Directional Flow Graph Framework for Detecting Known and Unknown Malicious Traffic

Yang Wu[1], Ruijia Wang[1], Mingyuan Zang[1], and Jie Wu[1,2⋆]

[1] China Telecom Cloud Computing Research Institute, China
{wuy92,wangrj12,zangmy1}@chinatelecom.cn
[2] Temple University, USA
jiewu@temple.edu

Abstract. Malicious traffic detection is a major challenge due to the concealed and varied nature of real-world attacks. Current methods face two main limitations: first, they often represent bidirectional traffic as undirected graphs or sequences, which limits their ability to capture both temporal information and interaction patterns that provide critical attack clues. Second, supervised methods excel at identifying known malicious traffic, while unsupervised methods are more suitable for unknown attacks. Most studies focus on a single learning method, neglecting the benefits of integrating both approaches. To address these issues, we introduce Directional Graph Network (DGNet), the first framework combining directional graph modeling with hybrid learning paradigms for malicious traffic detection. To capture directional traffic information and interaction patterns in bi-flow traffic, we propose *Directional Flow Graph* modeling and a tailored DIRectional Graph Isomorphism Networks (DIRGIN) learning directional graph representation. To address the performance degradation caused by the inconsistent distribution of known and unknown attack data, we propose a hybrid learning framework based on the tailored graph neural network that combines the advantages of supervised and unsupervised learning. Comprehensive evaluations on two public datasets demonstrate that DGNet consistently outperforms state-of-the-art baselines across both detection scenarios.

Keywords: Malicious traffic detection · Directional flow graph · Known malicious traffic · Unknown malicious traffic.

1 Introduction

With the rapid development of information systems and data-driven services, data security and service reliability face increasingly sophisticated and diverse threats. Malicious attacks such as command-and-control (C&C) communications, malicious scanning, and Distributed Denial of Service (DDoS) attacks can lead to data breaches, service interruptions, delays, or performance degradation. In multi-tenant cloud environments, such attacks and other noisy workloads

⋆ Corresponding author

exacerbate performance interference, as co-located tenants compete for shared hardware resources that adversaries can exploit to degrade critical services [20]. Detecting malicious traffic is vital to maintaining network and information system security. The search for effective methods to identify malicious traffic has attracted significant interest from academia and industry.

Traditional rule-based network traffic detection methods are relatively easy to implement and offer low-latency responses. However, they rely heavily on domain expertise and exhibit limited adaptability to novel or evolving attack patterns [19]. Recently, machine learning and deep learning techniques have been extensively adopted, leveraging their strong feature representation to improve detection performance [1, 7]. However, these methods still have shortcomings, especially regarding feature representation and the learning paradigm.

In real-world network communications, the uplink and downlink communication interactions of bidirectional traffic contain important clues to distinguish between benign and malicious traffic. For example, legitimate applications (such as web browsing, streaming, and downloading) involve large amounts of data transmission, with small uplink requests and large downlink responses. Meanwhile, malicious programs (such as C&C attacks) are characterized by command-response communication, typically sending small packets to avoid detection and lacking continuous unidirectional data flow. Most existing detection methods rely on vector sequences or graph structures to model traffic behavior. However, graph-based methods mostly use undirected graphs, which weaken the fluidity and directionality of information. While sequence-based models capture temporal order, they lose the structural interactions and path dependencies between upstream and downstream traffic. Therefore, existing research lacks a unified representation method that can simultaneously represent the structural and temporal characteristics of traffic.

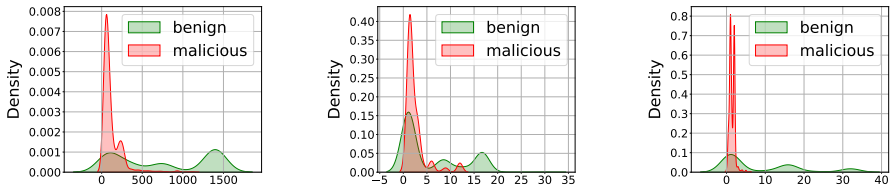
In addition to shortcomings in feature representation, existing methods also suffer from limitations in the learning paradigm, which can be divided into supervised learning, unsupervised learning, and hybrid learning paradigms. Supervised methods demonstrate high performance in detecting known malicious traffic but require substantial amounts of labeled training data [13, 18, 27, 28]. For previously unknown malicious traffic, model generalization performance may be limited and often requires retraining. Conversely, unsupervised learning methods do not rely on labeled data. By analyzing normal traffic patterns, they can detect unknown abnormal outliers, although their accuracy generally falls short compared to supervised learning for known malicious traffic [15, 5, 8]. Hybrid learning approaches combine the advantages of supervised and unsupervised learning. However, existing hybrid methods either use statistic-based methods that struggle to handle complex patterns or are based on Transformer [21] that require complex attention weight calculations [23, 30].

To address these limitations, we propose DGNet, the first framework combining directional graph modeling with hybrid learning paradigms for detecting malicious traffic. We propose a *Directional Flow Graph* to represent the structural interactions and temporal sequences of packets in bidirectional traffic. The

Directional Flow Graph treats packets as nodes and identifies the relationships between packets based on multiple types of edges in directionality and temporal order between packets. To better model the causality and hierarchy of directed flow graphs, we propose the DIRGIN to obtain feature representations of directed flow graphs. Although existing Graph Neural Network (GNN) methods [11, 22, 25] exhibit remarkable success on graph representation learning, they ignore directional information and cannot distinguish between incoming and outgoing edges in directed graphs. Our proposed DIRGIN module aggregates the neighbor information of incoming and outgoing edges separately and then combines them. This allows the model to capture information flow patterns and distinguish between request and response behaviors.

To ensure our model maintains high classification performance against malicious traffic attacks, even zero-day attacks, we propose an end-to-end hybrid learning framework to address performance degradation caused by inconsistent data distribution, balancing performance and complexity. First, we use DIRGIN as the encoder to extract features through supervised learning and train a strong-performance classifier. At this stage, DIRGIN and the classifier are excellent at classifying known malicious traffic but struggle to identify unknown malicious traffic outside their learned distribution. To improve this, we retain the DIRGIN parameters and introduce a decoder for unsupervised learning based on the autoencoder structure. This approach allows the model to recognize unknown malicious traffic that exceeds normal baselines by learning from benign traffic patterns. During detection, input traffic is classified based on hybrid results, enabling the identification of both known and unknown attacks. **The main contributions of this paper are summarized as follows**:

- According to our knowledge, we are the first to propose a *Directional Flow Graph* to represent the packet interaction of each bidirectional flow. The directional flow graph includes both structural and temporal information of packet interactions, which facilitates a more detailed mining of the differences between benign and malicious traffic.
- We design a directed graph neural network, DIRGIN, which develops Graph Isomorphism Networks (GIN) [25] to capture the flow graph's directional information. DIRGIN aggregates the information of incoming and outgoing edge neighbors separately, enabling the ability to model the causality and hierarchy of directional flow graphs.
- We propose a hybrid learning framework based on DIRGIN, combining the benefits of supervised and unsupervised learning to detect known and unknown attacks. We first develop the ability of DIRGIN to identify known malicious traffic through supervised training. Then, we retain the DIRGIN parameters and build an anomaly detection model based on an autoencoder structure to identify unknown malicious traffic.
- We conduct experiments using two reorganized public datasets for malicious traffic detection in known and unknown attack scenarios. Experimental results demonstrate that DGNet outperforms existing state-of-the-art approaches, achieving the highest accuracy and F1 scores in both scenarios.



(a) Packet Length (b) Burst (c) Down/Up Ratio

Fig. 1: The kernel density estimation (KDE) of benign and malicious traffic across three features in the USTC-TFC2016 [24] dataset.

(a) Benign (b) Malicious

Fig. 2: Examples of directional flow graphs for benign and malicious traffic in the USTC-TFC2016 dataset. Green nodes are downlink packets, and orange nodes are uplink packets.

2 Design Intuition

Although malicious traffic may be disguised or evade detection, it still exhibits distinct packet interaction patterns compared to benign traffic due to its underlying malicious behavior. To validate this, we analyze the USTC-TFC2016 dataset by randomly selecting 1,000 samples from each of 10 benign and malicious categories. Each sample is represented as a bidirectional session defined by a five-tuple, from which the average packet length, burst length, and down/up ratio are computed. As shown in Figure 1, these features reveal clear differences between benign and malicious traffic.

Based on the above observations, we consider extracting information such as packet length, burst, packet direction, and packet order for each session to differentiate malicious from benign traffic. Inspired by GraphDApp [18], we construct a *Directional Flow Graph* based on bi-flow interactions. The nodes represent packets, and the edges depict interaction relationships between packets. Figure 2 is a representative example of directional flow graphs for benign and malicious traffic. It reveals apparent differences between the two categories. The construction of the directed flow graph is described in detail in the next section.

We can train a graph-based encoder and a classifier using labeled data to effectively recognize known malicious traffic. However, since the model lacks prior knowledge of unknown attacks, it may incorrectly classify them as benign.

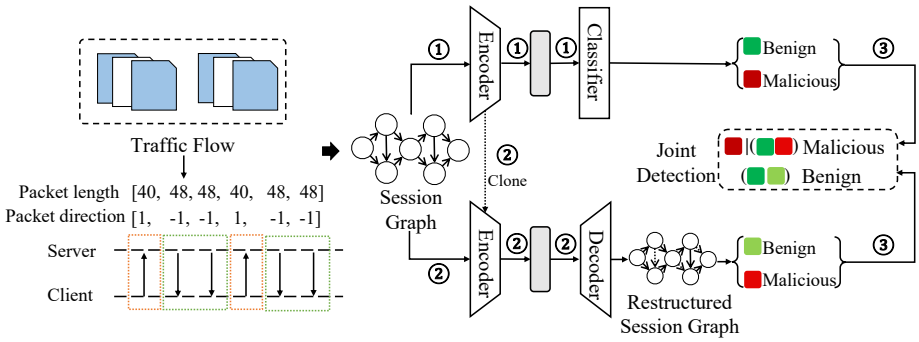


Fig. 3: The overview of DGNet. ①, ②, and ③ represent the supervised training, the unsupervised training, and the inference stage, respectively.

Thus, we introduce an encoder-decoder structure that identifies unknown attacks by measuring reconstruction errors, providing an additional layer of detection when the classifier alone is insufficient.

3 Proposed Method

3.1 DGNet Overview

The overview of training DGNet is presented in Figure 3. We collect bi-flow traffic based on 5-tuples and construct a directional flow graph for each session. In the supervised training stage, the directional flow graph is fed into the DIRGIN encoder to learn an effective classifier. In the unsupervised training stage, we clone the DIRGIN parameters and feed the hidden features produced by DIRGIN into the decoder, which is made up of Multilayer Perceptron (MLP), to reconstruct the directed graph's edges. During testing, the trained parameters of the supervised and unsupervised encoders are not shared. Each performs reasoning individually and summarizes the detection results from both stages.

3.2 Directional Flow Graph Construction

Benign and malicious traffic differ in packet length, burst patterns, and down/up ratios. To capture these traits, the directional flow graph is constructed by dividing the packet length sequence into bursts. As shown in Figure 4 and Figure 2b, edges connect intro-burst and inter-burst packets. Node features are initially set to the packet length multiplied by the direction (1 for uplink, -1 for downlink). A large burst followed by a small burst and a small burst followed by a large burst are distinct behaviors, but prior work using undirected graphs overlooks these packet order patterns [18, 10]. We construct directed graphs to preserve temporal and structural information. Packets within a burst are concatenated in order. Adjacent bursts are connected sequentially, with the first and last packets of one burst pointing to those of the next.

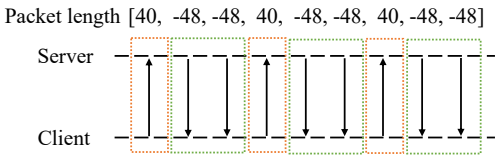


Fig. 4: The bi-flow interaction behavior corresponding to Figure 2b.

3.3 Supervised Learning Stage

Given a set of directional flow graphs $\mathcal{G} = \{G_1, ..., G_N\}$ and labels $\{y_1, ..., y_N\}$, we aim to learn a representation vector h_G of each directed graph that can be correctly predicted by a classifier in the supervised learning stage. The main body of DGNet in the supervised learning stage consists of a GNN-based encoder and an MLP classifier.

DIRGIN Encoder. GNNs effectively model irregular data by capturing complex topological dependencies, making them well-suited for analyzing session graphs. GIN is a variant of GNN that employs an injective neighborhood aggregation scheme, enabling the model to distinguish subtle structural variations in the graphs. However, vanilla GIN does not support learning directed graphs. Therefore, we improve GIN to aggregate the information of incoming and outgoing edge neighbors separately, thereby enabling the ability to learn directed graphs. We define the improved GIN as *DIRGIN*.

By stacking multiple DIRGIN convolution layers followed by pooling, we extract a discriminative representation for each directional flow graph, which is then utilized to identify malicious traffic. We employ a K-layer DIRGIN, where each convolution layer is defined as:

$$\mathbf{m}_v^{\text{in}} = \sum_{u \in \mathcal{N}_{\text{in}}(v)} \mathbf{h}_u^{k-1} \mathbf{W}_{\text{in}} \tag{1}$$

$$\mathbf{m}_v^{\text{out}} = \sum_{u \in \mathcal{N}_{\text{out}}(v)} \mathbf{h}_u^{k-1} \mathbf{W}_{\text{out}} \tag{2}$$

$$\mathbf{h}_v^{\text{self}} = \mathbf{h}_v^{k-1} \mathbf{W}_h \tag{3}$$

$$\mathbf{m}_v^{\text{com}} = \sigma\left([\mathbf{m}_v^{\text{in}} \parallel \mathbf{m}_v^{\text{out}}]\right) \mathbf{W}_{\text{com}} \tag{4}$$

$$\mathbf{h}_v^k = \sigma\left(\mathbf{m}_v^{\text{com}} + (1+\epsilon^k) \cdot \mathbf{h}_v^{\text{self}}\right) \mathbf{W}_{\text{final}} \tag{5}$$

where $\mathbf{h}_v^k \in \mathbb{R}^d$ is the feature of node v at layer k, d denotes the dimensionality of $\mathbf{h}_v^k$, and $\mathcal{N}_{\text{in}}(v)$ and $\mathcal{N}_{\text{out}}(v)$ denote the in-degree and out-degree neighbors of node v respectively. ϵ^k is a learnable scalar. σ is a ReLU function.

After K DIRGIN convolution layers, a pooling operation is applied:

$$\mathbf{h}_G^s = \text{Pooling}\left(\left\{\mathbf{h}_v^{(k)}\right\}_{v \in G}\right) \tag{6}$$

where $\mathbf{h}_G^s \in \mathbb{R}^d$ is the hidden features of the directional flow graph G.

Classifier. We send $\mathbf{h}_G^s$ to a classifier composed of two fully connected layers for binary classification.

$$\hat{\mathbf{y}} = \text{softmax}(\sigma(\mathbf{h}_G^s \mathbf{W}_1)\mathbf{W}_2) \tag{7}$$

where $\mathbf{W}_1$ and $\mathbf{W}_2$ are parameters of the classifier, σ is the ReLU function. The loss function is devised to minimize the cross-entropy value:

$$\mathcal{L}_s = -\mathbf{y} \log(\hat{\mathbf{y}}) - (1 - \mathbf{y}) \log(1 - \hat{\mathbf{y}}) \tag{8}$$

where $\mathbf{y}$ is the ground truth, with 1 representing malicious traffic and 0 representing benign traffic.

3.4 Unsupervised Learning Stage

The core idea of unsupervised training is to detect unknown anomalies by learning the structural patterns of benign traffic. Specifically, the model learns to reconstruct benign directed graphs based on encoded features. When inputting malicious traffic, the model, trained only on benign data, struggles to accurately reconstruct the graph, resulting in a higher reconstruction loss. The unsupervised model consists of an encoder and a decoder: the encoder is shared with the supervised setting, while the decoder is a fully connected layer that predicts edge probabilities between nodes.

DIRGIN Encoder. The encoder is identical to that used in the supervised learning stage, with its parameters initialized from the pretrained supervised encoder. The directional flow graph G is then encoded into a representation $\mathbf{h}_G^u$ by stacking K layers of DIRGIN convolution.

Decoder. Benign and malicious directed graphs exhibit distinct structural patterns, making graph reconstruction an effective signal for anomaly detection. To reconstruct the directional flow graph, we utilize a decoder to predict the existence of edges based on node embeddings. Given node representations $\mathbf{H} = [\mathbf{h}_1^u, \mathbf{h}_2^u, \ldots, \mathbf{h}_m^u]$, the edge score between nodes (u, v) is computed as:

$$\mathbf{s}_{uv} = [\mathbf{h}_u^u \| \mathbf{h}_v^u]\mathbf{W}_u \tag{9}$$

where $[\cdot \parallel \cdot]$ denotes concatenation and $\mathbf{W}_u$ is a trainable weight vector. The scores $\mathbf{s}_{uv}$ are used as logits for a binary classification task (i.e., whether an edge exists or not).

To help the decoder capture meaningful structural patterns, we apply negative sampling by choosing node pairs that are distant in the embedding space and not connected in the original graph. The number of negative samples is kept equal to that of positive edges to maintain balance.

Given positive edges $\mathcal{E}_{\text{pos}}$ and hard negative samples $\mathcal{E}_{\text{neg}}$, the decoder is optimized using a binary cross-entropy (BCE) loss over both positive and negative edge logits:

$$\mathcal{L}_{\text{u}} = \mathbb{E}_{(u,v)\in\mathcal{E}_{\text{pos}}}\left[\text{BCE}(\mathbf{s}_{uv}, 1)\right] + \mathbb{E}_{(u,v)\in\mathcal{E}_{\text{neg}}}\left[\text{BCE}(\mathbf{s}_{uv}, 0)\right] \tag{10}$$

3.5 Inference and Threshold Selection

During inference, we combine the outputs of the supervised classifier and the unsupervised reconstruction module to make the final prediction. This hybrid inference allows the model to identify suspicious, previously unseen attacks that a purely supervised model might misclassify as benign.

Supervised Prediction. Using the trained DIRGIN encoder and classifier in the supervised learning stage, we obtain a coarse prediction:

$$\hat{\mathbf{y}}^s = \arg\max(\text{Classifier}(\mathbf{h}_G^s)) \tag{11}$$

Unsupervised Prediction. We forward the directional flow graph to the whole model trained in the unsupervised learning stage to obtain edge-level reconstruction scores.

$$\{\mathbf{s}_{\text{pos}}, \mathbf{s}_{\text{neg}}\} = \text{Decoder}(\mathbf{h}_G^u, \mathcal{E}_{\text{pos}}, \mathcal{E}_{\text{neg}}) \tag{12}$$

For each graph G in the batch, we compute its per-graph reconstruction loss:

$$\ell^u = \frac{1}{2}\left(\text{BCE}(\mathbf{s}_{\text{pos}}, 1) + \text{BCE}(\mathbf{s}_{\text{neg}}, 0)\right) \tag{13}$$

Final Decision. The final prediction $\hat{\mathbf{y}}'$ is decided as:

$$\hat{\mathbf{y}}' = \begin{cases} 1 & \text{if } \hat{\mathbf{y}}^s = 1 \\ 1 & \text{if } \hat{\mathbf{y}}^s = 0 \text{ and } \ell^u > \delta \\ 0 & \text{if } \hat{\mathbf{y}}^s = 0 \text{ and } \ell^u < \delta \end{cases} \tag{14}$$

where δ is the reconstruction threshold.

To enable auto-tuning, we estimate the reconstruction threshold δ in a data-driven and unsupervised manner using KDE on the reconstruction loss distribution observed in the validation set. Let $\ell_1, \ldots, \ell_B$ denote the validation reconstruction losses. We fit a kernel density estimator $f(\cdot)$ and identify peaks (local maxima) and valleys (local minima) on the density curve. If two peaks are detected, we select the first valley between the first two peaks as the threshold:

$$\delta = \underset{\ell\in(\ell_{\text{peak}_1}, \ell_{\text{peak}_2})}{\arg\min} f(\ell) \tag{15}$$

This threshold effectively separates the bimodal distribution of benign and malicious reconstruction losses. If a clear bimodal pattern is not detected (e.g., only one peak or no valleys), we fall back to a high-percentile heuristic, such as the 95th percentile:

$$\delta = \text{Percentile}_{95}(\ell_1, \ldots, \ell_B) \tag{16}$$

Table 1: The statistics of two datasets.

Dataset		Train		Valid	Test	Total
		Supervised	Unsupervised			
USTC-TFC2016	Benign	108,782	82,000	21,199	90,850	302,831
	Malicious (I)	108,763	0	12,085	51,792	172,640
	Malicious (II)	107,345	0	11,928	53,367	172,640
CICIoT2024	Benign	370,000	156,793	58,533	250,854	836,180
	Malicious (I)	373667	0	41,519	177,937	593,123
	Malicious (II)	335,380	0	37,265	220,478	593,123

Table 2: Reorganization details of malicious traffic of the two datasets.

Dataset		Train	Test
USTC-TFC2016	I	Cridex, Geodo, Htbot, Miuref, Neris, Nsis-ay, Shifu, Tinba, Virut, Zeus	
	II	Cridex, Geodo, Htbot, Miuref, Neris, Zeus	Nsis-ay, Shifu, Tinba, Virut
CICIoT2024	I	DoS, BruteForce, Recon, Web-based, Spoofing, DDoS, Mirai	
	II	DDoS, BruteForce, Recon, Mirai	Web-based, Spoofing, DoS

4 Experiments

4.1 Experiment Settings

Dataset. To evaluate DGNet, we conduct experiments on two public datasets: USTC-TFC2016 [24] and CICIoT2024 [17]. Due to the limited proportion of benign traffic in CICIoT2024, benign samples from CICIoT2022 [3] are incorporated to improve training stability and generalization. Both datasets are reorganized to construct known and unknown malicious traffic scenarios. For known attacks, all traffic is randomly mixed and split into training and test sets according to the ratio of 7:3, with 10% of the training set as a validation set, yielding USTC-TFC2016-I and CICIoT2024-I. For unknown attacks, different attack types are separated between training and testing sets to ensure no overlap, resulting in USTC-TFC2016-II and CICIoT2024-II. Dataset statistics are summarized in Tables 1 and 2.

Implementation Details. In the hybrid learning, the batch size is 150, and the total number of steps is 20 and 100, respectively. We apply early stopping when the validation loss stops decreasing for 5 epochs. We set the learning rate to 5e-4 with the Adam optimizer. The dimensions of hidden features are 128. The number of layers in DIRGIN is 2. The minimum and maximum nodes in each directional flow graph are 2 and 100, respectively.

4.2 Performance Comparison

To thoroughly evaluate the performance of DGNet, we establish seven baseline methods for comparison, including supervised methods (*FS-Net* [13], *Graph-DAPP* [18], *Graph2Vec* [10]), *FG-SAT* [2], an unsupervised method (*Kitsune* [15]),

Table 3: Performance comparison on known attacks on the two datasets.

Method	USTC-TFC2016-I				CICIoT2024-I			
	Acc	F1	Precision	Recall	Acc	F1	Precision	Recall
FSNet	0.9966	0.9964	0.9960	0.9938	0.9689	0.9676	0.9695	0.9660
GraphDAPP	0.9697	0.9669	0.9729	0.9618	0.7167	0.6741	0.7445	0.6747
Graph2Vec	0.8851	0.8531	0.9237	0.8255	0.8233	0.6942	0.9033	0.6646
Kitsune	0.8331	0.8061	0.8458	0.7901	0.7746	0.7723	0.7721	0.7798
IoTArgos	0.9732	0.9657	0.9788	0.9714	0.9423	0.9383	0.9454	0.9411
CVAE-EVT	0.9962	0.9958	0.9956	0.9960	0.9626	0.9602	0.9504	0.9677
ECNet	0.9937	0.9932	0.9925	0.9938	0.9416	0.9396	0.9418	0.9376
FG-SAT	0.9969	0.9967	**0.9972**	0.9962	0.9607	0.9592	0.9644	0.9552
DGNet(Mean Pool)	**0.9975**	**0.9973**	0.9967	**0.9979**	0.9714	0.9704	0.9716	0.9694
DGNet(Add Pool)	0.9959	0.9956	0.9957	0.9955	0.9366	0.9387	0.9307	0.9341
DGNet(Max Pool)	0.9926	0.9920	0.9917	0.9924	**0.9739**	**0.9741**	**0.9722**	**0.9731**

Table 4: Performance comparison on unknown attacks on the two datasets.

Method	USTC-TFC2016-II				CICIoT2024-II			
	Acc	F1	Precision	Recall	Acc	F1	Precision	Recall
FSNet	0.9938	0.9934	0.9924	0.9944	0.9330	0.9311	0.9434	0.9258
GraphDAPP	0.9209	0.9110	0.9406	0.8948	0.6157	0.5574	0.6678	0.5948
Graph2Vec	0.9372	0.9239	0.9575	0.9027	0.9023	0.8674	0.9394	0.8328
Kitsune	0.7906	0.7593	0.7931	0.7473	0.8051	0.8051	0.8096	0.8090
IoTArgos	0.9745	0.9677	0.9797	0.9730	0.9357	0.9352	0.9368	0.9356
CVAE-EVT	0.9935	0.9930	0.9925	0.9936	0.9457	0.9420	0.9375	0.9474
ECNet	0.9964	0.9961	0.9969	0.9954	0.8613	0.8570	0.8825	0.8537
FG-SAT	0.9945	0.9941	0.9955	0.9927	0.9425	0.9419	0.9474	0.9397
DGNet(Mean Pool)	**0.9976**	**0.9974**	**0.9971**	**0.9977**	0.9581	0.9579	0.9594	0.9569
DGNet(Add Pool)	0.9975	0.9973	0.9971	0.9976	0.9228	0.9218	0.9280	0.9195
DGNet(Max Pool)	0.9961	0.9958	0.9948	0.9969	**0.9646**	**0.9644**	**0.9661**	**0.9633**

a hybrid method (*IoTArgos* [23]), and other methods specifically focused on robust malicious traffic detection (*CVAE-EVT* [26], *ECNet* [9]). We also compared the impact of different pooling methods on the performance of the entire model after the supervised DIRGIN encoder. We conduct experiments on the USTC-TFC2016 and CICIoT2024 for detecting known and unknown malicious traffic tasks. The comparison results are shown in Tables 3 and 4, respectively.

Known Malicious Traffic Detection. According to Table 3, several conclusions can be drawn. First, DGNet achieved the best results in almost all metrics. Specifically, the best results were achieved when the pooling methods used were mean pooling on the USTC-TFC2016 dataset and max pooling on the CICIoT2024 dataset, respectively. Second, supervised learning methods are generally much better than unsupervised learning methods. Compared with the traditional machine learning method IoTArgos, our hybrid method is based on

deep learning and has dramatically improved performance. Third, GraphDAPP, Graph2Vec, and FG-SAT also detect malicious traffic based on graphs, but the overall performance is worse than that of DGNet. In addition to the reason that we added unsupervised learning, it is also related to their neglect of the packet temporal information. FSNet is a model that captures packet sequence features, and its performance is second only to our model, indicating that the packet sequence pattern feature is crucial for distinguishing between benign and malicious traffic. Fourth, compared to CVAE-EVT and ECNet, models that also focus on detecting both known and unknown malicious traffic, our model performs better, especially on the CICIoT2024 dataset. ECNet limits the burst size to 3 and lacks the ability to distinguish benign and malicious traffic according to burst mode, which may be the reason for its poor performance.

Unknown Malicious Traffic Detection. Detecting unknown malicious traffic is more challenging than detecting known attacks due to the higher requirement for model generalization. Experiments on USTC-TFC2016-II and CICIoT2024-II evaluate model robustness under increasing distribution shifts between training and testing data (Table 4). Consistent with Table 3, most methods perform worse on CICIoT2024-II than on CICIoT2024-I, a trend less evident on USTC-TFC2016 due to its smaller scale and lower complexity. When facing unknown attacks, supervised methods generally degrade, as they struggle to identify unseen patterns, whereas unsupervised approaches such as Kitsune improve. Graph2Vec benefits from unsupervised graph embedding, yielding more robust representations, while CVAE-EVT maintains stable performance due to its generative latent-variable modeling. By integrating supervised and unsupervised learning, our method shows only a slight performance drop in unknown malicious traffic detection.

4.3 Complexity Analysis

To evaluate the trade-off between performance and complexity, we present model parameters and per-session processing times in Table 5. Our findings indicate that the computational complexity of DGNet is moderate. The time overhead of DGNet is higher than that of supervised methods (FSNet, GraphDAPP, Graph2Vec, CVAE-EVT, ECNet, and FG-SAT) and lower than that of the unsupervised method Kitsune. While the inference process of our hybrid framework takes longer than a single inference process, it significantly improves performance with only a modest increase in runtime.

4.4 Ablation Study

To quantify the individual contribution of each component in DGNet, we perform ablation experiments, and the results are shown in Table 6. Both supervised and unsupervised learning play vital roles. Supervised learning generally outperforms unsupervised learning when labels are sufficient and data distributions are

Table 5: Model parameters and time overhead.

Method	FSNet	GraphDAPP	Graph2Vec	Kitsune	IoTArgos	CVAE-EVT	ECNet	FG-SAT	DGNet
Parameters	4.11e+05	9.21e+03	/	/	/	3.78e+03	3.66e+04	8.89e+03	4.63e+04
Times (ms)	1.06e-02	2.37e-02	5.32e-02	4.06e-01	6.27e-03	1.08e-02	1.49e-02	2.64e-02	3.16e-01

"/" indicates that the number is either non-existent or inaccessible.

Table 6: Ablation study. For convenience, we denote using only supervised learning and unsupervised learning as 'w/ Super' and 'w/ Unsuper'.

Scenario	Method	USTC-TFC2016		CICIoT2024	
		Acc	F1	Acc	F1
Known	DGNet	0.9975	0.9973	0.9739	0.9741
	w/ Super	0.9741	0.9716	0.9668	0.9658
	w/ Unsuper	0.9544	0.9493	0.8703	0.8575
Unknown	DGNet	0.9976	0.9974	0.9646	0.9644
	w/ Super	0.9355	0.9281	0.9379	0.9374
	w/ Unsuper	0.9289	0.9202	0.9258	0.9243

stable, as it provides clear objectives and learns more discriminative features. In contrast, unsupervised learning, with its vague objectives, may produce nondiscriminative features and higher false alarms, but it excels at detecting zero-day attacks since it does not rely on known categories. In known attack scenarios, supervised learning performs significantly better, while in unknown attack scenarios, their performance is comparable. Our DGNet integrates the advantages of supervised learning and unsupervised learning. In both malicious attack scenarios, the false positives and missed samples generated by the two types of learning are correctly detected through their complementary advantages.

4.5 Impact of GNN Backbone

To evaluate the impact of different graph neural networks on our framework, we experiment by replacing the DIRGIN encoder in our model with other popular GNN variants: GIN, Graph Convolutional Network (GCN) [11] and Graph Attention Network (GAT) [22]. All other components of the framework remain unchanged, which ensures that performance differences can be attributed solely to the choice of GNN. These models all apply max pooling on the CICIoT2024 dataset.

As shown in Table 7, the best detection performance is consistently achieved using DIRGIN on all metrics, followed closely by GIN. GIN has higher representational capabilities in capturing structural patterns. Our designed DIRGIN captures the temporal features of traffic sessions, which are crucial for distinguishing subtle differences in the directed graphs of benign and malicious traffic. GAT performs moderately but consistently lags behind GIN. GCN has the weakest performance, which may be due to the fact that benign and malicious traffic still share some similarities in the session graph structure. When the number of

Table 7: Ablation study of GNN backbone in DGNet on the two datasets.

Scenario	Method	USTC-TFC2016		CICIoT2024	
		Acc	F1	Acc	F1
Known	DGNet_{DIRGIN}	0.9975	0.9973	0.9739	0.9741
	DGNet_{GIN}	0.9862	0.9851	0.8957	0.8933
	DGNet_{GAT}	0.9617	0.9592	0.8972	0.8948
	DGNet_{GCN}	0.9454	0.9397	0.8544	0.8505
Unknown	DGNet_{DIRGIN}	0.9976	0.9974	0.9646	0.9644
	DGNet_{GIN}	0.9936	0.9931	0.8548	0.8518
	DGNet_{GAT}	0.9645	0.9625	0.7636	0.7620
	DGNet_{GCN}	0.9216	0.9150	0.8807	0.8798

nodes is small, such as only 2 or 3 nodes, the differences become very subtle, especially in terms of node characteristics. At this point, GCN is unable to capture these slight differences due to its overly simple message passing approach.

4.6 Missed Detection Recovery Analysis

Supervised learning methods, trained on labeled data, are generally effective in detecting known malicious traffic. However, when facing unseen attacks or evasion tactics by adversaries, these models often lack generalization capability, leading to misclassification of malicious traffic as benign, i.e., false negatives. This limitation motivates our hybrid detection framework, which integrates unsupervised learning to complement supervised models and reduce the number of false negatives. Figure 5 shows the experimental results on the USTC-TFC2016 testing set. Figures 5a and 5b compare the confusion matrices from the supervised output and the final output under both known and unknown attack scenarios, respectively. Many malicious samples misclassified as benign in the supervised stage are correctly identified after correction by the unsupervised module, significantly reducing the false negative rate.

Figures 5c and 5d present the reconstruction loss distributions of false negative (FN) samples and true positive (TP) samples that were initially misclassified but recovered by the second stage. TP samples are randomly selected to match the number of FN samples. The results show that the corrected malicious samples exhibit higher reconstruction losses, making them easier to detect. This demonstrates that the unsupervised module plays a critical role in recovering missed detections and substantially enhances overall performance.

5 Related Work

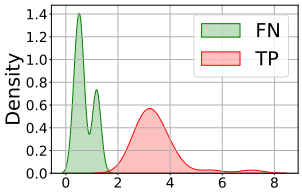
Existing malicious traffic detection methods can be broadly categorized into supervised, unsupervised, and hybrid approaches. Supervised methods include traditional machine learning techniques that rely on handcrafted traffic features and classifiers such as SVM and random forest, as well as deep learning models

	Predicted Label	
True Label	Super	1	0
	1	48426	3366
	0	323	90527
	Final	1	0
	1	51760↑	32↓
	0	323	90527

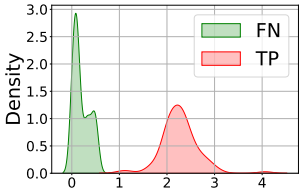
(a) First-stage and final classification confusion matrices for known attack scenarios.

	Predicted Label	
True Label	Super	1	0
	1	44330	9037
	0	256	90594
	Final	1	0
	1	53273↑	94↓
	0	256	90594

(b) First-stage and final classification confusion matrices for unknown attack scenarios.



(c) Reconstruction loss distribution of FN and TP samples corresponding to (a).



(d) Reconstruction loss distribution of FN and TP samples corresponding to (b).

Fig. 5: The confusion matrix of the two-stage detection in the USTC-TFC2016 testing set and the final false negative and true positive sample reconstruction loss distribution.

(e.g., CNNs, RNNs, GNNs, and Transformer-based models) that capture spatial, temporal, and topological patterns through end-to-end learning [1, 12, 7, 14, 28–30]. While these methods achieve strong performance on known attacks, they generalize poorly to unseen threats. Unsupervised methods, which do not require labeled data, are typically based on clustering-based outlier detection or self-supervised representation learning, enabling limited detection of unknown attacks at the cost of lower overall accuracy [4, 6, 15, 8]. Hybrid approaches enhance robustness by combining supervised and unsupervised learning or incorporating generative models and confidence estimation [26, 9, 23, 16], but they often depend on labeled data or manual feature engineering. In contrast, our method integrates supervised and unsupervised learning in an end-to-end framework to effectively detect both known and unknown attacks.

6 Conclusion

This paper presents DGNet, a hybrid directional graph-based method for malicious traffic detection that integrates supervised and unsupervised learning. It builds a directional flow graph from bidirectional traffic flows, capturing the structural and temporal characteristics of packet interactions. Each directional flow graph is encoded using DIRGIN, our custom-designed directed graph neural network, and subsequently processed by a classifier and a decoder in the

supervised and unsupervised learning processes, respectively. By combining the strengths of the hybrid approach, DGNet effectively identifies known and unknown attacks. We evaluate DGNet on four reorganized datasets, and results show that it consistently outperforms existing state-of-the-art methods in detecting both known and unknown malicious traffic.

References

1. Anderson, B., McGrew, D.: Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity. In: Proceedings of the 23rd ACM SIGKDD International Conference on knowledge discovery and data mining. pp. 1723–1732 (2017)
2. Cui, S., Han, X., Han, D., Wang, Z., Wang, W., Jiang, B., Liu, B., Lu, Z.: Fg-sat: Efficient flow graph for encrypted traffic classification under environment shifts. IEEE Transactions on Information Forensics and Security **20**, 5326–5339 (2025). https://doi.org/10.1109/TIFS.2025.3571663
3. Dadkhah, S., Mahdikhani, H., Danso, P.K., Zohourian, A., Truong, K.A., Ghorbani, A.A.: Towards the development of a realistic multidimensional iot profiling dataset. In: 2022 19th Annual International Conference on Privacy, Security Trust (PST). pp. 1–11 (2022). https://doi.org/10.1109/PST55820.2022.9851966
4. Fu, C., Li, Q., Shen, M., Xu, K.: Realtime robust malicious traffic detection via frequency domain analysis. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. pp. 3431–3446 (2021)
5. Fu, C., Li, Q., Xu, K.: Detecting unknown encrypted malicious traffic in real time via flow interaction graph analysis. arXiv preprint arXiv:2301.13686 (2023)
6. Gao, L., Fu, C., Deng, X., Xu, K., Li, Q.: Wedjat: Detecting sophisticated evasion attacks via real-time causal analysis. In: Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1. pp. 342–353 (2025)
7. Gu, J., Lu, S.: An effective intrusion detection approach using svm with naïve bayes feature embedding. Computers & Security **103**, 102158 (2021)
8. Han, X., Cui, S., Qin, J., Liu, S., Jiang, B., Dong, C., Lu, Z., Liu, B.: Contramtd: An unsupervised malicious network traffic detection method based on contrastive learning. In: Proceedings of the ACM Web Conference 2024. pp. 1680–1689 (2024)
9. Han, X., Liu, S., Liu, J., Jiang, B., Lu, Z., Liu, B.: Ecnet: Robust malicious network traffic detection with multi-view feature and confidence mechanism. IEEE Transactions on Information Forensics and Security (2024)
10. Hu, X., Gao, W., Cheng, G., Li, R., Zhou, Y., Wu, H.: Toward early and accurate network intrusion detection using graph embedding. IEEE Transactions on Information Forensics and Security **18**, 5817–5831 (2023)
11. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
12. Li, X., Chen, W., Zhang, Q., Wu, L.: Building auto-encoder intrusion detection system based on random forest feature selection. Computers & Security **95**, 101851 (2020)
13. Liu, C., He, L., Xiong, G., Cao, Z., Li, Z.: Fs-net: A flow sequence network for encrypted traffic classification. In: IEEE INFOCOM 2019-IEEE Conference On Computer Communications. pp. 1171–1179. IEEE (2019)
14. Liu, Y., Wang, X., Qu, B., Zhao, F.: Atvitsc: A novel encrypted traffic classification method based on deep learning. IEEE Transactions on Information Forensics and Security (2024)

15. Mirsky, Y., Doitshman, T., Elovici, Y., Shabtai, A.: Kitsune: an ensemble of autoencoders for online network intrusion detection. arXiv preprint arXiv:1802.09089 (2018)
16. Qiu, Z., Zhou, D., Zhai, Y., Liu, B., He, L., Cao, J.: Vaemax: Open-set intrusion detection based on openmax and variational autoencoder. In: 2024 5th Information Communication Technologies Conference (ICTC). pp. 98–105. IEEE (2024)
17. Rabbani, M., Gui, J., Nejati, F., Zhou, Z., Kaniyamattam, A., Mirani, M., Piya, G., Opushnyev, I., Lu, R., Ghorbani, A.A.: Device identification and anomaly detection in iot environments. IEEE Internet of Things Journal (2024)
18. Shen, M., Zhang, J., Zhu, L., Xu, K., Du, X.: Accurate decentralized application identification via encrypted traffic analysis using graph neural networks. IEEE Transactions on Information Forensics and Security **16**, 2367–2380 (2021)
19. Tang, R., Yang, Z., Li, Z., Meng, W., Wang, H., Li, Q., Sun, Y., Pei, D., Wei, T., Xu, Y., et al.: Zerowall: Detecting zero-day web attacks through encoder-decoder recurrent neural networks. In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications. pp. 2479–2488. IEEE (2020)
20. Tang, W., Fu, S., Ke, Y., Peng, Q., Gao, F.: Themis: Fair memory subsystem resource sharing with differentiated qos in public clouds. In: Proceedings of the 51st International Conference on Parallel Processing. ICPP '22
21. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
22. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
23. Wan, Y., Xu, K., Xue, G., Wang, F.: Iotargos: A multi-layer security monitoring system for internet-of-things in smart homes. In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications. pp. 874–883. IEEE (2020)
24. Wang, W., Zhu, M., Zeng, X., Ye, X., Sheng, Y.: Malware traffic classification using convolutional neural network for representation learning. In: 2017 International conference on information networking (ICOIN). pp. 712–717. IEEE (2017)
25. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? arXiv preprint arXiv:1810.00826 (2018)
26. Yang, J., Chen, X., Chen, S., Jiang, X., Tan, X.: Conditional variational autoencoder and extreme value theory aided two-stage learning approach for intelligent fine-grained known/unknown intrusion detection. IEEE Transactions on Information Forensics and Security **16**, 3538–3553 (2021)
27. 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. pp. 2066–2075 (2023)
28. Zhang, H., Yue, H., Xiao, X., Yu, L., Li, Q., Ling, Z., Zhang, Y.: Revolutionizing encrypted traffic classification with mh-net: A multi-view heterogeneous graph model. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 39, pp. 1048–1056 (2025)
29. Zhao, R., Zhan, M., Deng, X., Wang, Y., Wang, Y., Gui, G., Xue, Z.: Yet another traffic classifier: A masked autoencoder based traffic transformer with multi-level flow representation. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 5420–5427 (2023)
30. Zhou, G., Guo, X., Liu, Z., Li, T., Li, Q., Xu, K.: Trafficformer: an efficient pretrained model for traffic data. In: 2025 IEEE Symposium on Security and Privacy (SP). pp. 1844–1860. IEEE (2025)