MDPI

Article

Adversarial Hierarchical-Aware Edge Attention Learning Method for Network Intrusion Detection

Hao Yan [1,2], Jianming Li [1,2], Lei Du [1,2], Binxing Fang [1,2], Yan Jia [3], Zhaoquan Gu[1,2]*

[1] School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, Guangdong, China; 22b951028@stu.hit.edu.cn
[2] Department of New Networks, Peng Cheng Laboratory, Shenzhen, Guangdong, China;
[3] National University of Defense Technology, Changsha, Hunan, China;
* Correspondence: guzhaoquan@hit.edu.cn; Zhaoquan Gu is corresponding author.

Abstract: The rapid development of information technology has made cyberspace security even more serious. Network intrusion detection method is a practical scheme to protect network systems from cyberattacks. However, as network flows have natural topological relationships and anomaly detection cannot simultaneously detect the attack types, the existing detection methods cannot handle these challenges in the real world. To handle these problems, we propose a graph neural network based network intrusion detection method called Adversarial Hierarchical-Aware Edge Attention Learning Method (AH-EAT for short), which can utilize computer networks naturally exhibiting a graph structure with robust and multi-grained intrusion detection ability. AH-EAT includes an edge-based graph attention mechanism embedding module, a hierarchical multi-grained detection module and an adversarial training module. The edge-based graph attention mechanism embedding module adopts graph attention networks to aggregate node and edge features according to their importance, effectively capturing key topological structural information. Moreover, unlike traditional anomaly detection methods, the hierarchical multi-grained detection module uses coarse-grained detection to distinguish malicious and benign flow, while fine-grained classification identifies specific attack types simultaneously. Specifically, the adversarial training module uses projected gradient descent to generate adversarial edge perturbations during training, which improves robustness and resilience to evasion attacks. We conduct extensive experiments on four popular network intrusion detection benchmark datasets and the results demonstrate that AH-EAT outperforms state-of-the-art methods, achieving higher accuracy in multi-grained detection and robustness in both standard and adversarial scenarios.

Keywords: Graph Neural Networks, Network Intrusion Detection System, Hierarchical Intrusion Detection, Adversarial Robustness, Edge Attention Network

Received:
Revised:
Accepted:
Published:

Citation: . . *Journal Not Specified* **2025**, *1*, 0. https://doi.org/

1. Introduction

With the emergence of innovative information technology, the increasing frequency and complexity of attacks on computer networks continue to threaten the security of information within computer systems. The rapid evolution of cyber threats necessitates advanced intrusion detection methods that are able to model complex network interactions. Cyber attacks significantly threaten the assets and data security of individuals, enterprises, and even the country [1,2]. In order to combat this, enterprises utilize network intrusion detection systems (NIDSs) to protect critical infrastructure, data, and networks.

Network intrusion detection is a practical scheme for detecting attacks. Traditionally, NIDSs utilize rule-based methods to detect intrusions. The detection rule is generated from

attack samples [3,4]. [3] leverages both benign and malicious flow samples to generate detection rules. Meanwhile, [4] employs Bayesian networks to learn attack patterns and convert them into "IF-THEN" detection rules. These methods can detect attack types, but they cannot handle the interaction and variants between attack flows. In fact, attackers often use various techniques to conceal their attack behaviors to evade rule detection, making attack detection more difficult. Recently, there has been great progress in the application of new learning-based models for the development of new solutions [5,6]. Learning-based methods, on the other hand, rely on more complex operations, commonly involving machine learning (ML) to identify sophisticated attacks [7–9]. However, these approaches have generally barely considered the topological patterns of both benign and attack network flow, which is an important factor in network intrusion detection. NIDSs often fail to capture interdependencies between network entities, resulting in high false positive rates and limited interpretability. Therefore, we advocate using the network flow topology patterns to detect complex attacks such as Advanced Persistent Threat (APT) because APT attacks need to consider both the overall network graph topology patterns and lateral movement paths.

In order to explore topology patterns, the graph neural networks (GNNs) [10] is a potential method in deep learning. GNNs take advantage of the graph structures through the use of message passing between nodes or edges. This enables the neural network to learn and generalize effectively on graph-based data to output low-dimensional vector representations [11]. For instance, studies such as [12–14] have demonstrated the potential of GNNs in identifying attack patterns. However, despite the progress made by these methods, significant limitations remain when dealing with the importance of structure and detection grains.

Based on detection grains, the existing methods of detecting cyber attacks can be divided into two categories: anomaly-based binary classification [7,9,13] and fine-grained multi-classification [12,15,16]. Anomaly detection methods use benign samples for modeling, and any behavior patterns that deviate from expectations are identified as anomalies. For example, regularization and autoencoders are introduced in [7] to improve potential representations for anomaly detection. OnlineBPCA is proposed in [9], which converts network flow into a two-dimensional tensor and analyzes the changes of principal directions for anomaly detection. However, anomaly detection is rarely deployed in realistic situations due to the inability to provide detailed diagnostic information about attack types and the high false positive rate. Fine-grained detection methods are trained and evaluated using supervised learning based on predefined classes. In the study [12], the embedded representation of network flow is obtained by splicing the aggregated information of nodes. However, these methods require training from the scratch and pre-defined classes. In a realistic network, attack classes sometimes only need to know benign and malicious, and sometimes need to know the specific attack category. If you train from scratch every time, it leads to a lot of computing power and time wasted.

In recent years, adversarial attacks [17–19] have emerged in several research fields, such as in computer vision [18] and natural language processing [19]. Attackers can induce deep learning models to make incorrect predictions by injecting tiny perturbations that are imperceptible to the human eye into the input data. We envision that when dealing with real network intrusion detection, network attacks change their features to evade existing detection rules to evade intrusion detection methods. For example, obfuscation techniques have been used to generate variants from known attacks that evade rule-based detection methods [20]. Attackers often use techniques such as protocol camouflage (such as DNS covert tunnels) and flow feature obfuscation (such as modifying TCP window size) to construct adversarial network flow in order to evade detection systems. Such attacks can

be regarded as edge feature perturbations on graph-structured data. Their goal is to modify the statistical features or protocol semantics of flow interactions so that malicious flow is misclassified as benign in the GNN model. Existing GNN-based intrusion detection methods (such as [12,13]) generally have adversarial vulnerabilities. In the model training process, these methods [12,13] assume that edge features are fixed and ignore the escape behavior of attackers' actively perturbing features.

Overall, to propose a robust and multi-grained GNN intrusion detection method, three key challenges must be addressed: (1) Existing methods do not distinguish the importance of nodes and edges in aggregation and message passing, using static aggregation strategies (such as mean/max pooling). So they can not adaptively distinguish key attack paths from noise edges. (2) Most studies only focus on anomaly classification and lack a hierarchical detection framework to balance efficiency (coarse-grained screening) and accuracy (fine-grained classification). (3) Traditional intrusion detection methods do not consider the specificity of graph structure perturbations, making the model vulnerable to topology-aware escape attacks.

In this paper, we propose an adversarial hierarchical-aware edge attention learning method for network intrusion detection, named AH-EAT, which systematically addresses the aforementioned challenges through three core mechanisms. The proposed method is able to learn an edge feature representation and exactly classify this edge into coarse-grained and fine-grained types with robustness against adversarial attacks. In detail, the method consists of three modules: an edge-based graph attention mechanism embedding module, a hierarchical multi-grained detection module and an adversarial training module. The edge-based graph attention mechanism embedding module dynamically weights node-neighbor interactions by integrating edge features (e.g., flow protocols, interaction frequency) with attention mechanisms, adaptively distinguishing critical attack paths from noisy edges and capturing structural-semantic patterns vital for attack detection. The hierarchical multi-grained detection module is constructed with a two-stage pipeline: coarse-grained filtering of malicious flow followed by fine-grained classification of specific attack types, balancing detection efficiency and precision to overcome the limitation of single-granularity models. In the adversarial training module, a Projected Gradient Descent (PGD) [21] based adversarial training strategy is proposed, adding edge feature perturbations and graph structure noise during training. This module can enhance the model's robustness against graph-structural perturbations and edge-feature evasion attacks, significantly improving its resilience to evasion tactics in adversarial network environments. The synergistic effect of the three modules enables AH-EAT to efficiently capture deep features of critical attack interactions while maintaining stable detection performance in adversarial environments.

The contribution of this work is summarized as follows:

- Edge-based Graph Attention Network: By integrating edge features and graph attention network mechanisms, AH-EAT dynamically weights node-neighbor interactions, capturing both structural and semantic patterns critical for attack detection.
- Hierarchical Multi-grained Detection: A two-stage framework first filters malicious flows and then classifies specific attack types, addressing the limitations of single-grained models.
- Adversarial Robustness: Leveraging PGD-based adversarial training, AH-EAT mitigates graph-structural perturbations and edge-feature evasion attacks and improves resilience for attack evasion tactics.

We also conduct extensive experiments on four benchmarks and the results demonstrate that AH-EAT significantly outperforms the baselines with significant advantages.

The rest of the paper is organized as follows. Section 2 discusses related works, and Section 3 provides an overview of the relevant background. Our proposed AH-EAT

algorithm is introduced in Section 4. Experimental evaluation results are presented in Section 5. Section 6 makes a conclusion and discusses future research directions.

2. Related Work

2.1. Application of Graph Neural Network in NIDSs

In recent years, GNNs have been widely used in network intrusion detection systems. Zhou et al. [22] introduce a GNN-based framework for end-to-end botnet detection, focusing exclusively on network topological structures rather than incorporating node or edge features. This approach enabled the model to aggregate features across the graph purely based on structural relationships, formulating botnet detection as a binary classification task. Experimental results demonstrate that their GNN models significantly outperform the logistic regression method. Subsequently, traditional graph embedding approaches [23,24] are applied to network intrusion detection by converting network flows into graphs using source-destination IP and port pairings. Methods like DeepWalk (skip-gram) are employed to generate node embeddings for anomaly detection. However, this shortcoming renders them impractical for most real-world NIDSs scenarios. Because real network flow detection cannot only target nodes, but should target the network flow between two nodes.

Therefore, we investigate the edge determination method. E-GraphSAGE [12] introduces an edge feature encoding mechanism using the GraphSAGE [25] framework, which achieves inductive learning of dynamic network flow by aggregating information about neighbor nodes and edges. However, this method adopts a static mean aggregation strategy and fails to distinguish the importance between key attack paths and normal flow interactions, resulting in insufficient detection sensitivity for covert attack patterns (such as APT). Similarly, GNN-IDS [26] proposes a flow-level anomaly detection framework based on graph convolution, but its homogeneous graph assumption oversimplifies the heterogeneity of network entities (such as differences between hosts, protocols, and services). Recent studies attempt to model edge attributes through GNNs [13], but they mainly focus on binary classification (malicious vs. benign flow) and ignore the need for hierarchical identification of fine-grained attack types (such as DDoS, port scanning, and malware propagation).

In summary, the above methods generally have two key problems: firstly, the dynamic fusion mechanism of edge features and topological structures is missing, making it difficult to capture the synergy between edge semantics (such as flow statistics) and topological patterns (such as attack propagation paths) in attack behaviors; secondly, the detection granularity is single, which cannot meet the synergy requirements of coarse-grained screening and fine-grained classification in actual network environments. The AH-EAT framework proposed in this study effectively solves the above problems through attention mechanisms and hierarchical multi-grained detection.

2.2. Adversarial Training in Cybersecurity

Adversarial training has emerged as a critical technique to enhance the robustness of machine learning models against adversarial attacks in cybersecurity. Traditional machine learning models, including deep neural networks (DNNs) and GNNs, are vulnerable to adversarial perturbations that manipulate input data to deceive detection systems.

Wang et al. [14] propose the vulnerability of GNNs under structural perturbations and use a certified robustness framework based on randomized smoothing. This method mathematically proves the prediction stability of GNNs under structural attacks such as node deletion or addition. Its core innovation lies in extending the randomized smoothing technology to the field of graph data. Specifically, by adding random noise to the input graph and smoothing it, the model can theoretically guarantee robustness to structural

perturbations within a certain range. Angion et al. [27] focuses on the problem of robustness degradation during model updating and proposes an adversarial training strategy based on non-regression constraints. This method effectively reduces the phenomenon of "negative flips" by forcing the model to retain robustness to historical adversarial samples when updating the model. The updated model misjudges adversarial samples that were previously correctly classified. Specifically, by introducing the robustness-congruent adversarial training loss function, the model must simultaneously meet the prediction consistency of historical adversarial samples when optimizing new data, so that the model has good results for both post-adversarial samples and historical samples.

In order to obtain a highly robust graph representation learning network for intrusion detection methods. We need to delete/add nodes and edges or perturb features of the graph data of the graph neural network model to achieve adversarial training results. At the same time, the balance between adversarial samples and historical samples should be fully considered in the training objectives, and negative flips should not occur. It is necessary to effectively design the adversarial method and loss function.

In summary, the limitation of existing research is mainly reflected in the vulnerability of GNNs. As adversarial attacks are a practical strategy to improve the robustness of the model, the AH-EAT proposed in this study uses PGD-based adversarial training to inject edge feature perturbations and graph structure noise.

3. Problem Formulation and Preliminaries

Graph learning is a rapidly evolving research field with a wide range of applications, including social and molecular networks. In recent years, GNNs have demonstrated outstanding performance in network attack detection, achieving state-of-the-art results, especially in the area of network intrusion detection [12,13].

3.1. Edge-based Intrusion Detection

While existing GNNs excel at node-level tasks like social network analysis, they face two key limitations in edge-based intrusion detection:

- Edge Feature Neglect: Most GNNs (e.g., GCN [10]) treat edge features as static topological structures, failing to dynamically fuse them with node representations during message passing.
- Adversarial Vulnerability: Traditional GNNs are brittle against topology-aware attacks where adversaries perturb edge features (e.g., packet size, protocol flags) to evade detection.

GNNs extend deep learning to non-Euclidean graph data by propagating messages along topological structures. Formally, given a graph $G = (V, E)$ with node features $X_v \in R_{dn}$ and edge features $X_e \in R_{de}$, the ll-th layer of a GNN can be defined as:

$$\mathbf{h}_v^{(l)} = \phi^{(l)}\left(\mathbf{h}_v^{(l-1)}, \bigoplus_{u \in \mathcal{N}(v)} \psi^{(l)}(\mathbf{h}_u^{(l-1)}, \mathbf{h}e_{uv}{}^{(l-1)})\right)$$

where $\bigoplus$ is a permutation-invariant aggregation operator (e.g., mean, sum), ϕ and ψ are learnable functions.

Meanwhile, network intrusion detection systems (NIDSs) can be naturally formulated as an edge classification problem on network flow graphs. Nodes represent IP addresses with features X_v. Edges are network flows $e_{uv} \in E$, with features $X_e = [packetcount, duration, TLSversion.....]$. The objective is learning a function $f : E \to \mathcal{Y}_e$ which maps edge to labels $\mathcal{Y}_e \in Benign, DDoS, Malware, ...$

Existing GNN-based NIDSs [12] primarily focus on binary edge classification (malicious vs. benign) or multi-class edge classification (different types of attacks) separately, by training two models. Therefore, we aim at hierarchical detection, which means to classify coarse-grained and fine-grained models with one model at the same time.

3.2. GNNs for Graph Edge Classification

GNNs can be treated as a deep learning approach for graph-based data. GNN acts as an extension of convolutional neural networks (CNN) and can be generalized to non-euclidean data. For each node in the graph, this means aggregating the adjacent node features as a new embedding of the current node, which takes into account the adjacent information. Therefore, node embeddings are capable of capturing the topological characteristics of the network structure where the nodes are located, as well as the unique properties of each individual node. Similarly, the embedding process can be applied to edges and graphs. By doing so, we can obtain edge embeddings and graph embeddings, which encode the relevant features of edges and entire graphs respectively.

Since computer networks and the network flow can inherently be depicted in the form of graphs, leveraging GNNs for network intrusion and anomaly detection presents a highly promising method for identifying novel and sophisticated network attacks, like APT (Advanced Persistent Threat) attacks. In this scenario, we form a network flow graph. IP addresses can serve as representations of nodes, while the communication of packets or traffic flow between nodes can be used to represent edges.

Traditional GNNs, such as GCN, GraphSAGE and GAT, have been successfully applied to a wide range of applications. However, these methods mainly focus on applying node features to node classification and graph classification tasks. Generally, the features of the edges in the graph cannot be considered in message passing and aggregation, so edge classification cannot be considered.

3.3. PGD Adversarial Training

Projected Gradient Descent (PGD) [21] is a landmark adversarial training method that enhances model robustness. Deep neural networks are vulnerable to adversarial examples. Input samples are nearly indistinguishable from natural data but are misclassified by the network. The existence of adversarial attacks may be an inherent weakness of deep learning models. Madry et al. [21] studies the adversarial robustness of neural networks from the perspective of robust optimization. What's more, in the field of cybersecurity, in order to avoid being detected, attacks usually obfuscate their behavioral characteristics, making them more similar to normal flow and thus evading network attack detection methods. Therefore, adversarial training is of great importance in cybersecurity.

The core idea is a natural saddle point (min-max) formulation to capture the notion of security against adversarial attacks to enhance the robustness:

$$\min_{\theta} \max_{\|\delta\| * \infty \leq \epsilon} \mathcal{L}(f_\theta(\mathbf{X} + \delta), y)$$

Where δ is the adversarial perturbation, which is bounded by ϵ. $f\theta$ is model parameter and $\mathcal{L}$ is loss function. PGD generates adversarial examples through multi-step iteration.

4. Methodology

AH-EAT integrates a graph attention mechanism and adversarial training to solve the problems encountered by existing graph-based NIDSs, such as the lack of a fusion mechanism of node embedding edge features and topology, single detection granularity, and adversarial vulnerability. Therefore, we aim to leverage edge-level feature information and learning edge embedding by GAT, hierarchical detection and adversarial learning.

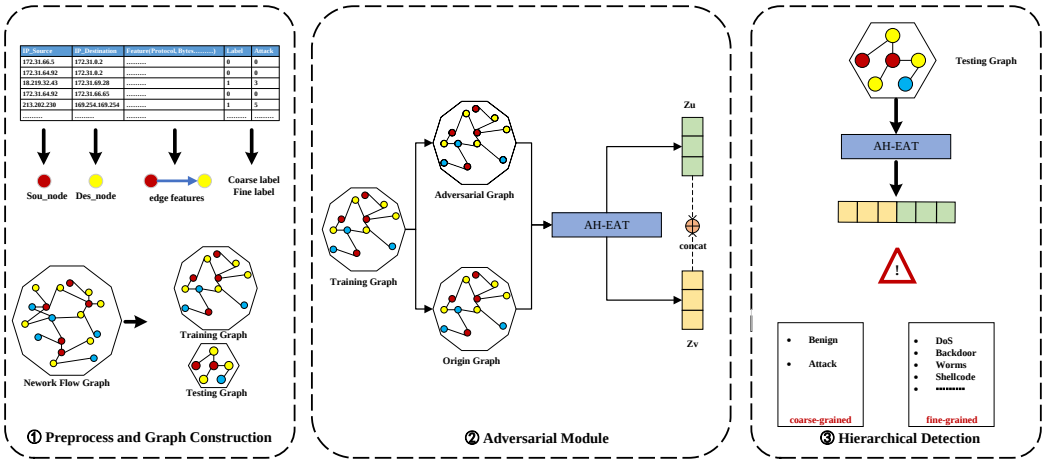

Figure 1. Proposed AH-EAT NIDSs pipeline. A given NIDSs dataset is pre-processed into training and testing graphs (left). The training graph is used to generate an adversarial graph, then train the AH-EAT model (middle). Finally, the testing graph can be detected in coarse-grained and fine-grained (right).

Fig. 1 presents an overview of the anomaly detection process performed using AH-EAT, which comprises three organized modules designed to address multi-grained network intrusion detection. The pipeline initiates with a sophisticated graph construction phase (left panel), where raw network flows transform into network flow graph representations. We use the IP address of each network flow as a node and other features as edge features to construct the graph. At the same time, we construct training and testing graphs while preserving both coarse-grained (e.g., attack categories) and fine-grained labels (e.g., specific attack variants), thus maintaining hierarchical threat intelligence crucial for multi-level detection. Finally, we divide the training graph and test graph according to the defined ratio. The core innovation resides in the adversarial graph generation module (central panel), where we implement a graph adversarial training that injects perturbation patterns constrained by attack semantics. Unlike conventional adversarial methods, our model employs edge feature perturbation and feature masking to simulate sophisticated evasion tactics while preserving the integrity of genuine attack flow signatures. This adversarially augmented graph is subsequently fed into a graph neural network equipped with edge-based attention mechanisms. Afterwards, the embedding vectors obtained from the origin graph and the adversarial graph are concatenated and used as the embedding representation of the edge at the same time, so that the representation features obtained by each network flow training have both the original features and the adversarial perturbation feature information. The final detection phase (right panel) implements a hierarchical classification architecture that synergistically combines coarse-grained anomaly detection with fine-grained label prediction. The system first performs coarse-grained detection to identify fundamental attack categories (Benign and Attack flow), then executes fine-grained classification using multi-scale feature fusion that correlates with global graph properties. This dual-detection approach enables simultaneous macro-pattern (coarse-grained) recognition and micro-anomaly (fine-grained) localization, effectively addressing the challenge of detecting both malicious or not and their broad attack types. The entire process is optimized through adversarial training, ensuring robustness against adversarial attack patterns. This results in a highly robust model that considers the possibility of being applied to actual NIDSs in the future to protect network security.

4.1. Graph Construction

Before training, convert the data into the table format using the raw Netflow format. NetFlow format is an IP flow collection tool that utilizes network elements (such as routers and switches) to collect encountered IP flows and export them to external devices. These IP flows can be defined as a unidirectional sequence of packets encountered on a network device and contain a variety of important network information, including IP addresses, port numbers, number of packets and bytes, and other useful packet statistics. Preprocess the tabular data, including operations such as data standardization. Use IP as a node, and all attributes except IP as edge attributes to construct a graph, and divide it into training graphs and testing graphs. While building the training graph and testing graph, retain the coarse-grained labels and fine-grained labels. So that it is possible to detect whether it is an attack flow and what kind of attack flow it is during the training process.

Here is the process of Netflow format from each dataset into training and testing graphs. First, we split the tabular data into train and test samples. Then, source and destination port information is removed from each flow record. After that, target encoding on categorical features in both the training and testing set. The training set is used to train a target encoder on categorical data. Encoding is then performed again using the fully trained encoder on both the training and testing sets. Meanwhile, any empty or infinite values arising from this process are replaced with a value of 0. Before finally generating graphs, training and testing sets are normalized using an L2 normalization approach. Similarly to the approach taken in target encoding, the normalizer is trained using the training set to guarantee a learning process. Finally, each graph's node features are set to a constant vector containing ones with the same dimensions as those of the edge features, for example, if 10 edge features are used, node features will be vectors of 10 ones.

4.2. Edge Attention Mechanism Embedding

GNNs have been successfully applied in various application domains. However, these approaches mainly focus on node features for message propagation and are currently unable to consider edge features for edge classification. Therefore, we propose an edge attention mechanism that integrates edge-specific semantics into the message-passing process, enabling the model to dynamically weigh node-neighbor interactions while incorporating edge features. The attention mechanism provides a new idea for graph modeling by dynamically assigning node-edge interaction weights. For example, hierarchical Attention Network (HAN) [28] uses predefined meta-paths to capture the semantic information of heterogeneous graphs (such as the "author-paper-conference" relationship in academic networks), but its performance is highly dependent on manually designed meta-paths and is difficult to adapt to the variability and concealment of network attack patterns (such as the unknown propagation path of zero-day attacks).

The core of our approach is a two-stage process: attention coefficient calculation and edge-aware message aggregation, as formalized in Algorithm 1.

$$\mathbf{z}_{uv} \leftarrow \text{CONCAT}(\mathbf{h}_u^{(k-1)}, \mathbf{h}_v^{(k-1)})$$

$$\alpha_{uv} \leftarrow \text{LeakyReLU}(\mathbf{W}_{\text{attn}}\mathbf{z}_{uv})$$

where $\mathbf{h}_u^{(k-1)}$ and $\mathbf{h}_v^{(k-1)}$ are the node features of u and v at the $(k-1)$-th layer. This vector is then fed into a linear layer with a LeakyReLU activation to generate the attention coefficient. $\mathbf{W} * \text{attn} *$ is a weight matrix that parameterizes the attention mechanism, enabling the model to learn the importance of node pairs adaptively.

Algorithm 1 Attention-Based Graph Edge Embedding

Input: Graph $G(V, E)$
Parameter: Node features $\mathbf{H}^{(0)} = \{\mathbf{h}_v^{(0)} \in \mathbf{R}^{d_n}, \forall v \in V\}$
Parameter: Edge features $\mathbf{E} = \{\mathbf{e}_{uv} \in \mathbf{R}^{d_e}, \forall (u, v) \in E\}$
Parameter: Weight matrices $\mathbf{W}_{\text{msg}}$, $\mathbf{W}_{\text{apply}}$, $\mathbf{W}_{\text{attn}}$
Parameter: Nonlinearity σ
Output: Edge embeddings $\mathbf{Z}_{uv}^{(K)}$, Node embeddings $\mathbf{Z}_v$

for layer $k = 1$ **to** K **do**
  Compute attention coefficients:
  for edge $(u, v) \in E$ **do**
    $\mathbf{z}_{uv} \leftarrow \text{CONCAT}(\mathbf{h}_u^{(k-1)}, \mathbf{h}_v^{(k-1)})$
    $\alpha_{uv} \leftarrow \text{LeakyReLU}(\mathbf{W}_{\text{attn}} \mathbf{z}_{uv})$
  end for
  Normalize attention:
  $\alpha_{uv} \leftarrow \text{Softmax}(\{\alpha_{uv} | u \in \mathcal{N}(v)\})$
  Aggregate messages:
  for node $v \in V$ **do**
    $\mathbf{h}_v^{(k)} \leftarrow \sum \alpha_{uv} \cdot (\mathbf{W}_{\text{msg}} \text{CONCAT}(\mathbf{h}_u^{(k-1)}, \mathbf{e}_{uv}))$
  end for
  $\mathbf{z}_v = \mathbf{h}_v^K$
  Update Edge features:
  for node $u, v \in V$ **do**
    $\mathbf{z}_{uv} \leftarrow \text{CONCAT}(\mathbf{z}_u^{(k)}, \mathbf{z}_v^{(k)})$
  end for
end for
return $\mathbf{Z}_{uv}^K$, $\mathbf{Z}_v$

After softmax normalization for neighbor weights, we fuse node and edge features during message propagation, rather than traditional GNNs that ignore edge semantics.

$$\mathbf{h}_v^{(k)} \leftarrow \sum \alpha_{uv} \cdot (\mathbf{W}_{\text{msg}} \text{CONCAT}(\mathbf{h}_u^{(k-1)}, \mathbf{e}_{uv}))$$

This step embeds edge-specific information (e.g., flow protocols, interaction frequency) into the node representation, enhancing the model's ability to distinguish critical attack paths from noisy edges. This formulation explicitly incorporates edge features $\mathbf{e}_{uv}$ during neighborhood aggregation, preserving protocol-specific semantics critical for network intrusion detection.

Finally, K layers of message-passing, edge embeddings are generated by concatenating the features of their incident nodes.

$$\mathbf{z}_{uv} \leftarrow \text{CONCAT}(\mathbf{z}_u^{(k)}, \mathbf{z}_v^{(k)})$$

This ensures that each edge embedding captures the collaborative representation of its source and destination nodes, enriched by the edge attention mechanism.

The architecture achieves three key advantages: 1) Edge features preservation through $\mathbf{e}uv$ integration prevents information loss common in conventional GNNs; 2) Adaptive attention weights α_{uv} suppress benign flow patterns while amplifying suspicious connections; 3) Layer propagation captures multi-grained attack, with shallow layers detecting anomalies and deeper layers identifying attack types.

4.3. Hierarchical Detection Framework

Generally, NIDSs using graph embedding usually use coarse labels to predict whether the network flow is an attack or benign [13]. However, such a model is not able to further explore the attack type to facilitate network defenders in conducting research. Lo et al.[12] classifies the attack fine-grained labels while training coarse-grained detection, and then trains a second model for attack type recognition, but this will cause a lot of training resources to be wasted.

Therefore, we add both fine-grained and coarse-grained objectives to the training objective. Coarse-grained loss: L_{coarse} is calculated using the cross entropy loss function:

$$L_{coarse} = CrossEntropyLoss(\hat{\mathbf{y}}_{coarse}, \mathbf{y}_{coarse})$$

For fine-grained loss. First, create a mask $\mathbf{M}$ to mark samples whose coarse-grained labels are positive (assuming the positive label is 1):

$$\mathbf{M}_i = \begin{cases} 1, & \text{if } \mathbf{y}_{coarse}(i) = 1 \\ 0, & \text{otherwise} \end{cases}$$

If $\sum_{i=1}^{N} \mathbf{M}_i = 0$, that is, there are no samples with coarse-grained labels as positive, then the fine-grained loss $L_{fine} = 0$.

- Compute fine-grained prediction probability distribution $\mathbf{P}$:

$$\mathbf{P}_{ij} = \frac{\exp(\hat{\mathbf{y}}_{fine}(i,j))}{\sum_{k=1}^{C_{fine}} \exp(\hat{\mathbf{y}}_{fine}(i,k))}$$

  Where i is the coarse-grained positive sample index and j is the fine-grained category index, $j = 1, 2, ..., C_{fine}$.
- Calculate the information entropy of each sample $\mathbf{H}$:

$$\mathbf{H}_i = -\sum_{j=1}^{C_{fine}} \mathbf{P}_{ij} \log(\mathbf{P}_{ij})$$

- Calculate the weight of each sample $\mathbf{w}$:

$$\mathbf{w}_i = \frac{1}{1 + \exp(-\tau \mathbf{H}_i)}$$

  Where the τ is the temperature coefficient.
- Normalize weights and calculate fine-grained losses L_{fine}:

$$\mathbf{w}_i = \frac{\mathbf{w}_i}{\sum_{i=1}^{N} \mathbf{M}_k} \mathbf{w}_i$$

$$L_{fine} = \sum_{i=1}^{N} \mathbf{M}_i \mathbf{w}_i \text{CrossEntropyLoss}(\hat{\mathbf{y}}_{fine}(i), \mathbf{y}_{fine}(i))$$

Finally, automatically balance the weights of the loss function. The total loss function L_{total} is a combination of the coarse-grained loss function and the fine-grained loss function.

$$L_{total} = L_{coarse} + \frac{\sum_{i=1}^{N} \mathbf{M}_i}{N} L_{fine}$$

Algorithm 2 Adversarial Training with PGD

Input: Training graph G, model f_θ, perturbation bound ϵ, step size α, iterations T, total epochs N, Robust model parameters θ

for epoch $= 1$ **to** N **do**
  Generate adversarial examples:
  Initialize $\mathbf{\Delta}^{(0)} \sim \mathcal{U}(-\epsilon, +\epsilon)$
  for $t = 1$ **to** T **do**
    Compute gradient: $\mathbf{g} \leftarrow \nabla_{\mathbf{\Delta}} \mathcal{L}(f_\theta(G + \mathbf{\Delta}^{(t-1)}))$
    Update perturbation: $\mathbf{\Delta}^{(t)} \leftarrow \mathbf{\Delta}^{(t-1)} + \alpha \cdot \text{sign}(\mathbf{g})$
    Project to ℓ_∞ ball: $\mathbf{\Delta}^{(t)} \leftarrow \text{Clip}_{[-\epsilon,+\epsilon]}(\mathbf{\Delta}^{(t)})$
    Apply protocol constraints:
    $\mathbf{\Delta}^{(t)} \leftarrow \text{ProtocolProjection}(\mathbf{\Delta}^{(t)})$
  end for
  Update model parameters:
  $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(f_\theta(G + \mathbf{\Delta}^{(T)}))$
end for

4.4. Adversarial Training Module

In response to the defense requirements against escape attacks in network intrusion detection, this work proposes an edge feature perturbation generation framework based on projected gradient descent (PGD) [21]. PGD is adopted for its proven effectiveness as a universal first-order adversary, capable of generating strong adversarial examples that systematically expose model vulnerabilities. Unlike traditional nodes, focus on centric perturbation methods. Our approach leverages PGD to target edge features. Attackers often obfuscate protocol fields, port numbers, or flow timing to evade detection (e.g., DNS tunneling via modified UDP payloads or TCP flag order manipulation).

Given the original edge feature matrix $E \in R^{d_e}$, this paper generates adversarial perturbations $\Delta \in R^{d_e}$ through multiple steps of iteration. The perturbed edge feature $E^{adv} = E + \Delta$ maximizes the loss function of the detection model under the premise of satisfying the ℓ_∞ norm constraint $\|\Delta\|_\infty \leq \epsilon$.

$$\Delta^{(t+1)} = \prod_{[-\epsilon,+\epsilon]} \left(\Delta^{(t)} + \alpha \cdot \text{sign}(\nabla_{\Delta^{(t)}} \mathcal{L}(f_\theta(\mathbf{E} + \mathbf{\Delta}^{(t)}), \mathbf{Y})) \right)$$

where $\prod$ represents the projection operation, α is the step size, and $\mathcal{L}$ is the hierarchical cross entropy loss function. The adversarial training formalized in Algorithm 2, alternates between adversarial perturbation crafting and model optimization throughout each training epoch. The process is initiated by generating edge feature perturbations through multi-step projected gradient descent (PGD). Starting from uniformly sampled noise within the prescribed ℓ_∞ bound, the perturbation undergoes iterative refinement by ascending the gradient direction of the detection model's loss function. A critical distinction from conventional PGD implementations lies in the protocol-aware projection step, which enforces domain-specific constraints to maintain semantically valid network flows. Following perturbation generation, the model parameters are updated using the adversarially perturbed graph features, thereby hardening the detector against evasion tactics while preserving topological relationships essential for contextual analysis.

Our approach focuses on edge feature manipulation, such as payload content and inter-packet timing. These are key vectors for evasion in real-world attacks. By prioritizing edge features over static node attributes like IP geolocation, we ensure the training process aligns with adversarial-realistic tactics. We preserve the topological integrity of the network graph, such as IP communication patterns, through edge-centric perturbation. This allows

the model to learn how to distinguish malicious behaviors while retaining structural context. Additionally, the iterative nature of PGD. For example, 10–20 steps are designed to capture incremental edge feature modifications in advanced attacks. For example, it can model APTs' gradual protocol shifting from HTTP to DNS. This capability forces the model to generalize across evolving evasion strategies.

By integrating edge feature perturbation into the PGD framework, this module enhances the detection model's robustness against real-world escape tactics, ensuring reliable performance in adversarial network environments.

5. Experiments and Results

We evaluate the effectiveness and efficiency of AH-EAT using 4 datasets, including UNSW-NB15, CSE-CIC-IDS2018, BoT-IoT and ToN-IoT.

We first describe our experimental settings (Sec. 5.1). Then we elaborate on the performance compared with other state-of-art methods (Sec. 5.2). We conduct the robustness performance study (Sec. 5.3). Finally, we perform an ablation study (Sec. 5.4).

5.1. Experimental Setting

5.1.1. Implementation

We implement AH-EAT in Python 3.8. The graph representation module is implemented using PyTorch [29], PyG [30], Scikit-learn [31], and NetworkX [32]. For each data set, we divide the training set and testing set into the conventional 70% and 30% respectively. The number of training epochs is 2000 iterations. Experiments are run on a server with an Intel CPU and NVIDIA V100 GPU. The operating system is Linux. We will release the source code after the paper is published.

To clearly distinguish the differences and benefits of using AH-EAT embeddings in NIDSs, our results compare the accuracy and F1-score for both coarse-grained detection and fine-grained detection. F1-score is used as a key performance indicator for our experiments, as this metric combats the problem of potential imbalance within datasets. Additionally, accuracy (Acc) is also used for comparison.

In the robustness study, we use the accuracy degradation before and after adversarial attack and the adversarial adversarial success rate (ASR) as research indicators. The ASR is the proportion of input samples that are misclassified by the model under adversarial attacks. Specifically, it is the ratio of the number of perturbed samples that are misclassified by the model to the total number of perturbed samples after adding adversarial perturbations to the original samples. It measures the vulnerability of the model to adversarial attacks. The higher the ASR, the more vulnerable the model is to adversarial attacks and the worse its robustness.

$$\textbf{ASR} = \frac{N_{miss}}{N_{total}}$$

where the N_{miss} is the number of misclassified adversarial samples and N_{total} is the total number of adversarial samples

5.1.2. Datasets

We evaluate our approach on four NIDSs datasets.

UNSW-NB15 [33] has 43 standardised features. Out of the 2,390,275 flows, 2,295,222 (96.02%) samples are benign flows, and the remaining 95,053 (3.96%) samples are attack flows. Nine attack types are distributed across the 3.98% of the flows representing attacks. **CSE-CIC-IDS2018** [34] contains 18,893,708 flows distributed between 16,635,567 (88.05%) benign samples and 2,258,141 (11.95%) attack samples. Within the 11.95% of attack samples are 6 different attack methods. **ToN-IoT** [35] is a netflow-based IoT network dataset, which

includes different types of IoT data, such as operation system logs and telemetry data of IoT/IIoT services. The dataset consists of 796,380 (3.56%) benign flows and 21,542,641 (96.44%) attack flows, with a total of 22,339,021 flows. **BoT-IoT** [36] is comprised of 6 types of attacks and a total of 47 features with corresponding class labels. The dataset contains only 477 (0.01%) benign flows and 3,668,045 (99.99%) attack flows, with a total of 3,668,522 flows. These datasets are all updated versions of existing NIDSs datasets that have been standardized into a NetFlow format [37].

5.1.3. Baseline

Table 1. Comparison of the coarse-grained performance of the model on different datasets

Model	UNSW-NB15		CSE-CIC-IDS2018		BoT-IoT		ToN-IoT	
	Acc (%)	F1 (%)	Acc (%)	F1 (%)	Acc (%)	F1 (%)	Acc (%)	F1 (%)
E-GraphSAGE	98.73	98.81	98.70	98.72	93.76	95.45	81.47	82.99
E-GCN	98.73	98.81	79.01	82.46	94.35	95.73	79.79	80.20
Anomal-E	97.77	97.74	97.89	97.81	2.31	2.26	19.57	16.37
AH-EAT	96.02	94.08	88.03	82.43	98.44	98.05	80.42	71.68

Table 2. Comparison of the fine-grained performance of the model on different datasets

Model	UNSW-NB15		CSE-CIC-IDS2018		BoT-IoT		ToN-IoT	
	Acc (%)	F1 (%)	Acc (%)	F1 (%)	Acc (%)	F1 (%)	Acc (%)	F1 (%)
E-GraphSAGE	95.44	95.92	9.41	11.35	76.28	77.02	49.36	51.13
E-GCN	94.99	95.40	1.41	1.29	78.99	78.51	11.49	2.52
AH-EAT	95.57	95.52	22.77	34.34	81.46	83.84	42.25	25.38

We have four comparison methods. E-GraphSAGE [12] is an improved GNN model designed for edge features in IoT. Through the message-passing mechanism, source node features and edge features (such as protocol type, and flow statistics) are aggregated to generate node embeddings. The two node embeddings of an edge are concatenated to obtain edge embeddings. Anomal-E [13] is a GNN-based intrusion and anomaly detection method that uses edge features and graph topology in a self-supervised manner. This method allows for the incorporation of both edge features and topological patterns to detect attack patterns, without the need for labeled data. E-GCN is a model we modified based on E-GraphSAGE, changing the embedding model from GraphSAGE [25] to the standard convolution operation of GCN [10], which is used to compare the sampling effect of GraphSAGE. In network flow data with rich edge features, the complete aggregation of global topological information is richer than what can be learned by random sampling.

5.2. Comparison with SOTA Methods

The coarse-grained and fine-grained classification capabilities of GNNs are crucial in the field of network security. Table 1 and Table 2 show the performance comparison of different models on coarse-grained (binary classification: normal/attack) and fine-grained (multi-classification: specific attack type) tasks, respectively.

In coarse-grained detection, AH-EAT shows significant advantages on the IoT datasets BoT-IoT and ToN-IoT. For BoT-IoT, AH-EAT's accuracy (98.44%) and F1 score (98.05%) are both higher than E-GraphSAGE (93.76%, 95.45%) and E-GCN (94.35%, 95.73%), indicating that it has a stronger ability to identify complex attacks in IoT environments. On the ToN-IoT dataset, although E-GraphSAGE has a slightly higher accuracy (81.47% vs. 80.42%), AH-EAT's F1 score (71.68%) far exceeds Anomal-E (16.37%), showing better

classification balance. Anomal-E completely fails on BoT-IoT and ToN-IoT (F1<20%) due to its unsupervised design. Since these two datasets were not experimented on in the original Anomal-E paper, the unsupervised method Anomal-E is not adaptable enough to IoT data and the limitations of the algorithm.

Fine-grained classification can better reflect the model's ability to identify attack subclasses. Since Anomal-E cannot support fine-grained tasks, Table 2 focuses on the comparison of E-GraphSAGE, E-GCN, and AH-EAT. AH-EAT's fine-grained accuracy (22.77%) and F1 score (34.34%) on the CSE-CIC-IDS2018 dataset are significantly higher than E-GraphSAGE (9.41%, 11.35%) and E-GCN (1.41%, 1.29%), indicating that it has stronger feature extraction capabilities in complex attack scenarios. In BoT-IoT and ToN-IoT, AH-EAT's fine-grained F1 scores reach 83.84% and 25.38% respectively, far exceeding similar models.

Overall, AH-EAT effectively balances the generalization of coarse-grained detection and the specificity of fine-grained classification through hierarchical detection mechanisms, so that it can perform very well in most cases for both coarse and fine granularity scenarios. Compared with the self-supervised method Anomal-E, it shows better adaptability; compared with traditional GNN models (such as E-GraphSAGE, E-GCN), its innovative edge-based graph attention network and feature fusion strategies significantly improve the ability to identify attack types, providing a more reliable solution for actual network security deployment.

5.3. Compare Robustness

Table 3. Comparison of adversarial robustness performance on CIC-IDS2018

Model	Performance (%)		
	Clean Accuracy	Robust Accuracy	ASR
E-GraphSAGE	98.48	81.66	18.34
E-GCN	98.91	53.50	46.50
Anomal-E	97.83	11.97	88.03
AH-EAT	98.55	98.55	1.45

Table 4. Comparison of adversarial robustness performance on ToN-IoT

Model	Performance (%)		
	Clean Accuracy	Robust Accuracy	ASR
E-GraphSAGE	82.37	19.58	80.42
E-GCN	80.67	30.64	69.36
AH-EAT	84.38	84.38	15.62

Adversarial robustness is a core indicator to measure whether a model can maintain stable performance under adversarial attacks. Table. 3 and Table. 4 show the robustness performance of different models on the CIC-IDS2018 and ToN-IoT datasets, including clean sample accuracy (Clean Accuracy), adversarial sample robust accuracy (Robust Accuracy) and adversarial success rate (ASR).

E-GCN achieves the highest clean accuracy (98.91%) on clean samples but collapses under adversarial attacks, with robust accuracy dropping to 53.50% and ASR reaching 46.50%, indicating severe sensitivity to adversarial perturbations. E-GraphSAGE performs better than E-GCN with a robust accuracy of 81.66% and ASR of 18.34%, though still showing significant degradation. Anomal-E performs worst, with robust accuracy as low as 11.97% and ASR as high as 88.03%, reflecting its inability to maintain classification ability

in adversarial environments, likely due to poor feature extraction for IoT data using a self-supervised way. AH-EAT demonstrates overwhelming advantages on CIC-IDS2018: its clean accuracy (98.55%) is identical to robust accuracy, with an ASR of only 1.45%. This means the model's accuracy remains unchanged after adding adversarial perturbations, proving its strong resistance to adversarial attacks.

As a dataset for low-resource IoT devices, ToN-IoT imposes stricter robustness requirements. E-GraphSAGE and E-GCN suffer drastic drops in robust accuracy to 19.58% and 30.64%, respectively, with ASR exceeding 60%, indicating traditional GNNs struggle to defend against attacks in resource-constrained scenarios. In contrast, AH-EAT maintains consistent clean and robust accuracy (84.38%) with an ASR of only 15.62%, far lower than peer models. This benefits from its hierarchical adversarial training, which suppresses the impact of perturbations on classification decisions while preserving original feature representations.

AH-EAT constructs a multi-layer defense against adversarial perturbations through gradient alignment constraints and feature-space robust regularization. On CIC-IDS2018, its ASR is only $\frac{1}{32}$ of E-GCN, and on ToN-IoT, it reduces ASR by over 80% compared to E-GraphSAGE. This advantage is particularly critical for resource-constrained IoT scenarios, proving that AH-EAT can maintain stable detection performance under real-world attacks, providing a solid foundation for industrial deployment.

5.4. Ablation Study

Table 5. Ablation study of model components on BoT-IoT(fine-grained)

Model	Performance (%)				
	Accuracy	F1-score	Robust Acc	Robust F1	ASR
AH-EAT	81.46	83.84	81.15	83.52	18.85
AH-EAT-NA	82.26	82.68	82.03	82.54	17.97
AH-EAT-NH	78.54	69.09	78.18	68.92	21.82
AH-EAT-NADV	78.90	77.10	72.81	71.01	27.19

Table 6. Ablation study of model components on UNSW-NB15(coarse-grained)

Model	Performance (%)				
	Accuracy	F1-score	Robust Acc	Robust F1	ASR
AH-EAT	97.45	97.23	97.36	97.10	2.64
AH-EAT-NA	97.09	96.67	97.12	96.73	2.88
AH-EAT-NH	96.02	94.08	96.02	94.08	3.98
AH-EAT-NADV	11.97	2.56	11.97	2.56	88.03

The ablation experiment removes the core modules (attention, hierarchical detection framework, adversarial training) to quantify the independent contribution of each component to the model performance (accuracy, robustness) and reveal the synergy between modules. By comparing the performance of the complete model with the variant model, the necessity of each module design and its unique value in complex tasks are verified, providing a theoretical basis for model optimization. Table. 5 and Table. 6 show the results on the BoT-IoT (fine-grained) and UNSW-NB15 (coarse-grained) datasets, respectively.

The three models are designed as follows:

- **AH-EAT-NA** (removes attention mechanism). This variant removes the attention mechanism and uses mean aggregation of neighbor features instead. The goal is to verify the key role of hierarchical attention in the fusion of graph structure features.

By comparing the original model, it is observed whether mean aggregation leads to a decrease in feature discrimination (such as ignoring the structural information of key nodes). Therefore, it clarifies the irreplaceable role of the attention mechanism in screening high-value neighbor features and suppressing noise interference.

- **AH-EAT-NH** (remove hierarchical detection framework). This variant removes the hierarchical framework of "coarse-grained detection → fine-grained classification" and directly performs multi-classification on all samples. Its core goal is to verify the dual improvement of hierarchical detection on efficiency and accuracy: whether the coarse-grained stage can effectively filter normal samples to reduce the computational load, and whether the fine-grained stage can alleviate the category imbalance problem through hierarchical decision-making.
- **AH-EAT-NADV** (remove adversarial training). This variant turns off the adversarial training module and only uses the original clean data for training, aiming to verify the core impact of adversarial training on model robustness. By comparing the performance under adversarial samples (such as ASR, robust accuracy), quantifying the cornerstone role of adversarial training in resisting perturbations. It enhances the robustness of feature space and reveals its decisive impact on the stability of the model in real attack scenarios.

In Table 5, attention mechanism (AH-EAT-NA): after removal, F1-score dropped by 1.16% (83.84%→82.68%), but Accuracy unexpectedly increased by 0.8% (81.46%→82.26%). This shows that although mean aggregation improves overall accuracy, it weakens the ability to capture hidden attack features (such as the time series pattern of DNS tunnels). Its ASR decreased by 1.88% (18.85%→17.97%), reflecting that the attention mechanism may introduce slight vulnerabilities, so the effect of the attention mechanism is limited in a small-scale dataset such as BoT-IoT; Hierarchical detection framework (AH-EAT-NH): direct multi-classification caused F1-score to plummet by 14.75% (83.84%→69.09%), and ASR increased by 2.97% (18.85%→21.82%). It is proved that the hierarchical framework effectively protects fine-grained modules from noise interference through coarse-grained screening (filtering 85% of benign flow), and improves the identification accuracy of key attack paths; adversarial training (AH-EAT-NADV): after closing, Robust F1 drops by 12.51% (83.52%→71.01%), and ASR surges by 8.34% (18.85%→27.19%). It shows that adversarial training improves the model's defense capability against edge feature perturbations by more than 3 times, highlighting the necessity of adversarial training.

In Table 6, attention mechanism (AH-EAT-NA): F1-score drops by 0.56% (97.23%→96.67%), and ASR increases slightly by 0.24% (2.64%→2.88%). It shows that in coarse-grained tasks, global topological information is more important than local attention. However, the attention mechanism still contributes to the recognition ability of key information. Hierarchical detection framework (AH-EAT-NH): Direct multi-classification leads to a 3.15% drop in F1-score (97.23%→94.08%), and a 1.34% increase in ASR (2.64%→3.98%). Adversarial training (AH-EAT-NADV): After turning it off, the model completely collapses (Robust F1=2.56% vs 97.10%), and ASR soars to 88.03%. It proves that adversarial training is the core guarantee of robustness. It enables the model to learn the protocol semantic invariance (such as HTTP method legality constraints) through edge feature perturbation generation.

In summary, the attention mechanism provides high-discrimination features for hierarchical detection, and the hierarchical framework reduces the interference of invalid samples in adversarial training (such as fine-grained adversarial optimization only for samples that are judged as attacks at a coarse-grained level). Adversarial training feeds back feature learning, forming a positive cycle of "feature screening → hierarchical decision → robust enhancement".

6. Conclusion

In this paper, we propose AH-EAT, an adversarial hierarchical-aware edge attention learning method. AH-EAT collaborates an edge attention mechanism, hierarchical detection framework and adversarial training module to construct a detection system with accuracy, efficiency and robustness. It solves the defects of existing NIDSs in static feature aggregation, single-granularity detection and adversarial vulnerability. We integrate edge features (such as flow protocol and interaction frequency) with attention mechanism, adaptively distinguish key attack paths from noise edges, and effectively capture structures and feature patterns that are crucial for attack detection. It solves the problem that traditional static aggregation strategies (such as mean/max pooling) cannot intelligently filter features, and lays a feature foundation for accurate detection. Through the two-stage detection pipeline of "coarse-grained filtering and fine-grained classification". Malicious and benign flows are quickly distinguished through coarse-grained detection, and more than 90% of normal samples are filtered. Then fine-grained attack detections are identified for malicious flow. This framework breaks through the limitations of single-granularity detection, alleviates the problem of imbalanced attack class samples through joint optimization of hierarchical loss functions, and avoids the waste of resources of traditional methods of training from scratch. The adversarial training strategy based on projected gradient descent (PGD) is introduced to generate adversarial samples of graph structure perturbations and edge features, forcing the model to learn robust decision boundaries that are insensitive to perturbations, significantly improving the stability of the model in adversarial environments and filling the key gap in the adversarial vulnerability of traditional methods. Experiments have shown that AH-EAT has both high-precision detection and strong robustness in both normal sample and adversarial sample scenarios, providing a feasible technical solution for intrusion detection in complex network environments.

Acknowledgment

This work was supported in part by the Shenzhen Science and Technology Program (No. KJZD20231023094701003), the Major Key Project of PCL (Grant No. PCL2024A05), and the National Natural Science Foundation of China (Grant No. 62372137).

1. Anthi, E.; Williams, L.; Słowińska, M.; Theodorakopoulos, G.; Burnap, P. A supervised intrusion detection system for smart home IoT devices. *IEEE Internet of Things Journal* **2019**, *6*, 9042–9053.
2. Alani, M.M.; Awad, A.I. An intelligent two-layer intrusion detection system for the Internet of Things. *IEEE Transactions on Industrial Informatics* **2022**, *19*, 683–692.
3. Alcantara, L.; Padilha, G.; Abreu, R.; d'Amorim, M. Syrius: Synthesis of rules for intrusion detectors. *IEEE Transactions on Reliability* **2021**, *71*, 370–381.
4. Liu, Q.; Keller, H.B.; Hagenmeyer, V. A bayesian rule learning based intrusion detection system for the mqtt communication protocol. In Proceedings of the Proceedings of the 16th International Conference on Availability, Reliability and Security, 2021, pp. 1–10.
5. Saranya, T.; Sridevi, S.; Deisy, C.; Chung, T.D.; Khan, M.A. Performance analysis of machine learning algorithms in intrusion detection system: A review. *Procedia Computer Science* **2020**, *171*, 1251–1260.
6. Liu, H.; Lang, B. Machine learning and deep learning methods for intrusion detection systems: A survey. *applied sciences* **2019**, *9*, 4396.
7. Nicolau, M.; McDermott, J.; et al. Learning neural representations for network anomaly detection. *IEEE transactions on cybernetics* **2018**, *49*, 3074–3087.
8. Xu, C.; Shen, J.; Du, X. A method of few-shot network intrusion detection based on meta-learning framework. *IEEE Transactions on Information Forensics and Security* **2020**, *15*, 3540–3552.
9. Xie, K.; Li, X.; Wang, X.; Cao, J.; Xie, G.; Wen, J.; Zhang, D.; Qin, Z. On-line anomaly detection with high accuracy. *IEEE/ACM transactions on networking* **2018**, *26*, 1222–1235.

10. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* **2016**.
11. Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Yu, P.S. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* **2020**, *32*, 4–24.
12. Lo, W.W.; Layeghy, S.; Sarhan, M.; Gallagher, M.; Portmann, M. E-graphsage: A graph neural network based intrusion detection system for iot. In Proceedings of the NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium. IEEE, 2022, pp. 1–9.
13. Caville, E.; Lo, W.W.; Layeghy, S.; Portmann, M. Anomal-E: A self-supervised network intrusion detection system based on graph neural networks. *Knowledge-based systems* **2022**, *258*, 110030.
14. Wang, B.; Jia, J.; Cao, X.; Gong, N.Z. Certified robustness of graph neural networks against adversarial structural perturbation. In Proceedings of the Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 1645–1653.
15. 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 Transactions on Network and Service Management* **2022**, *20*, 684–696.
16. Yu, L.; Dong, J.; Chen, L.; Li, M.; Xu, B.; Li, Z.; Qiao, L.; Liu, L.; Zhao, B.; Zhang, C. PBCNN: Packet bytes-based convolutional neural network for network intrusion detection. *Computer Networks* **2021**, *194*, 108117.
17. Chakraborty, A.; Alam, M.; Dey, V.; Chattopadhyay, A.; Mukhopadhyay, D. A survey on adversarial attacks and defences. *CAAI Transactions on Intelligence Technology* **2021**, *6*, 25–45.
18. Long, T.; Gao, Q.; Xu, L.; Zhou, Z. A survey on adversarial attacks in computer vision: Taxonomy, visualization and future directions. *Computers & Security* **2022**, *121*, 102847.
19. Zhang, W.E.; Sheng, Q.Z.; Alhazmi, A.; Li, C. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)* **2020**, *11*, 1–41.
20. Beaver, J.M.; Symons, C.T.; Gillen, R.E. A learning system for discriminating variants of malicious network traffic. In Proceedings of the Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop, 2013, pp. 1–4.
21. Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* **2017**.
22. Nagaraja, S.; Mittal, P.; Hong, C.Y.; Caesar, M.; Borisov, N. {BotGrep}: finding {P2P} bots with structured graph analysis. In Proceedings of the 19th USENIX Security Symposium (USENIX Security 10), 2010.
23. Lopez-Martin, M.; Carro, B.; Arribas, J.I.; Sanchez-Esguevillas, A. Network intrusion detection with a novel hierarchy of distances between embeddings of hash IP addresses. *Knowledge-based systems* **2021**, *219*, 106887.
24. Xiao, Q.; Liu, J.; Wang, Q.; Jiang, Z.; Wang, X.; Yao, Y. Towards network anomaly detection using graph embedding. In Proceedings of the Computational Science–ICCS 2020: 20th International Conference, Amsterdam, The Netherlands, June 3–5, 2020, Proceedings, Part IV 20. Springer, 2020, pp. 156–169.
25. Hamilton, W.; Ying, Z.; Leskovec, J. Inductive representation learning on large graphs. *Advances in neural information processing systems* **2017**, *30*.
26. Sun, Z.; Teixeira, A.M.; Toor, S. GNN-IDS: Graph Neural Network based Intrusion Detection System. In Proceedings of the Proceedings of the 19th International Conference on Availability, Reliability and Security, 2024, pp. 1–12.
27. Angioni, D.; Demetrio, L.; Pintor, M.; Oneto, L.; Anguita, D.; Biggio, B.; Roli, F. Robustness-congruent adversarial training for secure machine learning model updates. *arXiv preprint arXiv:2402.17390* **2024**.
28. Wang, X.; Ji, H.; Shi, C.; Wang, B.; Ye, Y.; Cui, P.; Yu, P.S. Heterogeneous graph attention network. In Proceedings of the The world wide web conference, 2019, pp. 2022–2032.
29. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* **2019**, *32*.
30. Fey, M.; Lenssen, J.E. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428* **2019**.

31. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* **2011**, *12*, 2825–2830.
32. Hagberg, A.; Swart, P.; S Chult, D. Exploring network structure, dynamics, and function using NetworkX. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.
33. 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). IEEE, 2015, pp. 1–6.
34. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A.; et al. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp* **2018**, *1*, 108–116.
35. Koroniotis, N.; Moustafa, N.; Sitnikova, E.; Turnbull, B. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Generation Computer Systems* **2019**, *100*, 779–796.
36. 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.
37. Sarhan, M.; Layeghy, S.; Portmann, M. Towards a standard feature set for network intrusion detection system datasets. *Mobile networks and applications* **2022**, pp. 1–14.