big data and cognitive computing

MDPI

Review

Network Traffic Analysis Based on Graph Neural Networks: A Scoping Review

Ruonan Wang *, Jinjing Zhao, Hongzheng Zhang, Liqiang He, Hu Li and Minhuan Huang *

Institute of Systems Engineering, Academy of Military Sciences, People's Liberation Army, Beijing 100101, China; zhaojinjing@nudt.edu.cn (J.Z.); kuku13145@126.com (H.Z.); hlq0722@126.com (L.H.); lihu_lh@163.com (H.L.)
* Correspondence: strickland_wang@163.com (R.W.); darbean@126.com (M.H.)

Abstract

Network traffic analysis is crucial for understanding network behavior and identifying underlying applications, protocols, and service groups. The increasing complexity of network environments, driven by the evolution of the Internet, poses significant challenges to traditional analytical approaches. Graph Neural Networks (GNNs) have recently garnered considerable attention in network traffic analysis due to their ability to model complex relationships within network flows and between communicating entities. This scoping review systematically surveys major academic databases, employing predefined eligibility criteria to identify and synthesize key research in the field, following the Preferred Reporting Items for Systematic reviews and Meta-Analyses extension for Scoping Reviews (PRISMA-ScR) methodology. We present a comprehensive overview of a generalized architecture for GNN-based traffic analysis and categorize recent methods into three primary types: node prediction, edge prediction, and graph prediction. We discuss challenges in network traffic analysis, summarize solutions from various methods, and provide practical recommendations for model selection. This review also compiles publicly available datasets and open-source code, serving as valuable resources for further research. Finally, we outline future research directions to advance this field. This work offers an updated understanding of GNN applications in network traffic analysis and provides practical guidance for researchers and practitioners.

Keywords: network traffic analysis; network security; graph neural networks; graph feature aggregation; graph-based methods

Academic Editor: Magdalini Eirinaki

Received: 21 July 2025
Revised: 11 October 2025
Accepted: 21 October 2025
Published: 24 October 2025

Citation: Wang, R.; Zhao, J.; Zhang, H.; He, L.; Li, H.; Huang, M. Network Traffic Analysis Based on Graph Neural Networks: A Scoping Review. *Big Data Cogn. Comput.* **2025**, *9*, 270. https://doi.org/10.3390/bdcc9110270

1. Introduction

Driven by the ongoing developments in Internet technologies, the Internet has become an integral part of daily operations for both individuals and enterprises. According to the latest report from the International Telecommunication Union (ITU) [1], the global number of Internet users reached 5.5 billion in 2024, an increase of 227 million over the 2023 estimate. This explosive user growth presents substantial challenges to Internet management, necessitating effective traffic analysis methods. These methods are crucial for applications such as user traffic identification, encrypted traffic classification, malicious traffic identification, intrusion detection, network anomaly detection, and website fingerprinting, all essential for ensuring quality online experiences and maintaining network security.

Researchers have continuously adapted to technological advancements over the years, implementing new network traffic analysis methods. Early rule-based analysis methods were widely adopted [2], but their primary limitation was poor generalization to new

patterns. Minor modifications to attack sequences were often sufficient to bypass these rules [3]. For instance, port-based analysis methods became less effective with the advent of dynamic ports. Subsequently, Deep Packet Inspection (DPI) techniques emerged [4]. However, DPI became progressively ineffective with the continuous refinement of encryption technologies because of its inability to analyze encrypted packet content. Machine learning-based approaches achieved considerable success in addressing these issues [5–8]. However, feature engineering in machine learning necessitates domain experts to select useful features for downstream detection systems. This reliance on prior knowledge hinders adaptability to the rapidly changing nature of contemporary network traffic.

Deep learning-based methods have become the mainstream approach in contemporary network traffic analysis. Common deep learning methods include Multi-Layer Perceptrons (MLPs) [9], Recurrent Neural Networks (RNNs) [10], Autoencoders (AEs) [11], and Convolutional Neural Networks (CNN)s [12]. Current deep learning methods primarily focus on extracting features from individual traffic flows or packet sequences but exhibit limited awareness of the interactions between traffic flows. Network traffic data are inherently structured, with rich information embedded in the relationships between various elements such as IP addresses, protocols, and host events. Consequently, traditional deep learning architectures may not fully exploit the structural properties of network traffic in the domain of network traffic analysis. Researchers have extended deep learning methods to graph data to leverage graph structural information. A key characteristic of graph data, compared to other data structures, is its ability to effectively capture multiple interaction relationships within complex systems [13].

In numerous fields, GNN-based methods have demonstrated strong performance [14,15], effectively capturing intricate interactions and relationships within complex systems. Due to the inherently structured interaction properties of traffic data, industry practitioners have also started applying GNNs in the field of traffic analysis, resulting in a growing body of research. However, there is currently no systematic review in this area, and existing studies such as [16,17] emphasize the importance of structured overviews. A systematic review would summarize existing frameworks and provide guidance on selecting appropriate prediction methods, graph construction techniques, and feature aggregation strategies for different tasks. It would also help identify the limitations and future directions of current research and evaluate the advantages and disadvantages of applying GNNs in this context. Therefore, a comprehensive survey is urgently needed, as it is crucial for enhancing the theoretical understanding and practical applications of GNNs in network traffic analysis.

To address the research gap in systematic reviews of GNN-based traffic analysis, this work begins by revisiting fundamental concepts of GNNs and their applicability to traffic analysis. We then summarize the current challenges in traffic analysis and the inherent advantages of GNNs for this domain. We conduct a methodical review of 78 studies (details in Figure 1). We organize these studies into three prediction task categories: node-level, edge-level, and graph-level prediction. This allows us to analyze their graph construction techniques and feature aggregation strategies and conduct a comparative analysis across different prediction tasks. Finally, we compile publicly available datasets and open-source resources, and summarize the limitations of current approaches while outlining potential directions for future research.

The main contributions of this paper are as follows:

1. Technical Workflow Framework: A structured technical workflow framework for GNN-based traffic analysis is introduced, serving as a valuable guide for researchers and practitioners.
2. Comprehensive Task-Based Analysis: This review presents a thorough review of 78 studies categorized by prediction tasks. It summarizes common graph construction

methods and node aggregation strategies, along with a comparative analysis of techniques in typical task scenarios.

3. Challenge-Driven Perspective: A unique challenge-driven perspective is provided, analyzing how representative works tackle key traffic analysis challenges using GNN-based methods.
4. Curated Resource Collection: A selection of publicly available datasets and open-source implementations is provided to support reproducible research and benchmarking in GNN-based traffic analysis.

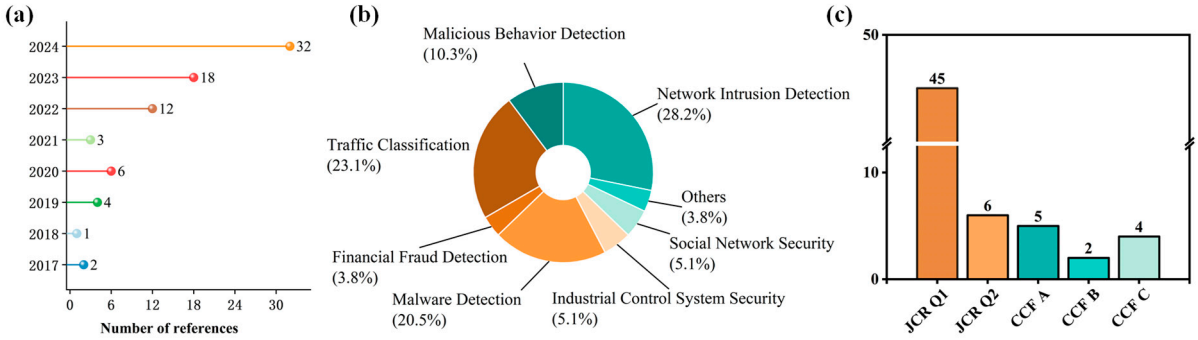

Figure 1. Statistical data of articles included in survey. (**a**) Number of articles categorized by year, (**b**) number of articles in each field, and (**c**) number of articles in each level of conference or journal.

The remainder of this paper is organized as follows. Section 2 shows the review methodology used for the survey. Section 3 reviews existing survey research on GNN-based traffic analysis. Section 4 discusses challenges and opportunities in network traffic analysis, emphasizing the advantages of GNNs. Section 5 presents background knowledge and general framework of GNNs. Section 6 comprehensively reviews GNN-based network traffic analysis methods. Section 7 summarizes public datasets and open-source code resources. Section 8 discusses the limitations of this review and future research directions. Finally, Section 9 concludes the paper.

2. Review Method

This scoping review was conducted and reported in accordance with the PRISMA-ScR guidelines. The primary objective was to systematically map the literature on GNN applications in network traffic analysis and network security, identifying research trends, methodological approaches, and knowledge gaps. No review protocol was registered prior to the commencement of this study.

We conducted systematic searches across major academic databases including IEEE Xplore, ACM Digital Library, Scopus, and Web of Science from January 2024 to April 2025. The search strategy combined keywords related to "network traffic analysis", "network security", "GNNs", "graph feature aggregation", and "graph-based methods", restricted to English-language publications. We included primary research studies that focused on GNN applications in network traffic analysis or security domains, proposed novel methodologies with clear technical innovation, and provided reliable experimental evaluations. Preference was given to studies with open-source datasets and code for reproducibility.

Two reviewers independently performed the study selection process. A third reviewer resolved any disagreements. Our initial search identified 388 records, as depicted in the PRISMA flow diagram (Figure 2). After removing duplicates and screening based on title and abstract, we assessed 147 reports for full-text eligibility. We excluded studies based on the following criteria: ineligible topics (n = 24), unclear methodology (n = 16), lack of source code (n = 13), outdated methods (n = 10), and conference abstracts (n = 6). In the end, 78 studies met the inclusion criteria and were included in the synthesis.

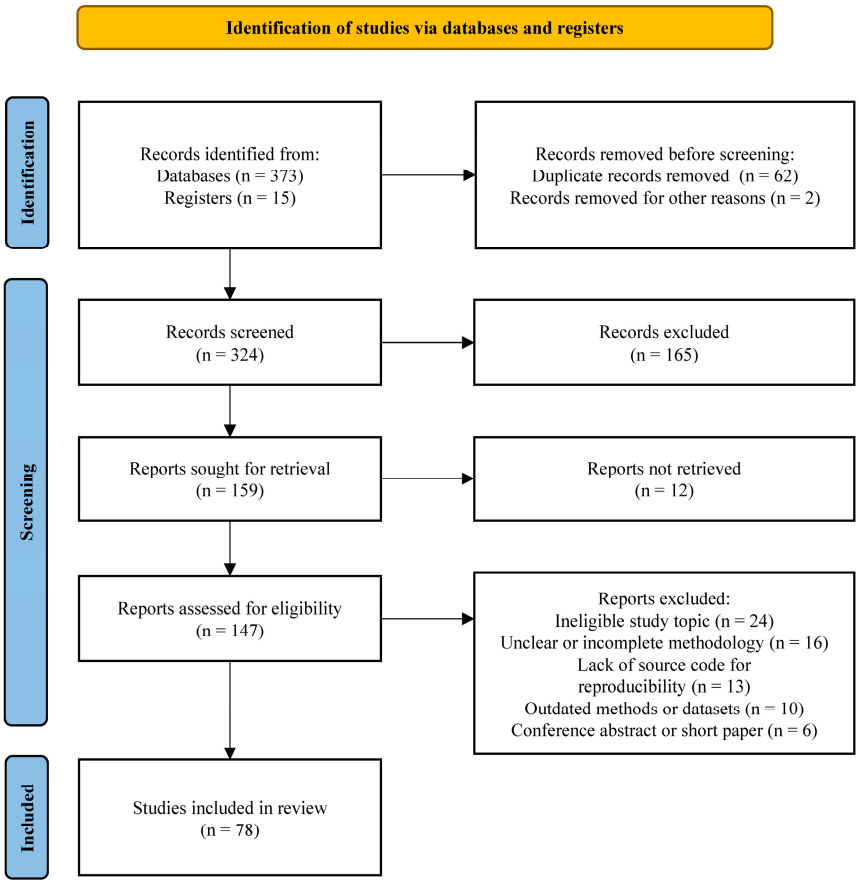

Figure 2. The scoping review procedure using PRISMA-ScR.

We performed data extraction using a standardized form. This form covered publication details, application domains, GNN architectures, graph construction methods, datasets used, evaluation metrics, and code availability. We performed a descriptive synthesis of the extracted data, presenting the results through descriptive statistics and visualizations. Quantitative meta-analysis was not deemed appropriate due to significant heterogeneity in methodologies, datasets, and evaluation metrics across studies. We used the excluded survey papers for contextualization in the Related Work chapter, but we did not include them in the primary synthesis.

3. Related Surveys

This chapter reviews existing surveys from two perspectives: (1) surveys on network traffic analysis, and (2) surveys on GNN-based traffic analysis tasks.

3.1. Surveys on Network Traffic Analysis

Existing surveys primarily focus on traditional techniques for network traffic analysis. For instance, Goli et al. [18] and Bhatla et al. [19] reviewed machine learning methods (e.g., supervised and unsupervised learning), while Azab et al. [20] and Getman et al. [21] compared the strengths and weaknesses of various approaches. Papadogiannaki et al. [22] explored encrypted traffic analysis, and Meng et al. [23] briefly mentioned GNNs in decentralized applications. However, these works lack a systematic analysis of GNN-based methodologies, which is a key focus of this article.

3.2. Surveys on GNN-Based Traffic Analysis Tasks

Other surveys focus on GNN-based approaches for specific traffic analysis tasks, including intrusion detection [24,25], anomaly detection [26,27], cybersecurity solutions [28],

and malware detection [29]. While these surveys provide valuable insights into GNN applications in specific areas, they lack a unified perspective and comprehensive framework for the broader field of traffic analysis.

Existing surveys predominantly address conventional methods with minimal discussion of GNNs, while GNN-related surveys are fragmented and lack a holistic perspective. To address these gaps, this article provides a systematic review of GNN-based traffic analysis, unifying diverse methodologies into a coherent framework and offering practical guidance for researchers and practitioners.

The key content of these surveys is summarized in Table 1, highlighting the gaps addressed by this article.

Table 1. Related Review Summary.

	Surveys	Year	Task	Survey Content
Traffic Analysis	Goli et al. [18]	2018	Real-time traffic analysis	Machine learning
	Bhatla et al. [19]	2023	Network traffic analysis	Machine learning
	Azab et al. [20]	2022	Encrypted traffic analysis	Machine learning
	Getman et al. [21]	2022	Encrypted traffic analysis	Deep learning
	Papadogiannaki et al. [22]	2021	Encrypted traffic analysis	Machine learning
	Meng et al. [23]	2023	Encrypted traffic analysis	Machine, Deep and Graph learning
GNN-based Traffic Analysis	Bilot et al. [24]	2024	Intrusion detection	GNN
	Zhong et al. [25]	2024	Intrusion detection	GNN
	Dong et al. [26]	2024	IoT security	GNN
	Pazho et al. [27]	2023	Anomaly detection	GNN
	Yan et al. [28]	2024	Network infrastructure security	GNN
	Madhoun et al. [29]	2023	Malware detection	GNN
Our Survey		2025	Traffic analysis	GNN

4. Challenges and Opportunities

This chapter introduces key challenges in network traffic analysis and discusses how GNNs address these challenges.

4.1. Challenges in Network Traffic Analysis

We categorize the research directions in the field of traffic analysis into three primary domains: network analysis, network security, and user privacy. Furthermore, we systematically classify the GNN-based traffic analysis literature encompassed in this survey according to their specific problem formulations, as illustrated in Figure 3. Although traffic analysis problems and their associated research directions exhibit considerable diversity, the majority of these problems encounter common fundamental challenges.

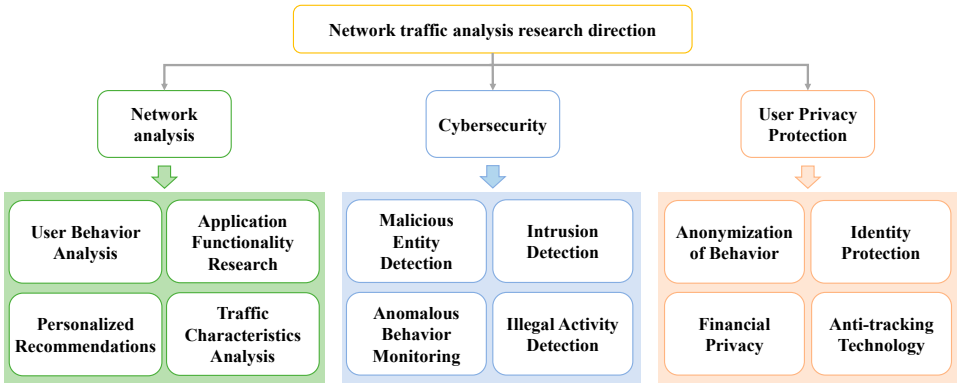

Figure 3. Research Directions in Network Traffic Analysis.

4.1.1. Training Data Challenges

Training data quality issues significantly compromise traffic analysis model performance. Key problems include insufficient data volumes, imbalanced distributions, and

annotation errors. Small sample problems are particularly severe in specialized domains like emerging network attacks or rare application types [30], where limited training samples cause model overfitting and poor generalization to real world scenarios.

Data imbalance affects many traffic analysis tasks. In anomaly detection [31], normal traffic samples vastly outnumber anomalous ones, causing models to favor majority classes and compromise minority class recognition. Traffic data annotation is labor-intensive and error-prone, resulting in mislabeling and inconsistent annotations that further degrade training quality. For instance, encrypted traffic classification [32] requires in-depth packet analysis, which becomes extremely challenging in encrypted environments.

Effectively addressing training data quality issues and developing robust algorithms that adapt to small samples, class imbalance, and annotation inaccuracies represents a critical challenge in traffic analysis research.

4.1.2. Feature Extraction Challenges

Feature selection involves identifying and extracting the most representative features from large amounts of network traffic data. This requires a deep understanding of network protocols and application behaviors. While deep learning approaches can automatically learn features, their lack of interpretability makes it difficult to understand and validate the model's decision-making process, increasing the complexity of model optimization and deployment.

Additionally, feature tuning becomes complex due to the need to continuously adapt to evolving network environments and emerging application types, requiring researchers to constantly update and refine feature extraction strategies. For instance, in encrypted traffic classification [33], effective features may evolve temporally as encryption algorithms advance. In anomaly detection tasks [34], the dynamic variations in normal traffic patterns further exacerbate the complexity of feature selection. Developing robust, efficient, and adaptive feature extraction methodologies represents a pivotal challenge in traffic analysis research.

4.1.3. Model Design Challenges

The high-dimensional, high-velocity, and high-variability characteristics of network traffic require models with robust feature extraction and learning capabilities. Traditional machine learning models often cannot handle complex non-linear relationships effectively. While deep learning models can automatically learn complex features, their "black-box" nature makes model interpretation and optimization more difficult.

Furthermore, real-time processing imposes substantial constraints on model design. For example, in intrusion detection systems (IDSs) [35], models must analyze massive traffic volumes and make decisions within extremely short timeframes. This demands both high accuracy and exceptional computational efficiency. Additionally, the dynamic nature of network environments presents significant design challenges. Emerging application types, attack vectors, and encryption protocols require models with strong adaptability and generalization capabilities. For instance, in anomaly detection [36], models must identify novel attack patterns.

Finally, model interpretability and trustworthiness constitute indispensable design considerations. In security-sensitive scenarios, high accuracy alone is insufficient; models must also provide decision rationales to support subsequent analysis and response actions [37]. Consequently, designing models that can efficiently process complex traffic data while maintaining superior adaptability, interpretability, and real-time performance represents one of the core challenges in traffic analysis research.

4.2. Advantages of GNNs

GNNs offer significant advantages in addressing the three challenges mentioned above.

For training data challenges, GNNs can better utilize limited data by leveraging graph structure and node relationships. The graph representation captures connectivity patterns that help improve learning from small datasets and handle class imbalance through neighborhood information.

For feature extraction challenges, GNNs automatically learn features from graph structure, eliminating manual feature engineering. The message-passing mechanism enables adaptive feature learning from node and edge information, reducing the need for domain expertise.

For model design challenges, GNNs provide flexible architectures that can scale to different network sizes and adapt to various traffic analysis tasks. Their modular design enables efficient processing while maintaining interpretability through graph visualization.

In the following chapters, we systematically review existing GNN-based traffic analysis work from three perspectives: node-level, edge-level, and graph-level analysis approaches.

5. Background Knowledge and General Framework

5.1. GNN Background Knowledge

5.1.1. General Graph Structure

Graphs constitute a fundamental concept in mathematics and computer science for representing relationships between objects. This subsection introduces graph structures commonly used in traffic analysis, categorized by node and edge types, graph directionality, and graph attributes. Figure 4 presents a summary of these graph structures.

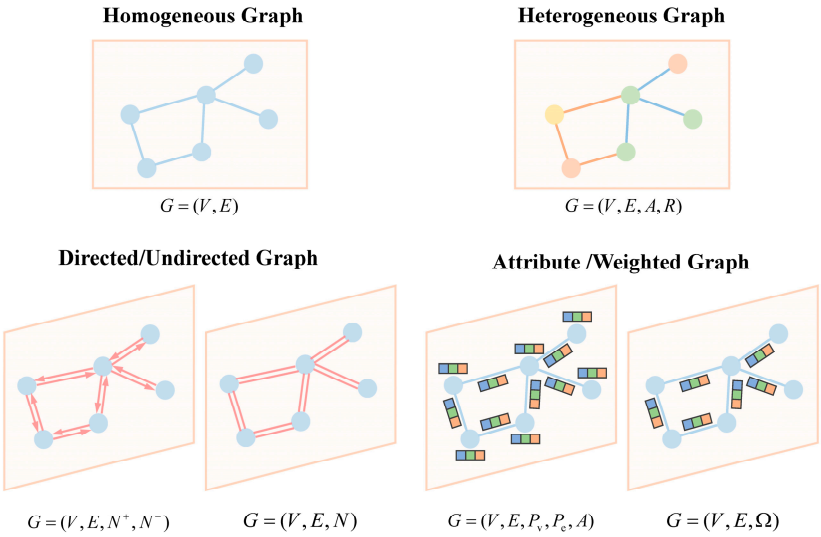

Figure 4. Common Graph Structures Used in Network Traffic Analysis.

Homogeneous Graph: A homogeneous graph can be defined as an ordered pair $G = (V, E)$, where V represents a non-empty finite set of nodes (vertices), and E represents a finite set of edges (relationships). In this definition, all nodes $v \in V$ belong to the same type (e.g., all are IP addresses), having the same attribute structure and semantic meaning, while all edges $e \in E$ represent the same type of relationship or interaction (e.g., all are TCP connections).

Heterogeneous Graph: A heterogeneous graph can be defined as an ordered four-tuple $G = (V, E, \mathcal{A}, \mathcal{R})$, where $\mathcal{A}$ represents a finite set of entity types (e.g., IP addresses, device types, user roles, etc.), and $\mathcal{R}$ represents a finite set of relationship types (e.g., TCP connections, HTTP requests, login behaviors, etc.). In heterogeneous graphs, there exist mapping functions

$\phi : V \to \mathcal{A}$ and $\psi : E \to \mathcal{R}$ that, respectively, map network entities and relationships to their corresponding types.

Directed/Undirected Graph: A directed graph can be defined as an ordered four-tuple $G = (V, E, \mathcal{N}^+, \mathcal{N}^-)$, where $\mathcal{N}^+ : V \to 2^V$ is the out-edge adjacency function. For any $v_i \in V$, $\mathcal{N}^+(v_i) = \{v_j \in V | e_{ij} \in E\}$, and $\mathcal{N}^- : V \to 2^V$ is the in-edge adjacency function. For any $v_i \in V$, $\mathcal{N}^-(v_i) = \{v_j \in V | e_{ji} \in E\}$. An undirected graph can be defined as an ordered three-tuple $G = (V, E, \mathcal{N})$, where $\mathcal{N} : V \to 2^V$ is the adjacency function. For any $v_i \in V$, $\mathcal{N}(v_i) = \{v_j \in V | e_{ij} \in E\}$. In directed graphs, e_{ij} represents a unidirectional connection from v_i to v_j, while in undirected graphs, e_{ij} represents a bidirectional connection. Edge directionality can be flexibly defined during specific computations according to task requirements; for example, in directed graphs, in-edges can be represented as positive numbers and out-edges as negative numbers, or vice versa.

Attributed/Weighted Graph: An attributed graph is defined as a five-tuple $G = (V, E, P_v, P_e, A)$, where $P_v : V \to 2^{A_v}$ maps nodes to attribute sets, $P_e : E \to 2^{A_e}$ maps edges to attribute sets, and $A = A_v \cup A_e$ represents the set of all possible attributes. This structure allows nodes and edges to possess multiple attributes, allowing the representation of complex network characteristics. A weighted graph can be defined as a three-tuple $G = (V, E, \Omega)$, where $\Omega : E \to \mathbb{R}$ is a function that maps edges to real-valued weights, primarily used to represent single numerical features such as traffic volume. In certain studies, to simplify computation, node attributes may be defined as constants, i.e., $P_v : V \to c$, where c is a constant value. This simplification can reduce computational complexity while preserving basic network structure, making it particularly suitable for analysis tasks that primarily focus on edge attributes or network topology.

Note that graph types are not mutually exclusive, and in practical applications, their properties can be combined to meet specific task requirements. For example, a weighted directed graph can be defined as a five-tuple $G = (V, E, \mathcal{N}^+, \mathcal{N}^-, \Omega)$. This structure combines the directionality of directed graphs with the numerical attributes of weighted graphs. This flexible approach enables graph models to adapt to various network traffic analysis scenarios, from simple traffic volume analysis to complex multi-dimensional network behavior modeling.

5.1.2. Graph Tasks

When constructing graphs, data can be represented as a single large graph for node or edge prediction through transductive learning. Alternatively, it can be represented as multiple graph structures, learning from these graphs and then inductively generalizing to new graphs for graph prediction.

Node Prediction: Node prediction aims to classify individual nodes in network traffic graphs. In traffic analysis, nodes typically represent individual data packets, flows, or hosts. The model classifies nodes by learning their features (such as protocol type, port number, packet size) and topological relationships within the graph. During training, GNNs aggregate information from neighboring nodes, capturing both local and global features of the nodes in the traffic graph. For large-scale traffic graphs, mini-batch training is commonly used, sampling subsets of nodes and their neighborhoods. During prediction, the model can classify unlabeled nodes in the graph, such as identifying malicious traffic or specific application types.

Edge Prediction: Edge prediction aims to predict the type or attributes of edges in traffic graphs. In network traffic analysis, edges may represent connections between hosts, sessions, or data transmissions. The model learns features of both endpoint nodes and their interaction patterns. Common approaches include representing edges as combinations of features from two nodes, or using GNNs to learn embedding representations of edges.

During training, negative sampling is typically used, randomly selecting non-existent edges as negative samples. During prediction, the model can classify potential edges between any two nodes in the graph, such as detecting anomalous connections or identifying specific types of network interactions.

Graph Prediction: Graph prediction aims to classify entire traffic graph structures. In network security and traffic analysis, each graph may represent network activity within a specific time window or a complete attack session. The model captures the global topological structure and node feature distribution of the entire traffic graph. A common approach uses graph pooling operations to aggregate node-level features into graph-level representations, which are then input to a classifier. During training, each complete traffic graph is treated as a sample. Training typically requires a large number of diverse traffic graphs. This enables the model to learn characteristics of various network behaviors and attack patterns. Graph classification is important for network anomaly detection, attack type identification, and network status assessment.

5.1.3. Representative GNN Architectures

In this section, we introduce the graph models commonly used in traffic analysis and summarize their characteristics, as shown in Table 2.

Table 2. GNN Architectures Comparison.

Architecture	Key Features	Aggregation Methods	Attention Mechanism	Use Cases
GNN	Message passing, iterative updates	Neighborhood aggregation	No	General tasks, node classification
GCN	Spectral convolution, localized	Neighborhood averaging	No	Node/graph classification, l ink prediction
GraphSAGE	Inductive learning, scalable	Sampling-based aggregation	No	Large-scale analysis, inductive learning
Edge-GraphSAGE	Edge-level prediction, combines node and edge features	Node and edge feature aggregation	No	Edge prediction, relationship analysis
GAT	Attention-based weighting	Attention-weighted aggregation	Yes (edge-level)	Node/graph classification
GIN	Graph isomorphism-inspired, learnable epsilon	Sum aggregation with epsilon weighting	No	Graph/node classification
HAN	Meta-path-based heterogeneous processing	Meta-path guided aggregation	Yes (node and semantic level)	Heterogeneous graph analysis

GNN [38]: The GNN was developed to extend deep learning methods to non-Euclidean data. GNN computation involves two main processes: feature aggregation and node update. In the feature aggregation stage, each node collects information from its neighbors to form updated node representations. This process can be represented as follows:

$$h_i^{(l+1)} = f\left(h_i^{(l)}, \text{AGGREGATE}\left(\left\{h_j^{(l)} \middle| j \in \mathcal{N}(i)\right\}\right)\right) \quad (1)$$

where $h_i^{(l)}$ is the feature vector of node i in the l-th layer, $\mathcal{N}(i)$ represents the neighbor node set of node i, and AGGREGATE is an aggregation function used to collect information from neighbors. Through multi-layer aggregation and updates, GNN can gradually integrate information from distant nodes into node representations, enabling global information propagation.

GCN [39]: GCN (Graph Convolutional Networks) is an important member of the GNN family that extends convolution operations to graph-structured data. The core GCN operation is as follows:

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

where $H^{(l)}$ is the node feature matrix of the l-th layer, $\overline{A} = A + I$ is the adjacency matrix with added self-loops, D is the degree matrix of $\overline{A}$, $W^{(l)}$ is the learnable weight matrix, and σ is the non-linear activation function. The key to this operation lies in the $D^{-\frac{1}{2}}\overline{A}D^{-\frac{1}{2}}H^{(l)}$ part, which actually performs a weighted average of each node's neighborhood information. Specifically, $\overline{A}$ ensures that each node considers information from itself and its direct neighbors, while $D^{-\frac{1}{2}}\overline{A}D^{-\frac{1}{2}}$ normalizes the adjacency matrix to prevent nodes with higher

degrees from dominating the aggregation process. GCN achieves multi-hop information propagation by stacking multiple layers, enabling the model to capture larger-scale graph structures. With each additional layer, the receptive field of nodes expands by one hop:

$$H^{(l+1)} = \sigma\left(D^{-\frac{1}{2}}\overline{A}D^{-\frac{1}{2}}\sigma\left(D^{-\frac{1}{2}}\overline{A}D^{-\frac{1}{2}}\ldots\sigma\left(D^{-\frac{1}{2}}\overline{A}D^{-\frac{1}{2}}H^{(0)}W^{(0)}\right)\ldots W^{(l-1)}\right)W^{(l)}\right) \quad (3)$$

This multi-layer structure allows GCN to progressively aggregate information from broader ranges, learning more abstract and global feature representations.

GraphSAGE [40]: GraphSAGE (Graph Sample and Aggregation) processes large-scale graph-structured data. Unlike traditional full-graph methods, GraphSAGE performs node embedding learning by sampling and aggregating features from neighbor nodes. This approach reduces computational complexity and enables efficient scaling to larger graph datasets. GraphSAGE involves two main steps: neighbor sampling and feature aggregation. First, for each target node, GraphSAGE samples a fixed number of nodes from the neighbor set using random sampling. Neighbor sampling can be represented as follows:

$$N_k(v) = \text{SAMPLE}(N(v), k) \quad (4)$$

where $N(v)$ is the set of all neighbors of node v, k is the sampling number, and $N_k(v)$ are the k neighbors obtained through sampling. This sampling strategy reduces dependency on full graph data and simplifies computation. Then, GraphSAGE performs feature aggregation on the sampled neighbor nodes. The feature aggregation formula is as follows:

$$h_v^k = \sigma\left(W^k \cdot \text{CONCAT}\left(h_v^{k-1}, \text{AGGREGATE}\left(\left\{h_u^{k-1}, \forall u \in N_k(v)\right\}\right)\right)\right) \quad (5)$$

h_v^k represents the embedding of node v in the k-th round, $N_k(v)$ is the sampled neighbor set of node v, W^k is the trainable weight matrix, and σ is the activation function. GraphSAGE employs various aggregation functions, such as mean, pooling, or LSTM. Different aggregation functions enable GraphSAGE to capture rich neighbor information. This flexible approach allows different contextual information to be reflected in node embeddings, improving the model's representational capability. This is especially beneficial for heterogeneous graphs and data with diverse connection patterns.

Edge-GraphSAGE [40]: Edge-GraphSAGE is a GNN variant specifically designed for edge-level prediction tasks. As an extension of GraphSAGE, it generates edge embeddings by aggregating node and neighbor features while considering edge attributes. The node representation update formula is as follows:

$$h_v^k = \sigma(W_k \cdot \text{CONCAT}(h_v^{(k-1)}, \text{AGG}_k(\left\{\phi(h_u^{(k-1)}, e_{vu}) : u \in N(v)\right\}))) \quad (6)$$

where σ is the activation function, AGG_k is the aggregation function, and ϕ is the function that combines node and edge features. Edge representations can be generated through $e_{(u,v)} = f(\text{CONCAT}(h_u^K, h_v^K, e_{uv}))$, combining the final representations of nodes at both ends of the edge with the original edge features.

GAT [41]: GAT (Graph Attention Network) dynamically allocates importance weights to neighbor nodes using attention mechanisms. Unlike methods that uniformly weight neighbor features, GAT enhances the flexibility and accuracy of node feature learning on heterogeneous graphs. The core computational steps of GAT include the definition of attention mechanisms and feature aggregation. For each node v and its neighbors $u \in N(v)$, GAT first computes pairwise attention coefficients. This is achieved through a shared linear transformation and attention mechanism:

$$e_{vu} = \text{LeakyReLU}\left(a^T[Wh_v \| Wh_u]\right) \quad (7)$$

where W is the learnable linear transformation matrix, h_v and h_u are the features of nodes v and u, a is the learnable attention mechanism parameter, and $\|$ represents the

vector concatenation operation. Next, GAT normalizes these using softmax to obtain attention coefficients:

$$\alpha_{vu} = \frac{\exp(e_{vu})}{\sum_{j \in N_v} \exp(e_{vj})} \tag{8}$$

This attention mechanism enables GAT to perform weighted aggregation, assigning greater weights to more important neighbors.

GIN [42]: The GIN (Graph Isomorphism Network) is inspired by graph isomorphism testing theory. It captures subtle differences in graph structures, enabling excellent performance in graph classification and node classification tasks. The GIN's core operation is a special aggregation and update function that combines a node's own features with neighbor features through weighted summation. The GIN update rule is as follows:

$$h_v^{(k)} = \mathrm{MLP}^{(k)}\left((1 + \epsilon^{(k)}) \cdot h_v^{(k-1)} + \sum_{u \in N(v)} h_u^{(k-1)}\right) \tag{9}$$

where $h_v^{(k)}$ is the representation of node v at the k-th layer, $N(v)$ is the neighbor set of node v, $\epsilon^{(k)}$ is a learnable scalar parameter, and $\mathrm{MLP}^{(k)}$ is a multi-layer perceptron. The key to this formula lies in the $(1 + \epsilon^{(k)})$ term, which allows the model to flexibly adjust the importance balance between the central node and its neighbors. The GIN also uses multi-layer structure and skip-connections. Stacking multiple GIN layers enables the model to capture broader structural information. Layer outputs are concatenated to form the final graph representation:

$$h_G = \mathrm{CONCAT}\left(\mathrm{READOUT}^{(1)}(\left\{h_v^{(1)} \middle| v \in G\right\}), \ldots, \mathrm{READOUT}^{(K)}(\left\{h_v^{(K)} \middle| v \in G\right\})\right) \tag{10}$$

The READOUT function can be summation, averaging, or pooling operations. This design enables the GIN to capture multi-scale structural information. These characteristics enable GIN to achieve excellent performance in graph classification and node classification, particularly when fine-grained structural distinction is required.

HAN [43]: The HAN (Heterogeneous Graph Attention Network) is designed for processing heterogeneous graph data. The HAN involves two main steps: meta-path-level aggregation and attention-weighted updates. First, for each meta-path M, the HAN performs aggregation in the corresponding view graph:

$$h_v^{M(k)} = \mathrm{AGGREGATE}^{M(k)}\left(\left\{h_u^{(k-1)} \middle| u \in N_M(v)\right\}\right) \tag{11}$$

$N_M(v)$ is the meta-path-based neighbor set, and $\mathrm{AGGREGATE}^{M(k)}$ is the meta-path-level aggregation function. Next, HAN uses attention mechanisms to weight different meta-path results, obtaining the final node representation:

$$h_v^{(k)} = \sum_M \alpha_M^{(k)} \cdot h_v^{M(k)} \tag{12}$$

$\alpha_M^{(k)}$ is the attention weight calculated for the path M. In path-level attention learning, HAN regulates the influence of each path on node representation by computing attention weights for different paths. The calculation formula for attention weights is as follows:

$$\alpha_M^{(k)} = \frac{\exp(\mathrm{LeakyReLU}(a^T [h_v^{M(k)} \| h_u^{M(k)}]))}{\sum_{M'} \exp(\mathrm{LeakyReLU}(a^T [h_v^{M'(k)} \| h_u^{M'(k)}]))} \tag{13}$$

a is the learnable attention parameter, and $\|$ represents the vector concatenation operation. This multi-level approach enables the HAN to capture broader attribute and relationship information, making it effective for heterogeneous graphs with multiple node types and relationships.

5.2. General Framework

Based on the literature of GNN-based traffic analysis, we summarize the general framework into the following steps. First, raw traffic data are collected from various sources including public datasets, real network traffic, and simulated traffic. Data preprocessing is then conducted, where preprocessing steps are tailored to specific task requirements, typically including data cleaning, segmentation, and normalization. Feature extraction is performed to obtain relevant node and edge features by analyzing the data characteristics. In the graph construction stage, appropriate graph construction strategies are selected according to specific tasks to build different types of graph structures. The graph embedding process vectorizes node and edge features and aggregates them through GNN architectures such as GCN and GraphSAGE, or through specialized aggregation mechanisms designed for specific tasks. After completing graph construction and embedding, classifiers are trained on the processed data to identify anomalous patterns. The general framework is shown in Figure 5.

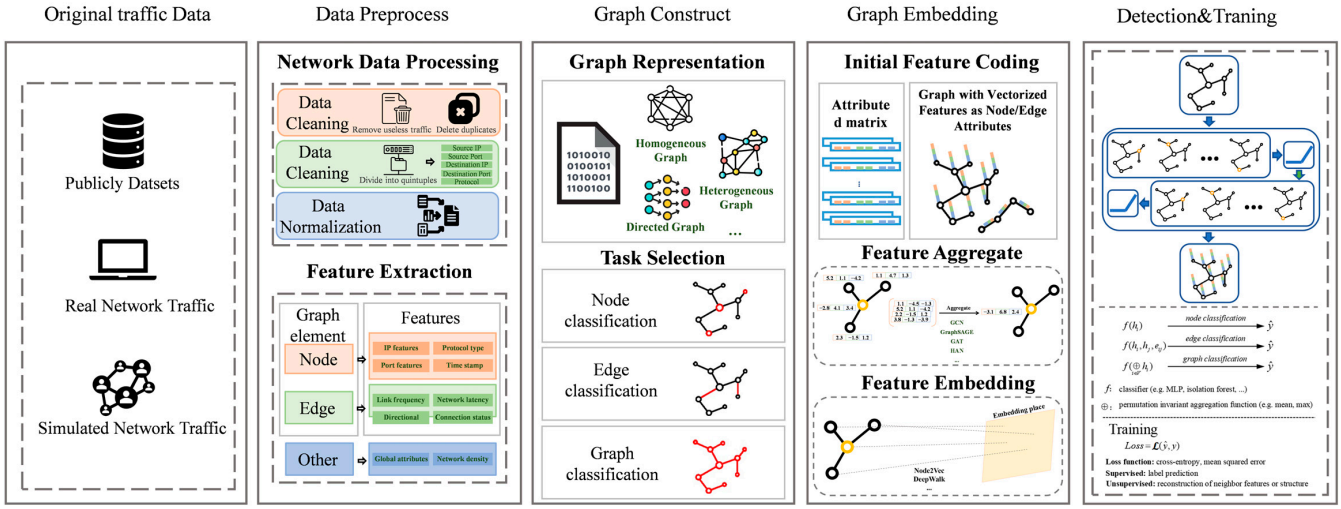

Figure 5. Graph-based Network Traffic Analysis General Framework.

6. GNN-Based Network Traffic Analysis

Section 6 adopts a three-tier taxonomy of node, edge, and graph prediction to systematically review GNN-based traffic analysis methods. This structure reflects fundamental differences in prediction granularity, addressing entity attribute analysis, interaction modeling, and global pattern recognition, respectively. Recognizing that graph construction and feature aggregation strategies often vary across prediction tasks due to differing granularity, we have compiled a list of commonly used graph structures and feature aggregation methods for each task, categorizing them based on their characteristics. These are common methods that have appeared in recent work. Each subsection follows a consistent 'graph construction, feature aggregation, challenge solutions' pipeline to establish comprehensive methodological frameworks. Section 6.4 provides a horizontal comparison of different approaches within the same prediction task across several common and impactful scenarios, directly comparing their performance, resource costs, efficiency, and other trade-offs. The overall distribution of the included studies—by publication year, research field, and publication venue—is summarized in the statistical charts presented in Figure 1 (Introduction Section). To assist readers in quickly locating the structure and content, we have constructed a structural classification tree for Section 6, as shown in Figure 6.

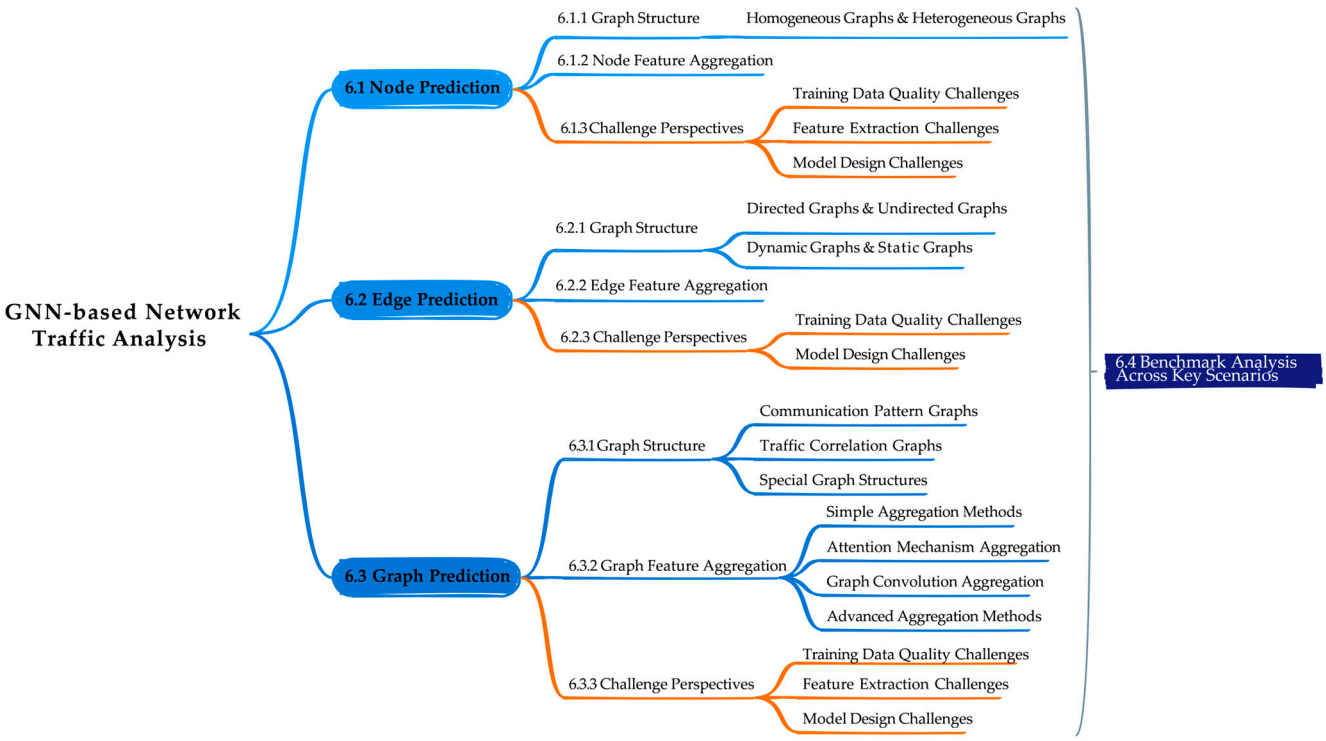

Figure 6. Structural Classification Tree.

In Section 4, we identified three general challenges in the field of network traffic analysis and highlighted the potential of GNNs to address them. The Challenge Perspectives in this section build upon that foundation, bridging the theoretical challenges of Section 4 with the practical solutions offered by GNN-based approaches. Incorporated within each prediction task subsection, these perspectives showcase specific methods that leverage GNNs to tackle these challenges in traffic analysis, offering insights and potential solutions to readers. Notably, in cases where representative traffic analysis solutions for a specific challenge within a particular prediction granularity were not found, they are not included.

6.1. Node Prediction

In network traffic analysis, node prediction is primarily used to analyze the attributes or behaviors of individual network entities. Node prediction is appropriate when the analysis focuses on individual network entities and sufficient node features and neighbor information are available. Typical applications include anomalous node identification, malicious device detection, and device or user classification.

6.1.1. Graph Structure

(1) Homogeneous Graph

Homogeneous graphs represent traffic data using a single node type, suitable for analyzing connections and communication patterns between entities of the same category.

For example, most network traffic analysis tasks (malicious entity identification [44], intrusion detection [45,46], anomaly detection [47]) represent IP addresses as nodes and packet-based interactions as edges. Node features typically include statistical metrics (packet count, byte count, traffic rate), temporal characteristics (activity time, connection duration), protocol information (protocol types, port usage), and behavioral patterns (connection patterns, traffic direction).

The specific graph construction depends on the prediction target and the task requirements. For example, in account labeling tasks [48], to model interactive behaviors between transactions, nodes are set as accounts, edges as transaction relationships between accounts,

and node features include transaction amounts, account balances, transaction counts, and transaction frequency. In content distribution network optimization [49], nodes can be set as servers or cache nodes, edges as data transmission paths, and node features include cache hit rates, response times, and bandwidth usage. In IoT device management [50], nodes can be IoT devices, edges as communication links between devices, and node features may include device types, data transmission frequency, and energy consumption levels. When performing Android application functionality scenario classification, researchers proposed [51] that transitions and dependencies between screens can serve as key information for classification, constructing a directed transition graph to preserve transition information, where nodes are screens, edges are transition behaviors, and useful information extracted from layout files, screenshots, and activity information serves as feature vectors for screens.

(2) Heterogeneous Graph

When network traffic data contain multiple entity types and complex relationships, researchers use heterogeneous graphs to capture diverse behavioral patterns and integrate multi-dimensional information. Meta-paths and meta-graphs are key concepts in heterogeneous graph representation that enhance the analysis of complex network structures. Meta-paths are a series of relationship sequences defined on meta-graphs, while meta-graphs describe different types of nodes in the graph and the possible relationship types between them.

For illegal mobile gambling detection, Gu et al. [52] constructed heterogeneous communication graphs to model gambling behaviors. This graph contains two types of nodes: application nodes and server nodes. Based on communication frequency and data volume information, they constructed meta-paths for four different communication patterns. This approach captures communication intentions between applications and servers using multiple edge types, integrating both server and application perspectives.

For botnet detection, Zhang et al. [53] applied heterogeneous graphs to model interactive behaviors between hosts. In their graph, fine-grained objects in network flows (such as source IP, protocol, and request) are modeled as nodes, with each node containing attribute information, such as session timestamps and user agents. Regarding meta-path formulation, the paper defines 10 symmetric meta paths to capture meaningful semantic relationships in network flows. Additionally, the paper introduces 7 symmetric meta-graphs to model higher-order semantic relationships. By using heterogeneous graphs to characterize the complex interaction patterns and behavioral characteristics of botnets, this work surpasses previous methods in botnet detection tasks.

For malicious domain detection, Simon et al. [54] used heterogeneous graphs where nodes represent network entities (domain names, IP addresses) and edges represent interactions (DNS resolution, WHOIS information sharing).

6.1.2. Node Feature Aggregation

After graph construction, GNNs learn node representations by aggregating the features of nodes and their neighbors.

Early GNN-based traffic analysis research primarily employed classical node aggregation algorithms. For example, Chowdhury et al. [55] utilized the classical self-organizing map algorithm to achieve adaptive detection of different types of bot behaviors in bot detection tasks through unsupervised learning without requiring labeled data.

Recent research has introduced more sophisticated feature aggregation methods for GNN-based traffic analysis. Hong et al. [56] proposed a method called MalDiscovery for detecting encrypted malicious traffic. MalDiscovery represents encrypted sessions as nodes and employs GraphSAGE to generate node embeddings through random neighbor sampling and feature aggregation. This approach exploits correlations between encrypted traffic sam-

ples to improve detection accuracy and efficiency. Zhang et al. [53] utilized GCN for node feature aggregation in botnet detection tasks. This method models network flow objects as multi-attribute heterogeneous information networks (AHIN). It defines semantic relationships between nodes using meta-paths and meta-graphs, then generates weighted adjacency matrices. Subsequently, GCN operations are performed on these adjacency matrices. This method simultaneously utilizes node attributes and topological information while capturing high-order semantic relationships through neighborhood aggregation. Compared with the method mentioned earlier by Chowdhury et al. [55], Zhang et al.'s [53] method achieved higher accuracy on the same task.

Researchers have also developed multi-level structures and hybrid models for complex traffic analysis scenarios. For detecting real money trading in online games, Tao et al. [57] proposed a Multi-View Attention Network (MVAN). This approach combines three attention mechanisms: Multi-Graph Attention Network (MGAT) for social relationships, Behavior Attention Network (BAN) for character behaviors, and Profile Attention Network (PAN) for attribute features. This method captures multi-dimensional characteristics of trading activities and outperforms traditional methods and other GNN approaches. Presekal et al. [49] proposed a hybrid deep learning-based node aggregation method for attack detection in cyber-physical power systems. This approach combines graph convolutional networks to capture spatial relationships between substations with long short-term memory networks to model temporal dependencies. By utilizing spatiotemporal features, this method improves both the performance and reliability of cyber-physical system attack detection.

6.1.3. Challenge Perspectives

(1) Training Data Quality Challenges

Zhang et al. [58] proposed a GNN-based framework for industrial IoT intrusion detection with class-imbalanced samples. The framework designs a network constructor that utilizes multi-head weighted cosine similarity and network reconstruction techniques to optimize graph structure and capture semantic relationships between data. Meanwhile, by optimizing network structure and prediction task loss, it better utilizes limited training data.

Deng et al. [59] proposed a Flow Topology-based Graph Convolutional Network (FT-GCN) for intrusion detection in IoT networks with limited labels. This method constructs an Interval-Constrained Traffic Graph (ICTG) to describe traffic flow topology and then designs a Node-Level Spatial Attention mechanism (NLS) to enhance key statistical features. Finally, it employs a Topology-Adaptive Graph Convolutional Network (TAGCN) to learn combined representations of statistical features and topological structure, achieving effective intrusion detection with limited labels.

Zhang et al. [60] proposed a model called Heterogeneous Graph Node Reweighting (HGNR) for detecting piracy video websites with imbalanced data. HGNR captures complex relationships between websites by constructing heterogeneous graphs containing multiple meta-relationships and uses a node reweighting mechanism to address node imbalance problems.

(2) Feature Extraction Challenges

Liu et al. [61] encountered the problem of high-dimensional node features that could not be effectively aggregated during node prediction when performing account classification in Ethereum transaction networks, due to the complex structure of the Ethereum network. Therefore, Liu et al. [61] proposed the FA-GNN model to address these feature overload problems through neighbor filtering and feature enhancement. Neighbor filtering helps screen out important relevant information and reduce noise interference, while feature enhancement supplements and enriches node features by utilizing high-order neighbor information, thereby improving classification accuracy in complex network environments.

Similarly, Lin et al. [62] found that simply aggregating features from neighboring accounts might introduce noise from different accounts when performing Ethereum transaction account classification, and similar types of accounts might be indirectly connected through multiple hops. Therefore, the authors used one-hop and multi-hop aggregators for feature aggregation, where the one-hop aggregator collects information from direct neighbors, and the multi-hop aggregator utilizes an importance-based sampling strategy to sample and aggregate information from more distant neighbors. By balancing local transaction patterns with a broader network context, account classification performance was significantly improved. However, these two methods were evaluated on different datasets, which limits direct performance comparisons.

Xie et al. [63] proposed a mobile application recommendation method based on GNN and multi-view learning (GVARec) to address the feature overload problem in mobile application recommendations. This method first constructs user and application similarity matrices and selectively fuses feature information from neighboring nodes. Finally, it selectively aggregates multi-view prediction scores through an attention mechanism. This method effectively solves the feature overload problem by fully utilizing rich semantic information and structural information from heterogeneous graphs.

Dou et al. [64] proposed the CARE-GNN model to address camouflaged fraudster detection through enhanced node feature processing. Specifically, CARE-GNN adopts a label-based similarity measurement method to identify feature-rich neighboring nodes. This approach selects neighbors that are most helpful for node classification by comparing the similarity of node features, thereby effectively addressing feature camouflage problems.

(3) Model Design Challenges

Due to the constantly changing internet environment, models need to possess the capability to handle dynamic networks. Therefore, Hei et al. [65] proposed an incremental learning model called MSGAT++, which effectively addresses the problem of dynamic application processing in Android malware detection. This model is specifically designed to handle out-of-sample nodes and can quickly generate embeddings for new applications without rebuilding the entire heterogeneous information network (HIN) and embedding model. This incremental design significantly improves training efficiency and model scalability.

We summarize the representative methods mentioned in this section in Table 3.

Table 3. Representative articles on node prediction based on GNNs.

Paper	Model	Year	Task	Graph Type	Modules	Learning
[55]	-	2017	Botnet detection	Homogeneous graph	SOM	Unsupervised
[63]	GVARec	2020	Mobile application recommendation	Heterogeneous graph	GCN, GAT	Supervised
[47]	-	2022	IoT network intrusion detection	Homogeneous graph	GCN, GIN	Supervised
[59]	FT-GCN	2023	Intrusion detection	Homogeneous graph	TAGCN	Supervised
[49]	CyResGrid	2023	Anomaly detection	Homogeneous graph	GCN, LSTM	Supervised
[56]	MalDiscovery	2023	Encrypted malicious traffic detection	Homogeneous graph	GraphSAGE	Supervised
[45]	FTG-Net-E	2024	DDoS attack detection	Homogeneous graph	GCN	Supervised
[62]	KYC-GCN	2024	Account labeling	Heterogeneous graph	GCN	Supervised
[52]	-	2024	Illegal mobile gambling apps	Heterogeneous graph	HAN	Supervised
[46]	GHGDroid	2024	Android malware detection	Homogeneous graph	GCN	Supervised
[65]	MSGAT++	2024	Rapid android malware detection	Heterogeneous graph	HIN, GAT	Supervised

6.2. Edge Prediction

Edge prediction in network traffic analysis primarily focuses on relationships or interactions between network entities. Edge prediction is suitable when research emphasizes connection patterns between entities and sufficient edge features and node information are available. However, in practical network traffic analysis applications, edge-level prediction is used less frequently than node-level prediction. First, edge-level prediction typically requires detailed edge attribute data, which may be difficult to obtain or annotate in network traffic analysis. For instance, specific link features or labels are often less accessible than node-level data. Second, edge-level prediction involves more complex computations

and higher resource consumption, especially in large-scale networks, where processing numerous edge features and relationships significantly increases model complexity and training time. With advancing GNN technology and expanding application scenarios, edge-level prediction research is expected to grow, particularly for applications requiring detailed relationship analysis, such as network security and traffic optimization.

6.2.1. Graph Structure

Unlike node prediction, edge prediction requires defining nodes and edges along with their features, while paying special attention to edge directionality and graph dynamics. Edge directionality may affect the nature of relationships and prediction results [66]; for example, in communication networks, the direction of data flow is a key feature. Additionally, the time dimension in edge prediction may directly affect the existence of edges, as certain connections may only exist during specific time periods [67].

In edge-level prediction, node features are often simplified to basic identifiers (such as IP addresses, port numbers) [68,69] or constant vectors [70–72] to highlight edge feature information and reduce computational complexity. Edge features include network traffic statistics [73–75], traffic interaction patterns [70], packet features [71], and communication link information [72].

(1) Directed Graph vs. Undirected Graph

Many tasks use directed graphs to more accurately capture interactions with strong directionality, sequentiality, or obvious causal relationships. For example, Yang et al. [73] utilized directed graphs to represent the directionality of HTTP requests, better characterizing the complex relationships in network traffic. Similarly, Fu et al. [70] and Nguyen et al. [76] used directed graphs to capture traffic interaction dynamics and directionality, improving detection performance and real-time update capabilities. In the field of network intrusion detection, Caville et al. [74] and Xu et al. [68] adopted directed graphs to show communication directions between hosts, thereby improving the ability to identify intrusions and anomalies. Altaf et al. [77] utilized directed graphs to capture communication flows, enhancing detection capability for sophisticated network attacks.

In contrast, undirected graphs are suitable for interactions without clear directionality or when simplifying model design to reduce computational complexity. They emphasize peer-to-peer relationships between devices. For example, Duan et al. [78] used dynamic undirected graphs, highlighting the importance of interaction patterns between devices. Song et al. [75] used undirected bipartite graph structures to flexibly adapt to the dynamics of e-commerce platforms, achieving fraud detection through subgraph analysis.

(2) Dynamic Graph vs. Static Graph

Network traffic is inherently dynamic, with connection establishment, disconnection, and traffic fluctuations that require real-time capture and analysis. Therefore, in edge-level prediction, researchers often use dynamic graphs to model these changes. For example, Fu et al. [70] and Nguyen et al. [76] utilized dynamic graphs to reflect real-time traffic pattern changes, improving system adaptability to newly emerging interactions. Caville et al. [74] relied on dynamic directed graphs to update and evaluate changes in network activities, thereby supporting self-supervised anomaly detection.

For stable network structures and relationships, researchers use static graphs to capture persistent patterns. Gu et al. [79] adopted static graphs to clarify topological relationships between network entities, which is particularly helpful for identifying global attack patterns. Xu et al. [68] used static graphs to capture long-term connections and communication characteristics between IoT devices, enhancing the detection and analysis capabilities for intrusions.

6.2.2. Edge Feature Aggregation

Researchers typically utilize classic GNN models or design their variants for edge feature aggregation. For example, Caville et al. [74] proposed the Anomal-E method, which combines Edge-GraphSAGE with Deep Graph Infomax (DGI) to achieve self-supervised learning, generating meaningful edge embeddings without requiring labeled data. Duan et al. [78] extended Edge-GraphSAGE to dynamic graphs by introducing a temporal dimension. Their method simultaneously considers spatial and temporal features of network traffic, improving intrusion detection performance. Song et al. [75] designed the R-GIN model for edge feature aggregation to address collective fraud detection on e-commerce platforms. When processing heterogeneous graphs, R-GIN considers edge types (normal and fraud) to be important features. In subgraphs, R-GIN aggregates information from neighboring nodes while combining edge type information to update node representations. Through this approach, R-GIN can effectively distinguish between normal and fraudulent behavior in edge classification tasks.

In addition to creating variants of classic graph models, researchers use model fusion and feature preprocessing approaches to enhance the effectiveness of edge feature aggregation. Altaf et al. [77] proposed a novel GNN framework that fuses spectral convolution and spatial convolution methods for edge feature aggregation, aiming to learn spectral and spatial features in graph structures. This method handles multi-edge and multi-dimensional edge features through a three-layer spectral–spatial–spectral structure, improving performance in network intrusion detection systems. Fu et al. [70] proposed an edge pre-clustering strategy that utilizes the sparsity of edge distribution in the graph structural feature space, employing the DBSCAN algorithm for density clustering. By extracting and normalizing key graph structural features, such as vertex in-degree and out-degree, this method can effectively distinguish between malicious and benign traffic.

6.2.3. Challenge Perspectives

(1) Training Data Quality Challenges

Xu et al. [68] proposed the EE-GCN model to address intrusion detection problems under insufficient sample conditions in Industrial Internet of Things (IIoT). They constructed a two-layer GCN network to extract edge features. The first layer GCN updates node features by aggregating information from direct neighbors through matrix operations. The second layer GCN further aggregates the updated node features, capturing broader neighbor information and thereby extracting deeper-level edge features. This addresses the challenge where insufficient training samples in IIoT environments lead to poor model performance, as existing GNN-based methods fail to fully utilize limited data information.

(2) Model Design Challenges

Baahmed et al. [71] conducted research on the “black box” properties of GNNs in network intrusion detection tasks. They adapted the existing GNNExplainer method to explain the decision-making process of GNNs in their specific application. Specifically, they adapted GNNExplainer for edge classification tasks to apply feature masks and edge masks on edges, thereby identifying the most important features and edges for predictions. Through this approach, they can better understand the topological relationships and features that GNN models utilize when detecting network intrusions, thus improving the model’s interpretability.

The references related to edge prediction mentioned in this subsection are summarized in Table 4.

Table 4. Articles on edge prediction based on GNNs.

Paper	Model	Year	Task	Graph Type		Modules	Learning
[75]	R-GIN	2021	Fraud detection	Undirected	Dynamic	GIN	Supervised
[73]	WTAGRAPH	2022	Web tracking detection	Directed	Static	GNN	Supervised
[74]	Anomal-E	2022	Intrusion detection	Directed	Dynamic	Edge-GraphSAGE	Self-supervised
[76]	TS-IDS	2023	IoT intrusion detection	Directed	Dynamic	GAT	Self-supervised
[68]	EE-GCN	2023	IoT intrusion detection	Directed	Static	GCN	Supervised
[70]	HyperVision	2023	Malicious traffic detecting	Directed	Dynamic	GNN	Unsupervised
[71]	-	2023	Intrusion detection	Directed	Static	Edge-GraphSAGE	Supervised
[78]	-	2023	Intrusion detection	Undirected	Dynamic	GNN	Semi-supervised
[77]	-	2023	IoT intrusion detection	Directed	Static	GCN, GAT	Supervised
[69]	-	2024	Intrusion detection	Undirected	Static	GAT	Self-supervised
[79]	-	2024	Intrusion detection	Directed	Static	Edge-GraphSAGE	Self-supervised
[72]	CTGNN	2024	Intrusion detection	Undirected	Dynamic	GNN	Supervised

6.3. Graph Prediction

Graph prediction aims to utilize GNNs for predicting and analyzing the entire network traffic graph structure or its global properties. This type of task focuses not only on individual nodes or edges, but on the behavior, patterns, or states of the entire network.

6.3.1. Graph Structure

In graph-level prediction, researchers often model traffic as a communication pattern graph, a traffic correlation graph, and a special graph structure according to the different characteristics of the network traffic to be analyzed [80,81].

(1) Communication Pattern Graph

Communication pattern graphs model direct interaction relationships between network entities by representing actual communication flows, temporal sequences, and behavioral patterns among network components.

The most common method is to use flows as graph nodes. For example, Shen et al. [82] in intrusion detection tasks used flows as nodes with edges representing temporal relationships between flows, effectively capturing attack sequence patterns. Chen et al. [83] also employed flows as nodes in DDoS attack detection tasks. However, their edges represented both temporal relationships and transmission direction information, enhancing attack traffic identification capabilities. In malicious software traffic identification tasks, Han et al. [84] further enriched node attributes to include features such as flow size and direction, thereby improving the model's sensitivity to malicious traffic.

Beyond standard flow-based approaches, some research has constructed task-specific communication pattern graphs to address domain-specific requirements. Gao et al. [85] used HTTP request content types as nodes in website fingerprinting tasks, employing multi-level edge structures (packet, traffic, and host edges) to improve website identification accuracy. Xu et al. [86] constructed a bipartite graph for mobile application encrypted traffic classification using IP addresses and application port numbers as distinct node types to capture application network behavior. Feng et al. [87] developed graph structures based on program execution flow for PHP Webshell detection, converting PHP scripts into Inter-procedural Control Flow Graphs, with nodes representing PHP statements and edges representing control flow relationships. Kisanga et al. [88] used various network entities (such as accounts) as nodes in network anomaly detection to construct comprehensive network behavior graphs for detecting potential anomalous activities.

Special graph structures further enrich communication pattern graph construction. Li et al. [89] modeled network devices and communication flows as heterogeneous graphs, where nodes represent different device types and edges represent network communications, introducing Express Edges to improve large-scale graph analysis efficiency. Cui et al. [90] addressed IPv6 user activity correlation attacks by constructing heterogeneous graphs for each IPv6 client address, where nodes include clients, servers, and

fingerprint nodes, and edges represent communication relationships, utilizing multi-type semantic metadata to accurately describe user activities.

(2) Traffic Correlation Graph

Traffic correlation graphs capture statistical correlations and dependencies between different network traffic features, focusing on data characteristics rather than direct communication interactions. We categorize the related work on traffic correlation graph construction methods into five main categories: using byte values as nodes, using data packets as nodes, using network addresses or flows as nodes, using events or messages as nodes, and methods that comprehensively consider traffic flow directions.

Methods using byte values as nodes can capture the statistical characteristics and correlations of data at a highly abstract level. For example, Zhang [91] and Hu et al. [92] used byte values as nodes in encrypted traffic classification tasks. They created edges by calculating Pointwise Mutual Information (PMI), establishing connections only between byte pairs with positive PMI values. This method demonstrates strong performance in tasks such as identifying overall data patterns, detecting anomalies, and performing correlation analysis.

Methods using data packets as nodes focus more on capturing content and structural information. In encrypted network traffic classification tasks, Okonkwo et al. [93] modeled the first ten data packets of each session as a ten-node graph. Edges bidirectionally connect adjacent preceding and following nodes. Huoh et al. [94] used raw byte data as node attributes, with edges representing temporal relationships and time intervals between data packets. They also introduced traffic meta-features as graph-level attributes. In malicious traffic classification tasks, Yang et al. [95] proposed the Malicious Traffic Interaction Graph (MTIG). This approach groups nodes based on data packet burst information and adds undirected edges within and between groups, effectively capturing features and structures within individual traffic flows.

Methods using network addresses or flows as nodes focus on analyzing communication interaction correlations. Fu et al. [96] used network addresses as nodes in unknown encrypted malicious traffic detection tasks. Edges represent communication interactions between addresses, enabling detection of potential malicious activities through traffic communication correlation learning. Zhao et al. [97] in anonymous network traffic identification tasks proposed two complementary graph structures: the Attribute Relation Graph (ARG) and the Temporal Relation Graph (TRG). ARG establishes edge connections through key flow attributes, capturing request-response patterns in network sessions. TRG focuses on temporal relationships between flows, calculating edge weights using Gaussian kernel functions based on the assumption that temporally close flows may belong to the same network activity.

Methods using events or messages as nodes focus on capturing correlations between attributes and message substructures. Wang et al. [98] used attribute values of behavioral events as graph nodes in behavioral event analysis and anomaly detection tasks. Edges represent co-occurrence relationships between attribute values, effectively capturing statistical associations between various behavioral event attributes. Zhang et al. [99] constructed directed attributed graphs from CAN message streams within given message intervals for the task of rapid anomaly detection in controller area networks (CAN), capturing substructural features of CAN messages using graph-level algorithms, and implementing anomaly detection and attack classification via a two-stage cascaded classifier.

Additionally, Zhu et al. [100] proposed a method that comprehensively considers flow directions, providing new insights for tasks such as darknet application classification. Their proposed Darknet Traffic Graph generates forward and reverse nodes for each flow. Node features include packet count, size, and arrival time intervals, with positive and negative values distinguishing flow directions. Nodes of the same type are placed in the

same layer and linked. Nodes between different layers are connected through a Cartesian product-like approach, preserving traffic direction information along with temporal and feature information.

(3) Special Graph Structure

Besides communication pattern graphs and traffic correlation graphs, researchers have also explored novel graph structures to address specific tasks.

For example, Pham et al. [101] constructed hybrid graphs that combine traffic correlation graphs with features from communication pattern graphs. This approach uses combinations of IP addresses and port numbers as nodes, with traffic interactions as edges. It focuses on modeling traffic statistical characteristics and correlations between network entities. Meanwhile, it also preserves some communication pattern elements such as temporal information, bidirectional traffic distinction, and protocol information. In Android malware detection and webpage classification tasks, researchers use natural language processing and semantic information to construct graph structures for predictions. Pei et al. [102] used NLP techniques to extract word-level, character-level, and lexical-level features of Android applications. They combined these with static features such as permissions, components, and API calls, constructing semantic relationship graphs for malware detection. Guo et al. [103] parsed webpage text into graphical representations for webpage classification. Nodes contain text, visual, and structural attributes, while edges are based on visual associations and DOM hierarchical relationships, constructing semantic relationship graphs.

6.3.2. Graph Feature Aggregation

Based on the development trajectory of GNNs and current academic consensus [104,105], graph feature aggregation methods can be categorized into the following four types: simple aggregation methods, attention mechanism aggregation, graph convolution aggregation, and advanced aggregation methods. This subsection reviews recent work on graph feature aggregation strategies based on this classification.

(1) Simple Aggregation

In network traffic analysis, graph-based prediction approaches using simple aggregation methods primarily employ average aggregation and its variants. Li et al. [106] used classic average aggregation methods to process vertex and neighbor features. This method is insensitive to neighbor ordering, suitable for non-Euclidean data structures, and effectively learns graph information through iterative updating of feature matrices. Busch et al. [107] proposed the NF-GNN model which adopts a more complex variant of average aggregation. This approach separately performs average aggregation on incoming and outgoing edge features, then concatenates them and obtains node representations through learnable transformations. This method can more effectively handle edge attributes and bidirectional traffic information in network flow graphs. Average aggregation methods demonstrate strong performance in handling non-sequential features.

(2) Attention Mechanism Aggregation

Wang et al. [98] proposed the BIG model which, adopts a variant of basic attention aggregation. This model learns attention weights to aggregate node information and generate event representations, effectively capturing complex behavioral patterns. Cui et al. [90] adopted a three-layer attention mechanism to aggregate heterogeneous graph information, specifically: node-level attention to capture meta-path specific neighbor information, semantic-level attention to fuse semantics from different meta-paths, and graph-level attention to aggregate global node information. By employing heterogeneous graph representation and hierarchical attention aggregation methods, this approach can efficiently learn correlations between IPv6 addresses.

(3) Graph Convolution Aggregation

In graph-based network traffic analysis, graph convolution aggregation is widely applied.

For encrypted traffic analysis tasks, Chen et al. [83] proposed an Ensemble Graph Neural Network (EGNN) architecture combining three models: the GCN, Gated Graph Neural Network (GGNN), and Capsule Graph Neural Network (CapsGNN). The GCN extracts features between adjacent nodes through a two-layer structure, the GGNN uses a GRU-like structure for message passing, and the CapsGNN uses dynamic routing algorithms for information processing. Voting and stacking mechanisms integrate results to improve classification accuracy and robustness. Zhang et al. [92] proposed improved graph feature aggregation methods based on GraphSAGE, performing average aggregation on neighbor node messages and concatenating them with node features to form updated representations. They introduced skip connection aggregation strategies to address over-smoothing problems and capture multi-scale information by stacking GraphSAGE up to 4 layers. Pham et al. [101] adopted the Deep Graph Convolutional Neural Network (DGCNN) for efficient application program classification, using 4 graph convolution layers to extract node representations, processing graphs through a SortPooling layer, and applying 1D convolution, MaxPooling, and fully connected layers for final classification. In anomaly detection tasks, Kisanga et al. [88] constructed a supervised GCN model that includes an input layer, 16 hidden layers, and an output layer. In the feature aggregation process, the input layer receives node features and embeddings, propagating and aggregating neighbor node information through graph convolution operations in the hidden layers.

In website fingerprinting tasks, Gao et al. [85] proposed using a customized GCN structure to handle heterogeneous graphs. This method first converts website browsing traffic into heterogeneous spatiotemporal graph representations, then aggregates and updates edge features and node features separately through multi-layer convolution structures. By adopting the message passing paradigm and multi-layer structure, this method can effectively capture complex relationships in graphs. Finally, it predicts the category of heterogeneous spatiotemporal graphs by processing and integrating the output from each layer.

(4) Advanced Aggregation

Advanced aggregation methods are typically used in deep GNNs to address issues such as information propagation, over-smoothing, and gradient vanishing. This approach implements aggregation using Equation (14).

$$h_G = \mathrm{CONCAT}(\mathrm{READOUT}(\left\{h_v^{(k)} \middle| v \in G\right\}) \middle| k = 0, 1, \ldots, K) \tag{14}$$

For example, Shen et al. [82] proposed using advanced aggregation methods to accurately identify distributed applications through encrypted traffic analysis. The specific steps include using multi-layer perceptron to update node features, performing READOUT operations on each layer, and finally concatenating the results of all layers. The main advantages of this method lie in its ability to capture graph structure information at different scales and adaptively select feature combinations. Compared to other methods, it preserves more structural information and is more flexible in utilizing multi-layer information.

6.3.3. Challenge Perspectives

(1) Training Data Quality Challenges

Yu et al. [108] encountered data imbalance problems in encrypted malicious traffic identification tasks, where malicious traffic constitutes an extremely small fraction of the total traffic. To address this issue, they modeled TLS sessions as state transition graphs and added statistical information to form attribute graphs as session fingerprints. By adopting the shortest path graph kernel method, they effectively measured the similarity between attribute graphs generated by TLS sessions. This approach maps data points to Hilbert

space for classification. This approach not only comprehensively captures the sequential and statistical features of encrypted traffic, but also demonstrates superior performance and robustness on extremely imbalanced datasets.

(2) Feature Extraction Challenges

To improve the effectiveness of feature extraction in anonymous network traffic identification tasks, Zhao et al. [97] adopted several innovative approaches. First, they selected flow sequences as input, thereby preserving the spatiotemporal correlations of traffic data. Second, by defining and utilizing attribute relationships and temporal relationships between flows, they further enhanced the expressive power of features. Based on these relationships, they proposed a method based on the Residual Graph Convolutional Network (ResGCN). This method represents flow sequences as graph structures for information exchange and feature extraction. Their experimental results demonstrated that this method extracted more discriminative and effective features compared to baseline approaches.

(3) Model Design Challenges

To address the challenge of model adaptability for unknown data, Fu et al. [96] proposed the HyperVision system for unknown malicious traffic detection tasks. They constructed a flow interaction graph to represent the interaction patterns of network traffic. Through aggregation of short traffic flows and recording of long traffic feature distributions, they effectively reduced graph complexity while preserving key information. HyperVision adopts unsupervised graph learning methods and achieves detection of abnormal interaction patterns through steps including graph preprocessing, key vertex identification, and edge clustering. This approach does not rely on prior knowledge of known attacks or labeled datasets, and can effectively identify encrypted malicious traffic with unknown patterns.

Similarly, Qin et al. [109] captured network interaction patterns by constructing Host Interaction Graphs (HIGs) and integrated them with traditional traffic features to form comprehensive network behavior representations. They further used unsupervised clustering methods to establish normal behavior models, then detected anomalies based on distance metrics. This approach combines an inductive learning framework, enabling the system to effectively capture complex network behavior patterns without labeled data and enhancing the detection capability for unknown attacks.

Wang et al. [110] proposed the TGPrint method, which uses a graph prediction approach to solve the problem of unknown encrypted traffic attack fingerprint classification. This method converts preprocessed traffic into attack graphs and uses graph convolutional attention networks to extract key attack behavior patterns from these graphs, generating attack fingerprints. This graph-based approach can capture the essential characteristics and high-level behavior patterns of attacks, rather than being limited to features of specific attack vectors, thus demonstrating strong generalization capability when facing unknown attacks.

This section summarizes recent graph prediction work in the field of traffic analysis, with representative methods summarized in Table 5.

Table 5. Representative articles on graph prediction based on GNNs.

Paper	Model	Year	Task	Graph Type		Modules	Learning
[90]	SIAMHAN	2020	IPv6 address correlation attacks	Undirected	Static	HAN	Supervised
[107]	NF-GNN	2021	Malware detection	Directed	Dynamic	GNN	Supervised
[109]	XNBAD	2022	Anomaly detection	Undirected	Dynamic	GraphSAGE	Supervised
[98]	-	2022	Anomaly detection	Undirected	Static	GAT	Supervised
[86]	TrafficGCN	2022	Encrypted traffic classification	Undirected	Static	GCN	Supervised
[88]	-	2023	Anomaly detection	Directed	Static	GCN	Supervised
[83]	WFF-EGNN	2023	Encrypted traffic classification	Directed	Static	GCN	Supervised
[91]	TFE-GNN	2023	Encrypted traffic classification	Undirected	Static	GraphSAGE	Supervised
[92]	TCGNN	2023	Network traffic classification	Undirected	Static	GCN	Supervised
[93]	-	2023	Network traffic classification	Directed	Static	GraphConv	Supervised
[106]	IBGC	2024	Encrypted traffic classification	Directed	Dynamic	GraphSAGE	Supervised
[85]	RKHSTGCN	2024	Website fingerprinting	Undirected	Dynamic	HAN	Supervised
[84]	-	2024	Intrusion detection	Undirected	Dynamic	GCN, GIN, GAT	Supervised

6.4. Benchmark Analysis Across Key Scenarios

To elucidate the practical trade-offs among GNN-based approaches, this section conducts a systematic comparison across three pivotal network security scenarios: malicious traffic detection (CTU-13 dataset), intrusion detection (UNSW-NB15 dataset), and IoT intrusion detection (ToN-IoT dataset). The selection criteria emphasize benchmark datasets with multiple published results to ensure methodological comparability.

Case Study 1: Malicious Traffic Detection (CTU-13 Dataset)

As shown in Table 6, While all three methods achieve 99% accuracy on CTU-13 dataset, Hong et al.'s multi-view GraphSAGE [56] demonstrates marginal superiority in detection precision. Zhang et al.'s heterogeneous graph approach [53] offers enhanced semantic modeling capabilities, albeit with a 0.63% accuracy trade-off relative to [56], suggesting a computational cost–benefit consideration for practitioners.

Table 6. Malicious Traffic Detection Methods Comparative Analysis.

Method	Graph Type	Model	Accuracy	F1-Score	Key Advantages
Chowdhury et al. [55]	Directed Graph	Basic GNN	99%	-	Efficient search (0.1% nodes)
Hong et al. [56]	Attributed Graph	GraphSAGE	99.9%	-	Multi-view feature fusion
Zhang et al. [53]	Heterogeneous Graph	HGCN	99.27%	93.26%	High-order semantic modeling

Case Study 2: Intrusion Detection (UNSW-NB15 Dataset)

As shown in Table 7, Deng et al.'s edge-level prediction [59] achieves 88.45% F1-score through self-supervised learning, demonstrating particular efficacy in label-scarce environments. This contrasts with Caville et al.'s node-level TAGCN [74], which attains 98.7% accuracy but requires full supervision, highlighting a critical trade-off between annotation requirements and performance.

Table 7. Intrusion Detection Methods Comparative Analysis.

Method	Prediction Level	Model	Accuracy	F1-Score	Training Paradigm
Deng et al. [59]	Edge	Edge-GraphSAGE	98.18%	88.45%	Self-supervised
Caville et al. [74]	Node	TAGCN	98.7%	-	Supervised
Han et al. [84]	Graph	GIN	98.42%	-	Supervised

Case Study 3: Intrusion Detection (ToN-IoT Dataset)

Table 8 uncovers divergent optimization paths for IoT environments: Duan et al.'s CTGNN [78] achieves near-perfect detection (99.98%) through comprehensive graph processing, whereas Gu et al.'s DLGNN [79] prioritizes computational efficiency with minimal performance degradation (0.18% accuracy difference). This dichotomy illustrates the growing specialization of GNN architectures for resource-constrained deployments.

Table 8. IoT Intrusion Detection Methods Comparative Analysis.

Method	Graph Type	Model	Accuracy	F1-Score	Specialization
Duan et al. [78]	Attributed Graph	CTGNN	99.98%	99.96%	New behavior detection
Gu et al. [79]	Line Graph	DLGNN	99.8%	99.85%	Label-efficient
Li et al. [89]	Heterogeneous Graph	RGCN	-	97.78%	Multi-graph fusion

The synthesized findings demonstrate several key trends in GNN-based security analytics. Node-level methods consistently deliver superior accuracy (98.7 to 99.98%) but necessitate extensive labeled data, making them suitable for well-instrumented networks. Edge-level approaches present a viable alternative for operational environments with limited ground truth, as evidenced by the 88.45% F1-score achieved through self-supervision in [59]. Heterogeneous graph modeling emerges as particularly effective for complex threat scenarios, with multi-view [56] and high-order semantic [53] approaches showing consistent advantages, albeit at increased computational overhead. The temporal progression of results (2017–2024) further indicates steady performance improvements against evolving attack patterns, with recent methods exhibiting enhanced robustness to novel threats as seen in [78,79].

7. Public Datasets and Open-Source Code

7.1. Public Datasets

Table 9 presents the datasets used for different tasks in the surveyed literature, including both publicly available datasets and datasets created by researchers, to facilitate future research and enable reproducible experiments.

Table 9. Public Dataset Summary.

Dataset	Category	Task	Link	Reference	Related Work
MCFP	Public	Malicious Traffic Detection	https://mcfp.felk.cvut.cz/publicDatasets, accessed on 22 October 2025.	-	[56]
JusticePC	Self-made	Malicious Account Detection	https://github.com/fuxiAIlab/JusticePC, accessed on 22 October 2025.	[48]	[48]
Pirated Video Websites Dataset	Self-made	Pirated Video Website Detection	https://github.com/lirenjieArthur/pirated-video-websites, accessed on 22 October 2025.	[51]	[51]
MAWI	Public	Detecting Encrypted Malicious Traffic	http://mawi.wide.ad.jp/mawi, accessed on 22 October 2025.	[62]	[62]
Malicious Traffic Encryption Dataset	Self-made	Malicious Traffic Encryption Detection	https://github.com/fuchuanpu/HyperVision, accessed on 22 October 2025.	[63]	[63]
Yelp	Public	E-commerce Fraud Detection	http://odds.cs.stonybrook.edu/yelpchi-dataset, accessed on 20 June 2025.	[66]	[66,75]
Amazon	Public	E-commerce Fraud Detection	http://jmcauley.ucsd.edu/data/amazon, accessed on 22 October 2025.	[66]	[66]
CERNET-2022-Service	Self-made	Encrypted Traffic Classification	https://data.iptas.edu.cn/web/tbps, accessed on 22 October 2025.	[77]	[77]
Mobile Application Traffic Dataset	Self-made	Encrypted network traffic	https://soeai.github.io/MAppGraph, accessed on 20 June 2025.	[88]	[88]
CTU-13	Public	Bot Detection	https://www.stratosphereips.org/datasets-ctu13, accessed on 22 October 2025.	[111]	[53,55,56]
CAN Signal Extraction and Translation	Public	CAN Intrusion Detection	https://ocslab.hksecurity.net/Datasets/can-signal-extraction-and-translation-dataset, accessed on 22 October 2025.	[112]	[77]
DREBIN	Public	Android Malware	https://www.sec.cs.tu-bs.de/people/arp/drebin, accessed on 22 October 2025.	[113]	[46]
CICAndMal2017	Public	Malware Detection	https://www.cic.gc.ca, accessed on 22 October 2025.	[114]	[65]
Praguard	Public	Malware Traffic	https://github.com/Praguard, accessed on 20 June 2025.	[115]	[107]
SGCC	Public	Anomaly Detection in Industrial IoT	https://github.com/henryRDlab/ElectricityTheftDetection, accessed on 22 October 2025.	[116]	[58]
BoT-IoT	Public	Intrusion Detection	https://ieee-dataport.org/documents/bot-iot-dataset, accessed on 22 October 2025.	[117]	[68,76,78]
ToN-IoT	Public	Intrusion Detection	https://research.unsw.edu.au/projects/toniot-dataset, accessed on 22 October 2025.	[117]	[68,72,78,79,89]
CSE-CIC-IDS2018	Public	Intrusion Detection	https://www.unb.ca/cic/datasets/ids-2018.html, accessed on 22 October 2025.	[117]	[66,76,110]
UNSW-NB15	Public	Intrusion Detection	https://www.unsw.edu.au/engineering/our-story/our-research/cyber-security-research/unsw-nb15-dataset, accessed on 20 June 2025.	[118]	[59,74,84]
BoT-IoT-V2	Public	Intrusion Detection	https://github.com/UNSW-NB15/NF-BoT-IoT-v2, accessed on 22 October 2025.	[119]	[78]
ToN-IoT-V2	Public	Intrusion Detection	https://github.com/UNSW-NB15/NF-ToN-IoT-v2, accessed on 22 October 2025.	[119]	[78]
CSE-CIC-IDS2018-V2	Public	Intrusion Detection	https://github.com/UNSW-NB15/NF-CSE-CIC-IDS2018-v2, accessed on 22 October 2025.	[119]	[74,78]
CIC-Darknet2020	Public	Intrusion Detection	https://www.unb.ca/cic/datasets/malmem-2020.html, accessed on 20 June 2025.	[120]	[59,100]
ISCX-Tor2016	Public	Network Traffic Classification	https://www.unb.ca/research/iscx/datasets/tor-dataset.html, accessed on 22 October 2025.	[121]	[59,97]
ISCX-Vpn2016	Public	Network Traffic Classification	https://www.unb.ca/cic/datasets/vpn-2016.html, accessed on 20 June 2025.	[121]	[83]
UNB-ISCX	Public	Intrusion Detection	https://www.unb.ca/cic/datasets/malmem-2020.html, accessed on 20 June 2025.	[122]	[92,94]
Kyoto	Public	Network Traffic Classification	http://www.takakura.com/Kyoto_data, accessed on 22 October 2025.	[123]	[98]
AndroZoo	Public	Encrypted Mobile App Traffic	https://androzoo.uni.lu, accessed on 22 October 2025.	[124]	[102]

7.2. Open-Source Code

Table 10 summarizes the available open-source code from the reviewed studies to support reproducible research and comparative analysis in network traffic analysis.

Table 10. Open-Source Code Summary.

Year	Model	Task	Framework	GitHub	Related Work
2023	MalDiscovery	Malicious Traffic Detection	-	http://github.com/NatsuHB/MalDiscovery, accessed on 22 October 2025.	[56]
2019	MVAN	Real Currency Transaction Detection	TensorFlow	https://github.com/fuxiAIlab/MVAN, accessed on 22 October 2025.	[57]
2020	CARE-GNN	Cybercrime Detection	Pytorch	https://github.com/safe-graph/DGFraud, accessed on 22 October 2025.	[64]
2024	HAWK	Android Malware Classification	TensorFlow	https://github.com/RingBDStack/HAWK, accessed on 22 October 2025.	[65]
2023	TS-IDS	IoT Network Intrusion Detection	Pytorch	https://github.com/hoangntc/TS-IDS, accessed on 22 October 2025.	[76]
2023	HyperVision	Encrypted Malicious Traffic Detection	-	https://github.com/fuchuanpu/HyperVision, accessed on 22 October 2025.	[77]
2022	E-GNNExplainer	Network Intrusion Detection	-	https://github.com/EagleEye1107/E-GNNExplainer, accessed on 22 October 2025.	[71]
2024	IBGC	Encrypted Network Traffic Classification	-	https://github.com/martinallee/ibgc, accessed on 22 October 2025.	[106]
2022	BIG	Anomaly Detection	-	https://github.com/SlothfulZhu/Wrongdoing-Monitor, accessed on 22 October 2025.	[98]
2021	MAppGraph	Encrypted Network Traffic Classification	-	https://soeai.github.io/MAppGraph, accessed on 22 October 2025.	[101]
2024	TLSFingerprint	Encrypted Malicious Traffic Detection	Pytorch	https://github.com/wesly2000/TLSFingerprint, accessed on 22 October 2025.	[108]

8. Discussion

8.1. Limitations

This review has several limitations. First, we only included research published in English and indexed in major databases, which may have resulted in the omission of some relevant studies. Second, heterogeneity in methods and evaluation metrics across studies may limit the comparability of results. Additionally, unpublished and gray literature were not considered.

Beyond these general limitations, the application of Graph Neural Networks (GNNs) to network traffic analysis introduces specific challenges that warrant further discussion. While GNNs offer unique advantages in capturing spatial and relational patterns in traffic data, their scalability remains a critical issue when dealing with large-scale network graphs. For instance, the computational overhead of training GNNs on graphs with millions of nodes and edges, as highlighted in the benchmark analysis (e.g., the trade-offs in computational efficiency for methods like Zhang et al.'s HGCN [53] and Gu et al.'s DLGNN [79]), can hinder real-time deployment. Moreover, the high training latency of GNNs poses challenges for scenarios requiring immediate analysis, such as real-time intrusion detection or dynamic traffic routing. The benchmark results further underscore these limitations, as methods achieving near-perfect accuracy (e.g., Duan et al.'s CTGNN [78]) often rely on extensive computational resources, making them less feasible for resource-constrained environments. Additionally, the interpretability of GNNs in traffic analysis remains underdeveloped, limiting their adoption in operational settings where transparency is crucial for decision-making. These challenges related to scalability, latency, and interpretability highlight the need for future research to address the practical constraints of GNNs in network traffic analysis.

8.2. Future Research Directions

As network environments rapidly evolve and traffic patterns become increasingly complex, GNN-based traffic analysis technologies still face numerous challenges and development opportunities. This section explores future development directions in this field, focusing on technological advances, practical implementation challenges, and emerging applications.

(1) Deepening Spatiotemporal Dynamic Traffic Modeling

Existing GNN traffic analysis methods have limitations in handling the spatiotemporal dynamic characteristics of network traffic. Future research should focus on developing novel spatiotemporal GNN architectures that can simultaneously capture traffic temporal

evolution and spatial topological relationships. This includes constructing multi-scale time-aware graph representation learning methods to understand traffic variation patterns at different temporal granularities (seconds, minutes, hours). Additionally, efficient streaming graph learning algorithms should be developed to enable models to quickly adapt when traffic patterns change, thereby supporting real-time traffic analysis and prediction tasks.

(2) Privacy Protection and Federated Traffic Learning

The sensitivity of network traffic data demands enhanced privacy protection, as traditional centralized learning methods face risks of data leakage and privacy violations. Future research should focus on developing federated GNN frameworks that support collaborative training across multiple network domains, achieving knowledge sharing and model performance improvement while protecting the privacy of traffic data in each domain. This includes applying differential privacy techniques to graph representation learning to protect the privacy of network topology and traffic patterns while ensuring analysis effectiveness, as well as exploring homomorphic encryption methods for GNN computation in encrypted states to achieve truly privacy-preserving traffic analysis.

(3) Integration of Interpretability and Causal Reasoning

Current GNN-based traffic analysis methods generally suffer from the "black box" problem, with model decision processes lacking transparency, which constitutes a major obstacle in network operations and fault diagnosis. Future research needs to develop GNN interpretability methods specifically for traffic analysis scenarios that can identify key network nodes, edges, and subgraph structures affecting prediction results, while combining causal reasoning techniques to not only predict traffic changes but also explain the fundamental causes of changes. Additionally, introducing uncertainty quantification mechanisms in traffic prediction and anomaly detection to provide confidence assessments of prediction results, offering reliable theoretical support for network optimization decisions.

(4) Large-scale Real-time Deployment Optimization

When considering practical network deployment requirements, existing GNN models encounter challenges such as high computational complexity and large memory consumption when processing large-scale network traffic. Future research should design distributed GNN training and inference frameworks suitable for large-scale networks, effectively handling network graphs with millions of nodes and edges. Simultaneously, lightweight GNN models should be developed to support real-time traffic analysis on network edge devices. Additionally, constructing incremental learning mechanisms that support online model updates, enabling systems to adapt to network topology changes and new traffic patterns while maintaining high-precision analysis performance with reduced network latency and bandwidth consumption.

(5) Robustness and Adversarial Defense

With the widespread application of GNNs in traffic analysis, adversarial attacks targeting these models are also increasing, making model robustness and security key challenges. Future research needs to deeply analyze the vulnerabilities of GNN models in traffic analysis scenarios and design GNN architectures with inherent adversarial resistance, possibly through integrating adversarial training techniques to improve model robustness. Meanwhile, it is necessary to establish security evaluation frameworks specifically for traffic analysis GNN models and develop real-time monitoring and defense mechanisms. Additionally, rapid response mechanisms for model updates and repairs should be established to ensure stable analysis performance when facing constantly evolving adversarial strategies.

9. Conclusions

This survey presents a general framework for GNN-based traffic analysis and systematically reviews recent advances in this field. Through our proposed three-level taxonomy,

we organize 78 representative works, revealing methodological differences across various prediction tasks. Compared to traditional traffic analysis methods, GNNs can effectively capture the interaction and structural information of traffic, thus achieving better performance in tasks such as anomaly detection and traffic classification. However, although GNNs have advantages in alleviating the challenges faced by traditional methods, they also introduce new problems, such as scalability, training latency, and model interpretability. Despite these challenges, GNNs, as an emerging technology, still hold great potential for development in the field of traffic analysis. By continuously optimizing model structures, improving training methods, and enhancing interpretability, GNNs are expected to become the mainstream technology in the field of traffic analysis in the future. We hope that this survey will provide researchers and practitioners with a comprehensive understanding of GNN applications in network traffic analysis and offer practical guidance for future research directions.

Author Contributions: Conceptualization, R.W. and M.H.; investigation, J.Z., H.Z., L.H. and H.L.; writing—original draft preparation, R.W., J.Z. and H.Z.; writing—review and editing, R.W., L.H., H.L. and M.H.; visualization, J.Z. and H.Z.; supervision, R.W. and M.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing is not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. International Telecommunication Union. Facts and Figures 2024. Available online: https://www.itu.int/itu-d/reports/statistics/facts-figures-2024/index/ (accessed on 27 June 2025).
2. Sperotto, A.; Sadre, R.; Van Vliet, F.; Pras, A. A Labeled Data Set for Flow-Based Intrusion Detection. In *IP Operations and Management*; Nunzi, G., Scoglio, C., Li, X., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2009; Volume 5843, pp. 39–50. [CrossRef]
3. Fogla, P.; Lee, W. Evading Network Anomaly Detection Systems: Formal Reasoning and Practical Techniques. In Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS 2006), Alexandria, VA, USA, 30 October–3 November 2006; pp. 59–68. [CrossRef]
4. Finsterbusch, M.; Richter, C.; Rocha, E. A Survey of Payload-Based Traffic Classification Approaches. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 1135–1156. [CrossRef]
5. Mishra, P.; Varadharajan, V.; Tupakula, U. A Detailed Investigation and Analysis of Using Machine Learning Techniques for Intrusion Detection. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 686–728. [CrossRef]
6. Gibert, D.; Mateu, C.; Planes, J. The Rise of Machine Learning for Detection and Classification of Malware: Research Developments, Trends and Challenges. *J. Netw. Comput. Appl.* **2020**, *153*, 102526. [CrossRef]
7. Wang, Q.; Xie, M.; Wu, Z.; Dong, Y. Network Intrusion Detection and Dynamic Defense Method Based on Unsupervised Machine Learning. In Proceedings of the 2023 International Conference on Computer Simulation and Modeling, Information Security (CSMIS), Buenos Aires, Argentina, 15–17 November 2023; pp. 75–80. [CrossRef]
8. Ren, Y. Analysis of Network Intrusion Detection Based on Machine Learning. In Proceedings of the 2022 4th International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM), Hamburg, Germany, 7–9 October 2022; pp. 699–702. [CrossRef]
9. Zhang, J.; Pan, L.; Han, Q.-L.; Chen, C.; Wen, S.; Xiang, Y. Deep Learning Based Attack Detection for Cyber-Physical System Cybersecurity: A Survey. *IEEE/CAA J. Autom. Sin.* **2022**, *9*, 377–391. [CrossRef]
10. Pan, Y.; Zhang, X.; Jiang, H.; Li, C. A Network Traffic Classification Method Based on Graph Convolution and LSTM. *IEEE Access* **2021**, *9*, 158261–158272. [CrossRef]
11. Xue, Y.; Wang, D.; Zhang, L. Traffic Classification: Issues and Challenges. *J. Commun.* **2013**, *8*, 240–248. [CrossRef]

12. Lotfollahi, M.; Jafari Siavoshani, M.; Shirali Hossein Zade, R. Deep Packet: A Novel Approach for Encrypted Traffic Classification Using Deep Learning. *Soft Comput.* **2020**, *24*, 1999–2012. [CrossRef]
13. Li, Y.; Xie, S.; Wan, Z.; Lv, H.; Song, H.; Lv, Z. Graph-powered Learning Methods in the Internet of Things: A Survey. *Mach. Learn. Appl.* **2023**, *11*, 100441. [CrossRef]
14. Shen, Y.; Li, Q.; Xu, C.; Chang, C.; Yin, Q. Graph-based Context Learning Network for Infrared Small Target Detection. *Neurocomputing* **2025**, *616*, 128949. [CrossRef]
15. Subedi, R.; Wei, L.; Gao, W.; Chakraborty, S.; Liu, Y. Empowering Active Learning for 3D Molecular Graphs with Geometric Graph Isomorphism. In Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS), Vancouver, BC, Canada, 10–16 December 2024; Curran Associates Inc.: Red Hook, NY, USA, 2024; pp. 1–14. Available online: https://dl.acm.org/doi/10.5555/3737916.3739679 (accessed on 27 June 2025).
16. Tan, J.; Jin, H.; Zhang, H.; Zhang, Y.; Chang, D.; Liu, X.; Zhang, H. A Survey: When Moving Target Defense Meets Game Theory. *Comput. Sci. Rev.* **2023**, *48*, 100544. [CrossRef]
17. He, W.; Tan, J.; Guo, Y.; Shang, K.; Zhang, H. A Deep Reinforcement Learning-based Deception Asset Selection Algorithm in Differential Games. *IEEE Trans. Inf. Forensics Secur.* **2024**, *19*, 8353–8368. [CrossRef]
18. Goli, Y.D.; Ambika, R. Network Traffic Classification Techniques—A Review. In Proceedings of the 2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS), Theni, India, 21–22 December 2018; IEEE: New York, NY, USA, 2018; pp. 231–236. [CrossRef]
19. Bhatla, N.; Malik, M. Network Traffic Classification Techniques: A Review. In *Computational Intelligence for Engineering and Management Applications*; Lecture Notes in Electrical Engineering, 1045; Springer: Singapore, 2023; pp. 371–388. [CrossRef]
20. Azab, A.; Khasawneh, M.; Alrabaee, S.; Choo, K.-K.R.; Sarsour, M. Network Traffic Classification: Techniques, Datasets, and Challenges. *Digit. Commun. Netw.* **2022**, *10*, 676–692. [CrossRef]
21. Getman, A.I.; Ikonnikova, M.K. A Survey of Network Traffic Classification Methods Using Machine Learning. *Program. Comput. Softw.* **2022**, *48*, 413–423. [CrossRef]
22. Papadogiannaki, E.; Ioannidis, S. A Survey on Encrypted Network Traffic Analysis Applications, Techniques, and Countermeasures. *ACM Comput. Surv.* **2021**, *54*, 123:1–123:35. [CrossRef]
23. Meng, S.; Ke, Y.; Li, X.; Zhang, L.; Kou, J.; Yu, S.; Li, Q.; Xu, K. Machine Learning-Powered Encrypted Network Traffic Analysis: A Comprehensive Survey. *IEEE Commun. Surv. Tutor.* **2023**, *25*, 791–824. [CrossRef]
24. Bilot, T.; El Madhoun, N.; Al Agha, K.; Zouaoui, A. Graph Neural Networks for Intrusion Detection: A Survey. *IEEE Access* **2023**, *11*, 49114–49139. [CrossRef]
25. Zhong, M.; Lin, M.; Zhang, C.; Xu, Z. A Survey on Graph Neural Networks for Intrusion Detection Systems: Methods, Trends and Challenges. *Comput. Secur.* **2024**, *141*, 103821. [CrossRef]
26. Dong, G.; Tang, M.; Wang, Z.; Gao, J.; Guo, S.; Cai, L.; Gutierrez, R.; Campbel, B.; Barnes, L.E.; Boukhechba, M. Graph Neural Networks in IoT: A Survey. *ACM Trans. Sens. Netw.* **2023**, *19*, 47. [CrossRef]
27. Danesh Pazho, A.; Alinezhad Noghre, G.; Purkayastha, A.A.; Vempati, J.; Martin, O.; Tabkhi, H. A Survey of Graph-based Deep Learning for Anomaly Detection in Distributed Systems. *IEEE Trans. Knowl. Data Eng.* **2024**, *36*, 1–20. [CrossRef]
28. Yan, B.; Yang, C.; Shi, C. Graph Mining for Cybersecurity: A Survey. *ACM Trans. Knowl. Discov. Data* **2024**, *18*, 47. [CrossRef]
29. Bilot, T.; El Madhoun, N.; Al Agha, K.; Zouaoui, A. A Survey on Malware Detection with Graph Representation Learning. *ACM Comput. Surv.* **2024**, *56*, 278. [CrossRef]
30. He, M.; Wang, X.; Zhou, J.; Xi, Y.; Jin, L.; Wang, X. Deep-Feature-Based Autoencoder Network for Few-Shot Malicious Traffic Detection. *Secur. Commun. Netw.* **2021**, *2021*, 6659022. [CrossRef]
31. Xu, C.; Shen, J.; Du, X. A Method of Few-Shot Network Intrusion Detection Based on Meta-Learning Framework. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 3540–3552. [CrossRef]
32. Liu, J.; Xiao, Q.; Xin, L.; Wang, Q.; Yao, Y.; Jiang, Z. M3F: A Novel Multi-Session and Multi-Protocol Based Malware Traffic Fingerprinting. *Comput. Netw.* **2023**, *227*, 109723. [CrossRef]
33. Zhao, J.; Li, Q.; Hong, Y.; Shen, M. MetaRockETC: Adaptive Encrypted Traffic Classification in Complex Network Environments via Time Series Analysis and Meta-Learning. *IEEE Trans. Netw. Serv. Manag.* **2024**, *21*, 2460–2476. [CrossRef]
34. Ma, W.; Liu, R.; Li, K.; Yan, S.; Guo, J. An Adversarial Domain Adaptation Approach Combining Dual Domain Pairing Strategy for IoT Intrusion Detection under Few-Shot Samples. *Inf. Sci.* **2023**, *629*, 719–745. [CrossRef]
35. Niu, W.; Ma, X.; Lin, S.; Wang, S.; Qian, X.; Lin, X.; Wang, Y.; Ren, B. PatDNN: Achieving Real-Time DNN Execution on Mobile Devices with Pattern-based Weight Pruning. In Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Lausanne, Switzerland, 16–20 March 2020; pp. 907–922. [CrossRef]
36. He, M.; Huang, Y.; Wang, X.; Wei, P.; Wang, X. A Lightweight and Efficient IoT Intrusion Detection Method Based on Feature Grouping. *IEEE Internet Things J.* **2024**, *11*, 2935–2949. [CrossRef]
37. Maniriho, P.; Mahmood, A.N.; Chowdhury, M.J.M. A Survey of Recent Advances in Deep Learning Models for Detecting Malware in Desktop and Mobile Platforms. *ACM Comput. Surv.* **2024**, *56*, 145. [CrossRef]

38. Hamilton, W.L. The Graph Neural Network Model. In *Graph Representation Learning*; Springer: Cham, Switzerland, 2020; pp. 51–70. [CrossRef]
39. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In Proceedings of the 5th International Conference on Learning Representations (ICLR), Toulon, France, 24–26 April 2017.
40. Kim, J.; Kim, T.; Kim, S.; Hong, S.; Shin, J. Edge-labeling Graph Neural Network for Few-Shot Learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 11575–11584. [CrossRef]
41. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph Attention Networks. In Proceedings of the 6th International Conference on Learning Representations (ICLR), Vancouver, BC, Canada, 30 April–3 May 2018.
42. Xu, K.; Hu, W.; Leskovec, J.; Jegelka, S. How Powerful Are Graph Neural Networks? In Proceedings of the 7th International Conference on Learning Representations (ICLR), New Orleans, LA, USA, 6–9 May 2019.
43. Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A.; Hovy, E. Hierarchical Attention Networks for Document Classification. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT), San Diego, CA, USA, 12–17 June 2016; pp. 1480–1489. [CrossRef]
44. Dvorak, S.; Prochazka, P.; Bajer, L. GNN-Based Malicious Network Entities Identification in Large-Scale Network Data. In Proceedings of the 2022 IEEE/IFIP Network Operations and Management Symposium (NOMS), Budapest, Hungary, 25–29 April 2022; pp. 1–4. [CrossRef]
45. Zhou, X.; Liang, W.; Li, W.; Yan, K.; Shimizu, S.; Wang, K.I.-K. Hierarchical Adversarial Attacks Against Graph-Neural-Network-Based IoT Network Intrusion Detection System. *IEEE Internet Things J.* **2022**, *9*, 9310–9319. [CrossRef]
46. Bakar, R.A.; De Marinis, L.; Cugini, F.; Tornatore, M.; Svaluto Moreolo, M.; Wang, J.; Chaux, C.; Velasco, L. FTG-net-e: A Hierarchical Ensemble Graph Neural Network for DDoS Attack Detection. *Comput. Netw.* **2024**, *245*, 110508. [CrossRef]
47. Shen, L.; Fang, M.; Xu, J. GHGDroid: Global Heterogeneous Graph-Based Android Malware Detection. *Comput. Secur.* **2024**, *141*, 103846. [CrossRef]
48. Liu, Z.; Chen, C.; Yang, X.; Hu, J.; Ding, K.; Wang, Y.; Xiong, H. Heterogeneous Graph Neural Networks for Malicious Account Detection. *arXiv* **2020**, arXiv:2002.12307. [CrossRef]
49. Presekal, A.; Stefanov, A.; Rajkumar, V.S.; Palensky, P. Attack Graph Model for Cyber-Physical Power Systems Using Hybrid Deep Learning. *IEEE Trans. Smart Grid* **2023**, *14*, 4007–4020. [CrossRef]
50. Wu, Y.; Dai, H.-N.; Tang, H. Graph Neural Networks for Anomaly Detection in Industrial Internet of Things. *IEEE Internet Things J.* **2022**, *9*, 9214–9231. [CrossRef]
51. Li, G.; Zhang, F.; Pang, J.; Zhang, T.; Pan, M.; Li, X.; Zhang, C. Functional Scenario Classification for Android Applications Using GNNs. In Proceedings of the 13th Asia-Pacific Symposium on Internetware (Internetware '22), Wuhan, China, 17–18 December 2022. [CrossRef]
52. Gu, Z.; Gou, G.; Liu, C.; Chen, Y.; Zhang, X.; Lin, Z.; Xiong, G.; Jia, W. Let Gambling Hide Nowhere: Detecting Illegal Mobile Gambling Apps Via Heterogeneous Graph-Based Encrypted Traffic Analysis. *Comput. Netw.* **2024**, *243*, 110278. [CrossRef]
53. Zhang, J.; Liu, X.; Yan, Q.; Liu, B.; Sun, M.; Peng, H.; Lu, J.; Yang, Q.; Wei, T. Multi-attributed Heterogeneous Graph Convolutional Network for Bot Detection. *Inf. Sci.* **2020**, *537*, 380–393. [CrossRef]
54. Simon, M.; Pitner, T.; Safarik, V.; Blazek, L.; Dobias, J.; Hanacek, P. Malicious Internet Entity Detection Using Local Graph Inference. *IEEE Trans. Inf. Forensics Secur.* **2024**, *19*, 3554–3566. [CrossRef]
55. Chowdhury, S.; Khanzadeh, M.; Akula, R.; Zhang, F.; Zhang, S.; Medal, H.; Marufuzzaman, M.; Bian, L. Botnet Detection Using Graph-Based Feature Clustering. *J. Big Data* **2017**, *4*, 14. [CrossRef]
56. Yueping, H.; Qi, L.; Yanqing, Y.; Meng, S. Graph Based Encrypted Malicious Traffic Detection with Hybrid Analysis of Multi-View Features. *Inf. Sci.* **2023**, *644*, 119229. [CrossRef]
57. Tao, J.; Lin, J.; Zhang, S. MVAN: Multi-View Attention Networks for Real Money Trading Detection in Online Games. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 2536–2546. [CrossRef]
58. Zhang, Y.; Yang, C.; Huang, K.; Li, Y. Intrusion Detection of Industrial Internet-of-Things Based on Reconstructed Graph Neural Networks. *IEEE Trans. Netw. Sci. Eng.* **2023**, *10*, 2894–2905. [CrossRef]
59. Deng, X.; Zhu, J.; Pei, X.; Zhang, L.; Ling, Z.; Xue, K. Flow Topology-Based Graph Convolutional Network for Intrusion Detection in Label-Limited IoT Networks. *IEEE Trans. Netw. Serv. Manag.* **2023**, *20*, 684–696. [CrossRef]
60. Zhang, S.; Yin, J.; Li, Z.; Yang, R.; Du, M.; Li, R. Node-Imbalance Learning on Heterogeneous Graph for Pirated Video Website Detection. In Proceedings of the 2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD), Hangzhou, China, 4–6 May 2022; pp. 834–840. [CrossRef]
61. Liu, J.; Zheng, J.; Wu, J.; Zheng, Z. FA-GNN: Filter and Augment Graph Neural Networks for Account Classification in Ethereum. *IEEE Trans. Netw. Sci. Eng.* **2022**, *9*, 2579–2588. [CrossRef]

62. Lin, D.; Wu, J.; Huang, T.; Lin, K.; Zheng, Z. Who is Who on Ethereum? Account Labeling Using Heterophilic Graph Convolutional Network. *IEEE Trans. Syst. Man Cybern. Syst.* **2024**, *54*, 1541–1553. [CrossRef]
63. Xie, F.; Cao, Z.; Xu, Y.; Chen, L.; Zheng, Z. Graph Neural Network and Multi-View Learning Based Mobile Application Recommendation in Heterogeneous Graphs. In Proceedings of the 2020 IEEE International Conference on Services Computing (SCC), Virtual, Beijing, China, 18–24 October 2020; pp. 100–107. [CrossRef]
64. Dou, Y.; Liu, Z.; Sun, L.; Deng, Y.; Peng, H.; Yu, P.S. Enhancing Graph Neural Network-Based Fraud Detectors Against Camouflaged Fraudsters. In Proceedings of the 29th ACM International Conference on Information & Knowledge Management (CIKM), Virtual Event, Ireland, 19–23 October 2020; pp. 315–324. [CrossRef]
65. Hei, Y.; Yang, R.; Peng, H.; Wang, L.; Liu, X.; Kang, Y. Hawk: Rapid Android Malware Detection Through Heterogeneous Graph Attention Networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2024**, *35*, 4703–4717. [CrossRef] [PubMed]
66. Li, A.; Qin, Z.; Liu, R.; Yang, Y.; Li, D. Spam Review Detection with Graph Convolutional Networks. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM), Beijing, China, 3–7 November 2019; pp. 2703–2711. [CrossRef]
67. Zheng, L.; Li, Z.; Li, J.; Li, Z.; Gao, J. AddGraph: Anomaly Detection in Dynamic Graph Using Attention-Based Temporal GCN. In Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI), Macao, China, 10–16 August 2019; pp. 4419–4425. [CrossRef]
68. Xu, P.; Lu, G.; Li, Y.; Xu, C. EE-GCN: A Graph Convolutional Network Based Intrusion Detection Method for IIoT. In Proceedings of the 2023 5th International Conference on Natural Language Processing (ICNLP), Guangzhou, China, 24–26 March 2023; pp. 338–344. [CrossRef]
69. Xu, R.; Wu, G.; Wang, W.; Gao, X.; He, A.; Zhang, Z. Applying Self-Supervised Learning to Network Intrusion Detection for Network Flows with Graph Neural Network. *Comput. Netw.* **2024**, *248*, 110495. [CrossRef]
70. Fu, C.; Li, Q.; Xu, K. Detecting Unknown Encrypted Malicious Traffic in Real Time via Flow Interaction Graph Analysis. In Proceedings of the 30th Network and Distributed System Security Symposium (NDSS), San Diego, CA, USA, 27 February–3 March 2023. [CrossRef]
71. Baahmed, A.R.E.; Andresini, G.; Robardet, C.; Appice, A. Using Graph Neural Networks for the Detection and Explanation of Network Intrusions. In Proceedings of the European Conference on Machine Learning (ECML 2023), Torino, Italy, 18–22 September 2023; pp. 201–216. [CrossRef]
72. Duan, G.; Lv, H.; Wang, H.; Feng, G.; Li, X. Practical Cyber Attack Detection with Continuous Temporal Graph in Dynamic Network System. *IEEE Trans. Inf. Forensics Secur.* **2024**, *19*, 4851–4864. [CrossRef]
73. Yang, Z.; Pei, W.; Chen, M.; Yue, C. WTAGRAPH: Web Tracking and Advertising Detection Using Graph Neural Networks. In Proceedings of the 43rd IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 23–26 May 2022; pp. 1540–1557. [CrossRef]
74. Caville, E.; Lo, W.W.; Layeghy, S.; Portmann, M. Anomal-E: A Self-Supervised Network Intrusion Detection System Based on Graph Neural Networks. *Knowl.-Based Syst.* **2022**, *258*, 110030. [CrossRef]
75. Song, J.; Qu, X.; Hu, Z.; Li, Z.; Gao, J.; Zhang, J. A Subgraph-Based Knowledge Reasoning Method for Collective Fraud Detection in E-Commerce. *Neurocomputing* **2021**, *461*, 587–597. [CrossRef]
76. Nguyen, H.C.; Kashef, R. TS-IDS: Traffic-Aware Self-Supervised Learning for IoT Network Intrusion Detection. *Knowl.-Based Syst.* **2023**, *279*, 110966. [CrossRef]
77. Altaf, T.; Wang, X.; Ni, W.; Yu, G.; Liu, R.P.; Braun, R. A New Concatenated Multigraph Neural Network for IoT Intrusion Detection. *Internet Things* **2023**, *22*, 100818. [CrossRef]
78. Duan, G.; Lv, H.; Wang, H.; Feng, G. Application of a Dynamic Line Graph Neural Network for Intrusion Detection with Semisupervised Learning. *IEEE Trans. Inform. Forensics Secur.* **2023**, *18*, 699–714. [CrossRef]
79. Gu, Z.; Lopez, D.T.; Alrahis, L.; Sinanoglu, O. Always be Pre-Training: Representation Learning for Network Intrusion Detection with GNNs. *arXiv* **2024**, arXiv:2402.18986. [CrossRef]
80. Chang, X.; Ren, P.; Xu, P.; Li, Z.; Chen, X.; Hauptmann, A. A Comprehensive Survey of Scene Graphs: Generation and Application. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 1–26. [CrossRef] [PubMed]
81. Wu, S.; Sun, F.; Zhang, W.; Xie, X.; Cui, B. Graph Neural Networks in Recommender Systems: A Survey. *ACM Comput. Surv.* **2022**, *55*, 97. [CrossRef]
82. Shen, M.; Zhang, J.; Zhu, L.; Xu, K.; Du, X. Accurate Decentralized Application Identification via Encrypted Traffic Analysis Using Graph Neural Networks. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 2367–2380. [CrossRef]
83. Chen, Z.; Cheng, G.; Niu, D.; Qiu, X.; Zhao, Y.; Zhou, Y. WFF-EGNN: Encrypted Traffic Classification Based on Weaved Flow Fragment via Ensemble Graph Neural Networks. *IEEE Trans. Mach. Learn. Commun. Netw.* **2023**, *1*, 389–411. [CrossRef]
84. Han, Y.; Wang, X.; He, M.; Wang, X.; Guo, S. Intrusion Detection for Encrypted Flows Using Single Feature Based on Graph Integration Theory. *IEEE Internet Things J.* **2024**, *11*, 17589–17601. [CrossRef]
85. Gao, B.; Liu, W.; Liu, G.; Nie, F. Resource Knowledge-Driven Heterogeneous Graph Learning for Website Fingerprinting. *IEEE Trans. Cogn. Commun. Netw.* **2024**, *10*, 968–981. [CrossRef]

86. Xu, H.; Li, S.; Cheng, Z.; Qin, R.; Xie, J.; Sun, P. TrafficGCN: Mobile Application Encrypted Traffic Classification Based on GCN. In Proceedings of the 2022 IEEE Global Communications Conference (GLOBECOM), Rio de Janeiro, Brazil, 4–8 December 2022; pp. 891–896. [CrossRef]
87. Feng, P.; Wei, D.; Li, Q.; Wang, Q.; Hu, Y.; Xi, N.; Ma, J. GlareShell: Graph Learning-Based PHP Webshell Detection for Web Server of Industrial Internet. *Comput. Netw.* **2024**, *245*, 110406. [CrossRef]
88. Kisanga, P.; Woungang, I.; Traoré, I.; Carvalho, G.H.S. Network Anomaly Detection Using a Graph Neural Network. In Proceedings of the 2023 International Conference on Computing, Networking and Communications (ICNC), Honolulu, HI, USA, 20–23 February 2023; pp. 61–65. [CrossRef]
89. Li, H.; Chasaki, D. Heterogeneous GNN with Express Edges for Intrusion Detection in Cyber-Physical Systems. In Proceedings of the 2024 International Conference on Computing, Networking and Communications (ICNC), Honolulu, HI, USA, 19–22 February 2024; pp. 523–529. [CrossRef]
90. Cui, T.; Gou, G.; Xiong, G.; Li, Z.; Cui, M.; Liu, C. SiamHAN: IPv6 Address Correlation Attacks on TLS Encrypted Traffic via Siamese Heterogeneous Graph Attention Network. In Proceedings of the 30th USENIX Security Symposium, Virtual Event, 11–13 August 2021; pp. 4329–4346. [CrossRef]
91. Zhang, H.; Yu, L.; Xiao, X.; Li, Q.; Mercaldo, F.; Luo, X.; Liu, Q. TFE-GNN: A Temporal Fusion Encoder Using Graph Neural Networks for Fine-Grained Encrypted Traffic Classification. In Proceedings of the ACM Web Conference 2023 (WWW '23), Austin, TX, USA, 30 April–4 May 2023; pp. 2066–2075. [CrossRef]
92. Hu, G.; Xiao, X.; Shen, M.; Zhang, B.; Yan, X.; Liu, Y. TCGNN: Packet-Grained Network Traffic Classification via Graph Neural Networks. *Eng. Appl. Artif. Intell.* **2023**, *123*, 106531. [CrossRef]
93. Okonkwo, Z.; Foo, E.; Hou, Z.; Li, Q.; Jadidi, Z. Encrypted Network Traffic Classification with Higher Order Graph Neural Network. In Proceedings of the 28th Australasian Conference on Information Security and Privacy (ACISP 2023), Melbourne, VIC, Australia, 5–7 July 2023; pp. 630–650. [CrossRef]
94. Huoh, T.-L.; Luo, Y.; Li, P.; Zhang, T. Flow-Based Encrypted Network Traffic Classification with Graph Neural Networks. *IEEE Trans. Netw. Serv. Manag.* **2023**, *20*, 1224–1237. [CrossRef]
95. Yang, J.; Jiang, X.; Lei, Y.; Liang, W.; Ma, Z.; Li, S. MTSecurity: Privacy-Preserving Malicious Traffic Classification Using Graph Neural Network and Transformer. *IEEE Trans. Netw. Serv. Manag.* **2024**, *21*, 3583–3597. [CrossRef]
96. Fu, C.; Li, Q.; Xu, K. Flow Interaction Graph Analysis: Unknown Encrypted Malicious Traffic Detection. *IEEE/ACM Trans. Netw.* **2024**, *32*, 2972–2987. [CrossRef]
97. Zhao, R.; Deng, X.; Wang, Y.; Chen, L.; Liu, M.; Xue, Z.; Wang, Y. Flow Sequence-Based Anonymity Network Traffic Identification with Residual Graph Convolutional Networks. In Proceedings of the 2022 IEEE/ACM 30th International Symposium on Quality of Service (IWQoS 2022), Oslo, Norway, 10–12 June 2022; pp. 1–10. [CrossRef]
98. Wang, C.; Zhu, H. Wrongdoing Monitor: A Graph-Based Behavioral Anomaly Detection in Cyber Security. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17*, 2703–2718. [CrossRef]
99. Zhang, H.; Zeng, K.; Lin, S. Federated Graph Neural Network for Fast Anomaly Detection in Controller Area Networks. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 1566–1579. [CrossRef]
100. Zhu, Y.; Tao, J.; Wang, H.; Yu, L.; Luo, Y.; Qi, T.; Wang, Z.; Xu, Y. DGNN: Accurate Darknet Application Classification Adopting Attention Graph Neural Network. *IEEE Trans. Netw. Serv. Manag.* **2024**, *21*, 1258–1273. [CrossRef]
101. Pham, T.-D.; Ho, T.-L.; Truong-Huu, T.; Cao, T.-D.; Truong, H.-L. MAppGraph: Mobile-App Classification on Encrypted Network Traffic Using Deep Graph Convolution Neural Networks. In Proceedings of the 37th Annual Computer Security Applications Conference (ACSAC 2021), Virtual Event, 6–10 December 2021; pp. 1025–1038. [CrossRef]
102. Pei, X.; Yu, L.; Tian, S. AMalNet: A Deep Learning Framework Based on Graph Convolutional Networks for Malware Detection. *Comput. Secur.* **2020**, *93*, 101792. [CrossRef]
103. Guo, T.; Cui, B. Web Page Classification Based on Graph Neural Network. In *Innovative Mobile and Internet Services in Ubiquitous Computing*; Barolli, L., Yim, K., Chen, H.-C., Eds.; Lecture Notes in Networks and Systems; Springer: Cham, Switzerland, 2022; Volume 279, pp. 188–198. [CrossRef]
104. Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Yu, P.S. A Comprehensive Survey on Graph Neural Networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 4–24. [CrossRef]
105. Fan, W.; Ma, Y.; Li, Q.; He, Y.; Zhao, E.; Tang, J.; Yin, D. Graph Neural Networks for Social Recommendation. In Proceedings of the 2019 World Wide Web Conference (WWW '19), San Francisco, CA, USA, 13–17 May 2019; pp. 417–426. [CrossRef]
106. Li, Y.; Chen, X.; Tang, W.; Zhu, Y.; Han, Z.; Yue, Y. Interaction Matters: Encrypted Traffic Classification via Status-Based Interactive Behavior Graph. *Appl. Soft Comput.* **2024**, *155*, 111423. [CrossRef]
107. Busch, J.; Kocheturov, A.; Tresp, V.; Seidl, T. NF-GNN: Network Flow Graph Neural Networks for Malware Detection and Classification. In Proceedings of the 33rd International Conference on Scientific and Statistical Database Management (SSDBM 2021), Tampa, FL, USA, 6–7 July 2021; pp. 121–132. [CrossRef]

108. Yu, L.; Tao, J.; Xu, Y.; Sun, W.; Wang, Z. TLS Fingerprint for Encrypted Malicious Traffic Detection with Attributed Graph Kernel. *Comput. Netw.* **2024**, *247*, 110475. [CrossRef]
109. Qin, Z.-Q.; Xu, H.-Z.; Ma, X.-K.; Wang, Y.-J. Interaction Context-Aware Network Behavior Anomaly Detection for Discovering Unknown Attacks. *Secur. Commun. Netw.* **2022**, *2022*, 3595304. [CrossRef]
110. Wang, L.; Ma, X.; Li, N.; Lv, Q.; Wang, Y.; Huang, W.; Chen, H. TGPrint: Attack Fingerprint Classification on Encrypted Network Traffic Based Graph Convolution Attention Networks. *Comput. Secur.* **2023**, *135*, 103466. [CrossRef]
111. Garcia, S.; Grill, M.; Stiborek, J.; Zunino, A. An Empirical Comparison of Botnet Detection Methods. *Comput. Secur.* **2014**, *45*, 100–123. [CrossRef]
112. Song, H.M.; Kim, H.K. *CAN Signal Extraction and Translation Dataset*; HCRL: Seoul, Republic of Korea, 2020. Available online: https://ocslab.hksecurity.net/Datasets/can-signal-extraction-and-translation-dataset (accessed on 27 June 2025).
113. Arp, D.; Spreitzenbarth, M.; Hübner, M.; Gascon, H.; Rieck, K.; Siemens, C.E.R.T. DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket. In Proceedings of the Network and Distributed System Security Symposium (NDSS), San Diego, CA, USA, 23–26 February 2014. [CrossRef]
114. Lashkari, A.H.; Kadir, A.F.A.; Taheri, L.; Ghorbani, A.A. Toward Developing a Systematic Approach to Generate Benchmark Android Malware Datasets and Classification. In Proceedings of the 2018 International Carnahan Conference on Security Technology (ICCST), Montreal, QC, Canada, 22–25 October 2018; pp. 1–7. [CrossRef]
115. Praguard. Public Malware Traffic Dataset. Available online: https://github.com/Praguard (accessed on 27 June 2025).
116. Li, E.; Wang, L.; Song, B. Fault Diagnosis of Power Transformers With Membership Degree. *IEEE Access* **2019**, *7*, 28791–28798. [CrossRef]
117. Alsaedi, A.; Moustafa, N.; Tari, Z.; Mahmood, A.; Anwar, A. TON_IoT Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems. *IEEE Access* **2020**, *8*, 165130–165150. [CrossRef]
118. Moustafa, N.; Slay, J. UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set). In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, ACT, Australia, 10–12 November 2015; pp. 1–6. [CrossRef]
119. Sarhan, M.; Layeghy, S.; Portmann, M. Evaluating standard feature sets towards increased generalisability and explainability of MLbased network intrusion detection. *Big Data Res.* **2020**, *30*, 100359. [CrossRef]
120. Lashkari, A.H.; Kaur, G.; Rahali, A. DIDarknet: A Contemporary Approach to Detect and Characterize the Darknet Traffic Using Deep Image Learning. In Proceedings of the 10th International Conference on Communication and Network Security (ICCNS), Tokyo, Japan, 27–29 November 2020; pp. 1–13. [CrossRef]
121. Lashkari, A.H.; Draper-Gil, G.; Mamun, M.S.I.; Ghorbani, A.A. Characterization of Tor Traffic Using Time Based Features. In Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP), Porto, Portugal, 19–21 February 2017; pp. 253–262. [CrossRef]
122. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward the Development of a New Intrusion Detection System for the Internet of Things. In Proceedings of the 2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; pp. 1–6.
123. Song, J.; Takakura, H.; Okabe, Y.; Eto, M.; Inoue, D.; Nakao, K. Statistical Analysis of Honeypot Data and Building of Kyoto 2006+ Dataset for NIDS Evaluation. In Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS '11), Chicago, IL, USA, 21 October 2011; pp. 29–36. [CrossRef]
124. Allix, K.; Bissyandé, T.F.; Klein, J.; Le Traon, Y. AndroZoo: Collecting Millions of Android Apps for the Research Community. In Proceedings of the 13th International Conference on Mining Software Repositories (MSR), Austin, TX, USA, 14–15 May 2016; pp. 468–471. [CrossRef]