss of Lin et al. [27] to reduces contributions from easily classified samples via a modulating factor $(1 - p_{i,y_i})^{\gamma}$:

$$\mathcal{L}_{\text{focal}} = -\sum_{i=1}^{N} \widetilde{w}_{y_i} \, (1 - p_{i,y_i})^{\gamma} \, \log p_{i,y_i} \tag{10}$$

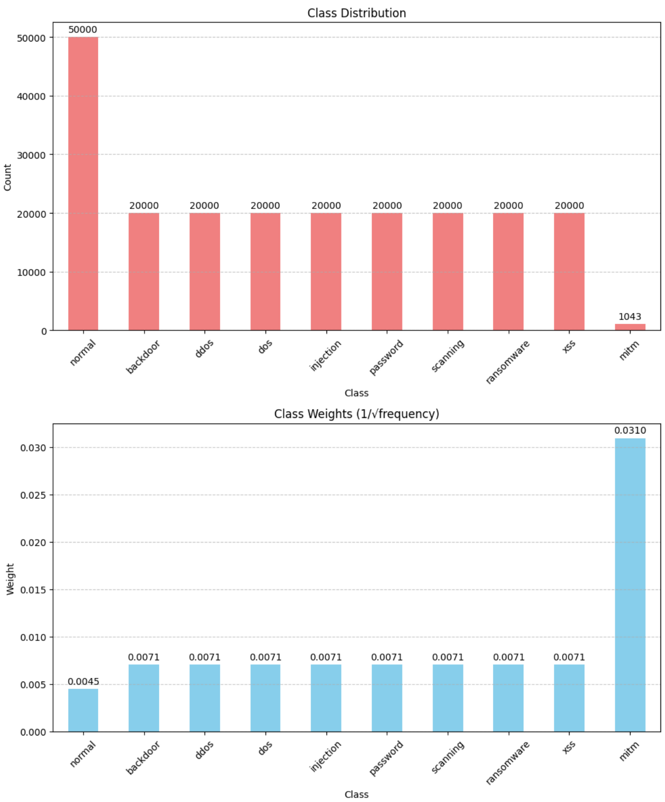


Fig. 12. Top panel: Class frequency distribution in the training set. Bottom panel: Inverse-square-root class weights.

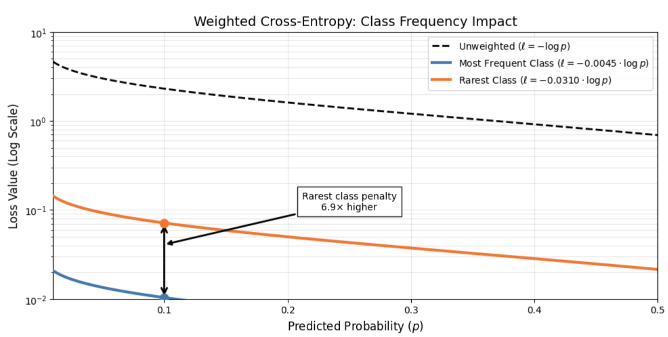


Fig. 13. Weighted cross-entropy loss $\ell = -\widetilde{w}_y \log p$ for the most frequent class (blue, $\widetilde{w}_y = 0.0045$) and the rarest class (orange, $\widetilde{w}_y = 0.0310$), compared with the unweighted loss (dashed black). At $p = 0.1$, the rarest class penalty is $6.9\times$ higher than the most frequent class penalty, helping training prioritize minority classes.

Fig. 14 demonstrates the focal modulator effect across $\gamma \in \{0, 1, 2, 5\}$ using the rarest class weight $\widetilde{w}_y = 0.0310$. Well-classified examples at $p = 0.9$ show progressive suppression with increasing γ, with $\gamma = 2$ achieving $100\times$ reduction versus $\gamma = 0$. Errors at $p = 0.1$ maintain high penalties across all γ. $\gamma = 2$ selected for the experiment because this value yielded the highest detection rates.

To jointly address class imbalance and hard-example learning, we combine class weighting and focal modulation into a total loss:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{wCE}} + \mathcal{L}_{\text{focal}} \tag{11}$$

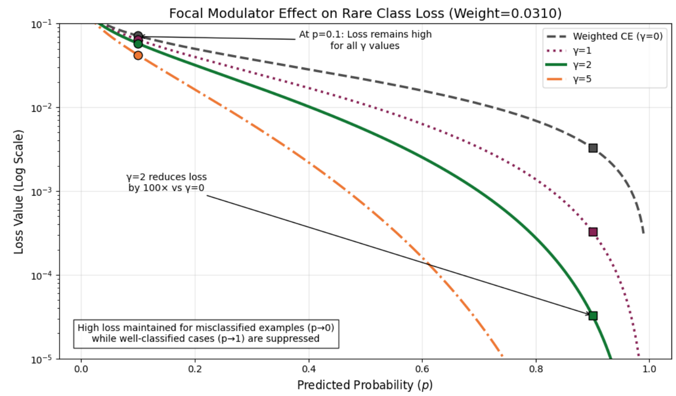


Fig. 14. Focal loss for the rarest class $\widetilde{w}_y = 0.0310$ across $\gamma \in \{0, 1, 2, 5\}$: $\gamma = 0$ (gray dashed), $\gamma = 1$ (purple dotted), $\gamma = 2$ (green solid), and $\gamma = 5$ (orange dash-dot). At $p = 0.9$, the $\gamma = 2$ curve yields a $100\times$ lower loss than $\gamma = 0$, while at $p = 0.1$, the loss remains high at 0.058 compared to 0.071 for $\gamma = 0$. Markers indicate critical points, with circles for $p = 0.1$ and squares for $p = 0.9$.

This integrated approach preserves high penalties for misclassified examples ($p \to 0$) while suppressing trivial contributions for confident predictions ($p \to 1$).

Fig. 15 demonstrates the combined loss components for the rarest class ($\widetilde{w}_y = 0.0310$, $\gamma = 2$). At $p = 0.1$:

- **Weighted CE**: $\mathcal{L}_{\text{wCE}} = 0.0714$
- **Focal term**: $\mathcal{L}_{\text{focal}} = 0.0578$ (≈81% of $\mathcal{L}_{\text{wCE}}$)
- **Total loss**: $\mathcal{L}_{\text{total}} = 0.1292$ (≈181% of $\mathcal{L}_{\text{wCE}}$)

The total loss amplifies penalties on hard examples by ≈81% beyond the class-weighted baseline.

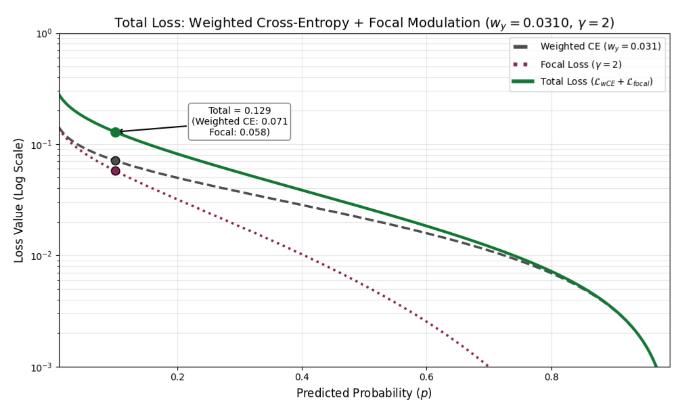


Fig. 15. Loss components for rarest class $\widetilde{w}_y = 0.0310$, $\gamma = 2$: Weighted cross-entropy (gray dashed), focal term (maroon dotted), and total loss (green solid). The loss maintains high penalties for errors ($p \to 0$) while suppressing the influence of confident predictions ($p \to 1$). At $p = 0.1$, the total loss of 0.129 is approximately 81% higher than $\mathcal{L}_{\text{wCE}} = 0.071$, and 123% higher than $\mathcal{L}_{\text{focal}} = 0.058$.

This dual strategy simultaneously mitigates global class imbalance through normalized weighting and adaptively focuses learning on hard examples via focal modulation. The result is improved detection of rare but critical attack patterns without sacrificing stability or overall performance.

E. Model Architecture

Three GNN variants, including GCN [14], GraphSAGE [15], and GAT [16], were implemented under a unified framework. All models share these core components:

a) **Graph Convolution Layers**: Two stacked layers with type-specific configurations:

- **GCN**:
  - Layer 1: `GCNConv` (input dimension $\rightarrow$ 64)
  - Layer 2: `GCNConv` (64 $\rightarrow$ 64)
  - Dropout: 30% after each convolution
- **GraphSAGE**:
  - Layer 1: `SAGEConv` (input dimension $\rightarrow$ 64)
  - Layer 2: `SAGEConv` (64 $\rightarrow$ 64)
  - Dropout: 30% after each convolution
- **GAT**:
  - Layer 1: `GATConv` (input dimension $\xrightarrow[4\ heads]{\text{concat}}$ 256, dropout=0.3)
  - Layer 2: `GATConv` (256 $\xrightarrow[2\ heads]{\text{average}}$ 64, dropout=0.3)
  - Dropout: 0% after each convolution
  - Attention Mechanism:
    * **Adaptive neighbor weighting**: Attention-based edge filtering
    * **Multi-perspective learning**: Independent heads capture distinct patterns
    * **Hierarchical abstraction**: Layer 1 (concatenated heads) preserves details. Layer 2 (averaged heads) summarizes

b) **Residual Connections with Batch Normalization (BN)**: Identical processing for both layers:

$$\mathbf{x}^{\text{out}} = \text{ELU}\left(\underbrace{\text{BN}(\text{Conv}(\mathbf{x}^{\text{in}}))}_{\text{BN output}} + \underbrace{\text{Linear}_{\text{res}}(\mathbf{x}^{\text{in}})}_{\text{Dim-matching proj.}}\right) \tag{12}$$

- Layer 1: Transforms input dimensions $\rightarrow$ conv1 output dimensions
- Layer 2: Projects conv1 output dimensions $\rightarrow$ conv2 output dimensions

c) **Edge Classification Head**: For each edge:

$$\text{MLP}\left(\ \mathbf{x}_{\text{src}} \parallel \mathbf{x}_{\text{dst}} \parallel \mathbf{e}_{\text{attr}}\ \right)$$

- Input: Source node embedding $\parallel$ Destination node embedding $\parallel$ Edge attributes
- 3-layer multilayer perceptron (MLP) configuration:
  - Linear(d_{in}, 512) $\rightarrow$ BN $\rightarrow$ ELU $\rightarrow$ Dropout (0.4)
  - Linear(512, 256) $\rightarrow$ BN $\rightarrow$ ELU $\rightarrow$ Dropout (0.3)
  - Linear(256, C)

  where $d_{\text{in}} = 2 \times d_{\text{node}} + d_{\text{edge}}$, C = num_classes

IV. Training and Evaluation

Building on the unified two-layer GNN architectures, we assessed model generalization using stratified 5-fold cross-validation. Edge indices were partitioned while maintaining the original class distribution proportions. During training, the Adam optimizer was configured with a learning rate of 0.001 and L2 weight decay of 10^{-5}, running for 200 epochs per fold. Training and test masks were applied at the edge level to maintain proper separation between splits. Model performance was assessed using weighted versions of classification metrics (accuracy, precision, recall, F1-score) to account for class imbalance. Each fold generated detailed diagnostic outputs including confusion matrices and classification reports with per-class statistics. Final model comparisons were based on averaged metrics across all folds, ensuring robust performance estimation under consistent experimental conditions. The complete pipeline maintained identical random seeds and preprocessing pipeline across architecture variants to enable direct comparison. In addition to performance metrics, the average runtime per fold was recorded to assess computational efficiency.

Table V presents the averaged results from five-fold cross-validation. The GAT model consistently outperformed the others, achieving an average of 96.09% accuracy, 96.64% precision, 96.09% recall, and 96.11% F1-score. However, this performance comes at the cost of the longest runtime. Fig. 16 shows a representative confusion matrix for GAT and class-wise results in Table VI highlight its robust detection across most of the categories, and better performance on minority classes like MITM (F1: 0.71 ± 0.02) compared to GCN and GraphSAGE. The attention mechanism enabled GAT to focus on irregular traffic patterns, potentially improving detection of stealthy and low-signal attacks.

TABLE V
Averaged 5-Fold Performance and Runtime

Model	Accuracy	Precision	Recall	F1-score	Runtime
GCN	88.87%	91.62%	88.87%	88.69%	408.12s
GraphSAGE	93.57%	95.26%	93.57%	93.49%	403.11s
GAT	96.09%	96.64%	96.09%	96.11%	464.91s

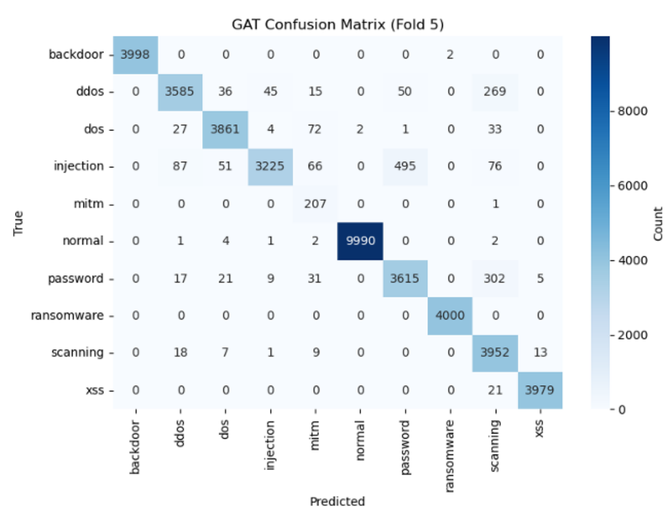


Fig. 16. Confusion matrix for GAT

GraphSAGE followed closely, with an average of 93.57% accuracy, 95.26% precision, 93.57% recall, and 93.49% F1-

TABLE VI
GAT PER-CLASS METRICS (MEAN ± STD OVER 5 FOLDS)

Class	Precision	Recall	F1-Score
Backdoor	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
DDoS	0.97 ± 0.01	0.93 ± 0.02	0.95 ± 0.01
DoS	0.97 ± 0.00	0.97 ± 0.00	0.97 ± 0.00
Injection	0.97 ± 0.01	0.77 ± 0.07	0.85 ± 0.05
MITM	0.56 ± 0.03	0.98 ± 0.01	0.71 ± 0.02
Normal	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
Password	0.81 ± 0.06	0.96 ± 0.03	0.88 ± 0.03
Ransomware	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
Scanning	0.95 ± 0.05	0.97 ± 0.02	0.96 ± 0.03
XSS	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00

score. Fig. 17 shows a representative confusion matrix and class-wise results in Table VII show good performance on common classes, but greater variance on threats such as Injection (F1: 0.76 ± 0.12) and MITM (F1: 0.70 ± 0.06). While it benefits from ≈13% faster training than GAT, its fixed aggregation scheme is less effective at capturing subtle, evolving behaviors of attacks like Injection and MITM.

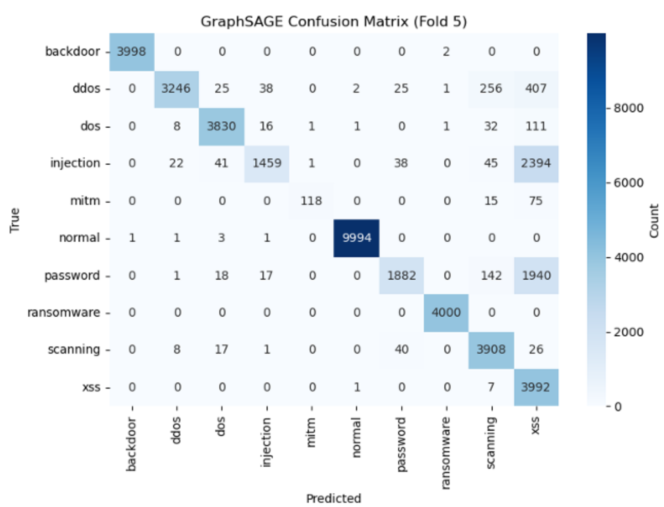


Fig. 17. GraphSAGE Confusion Matrix

TABLE VII
GRAPHSAGE PER-CLASS METRICS (MEAN ± STD OVER 5 FOLDS)

Class	Precision	Recall	F1-Score
Backdoor	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
DDoS	0.95 ± 0.05	0.92 ± 0.06	0.94 ± 0.03
DoS	0.97 ± 0.01	0.97 ± 0.02	0.97 ± 0.01
Injection	0.90 ± 0.10	0.70 ± 0.19	0.76 ± 0.12
MITM	0.69 ± 0.16	0.77 ± 0.12	0.70 ± 0.06
Normal	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
Password	0.87 ± 0.09	0.77 ± 0.21	0.79 ± 0.10
Ransomware	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
Scanning	0.94 ± 0.04	0.98 ± 0.00	0.96 ± 0.02
XSS	0.88 ± 0.22	1.00 ± 0.00	0.92 ± 0.15

GCN achieved the lowest scores, with an average of 88.87% accuracy, 91.62% precision, 88.87% recall, and 88.69% F1-score. Fig. 18 shows a representative confusion matrix and class-wise results in Table VIII reveal challenges in detecting attacks, such as Injection (F1: 0.72 ± 0.07), MITM (F1: 0.70 ± 0.04) and XSS attacks (F1: 0.72 ± 0.08), due to GCN's limited flexibility in modeling heterogeneous graph patterns.

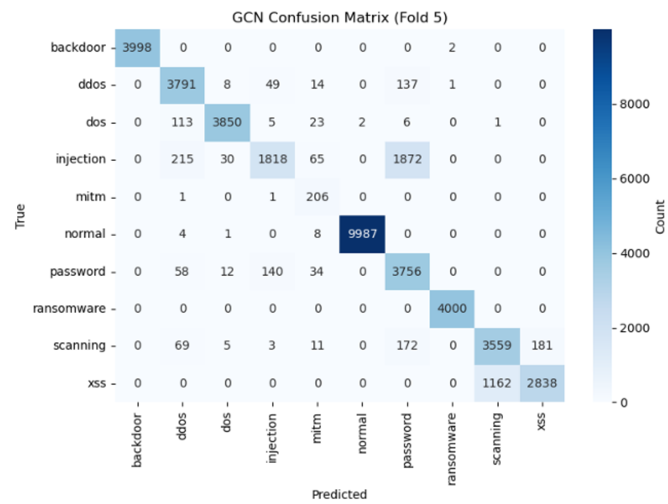


Fig. 18. GCN Confusion Matrix

TABLE VIII
GCN PER-CLASS METRICS (MEAN ± STD OVER 5 FOLDS)

Class	Precision	Recall	F1-Score
Backdoor	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
DDoS	0.96 ± 0.04	0.87 ± 0.08	0.91 ± 0.04
DoS	0.98 ± 0.01	0.96 ± 0.01	0.97 ± 0.00
Injection	0.79 ± 0.12	0.71 ± 0.17	0.72 ± 0.07
MITM	0.55 ± 0.05	0.98 ± 0.01	0.70 ± 0.04
Normal	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
Password	0.78 ± 0.12	0.75 ± 0.19	0.74 ± 0.08
Ransomware	1.00 ± 0.01	1.00 ± 0.00	1.00 ± 0.00
Scanning	0.65 ± 0.09	0.97 ± 0.04	0.77 ± 0.05
XSS	0.99 ± 0.02	0.57 ± 0.10	0.72 ± 0.08

V. DISCUSSION

A. Topological Drivers of Class-Level Performance

Section III-A identified 153 triangular subgraphs in the communication graph. These tightly interconnected structures create signature confusion patterns between Password, Injection, and XSS traffic classes. GraphSAGE demonstrates significant vulnerability to this effect, mislabeling 59.9% of true Injection samples as XSS and 48.5% of Password samples as XSS, achieving an F1 score of 0.76 ± 0.12 for Injection and an F1 score of 0.79 ± 0.10 for Password. The attention mechanism in GAT mitigates XSS-related confusion, eliminating Injection-to-XSS mislabeling while maintaining 12.4% Injection-to-Password errors, resulting in an F1 score of 0.85 ± 0.05 for Injection.

Benign traffic traversing bridge edges between communities mimics MITM traffic patterns. MITM precision remains constrained across models. As shown in Tables VII–VIII, GraphSAGE achieves 69.0 ± 0.16 precision, GAT 56.0 ± 0.03, and GCN 55.0 ± 0.05. Analysis of confusion matrices reveals that false positives originate from high-connectivity traffic patterns. In GAT, 195 non-MITM instances were labeled

MITM, with 66 corresponding to Injection at 33.8% and 31 to Password at 15.9%. This aligns with topological metrics showing Injection's external connectivity at 66.7% and Password at 42.1%, both substantially exceeding the 6.5% for true MITM traffic in Table IX. We therefore propose four topology-aware mitigations:

a) **External connectivity filter** to suppress MITM alerts on nodes with ExtConn above 9.6%, which is the normal traffic baseline.
b) **Edge penalty weighting** that adds each node's external-link weight W_i^{ext} to the MITM score so high-ExtConn nodes are down-weighted.
c) **Synthetic negatives** by injecting benign bridge-edge examples during training so the model learns to distinguish them.
d) **Bridge-edge validation** occurs in real-time by checking Louvain community membership. For example, an edge is tagged if it connects nodes from different communities.

TABLE IX
TOPOLOGICAL METRICS BY ATTACK TYPE

Type	Nodes	ExtConn (%)	MeanExt	PerfectIso (%)
backdoor	1	0.0	0.00	100.0
ddos	4	25.0	0.45	75.0
dos	58	1.7	0.23	98.3
injection	3	66.7	4.74	33.3
mitm	92	6.5	0.23	93.5
normal	582	9.6	0.73	90.4
password	19	42.1	0.39	57.9
ransomware	**6**	**33.3**	**8.64**	**66.7**
scanning	6	83.3	19.16	16.7
xss	5	60.0	7.41	40.0

B. Novel Findings Relative to Prior Work

Our analysis reveals several phenomena not previously documented in GNN-based intrusion detection literature. The triangular subgraph structures induce a previously unreported three-way confusion pattern between Password, Injection, and XSS classes. This represents a significant deviation from prior homogeneous-network studies [28] [29] where such cross-class confusion was unobserved. The effect is particularly pronounced in GraphSAGE which shows 59.9% Injection-to-XSS and 48.5% Password-to-XSS mislabeling, suggesting these attacks develop similar local connectivity signatures.

The ransomware connectivity patterns exhibit a tri-modal distribution that contradicts monolithic treatments in prior approaches [15] [16] and monolithic behavioral analyses [30]. As quantified in Table IX and Table X, ransomware nodes separate into three operational categories: isolated hosts with no external connections ($W_i^{\text{ext}} = 0$; 66.7%), lateral pivots with moderate external weights ($W_i^{\text{ext}} = 1.39$; 16.7%), and command-and-control servers with high external weights ($W_i^{\text{ext}} = 50.47$; 16.7%). This granular topology distribution explains the consistent detection achieved by models.

The peripheral-hub scanning dichotomy represents another novel finding. Contrary to degree-agnostic sampling approaches common in prior work, such as stochastic analyses of

TABLE X
RANSOMWARE NODE EXTERNAL WEIGHTS

Node	W_i^{ext}
192.168.1.193	50.4655
192.168.1.37	0.0000
192.168.1.33	0.0000
34.230.157.88	1.3863
13.55.50.68	0.0000
203.14.129.10	0.0000

IoT-generated scanning that nonetheless observe structural biases [31] and federated learning–based IDS frameworks where anomalies cluster in high-connectivity nodes [32], our analysis demonstrates that scanning activity concentrates exclusively in high-degree hubs. As shown in Fig. 19, zero scanning events originate from peripheral nodes with degree less than 5, while 75% occur at the 95th-percentile hub level with degree greater than 26,113. This distribution challenges the assumption of uniform or random scanning behavior and motivates hub-prioritized detection strategies.

The heavy-tailed degree distribution in Fig. 19 concentrates 75% of scanning activity in the hub region with degree greater than 26113 and none below the peripheral cutoff with degree less than 5. This hub-centric pattern underpins the exceptionally high scanning recall of our graph models. For example, GAT correctly detects 3952 of 4000 scanning nodes or 98.8% recall by leveraging its attention mechanism to emphasize high-degree connections. GraphSAGE and GCN also benefit, achieving 97.7% and 89.0% recall respectively. These results validate our security-weighting strategy based on the large weighted diameter of 1685 since it further accentuates critical hub-to-hub pathways where most scanning originates.

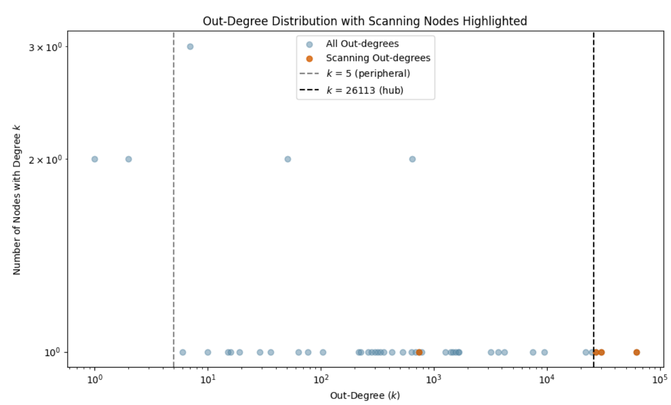


Fig. 19. Log–log scatter of the network out-degree distribution (light gray) overlaid with scanning-node frequencies (blue). Vertical dashed lines mark the peripheral cutoff $k = 5$ and the hub cutoff at the 95th percentile $k = 26{,}113$, illustrating that 75% of scanning activity originates from high-degree hubs.

C. Operational Framework for Topology-Aware Detection

Our framework's topology-aware feature engineering is informed by insights gained from graph analysis and the diagnostic evaluation of model performance conducted post-experiment:

a) **Bridge-edge tagging**: Automated identification of MITM-prone pathways via Louvain community checks
b) **Degree stratification**: Peripheral ($k < 5$) and hub ($k > 26,113$) classification
c) **Triangle density change monitoring**: Detection of confusion zones through rising changes in local triangle density $\triangle_{\text{local}}$

Alert thresholds are empirically grounded: the 0.97 probability cutoff for Injection preserves GAT's peak precision in confusion zones, while the 6.5% ExtConn threshold matches true MITM patterns. The 0.9 confidence requirement for scanning ensures operational safety in critical hubs. Table XI specifies optimized detection rules:

- **Password/Injection/XSS**: GAT with threshold $P(\text{Injection}) > 0.97$ (97% precision) + rising changes in $\triangle_{\text{local}}$ density
- **MITM**: GAT with bridge flags + $P(\text{MITM}) > 0.6$ & $\text{ExtConn} \leq 6.5\%$
- **Scanning**: GraphSAGE (hub-sampled) triggering at $k > 26,113$
- **Ransomware C2 Node**: Model-agnostic threshold $W_i^{\text{ext}} > 50$

TABLE XI
TOPOLOGY-AWARE DETECTION CONFIGURATION

Threat	Detection Protocol	Activation Criteria
Password/ Injection/XSS	GAT with triangle density monitoring	$P(\text{Inj}) > 0.97$ + rising $\triangle_{\text{local}}$
MITM	GAT with bridge-edge validation	$P(\text{mitm}) > 0.6$ + $\text{ExtConn} \leq 6.5\%$
Scanning	GraphSAGE hub prioritization	Confidence > 0.9 & Deg $> 26,113$
Ransomware C2 Node	Model-agnostic weight threshold	$W_i^{\text{ext}} > 50$

Notably, our framework omits specialized rules for backdoor, DDoS, DoS, and normal traffic due to their strong detection performance across all models. As comprehensively documented in Tables VI–VIII, these classes achieve near-perfect F1 scores ranging from 0.91 to 1.00 without requiring topological augmentation. This performance originates from their unambiguous structural signatures that models learn reliably. Backdoor attacks exhibit extreme isolation with zero external connectivity and perfect isolation. DDoS maintains moderate external connectivity at ≈25%. DoS mirrors benign traffic patterns with minimal external connectivity of 1.7%. Normal traffic demonstrates strong community alignment with over 90% perfect isolation. Consequently, the topology-aware thresholds and filters presented in Table XI specifically target only those threats where structural properties fundamentally impede detection: inducing cross-class confusion in Password/Injection/XSS clusters, enabling evasion in peripheral regions, or requiring disambiguation between malicious and benign bridge-edge traffic.

D. Future Work

Building on our topology-driven analysis, this research opens several critical avenues for advancing intrusion detection systems tailored for the unique challenges of IoT-enabled critical infrastructure, such as smart grids and and industrial control systems (ICS).

a) **Validating Topology-Based Detection in Operational Environments**: The first step is to empirically validate our topology-based filters in real-world, operational networks. Future work can focus on fine-tuning connectivity thresholds, refining edge penalty weighting to reduce trust in suspicious cross-network links, and generating synthetic bridge-edge training data to improve model robustness.
b) **Adapting Defenses to Evolving Network Topologies**: For sustainable defense in dynamic heterogeneous networks, it is crucial to monitor and adapt to topology changes. This includes tracking local triangle density and community stability over time. Adaptive technologies can leverage temporal GNNs to learn from changes in traffic patterns and enable cross-dataset knowledge transfer to improve detection on new networks.
c) **Investigating Cross-Domain Applicability to ICS**: While our study focused on a ToN-IoT dataset, the principles of topology-induced vulnerabilities and defenses are fundamentally applicable to other critical domains, including smart grid and SCADA systems. Future work should empirically test our framework's rules on specialized ICS datasets. This would involve adapting our rules to unique ICS protocols (e.g., Modbus, DNP3) and topological patterns, thereby extending our paradigm to critical infrastructure defense.

VI. CONCLUSION

This study bridges graph topology analysis and intrusion detection through a comprehensive investigation of structural properties governing threat visibility in heterogeneous IoT networks. Our graph analysis reveals how topological features including triangular subgraphs, diametric contrasts, community structures, and heavy-tailed degree distributions create operational challenges that shape detection strategies. By augmenting GCN, GraphSAGE, and GAT with dual imbalance mitigation strategies, our evaluation uncovered critical topology-driven diagnostics across hard-to-classify classes. Future research directions of validating topology-based detection in operational environments, adapting defenses to evolving network topologies, and investigating cross-domain applicability to ICS are essential for building advanced, context-aware defenses against sophisticated cyber threats in a world of proliferating IoT devices. By treating network structure as a detection parameter, this work establishes a new paradigm for operationalizing graph theory in the security of complex and heterogeneous interconnected ecosystems.

REFERENCES

[1] S. Arisdakessian, O. A. Wahab, A. Mourad, H. Otrok, and M. Guizani, "A Survey on IoT Intrusion Detection: Federated Learning, Game Theory, Social Psychology, and Explainable AI as Future Directions," *IEEE Internet of Things Journal*, vol. 10, no. 5, pp. 4059–4092, 2023, doi: 10.1109/JIOT.2022.3203249.

[2] D. Herranz-Oliveros, M. Tejedor-Romero, J. M. Gimenez-Guzman, and L. Cruz-Piris, "Unsupervised Learning for Lateral-Movement-Based Threat Mitigation in Active Directory Attack Graphs," *Electronics*, vol. 13, no. 19, 2024, doi: 10.3390/electronics13193944.

[3] M. M. Rahman, S. A. Shakil, and M. R. Mustakim, "A Survey on Intrusion Detection System in IoT Networks," *Cyber Security and Applications*, vol. 3, p. 100082, 2025, doi: 10.1016/j.csa.2024.100082.

[4] T. Bilot, N. E. Madhoun, K. A. Agha, and A. Zouaoui, "Graph Neural Networks for Intrusion Detection: A Survey," *IEEE Access*, vol. 11, pp. 49 114–49 139, 2023, doi: 10.1109/ACCESS.2023.3275789.

[5] S. K. Moorthy and J. Jagannath, "Survey of Graph Neural Network for Internet of Things and NextG Networks," 2024, doi: 10.48550/arXiv.2405.17309.

[6] N. Moustafa, "A New Distributed Architecture for Evaluating AI-Based Security Systems at the Edge: Network TON_IoT Datasets," *Sustainable Cities and Society*, vol. 72, p. 102994, 2021, doi: 10.1016/j.scs.2021.102994.

[7] T. M. Booij, I. Chiscop, E. Meeuwissen, N. Moustafa, and F. T. H. d. Hartog, "ToN_IoT: The Role of Heterogeneity and the Need for Standardization of Features and Attack Types in IoT Network Intrusion Data Sets," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 485–496, 2022, doi: 10.1109/JIOT.2021.3085194.

[8] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, "TON_IoT Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems," *IEEE Access*, vol. 8, pp. 165 130–165 150, 2020, doi: 10.1109/ACCESS.2020.3022862.

[9] N. Moustafa, M. Keshky, E. Debiez, and H. Janicke, "Federated TON_IoT Windows Datasets for Evaluating AI-Based Security Applications," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2020, pp. 848–855, doi: 10.1109/TrustCom50675.2020.00114.

[10] N. Moustafa, M. Ahmed, and S. Ahmed, "Data Analytics-Enabled Intrusion Detection: Evaluations of ToN_IoT Linux Datasets," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2020, pp. 727–735, doi: 10.1109/TrustCom50675.2020.00100.

[11] N. Moustafa, "New Generations of Internet of Things Datasets for Cybersecurity Applications Based Machine Learning: TON_IoT Datasets," in *Proceedings of the eResearch Australasia Conference*, 2019, pp. 21–25.

[12] N. Moustafa, "A Systemic IoT–Fog–Cloud Architecture for Big-Data Analytics and Cyber Security Systems: A Review of Fog Computing," *Secure Edge Computing*, pp. 41–50, 2021, doi: 10.1201/9781003028635-4.

[13] J. Ashraf, M. Keshk, N. Moustafa, M. Abdel-Basset, H. Khurshid, A. D. Bakhshi, and R. R. Mostafa, "IoTBoT-IDS: A Novel Statistical Learning-enabled Botnet Detection Framework for Protecting Networks of Smart Cities," *Sustainable Cities and Society*, vol. 72, p. 103041, 2021, doi: 10.1016/j.scs.2021.103041.

[14] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," 2016, doi: 10.48550/arXiv.1609.02907.

[15] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive Representation Learning on Large Graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, p. 1025–1035, doi: 10.5555/3294771.3294869.

[16] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," 2018, doi: 10.48550/arXiv.1710.10903.

[17] M. Sarhan, S. Layeghy, and M. Portmann, "Towards a Standard Feature Set for Network Intrusion Detection System Datasets," *Mobile Networks and Applications*, vol. 27, no. 1, pp. 357–370, 2022, doi: 10.1007/s11036-021-01843-0.

[18] C. Wu, J. Sun, J. Chen, M. Alazab, Y. Liu, and Y. Xiang, "TCG-IDS : Robust Network Intrusion Detection via Temporal Contrastive Graph Learning," *IEEE Transactions on Information Forensics and Security*, vol. 20, pp. 1475–1486, 2025, doi: 10.1109/TIFS.2025.3530702.

[19] S. Zhang, R. Chen, J. Chen, Y. Zhu, M. Hua, J. Yuan, and F. Xu, "L-GraphSAGE: A Graph Neural Network-Based Approach for IoV Application Encrypted Traffic Identification," *Electronics*, vol. 13, no. 21, 2024, doi: 10.3390/electronics13214222.

[20] A. J. Fofanah, D. Chen, L. Wen, and S. Zhang, "Addressing Imbalance in Graph Datasets: Introducing GATE-GNN With Graph Ensemble Weight Attention and Transfer Learning for Enhanced Node Classification," *Expert Systems with Applications*, vol. 255, p. 124602, 2024, doi: 10.1016/j.eswa.2024.124602.

[21] "ISCX NSL-KDD Dataset 2009," available on: https://www.unb.ca/cic/datasets/nsl.html, 2009.

[22] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6, doi: 10.1109/CISDA.2009.5356528.

[23] N. Moustafa and J. Slay, "UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1–6, doi: 10.1109/MilCIS.2015.7348942.

[24] E. W. Dijkstra, "A Note on Two Problems in Connexion With Graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959, doi: 10.1007/BF01386390.

[25] A. Hagberg, P. J. Swart, and D. A. Schult, "Exploring Network Structure, Dynamics, and Function Using NetworkX," Los Alamos National Laboratory, Tech. Rep., 2008.

[26] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast Unfolding of Communities in Large Networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, 2008, doi: 10.1088/1742-5468/2008/10/P10008.

[27] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2999–3007, doi: 10.1109/ICCV.2017.324.

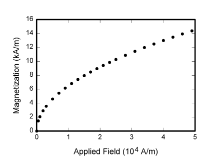
[28] H. Asad, S. Adhikari, and I. Gashi, "A Perspective–Retrospective Analysis of Diversity in Signature-Based Open-Source Network Intrusion Detection," *International Journal of Information Security*, vol. 23, no. 2, pp. 1331–1346, 2024, doi: 10.1007/s10207-023-00794-9.

[29] R. Holdbrook, O. Odeyomi, S. Yi, and K. Roy, "Network-Based Intrusion Detection for Industrial and Robotics Systems: A Comprehensive Survey," *Electronics*, vol. 13, no. 22, p. 4440, 2024, doi: 10.3390/electronics13224440.

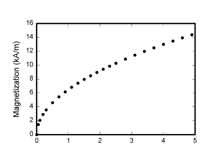
[30] H. Abu-Helo and H. Ashqar, "Early Ransomware Detection System Based on Network Behavior," in *Advanced Information Networking and Applications*, 2024, pp. 447–458, doi: 10.1007/978-3-031-57931-8_43.

[31] M. Safaei Pour, S. Torabi, E. Bou-Harb, C. Assi, and M. Debbabi, "Stochastic Modeling, Analysis and Investigation of IoT-Generated Internet Scanning Activities," *IEEE Networking Letters*, vol. 2, no. 3, pp. 159–163, 2020, doi: 10.1109/LNET.2020.2998045.

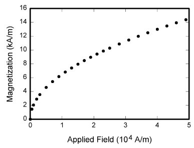
[32] X. S. de Cámara, J. L. Flores, C. Arellano, A. Urbieta, and U. Zurutuza, "Clustered Federated Learning Architecture for Network Anomaly Detection in Large Scale Heterogeneous IoT Networks," *Computers & Security*, vol. 131, p. 103299, 2023, doi: 10.1016/j.cose.2023.103299.



John Doe In this paragraph you can place John Doe's educational, professional background and research and other interests.



Jane Smith In this paragraph you can place Jane Smith's educational, professional background and research and other interests.



Peter Jones In this paragraph you can place Peter Jones's educational, professional background and research and other interests.