

Robust Intrusion Detection with GCN-AE: A One-Class GNN Approach

Zahra Eskandari[†], Non-member

ABSTRACT

Designing Intrusion Detection Systems (IDS) that are robust against both known and emerging threats remains a critical challenge in cybersecurity. Traditional graph-based IDS approaches often rely on supervised learning, which requires large labeled datasets and struggles against evasion tactics that deviate from known attack patterns. To overcome these limitations, recent research has explored unsupervised, self-supervised, and semi-supervised learning methods that focus on modeling normal behavior to detect anomalies. Autoencoders have proven effective in learning compact and informative representations. Building on this, we propose GCN-AE, a novel one-class Graph Neural Network (GNN) model that integrates Graph Convolutional Network (GCN) layers within an Autoencoder framework. The model incorporates a specialized message-passing mechanism designed to capture normative network traffic behavior, enabling it to detect deviations without prior knowledge of attack signatures. We evaluate GCN-AE on the NF-UQ-NIDS-v1 dataset under a binary classification setting. Experimental results show that the proposed model achieves a weighted F1-score of 98.8% and an accuracy of 98.8%, outperforming existing approaches while exhibiting strong robustness against zero-day, adversarial, and camouflage attacks. These findings highlight the effectiveness of integrating GCNs with Autoencoders to enhance the adaptability and resilience of network-based intrusion detection systems (IDS), providing a robust solution for detecting both known and previously unseen threats in real-world network environments.

Keywords: Autoencoder, Graph Convolutional Networks (GCNs), Graph Neural Networks (GNNs), Graph Semi-Supervised Learning, Zero-day attacks, Evasion Attacks

1. INTRODUCTION

Cyber attacks have increasingly posed substantial threats to organizational data and user privacy [1]. In

addition to existing vulnerabilities, adversaries continuously modify their behavioral patterns to bypass security mechanisms, allowing them to remain undetected within organizational networks [2]. Consequently, a critical challenge for security professionals is to design intrusion detection systems (IDS) that are not only robust against known threats but also adaptable to emerging attacks exhibiting previously unseen behaviors [2].

Intrusion Detection System (IDS) approaches are generally categorized into signature-based and anomaly-based methods. Signature-based techniques rely on predefined patterns of known attacks, offering high accuracy in detecting previously observed threats. However, they are ineffective against novel or previously unseen attacks. In contrast, anomaly-based detection methods utilize machine learning (ML) and deep learning (DL) to identify malicious behaviors by modeling normal activity patterns from data [3–5]. While traditional ML methods depend on manual feature extraction, DL approaches—such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs)—can automatically learn spatial and temporal features, thereby enhancing detection accuracy. Nonetheless, these models often analyze network flows independently, overlooking their dependencies. As a result, they struggle to capture the underlying network topology and communication patterns, which ultimately constrain their detection performance.

Graph Neural Networks (GNNs) have recently emerged as a powerful framework for learning from graph-structured data, with successful applications in various domains including social networks, natural language processing, and beyond [6]. Within the domain of network communication, modeling the network as a graph—where nodes represent endpoint devices and edges represent communication flows—provides a more holistic perspective moves beyond treating flows as independent entities. This graph-based representation enables a deeper understanding of the overall network state and facilitates the development of methods that more effectively capture and address complex network behaviors and associated security challenges.

Existing graph-based intrusion detection methods predominantly rely on supervised learning techniques [7, 8]. However, in addition to requiring large training datasets, these methods are often ineffective against evasion strategies that deliberately deviate from known attack patterns [2]. To address these limitations, alternative approaches such as unsupervised [9], self-supervised [10, 11], and semi-supervised learning [12] have been

Manuscript received on November 23, 2025; revised on June 8, 2026; accepted on July 23, 2026. This paper was recommended by Associate Editor Kampol Woradit.

The author is with Department of Computer Engineering, Faculty of Electrical and Computer Engineering, Quchan University of Technology, Quchan, Iran.

[†]Corresponding author: z.eskandari@qiet.ac.ir

©2026 Author(s). This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 4.0 License. To view a copy of this license visit: <https://creativecommons.org/licenses/by-nc-nd/4.0/>.

Digital Object Identifier: 10.37936/ecti-ec.2026243.262491

proposed. These approaches focus on modeling normal user behavior within a network and identifying deviations as potential anomalies. By design, such models are inherently more robust against deviation-based attacks, making them particularly effective in detecting evasion and zero-day attacks. Recently, Graph Semi-Supervised Learning (GSSL) [13] has been introduced and demonstrated significant potential in Bot detection [12].

Autoencoders [14] are a notable example of semi-supervised learning methods. They compress input data into a latent space during the encoding phase, capturing essential features, and subsequently reconstruct the input in the decoding phase with minimal reconstruction error. This reconstruction error serves as an anomaly score, where inputs that deviate from normal behavior yield higher reconstruction errors, thereby effectively distinguishing between normal and abnormal network traffic. Given their demonstrated effectiveness in extracting compact and informative representations, integrating Autoencoders with Graph Neural Networks (GNNs) offers a promising direction by leveraging the complementary strengths of both approaches [9, 15].

Integrating Graph Convolutional Networks (GCNs) with Autoencoder architectures, we propose a novel one-class GNN model, termed **GCN-AE**, which employs an Autoencoder structure combined with Graph Convolutional layers. The model utilizes a nonstandard message-passing mechanism specifically designed to learn the normal behavior of network states. In particular, the proposed model leverages both node and edge features within a network graph to extract embedding representations of normal activity. As a result, it can effectively detect deviations from this learned normality, thereby distinguishing between normal and abnormal behavior, including both known and previously unseen (zero-day and evasion) attacks. Compared to traditional flow-based methods, the proposed approach captures the interactions between source and destination nodes, as well as the network topology. This capability enables the model to extract rich representations from complex graph structures comprising nodes and edges.

Our main contributions are summarized as follows:

- We propose a novel one-class Graph Neural Network framework by integrating Graph Convolutional Network (GCN) layers with Autoencoder architecture (GCN-AE).
- The model is trained under a semi-supervised learning paradigm that relies solely on a small set of normal traffic samples. This approach enables the model to learn the normal behavior of network traffic and effectively identify deviations. Consequently, the proposed model demonstrates inherent robustness against previously unseen attacks, including zero-day and evasion attacks.
- Extensive experiments are conducted using NetFlow-based NF-UQ-NIDS-v1 dataset under a binary classification setting. Our method significantly outperforms existing approaches. Furthermore, the experiments demonstrate the proposed model's potential to provide

a robust solution for Network-based Intrusion Detection Systems (NIDS).

The structure of the paper is organized as follows: Section 2 presents a concise review of related work. Section 3 details the proposed methodology, including a step-by-step description and an in-depth explanation of the model's operation. Section 4 discusses the experimental setup and presents the evaluation results. Finally, Section 5 concludes the study and outlines potential directions for future research.

2. RELATED WORKS

Graph Neural Networks (GNNs) have gained increasing prominence in network intrusion detection systems (NIDSs) due to their capacity to model complex relationships within network data. Numerous GNN-based methods have been proposed for anomaly and intrusion detection tasks [16]. Among the recent studies, we briefly examine those that share the closest similarities with our proposed method.

Recently, Lo et al. [7] formulated intrusion detection as a link classification problem. They proposed E-GraphSAGE, an extension of GraphSAGE that performs message passing based on edge features rather than node features. The authors demonstrated that their model outperforms traditional machine learning approaches. However, a key limitation of this method lies in its scalability, particularly as the number of nodes increases substantially.

Anomal-E [10] explores the use of Graph Neural Networks (GNNs) combined with Self-Supervised Learning (SSL) to generate graph representations and output embeddings for binary intrusion detection (i.e., distinguishing between normal and abnormal behavior). The SSL module is integrated into the network encoding process to enhance the representation of edge information.

TS-IDS [11] introduces a graph-based, traffic-aware self-supervised model for network intrusion detection. The authors formulate the network intrusion detection (NID) problem as a link classification task, utilizing a graph representation of the flow network. In TS-IDS, node features are extracted based on prior knowledge of traffic behavior at endpoints, while the traffic flows between two nodes define the edge features.

In [17], graph representation learning (GRL) is applied to network intrusion detection by modeling NIDS data as graphs to capture contextual relationships. Leveraging the CIC-IDS2017 dataset, the study employs graph neural networks and embedding techniques to enhance detection at both the flow and packet levels, resulting in significant performance improvements.

In [8], a GNN-based intrusion detection system (GNN-IDS) is proposed. The system integrates attack graphs and real-time measurements to capture both static and dynamic attributes, enabling a comprehensive representation of complex network environments. By learning the network's connectivity structure, the GNN model quantifies the influence of neighboring nodes and node

features to make more reliable predictions. Experimental results demonstrate that the proposed GNN-IDS performs well in terms of uncertainty, explainability, and robustness.

A GNN-based self-supervised representation learning approach for NIDS was proposed in [18], utilizing an attention-based encoder to generate graph embeddings using only edge information. A contrastive learning framework is employed to construct sub-graphs and generate positive and negative samples for representation learning. Unlike traditional deep learning methods, this approach is capable of learning meaningful patterns without the need for manual labels or prior knowledge.

The study in [15] presents GTAE-IDS, an unsupervised, packet-based GNN designed for early and accurate anomaly detection in network traffic. GTAE-IDS constructs sequential packet-based graphs and employs graph Autoencoders to learn structural patterns from benign traffic, eliminating the need for labeled data. Operating in near real-time, GTAE-IDS achieves over 98% accuracy on benchmark datasets, thereby addressing critical challenges in identifying emerging threats.

As highlighted in [8, 19], Graph Neural Networks (GNNs) have demonstrated strong robustness against adversarial attacks. The results reported in [19] indicate that GNN-based Network Intrusion Detection Systems (NIDS) maintain full resilience against common evasion strategies employed by attackers. In [8], to evaluate generalization to unseen inputs, Gaussian noise was introduced to malicious samples in the test set. The result indicates that GNNs exhibit greater resilience to network variations, effectively mitigating evasion attempts.

In cybersecurity, where labeled data is often limited, semi-supervised learning (SSL) offers a viable approach that addresses the limitations of traditional supervised algorithms [20]. Current intrusion detection systems (IDS) predominantly rely on supervised machine learning models trained on historical labeled datasets to classify network flows. However, modern networks are inherently heterogeneous and dynamic, with heterogeneity causing substantial variability across different network environments and dynamic behavior leading to the continuous emergence of novel threats. These factors challenge the effectiveness of supervised learning-based IDS in such environments. Consequently, semi-supervised learning (SSL) has been increasingly adopted to address these challenges in IDS development [21–23].

In [12], the authors propose Bot-AHGCN, a graph-based semi-supervised learning framework designed for Bot detection. The proposed approach models network flow data as a multi-attributed heterogeneous graph. The Bot detection problem is formulated as a semi-supervised node classification task within an attributed heterogeneous information network (AHIN). Experimental results on real-world datasets demonstrate that Bot-AHGCN outperforms state-of-the-art methods and provides enhanced interpretability.

Most existing graph-based semi-supervised learning

(GSSL) approaches operate at the node level. In contrast, our approach advances this by initially utilizing node-level information to categorize nodes according to their behavioral patterns. Subsequently, this information is leveraged to analyze the edges connected to these nodes, enabling the classification of each edge as representing normal or abnormal behavior. To the best of our knowledge, GSSL has not previously been applied within an intrusion detection system (IDS) using a one-class GCN-AE architecture specifically designed to address zero-day and evasion attacks.

3. METHODOLOGY

As previously discussed, to overcome the primary limitation of existing models in detecting attacks, our objective is to develop a model for detecting anomalous network traffic that: (1) integrates not only flow-level information between source and destination nodes but also the overall behavioral patterns of these nodes within the network, and (2) prioritizes the model’s capability to identify any deviation from normal behavior, including zero-day and evasion attacks.

To achieve this, we propose GCN-AE, an Autoencoder model incorporating Graph Convolutional Network (GCN) layers. This section details the proposed approach, which comprises: (1) Construction of the network graph from network flows, and (2) Presentation of the proposed model along with its procedural steps for detecting anomalous network traffic.

3.1 Network Flow Graph Construction

The network graph is represented as: $G = (V, E, Z, X)$, where:

- $V = \{v_1, v_2, v_3, \dots, v_n\}$: The set of n nodes in the graph.
- $E = \{e_1, e_2, e_3, \dots, e_m\}$: The set of m edges in the graph.

The feature matrices corresponding to the nodes and edges are defined as follows:

- Node Feature Matrix (Z): $Z = \{z_i \mid i \in \{1, \dots, n\}\} \in \mathbb{R}^{n \times D_v}$, where D_v is the dimension of the feature vector for each node.
- Edge Feature Matrix (X): $X = \{x_i \mid i \in \{1, \dots, m\}\} \in \mathbb{R}^{m \times D_e}$, where D_e is the dimension of the feature vector for each edge.

Let $\mathcal{F}$ denote the NetFlow dataset, where each flow $f \in \mathcal{F}$ corresponds to a network flow characterized by a specific tuple (Source-IP, Source-Port, Destination-IP, Destination-Port, Protocol) within a given time interval. Each flow is represented by a feature vector $x \in \mathbb{R}^{D_e}$, which includes attributes such as the number of packets sent and received, the number of bytes transmitted and received, the flow duration, among others [24].

From the set of flows $\mathcal{F}$, we construct a network graph $G = (V, E, Z, X)$. As discussed in [9, 11], to mitigate the sparsity inherent in (IP, Port)-graphs—which tend to be less informative than IP-only graphs—the directed graph G is defined such that the vertex set V comprises all IP

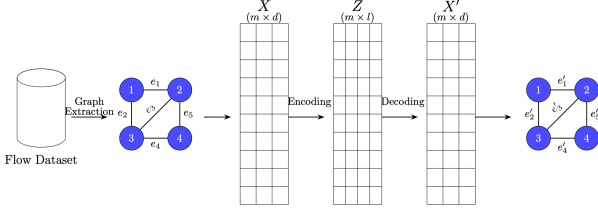


Fig. 1: Architecture of the GCN-AE Model.

addresses involved in any flow $f \in \mathcal{F}$. For each flow in $\mathcal{F}$, a directed edge is established from the Source-IP node to the Destination-IP node.

Since the edge feature vector must encapsulate the entire traffic behavior between two nodes, all flows occurring between the source and target nodes are aggregated into a single feature vector. This vector is constructed by applying five statistical functions—minimum, maximum, mean, standard deviation, and median—to the aggregated flows. The resulting values are concatenated to form the feature vector $x_i \in \mathbb{R}^{D_e}$ for each edge $e_i \in E$.

3.2 Proposed GCN-AE Model

In this section, we present the GCN-AE model, whose architecture is illustrated in Figure 1. Initially, the flow dataset is transformed into a graph $G = (V, E, X, Z)$ as described in Section 3.1. The constructed graph is subsequently used as input to the model.

Given the graph $G = (V, E, X, Z)$ with n nodes and m edges, let $A \in \{0, 1\}^{n \times n}$ denote the adjacency matrix and X represent the edge feature matrix. The layers of the proposed GCN-AE are defined as follows:

$$Z = f_1(\tilde{B}XW_1) \quad (1)$$

$$L = f_2(\tilde{B}^T Z, XW_2) \quad (2)$$

$$Z' = f_3(\tilde{B}LW_3) \quad (3)$$

$$X' = f_4(\tilde{B}^T Z', ZW_4) \quad (4)$$

Here, $X \in \mathbb{R}^{m \times D_e}$ and $X' \in \mathbb{R}^{m \times D_e}$ denote the input and output of the Autoencoder, respectively.

The proposed Autoencoder replaces conventional layers with graph convolution (GC) layers based on a non-standard message passing mechanism (Formulas (1)–(4)). In the encoder, incoming and outgoing edge features are aggregated per node to generate node embeddings (Formula (1)). These embeddings are then combined with original edge features to produce final edge embeddings (Formula (2)), resulting in a latent representation $L \in \mathbb{R}^{m \times h_e}$. The decoder mirrors this structure (Formulas (3)–(4)) to reconstruct X' from L by minimizing the reconstruction error.

In these formulations, the ReLU activation function is used as f_i . The incidence matrix $B \in \mathbb{R}^{n \times m}$ is defined such that, for each edge $e_k = (v_i, v_j)$, the entries are set as $B[i, k] = 1$ and $B[j, k] = -1$. This matrix B is then normalized to obtain $\tilde{B} \in \mathbb{R}^{n \times m}$.

The normalization incorporates the adjacency matrix A and degree matrix D , where $D_{ii} = \sum_j A'_{ij}$. By adding self-loops, $A' = A + I$, the normalized incidence matrix is computed as:

$$\tilde{B} = D^{-\frac{1}{2}} A' D^{-\frac{1}{2}} B. \quad (5)$$

This degree-aware normalization mitigates the dominance of high-degree nodes in aggregation, preserves the structural properties of the graph, and enables balanced message propagation for stable training [7].

Inspired by previous studies [9], incorporating the input of the first layer into formula (2) effectively introduces a residual connection. This design ensures that edge features are preserved and integrated with the embeddings of their source and destination nodes when computing the final edge representations, thereby improving the overall performance of the GCN.

The reconstruction error is defined as:

$$\mathcal{L}_G = \frac{1}{m} \sum_{i=1}^m \|x_i - x'_i\|_F^2, \quad (6)$$

where $\|\cdot\|_F^2$ denotes the squared Frobenius norm [9]. All weight matrices W_i are learnable parameters optimized by minimizing this reconstruction error on benign traffic during training. For attack flows—where both flow characteristics and source/destination node behaviors deviate from the benign baseline—the reconstruction of X' from L results in higher errors. Accordingly, the proposed method leverages the separability of reconstruction-error-based Autoencoders and formulates anomaly detection as a binary classification task:

$$\mathcal{F} : e \rightarrow Y, \quad e \in E, \quad Y \in \{0, 1\}, \quad (7)$$

where normal flows are assigned to class 0 and anomalous flows to class 1. The final output is a binary label $Y \in \{0, 1\}$ per edge $e \in E$, obtained by comparing its reconstruction error with an optimal threshold, as described in the following section.

3.3 Detection of Abnormal Network Traffic

Since the GCN-AE is trained exclusively on normal traffic graphs to minimize reconstruction error, deviations from this baseline produce higher errors that serve as anomaly scores. A key component of this process is the accurate determination of the threshold that separates normal from anomalous behavior. Threshold optimization is performed automatically using 20% of the data while preserving the attack-to-benign ratio. Specifically, the optimal threshold is selected by maximizing the weighted F1-score (W-F1) on a labeled validation set, following standard semi-supervised practices for reconstruction-based anomaly detection [25]. The procedure is defined as follows:

1. After training on normal traffic, compute the range of reconstruction errors using the benign validation set.

2. Generate candidate thresholds based on sorted error values across selected percentiles.
3. Evaluate each candidate threshold on the labeled validation set and select the one that maximizes the W-F1 score.

This approach trades increased computational cost for improved threshold optimality, which can be controlled by adjusting the percentile resolution.

As a summary, the proposed model operates in an inductive manner [13]. During training, the model weights are optimized for identifying normal traffic. During testing, to distinguish between normal and abnormal traffic, the model operates as a mask that traverses all nodes in the network. It begins by applying the learned weights to the feature vector of each edge. Then, it aggregates edge-level information at the node level to capture the sending and receiving behavior of each node. Based on these aggregated features, the model infers the behavior of each node in the context of its connected edges. Finally, it combines the features of each edge with those of its source and destination nodes to determine whether the edge exhibits anomalous behavior. Notably, this process is not constrained by the topology of the training graph and can be applied to any graph of arbitrary size and topology during the testing phase.

4. EXPERIMENTAL RESULTS

This section presents a comprehensive evaluation of the proposed approach. It begins with a description of the dataset used in the experiments, followed by an explanation of the performance evaluation metrics. Finally, the comprehensive experimental results are reported. All experiments were conducted on a Windows 10 machine equipped with a 13th Gen Intel(R) Core(TM) i7-13700K CPU @ 3.40 GHz and 62 GB of RAM. The experiments were implemented in Python 3.9.17 using TensorFlow 2.10.1 and PyTorch 2.4.1.

4.1 Dataset

We utilize the NetFlow dataset **NF-UQ-NIDS-v1**, a comprehensive dataset that integrates multiple publicly available NetFlow-based intrusion detection datasets [24]. This integration results in a large scale and more generalizable NIDS dataset that encompasses network traffic from diverse environments and various types of attack scenarios. The **NF-UQ-NIDS-v1** dataset contains a total of 11,994,893 flow records, of which 9,208,048 (76.77%) are benign and 2,786,845 (23.23%) are malicious. A summary of the distribution of attack categories is provided in Table 1.

To enable a comparative analysis of our proposed model with prior works, we additionally employ the **NF-CSE-CIC-IDS2018-v1** dataset, which is a subset of **NF-UQ-NIDS-v1**. This dataset consists of 8,392,401 flow records, including 7,373,198 benign samples (87.86%) and 1,019,203 attack samples (12.14%). The attack classes represented in this dataset include Brute Force, Bot, DoS, DDoS, Infiltration, and Web Attacks.

Table 1: *Distribution of Attacks: Pre- and Post-Aggregation.*

Type	# of samples	
	Pre-Aggregation	Post-Aggregation
Benign	9,208,048	466,721
DDoS	763,285	87
Reconnaissance	482,946	72
Injection	468,575	117
DoS	348,962	80
Brute Force	291,955	9
Password	156,299	25
XSS	99,944	12
Infiltration	62,072	4,022
Exploits	24,736	40
Scanning	21,467	13
Fuzzers	19,463	40
Backdoor	17,247	56
Bot	15,683	1
Generic	5,570	40
Analysis	1,995	25
Theft	1,909	6
Shellcode	1,365	40
MITM	1,295	157
Worms	153	40
Ransomware	142	3

Considering the aggregation process described in Section 3.1, and as illustrated in Table 1, the traffic volume associated with certain attack types is significantly reduced. However, by carefully selecting a feature set that effectively discriminates between normal and anomalous traffic, the aggregation process remains lossless, ensuring that no meaningful information is discarded.

To reduce input dimensionality while preserving discriminative information, we employ the Kolmogorov-Smirnov (KS) test [26] to identify features that exhibit the greatest separability between benign (class 0) and attack (class 1) traffic. This feature selection process reduces model complexity, accelerates convergence, and maintains high detection performance. As illustrated in Figure 2, among the original features, only Dur and OutB demonstrate strong separability. However, edge-based flow aggregation (Section 3.1) further enhances this capability, increasing both feature count and separability. Notably, more features now achieve KS statistics > 0.25 , including connection-level attributes such as flow count and active port counts alongside statistical features.

As shown in Figure 2, aggregating edge features at source and destination nodes during GCN training (Section 3.2) yields even greater separability than raw feature aggregation. This improvement arises implicitly from the graph-based learning process and highlights the effectiveness of GCN models in capturing structural dependencies, thereby improving the distinction between normal and anomalous traffic.

4.2 Performance Evaluation Metrics

In this study, we employ an Autoencoder as a binary classifier, designating attack samples as the positive class and normal samples as the negative class. Given the class

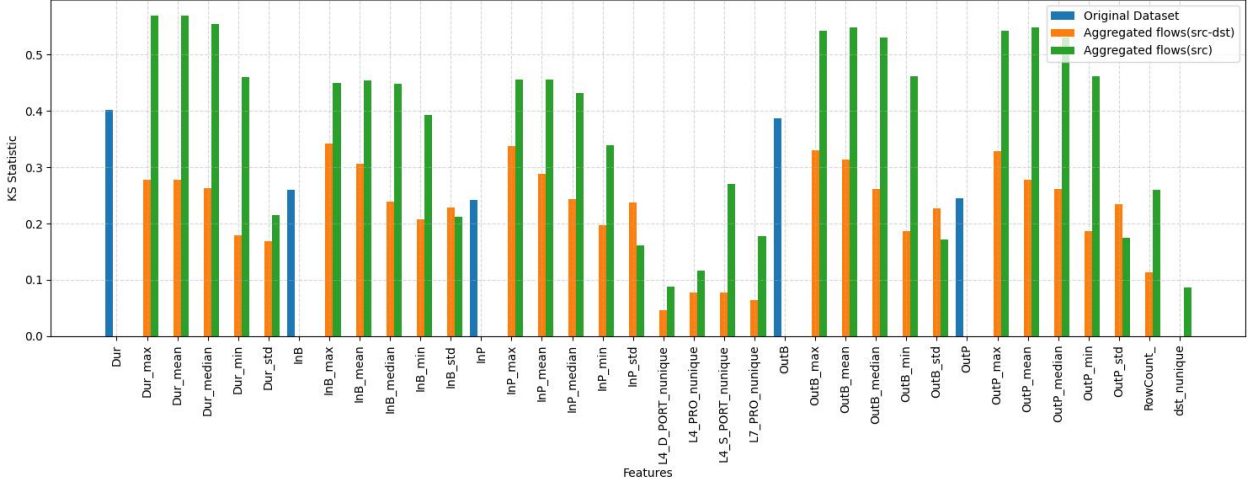


Fig. 2: KS Test Results: Original Dataset, Aggregated Dataset (Edges), and Aggregated Dataset (Source Nodes).

imbalance in the dataset—where approximately 76% of the traffic is normal—the selection of appropriate evaluation metrics is crucial. Let true positives (TP) denote correctly identified attacks; true negatives (TN) correctly identified normal samples, false positives (FP) normal samples misclassified as attacks, and false negatives (FN) attacks missed by the model. The definitions are as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (8)$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (9)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (10)$$

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (11)$$

Although accuracy reflects the overall correctness of predictions, it can be misleading in the presence of imbalanced datasets. Precision focuses on reducing false positives but may overlook missed attacks, while recall ensures the detection of all attacks but ignores false alarms. The F1-score, which balances precision and recall, is therefore more appropriate for imbalanced settings.

Considering the class imbalance in our dataset, we adopt the weighted F1-score (W-F1) as the primary evaluation metric. The W-F1 accounts for the contributions of both the attack (positive) and normal (negative) classes, weighted according to their respective sample sizes:

$$\text{W-F1} = \frac{\text{F1}_p \times N_p + \text{F1}_n \times N_n}{N_p + N_n} \quad (12)$$

Here, F1_p and F1_n are the F1-scores for the attack (positive) and normal (negative) classes, respectively, while N_p and N_n denotes the number of samples in the attack and normal classes, respectively. To ensure the Autoencoder's performance reflects the class imbalance, we select model parameters and the final model by maximizing the W-F1 as our objective.

Table 2: GCN-AE Model Settings.

Hyper-parameter	Value
Number of epochs	300
Optimizer	Adam
Hidden activation	ReLU
Input feature dimension	30
Node feature dimension	32
Latent feature dimension	32
Learning Rate (Lr)	[0.1, 0.01]
Dropout Rate (Dr)	[0, 0.1, 0.01]
Number of Encoding Layers	2
Number of Decoding Layers	2

4.3 Results

The numerical results are presented in three main sections. The first section details dataset partitioning. The second section provides a comprehensive evaluation of the proposed model's performance across various metrics. Finally, the third section assesses the robustness of the model under attacks such as zero-day and evasion attacks.

Six different architectures were evaluated under varying learning rates (LRs) and dropout rates (DRs). All models were trained using the settings listed in Table 2. For models that include a dropout layer, it is applied after the graph convolutional layer in the encoder. This strategy helps mitigate over-fitting and improves generalization.

The proposed model consists of two GCN-based encoding layers and two decoding layers, with layer dimensions defined by the corresponding input, latent, and node feature sizes. It should be noted that the total number of learnable parameters is determined by the sizes of the encoder and decoder weight matrices, which depend on the model architecture parameters D_e , h_e , and h_n , corresponding to the input size, latent size, and

node feature size, respectively, rather than the graph size. Therefore, the number of learnable parameters remains fixed across all evaluated scenarios.

4.3.1 Dataset Partitioning: Training, Validation, and Test Sets

The dataset comprises approximately 93,000 IP addresses, the majority of which exhibit sparse activity. As illustrated in Figure 3, the cumulative contribution of nodes to the network traffic is presented. Notably, approximately 50% of the total network traffic can be accounted for by considering only 270 nodes. Including nodes with sparse behavior increases the computational complexity of model training due to the enlarged graph size, and it also impedes convergence since these nodes contribute limited informative content to the model. Conversely, utilizing an insufficient number of edges for training results in inadequate learning of normal behavior, thereby diminishing the model’s overall effectiveness.

To select an appropriate number of nodes—and consequently edges—a predefined threshold is applied to exclude nodes exhibiting sparse behavior. This threshold serves as a criterion to identify nodes with normal behavior that participate sufficiently in the network to be included in the training dataset. The optimal threshold value is determined through a grid search procedure. For each candidate threshold, the corresponding number of nodes and the normal edges connecting them define the network graph utilized during the training phase.

Figure 4 illustrates the performance of different models evaluated with node counts of 230, 270, and 285, based on the parameters outlined in Table 2. As shown, the configuration with $n = 270$ nodes outperforms both the smaller network with $n = 230$, which suffers from an insufficient number of edges, and the larger network with $n = 285$, where the detrimental effects of sparsity on performance become evident. Therefore, $n = 270$ is selected as the optimal configuration for all subsequent experiments and result reporting.

Table 3 reports the average training time, test time, and memory consumption for the three evaluated configurations. As expected, increasing the number of nodes, and consequently the number of edges, results in higher computational cost during both training and inference. Nevertheless, the observed test times remain low across all configurations, indicating that the proposed model is suitable for efficient real-world detection. It is important to note that all configurations share an identical number of learnable parameters; therefore, the observed differences in execution time arise solely from variations in graph size (i.e., the number of nodes and edges).

After selecting the optimal configuration, 70% of benign edges were allocated for training, 10% for validation, and the remaining 20%, together with all attack-related traffic in the dataset, were reserved for testing. It should be noted that 10% of the attack traffic was included in the validation set. All results were averaged over 10-

fold cross-validation, with data splitting performed using stratified random sampling for each fold. Additionally, since the model operates within a binary classification framework, traffic is categorized as either normal or abnormal. All attack flows, regardless of attack type, are treated as abnormal. Consequently, the attack traffic in the dataset, comprising 3,489 flows, involves 1,526 distinct nodes.

4.3.2 Performance Evaluation of the Proposed Model

The results presented here correspond to the configuration with 270 nodes. Initially, model performance is evaluated by analyzing the average loss across training epochs, as illustrated in Figure 5. It can be observed that all models converge within the first 100 epochs. A more detailed examination of the subsequent two sets of 100 epochs indicates that higher learning rates accelerate convergence, while lower dropout rates reduce oscillations and further improve convergence stability. Despite differences in learning rates and dropout rates, all models ultimately converge, exhibiting comparable performance.

Following training, as described in Section 3.3, the range of reconstruction errors for normal traffic is estimated. The optimal threshold is then determined by evaluating model performance across this range using the validation set (Section 3.3).

Figure 7 illustrates this process for the top-performing GCN-AE model (Table 4). As discussed in Section 3.3, this percentile-based search trades computational complexity for optimality; Due to the limited number of outliers in the benign validation set, performance remains relatively stable at lower thresholds and increases sharply in the mid-range, where the optimal threshold is identified, as illustrated in Figure 7. Based on these optimal thresholds, the evaluation results for all models are reported in Table 4.

As presented in Table 4, the models demonstrate relatively strong performance. The reported results represent the average performance obtained through 10-fold cross-validation, facilitating a more robust and reliable comparison. To provide additional insight into the models’ behavior, the best result among the 10 folds for each model is also included. The highest performance was achieved with the configuration $LR = 0.01$ and $DR = 0.01$. Furthermore, Figure 6 displays the ROC and precision-recall curves, offering a comparative assessment of model performance. As observed, the proposed GCN-AE model maintains consistent performance across various configurations and exhibits stable behavior.

Comparison with Existing Approaches: The primary dataset employed in this study is Version 1, whereas most existing works [11, 18, 24] have utilized Version 2, which includes a greater number of features and, consequently, more comprehensive information, and have reported their results accordingly. In contrast, our approach achieves notable performance using a dataset

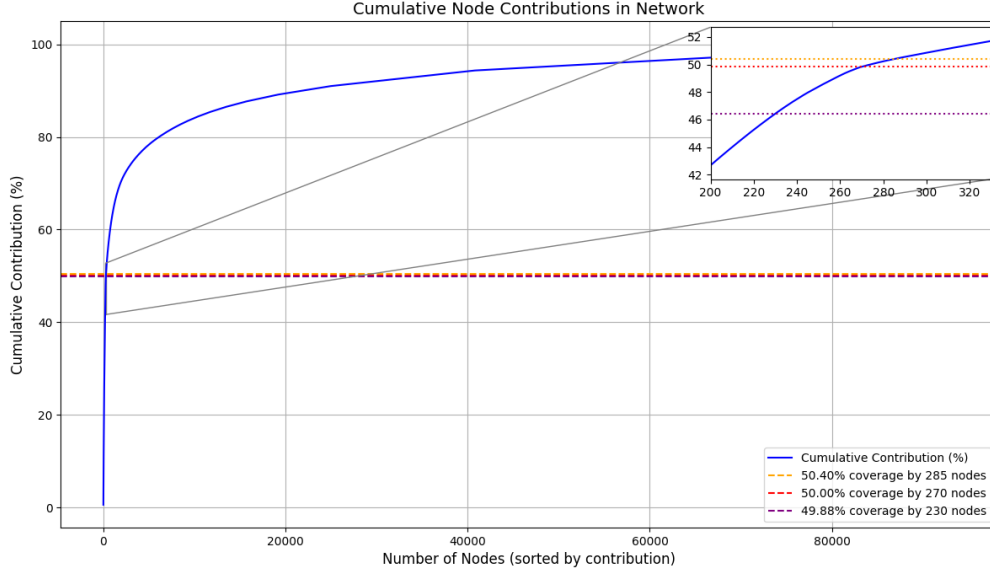


Fig. 3: Cumulative Contribution of Nodes in the Network.

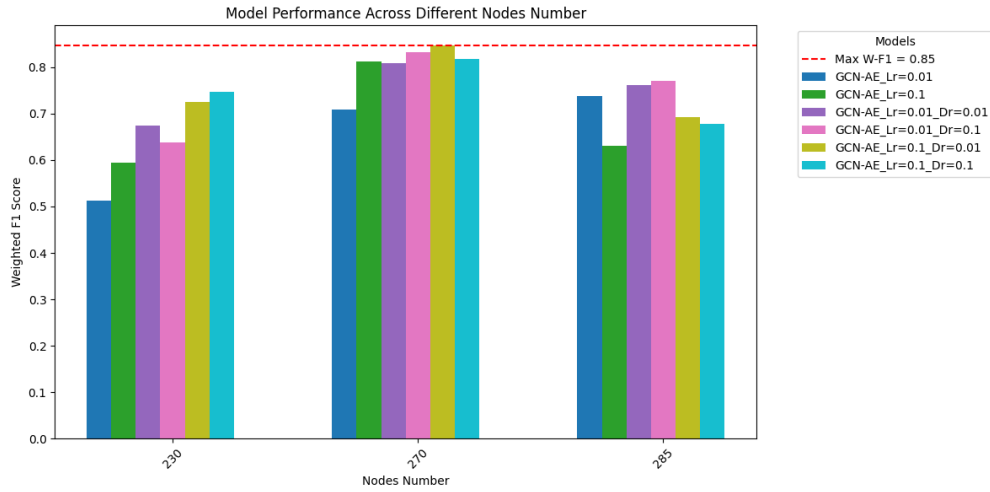


Fig. 4: Model Performance Across Different Node Counts.

Table 3: Time and Memory Complexity of Models.

# of nodes	Train Time (S)	Train Mem (B)	Benign Edges	Train Size	Test Time (S)	Test Mem (B)
230	0.43	493.77	783	548	0.005	494
270	0.57	511.37	1765	1235	0.006	524
285	0.74	618.83	4827	3378	0.007	614

with a minimum of only five features, yielding substantial improvements as summarized in Table 5.

To ensure a fair evaluation and to demonstrate the effectiveness of the proposed model relative to prior studies, additional experiments were conducted on the NF-CSE-CIC-IDS2018-v1 dataset, which constitutes a subset of our primary dataset. The results, presented in Table 5, indicate that the proposed model achieves significant performance improvements compared to existing

methods.

4.4 Robustness Evaluation of the GCN-AE Model

In this section, to further substantiate the efficacy of the proposed model, we examine the resilience of the GCN-AE against (1) zero-day attacks and (2) evasion attacks, considering two representative scenarios within these categories.

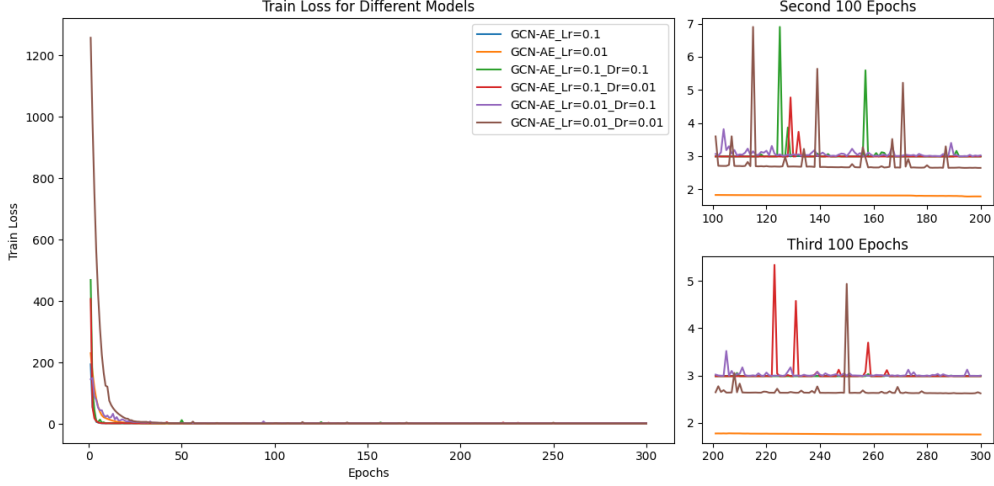


Fig. 5: Average Training Loss Over Epochs.

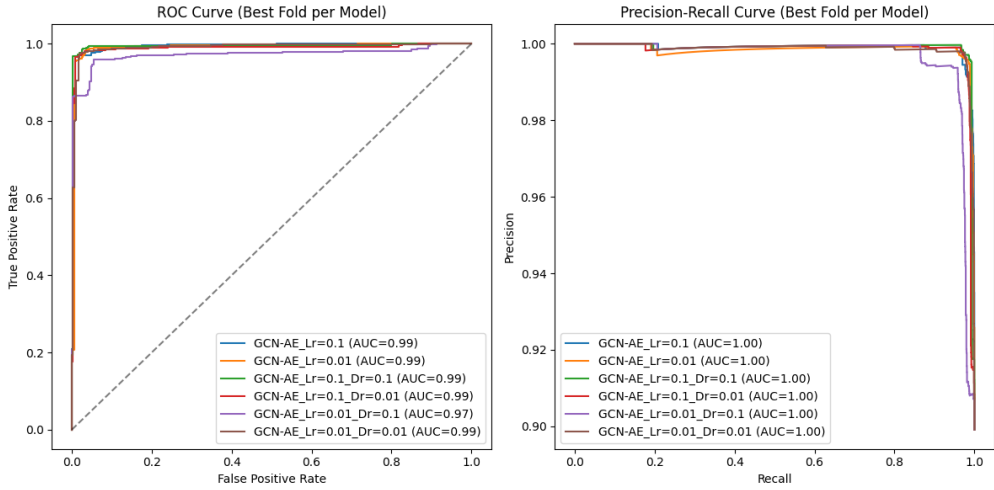


Fig. 6: ROC and Precision-Recall Curves of the GCN-AE.

Table 4: Evaluation Results of GCN-AE.

Model	Mean $\pm$ Std				Best			
	Accuracy	W-F1	Precision	Recall	Accuracy	W-F1	Precision	Recall
Lr = 0.1	0.966 $\pm$ 0.013	0.964 $\pm$ 0.014	0.964 $\pm$ 0.014	0.966 $\pm$ 0.013	0.981	0.980	0.980	0.981
Lr = 0.1, Dr = 0.1	0.959 $\pm$ 0.020	0.955 $\pm$ 0.028	0.958 $\pm$ 0.022	0.959 $\pm$ 0.020	0.979	0.979	0.979	0.979
Lr = 0.1, Dr = 0.01	0.957 $\pm$ 0.013	0.955 $\pm$ 0.015	0.955 $\pm$ 0.016	0.957 $\pm$ 0.013	0.976	0.976	0.976	0.976
Lr = 0.01	0.935 $\pm$ 0.022	0.919 $\pm$ 0.039	0.915 $\pm$ 0.057	0.935 $\pm$ 0.022	0.969	0.967	0.968	0.969
Lr = 0.01, Dr = 0.1	0.937 $\pm$ 0.025	0.934 $\pm$ 0.026	0.937 $\pm$ 0.022	0.937 $\pm$ 0.025	0.966	0.967	0.968	0.966
Lr = 0.01, Dr = 0.01	0.971 $\pm$ 0.018	0.968 $\pm$ 0.023	0.971 $\pm$ 0.016	0.971 $\pm$ 0.018	0.987	0.987	0.987	0.987

4.4.1 Robustness to Zero-day Attacks

This section evaluates the generalization capability of the proposed model to unseen attack types. In this setting, one attack type is excluded from the validation data during threshold optimization, while the remaining known attack types are retained. The excluded attack type is subsequently used only for testing. This leave-one-attack-type-out evaluation assesses whether the model can detect zero-day attacks without incor-

porating their reconstruction errors into the threshold selection process. The results are presented in Table 6. The reported results are averaged over 10-fold cross-validation. As shown, the model maintains stable performance under zero-day attack scenarios. This robustness stems from its ability to learn a compact representation of normal network behavior during training, allowing deviations from this baseline to be effectively identified as anomalies.

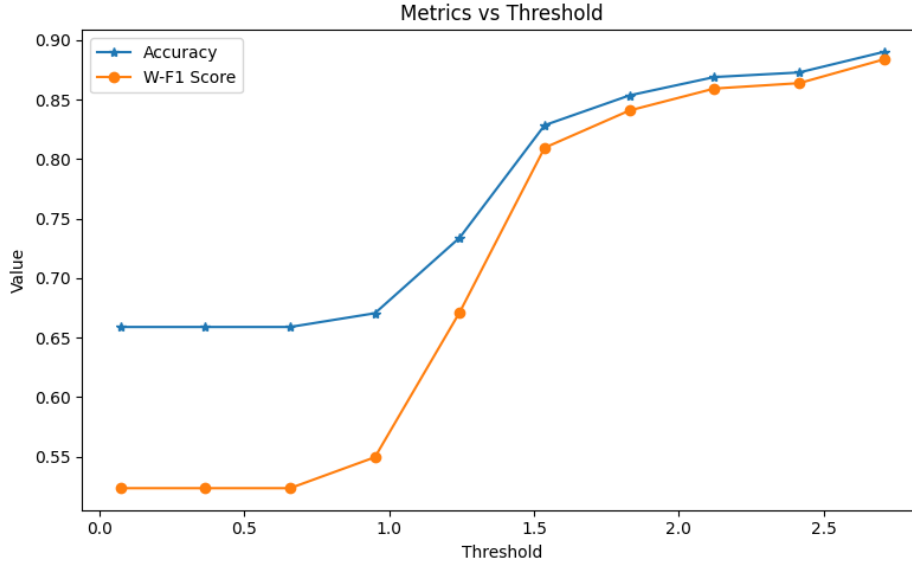


Fig. 7: Maximizing W-F1 score during Threshold Selection.

Table 5: GCN-AE and Existing Approaches Comparison.

Dataset	Accuracy	Precision	Recall	W-F1 Score	Reference
NF-CSE-CIC-IDS2018-v1	95.33	0.00	94.71	83.00	[24]
	97.66	96.75	96.77	96.76	[18]
	0.00	0.00	96.82	98.23	MLP-[18]
	98.8	98.8	98.8	98.8	GCN-AE
NF-UQ-NIDS-v1	97.25	0.00	95.66	94.00	[24]
	98.8	98.8	98.8	98.8	GCN-AE

Table 6: Performance Metrics for Zero-Day Attack Scenarios.

ZeroDay Type	W-F1 (mean $\pm$ std)	Accuracy (mean $\pm$ std)	Precision (mean $\pm$ std)	Recall (mean $\pm$ std)
Analysis	0.9799 $\pm$ 0.0104	0.9802 $\pm$ 0.0097	0.9799 $\pm$ 0.0104	0.9802 $\pm$ 0.0097
Backdoor	0.9801 $\pm$ 0.0105	0.9805 $\pm$ 0.0098	0.9802 $\pm$ 0.0105	0.9805 $\pm$ 0.0098
Bot	0.9804 $\pm$ 0.0103	0.9807 $\pm$ 0.0096	0.9804 $\pm$ 0.0103	0.9807 $\pm$ 0.0096
Brute Force	0.9793 $\pm$ 0.0102	0.9797 $\pm$ 0.0095	0.9793 $\pm$ 0.0102	0.9797 $\pm$ 0.0095
DDoS	0.9801 $\pm$ 0.0104	0.9804 $\pm$ 0.0097	0.9801 $\pm$ 0.0104	0.9804 $\pm$ 0.0097
DoS	0.9803 $\pm$ 0.0099	0.9807 $\pm$ 0.0092	0.9804 $\pm$ 0.0099	0.9807 $\pm$ 0.0092
Exploits	0.9801 $\pm$ 0.0101	0.9804 $\pm$ 0.0095	0.9801 $\pm$ 0.0102	0.9804 $\pm$ 0.0095
Fuzzers	0.9803 $\pm$ 0.0091	0.9806 $\pm$ 0.0084	0.9803 $\pm$ 0.0089	0.9806 $\pm$ 0.0084
Generic	0.9800 $\pm$ 0.0110	0.9803 $\pm$ 0.0103	0.9800 $\pm$ 0.0111	0.9803 $\pm$ 0.0103
Infiltration	0.9829 $\pm$ 0.0078	0.9832 $\pm$ 0.0070	0.9829 $\pm$ 0.0074	0.9832 $\pm$ 0.0070
Reconnaissance	0.9801 $\pm$ 0.0109	0.9804 $\pm$ 0.0103	0.9801 $\pm$ 0.0110	0.9804 $\pm$ 0.0103
Shellcode	0.9802 $\pm$ 0.0106	0.9806 $\pm$ 0.0099	0.9802 $\pm$ 0.0106	0.9806 $\pm$ 0.0099
Theft	0.9793 $\pm$ 0.0109	0.9796 $\pm$ 0.0102	0.9792 $\pm$ 0.0110	0.9796 $\pm$ 0.0102
Worms	0.9801 $\pm$ 0.0103	0.9804 $\pm$ 0.0096	0.9801 $\pm$ 0.0103	0.9804 $\pm$ 0.0096
Injection	0.9795 $\pm$ 0.0104	0.9798 $\pm$ 0.0097	0.9795 $\pm$ 0.0104	0.9798 $\pm$ 0.0097
MITM	0.9811 $\pm$ 0.0103	0.9814 $\pm$ 0.0097	0.9811 $\pm$ 0.0103	0.9814 $\pm$ 0.0097
Password	0.9796 $\pm$ 0.0105	0.9799 $\pm$ 0.0098	0.9796 $\pm$ 0.0105	0.9799 $\pm$ 0.0098
Ransomware	0.9807 $\pm$ 0.0103	0.9810 $\pm$ 0.0096	0.9807 $\pm$ 0.0103	0.9810 $\pm$ 0.0096
Scanning	0.9797 $\pm$ 0.0106	0.9800 $\pm$ 0.0099	0.9796 $\pm$ 0.0106	0.9800 $\pm$ 0.0099
XSS	0.9801 $\pm$ 0.0103	0.9804 $\pm$ 0.0096	0.9801 $\pm$ 0.0102	0.9804 $\pm$ 0.0096

Table 7: Accuracy and Weighted F1 Score at Varying Noise Levels in Adversarial Attacks.

Features	Metric	Noise				
		0	0.25	0.5	0.75	1
Dur	Accuracy	98.30	98.24	98.26	98.23	98.24
	W-F1 Score	98.30	98.23	98.27	98.23	98.24
Dur and OutB	Accuracy	98.28	90.36	90.36	90.36	90.36
	W-F1 Score	98.28	86.14	86.14	86.14	86.14

Table 8: Accuracy and Weighted F1 Score at Varying α Values in Camouflage Attacks.

Features	Metric	α				
		0	0.25	0.5	0.75	1
Dur	Accuracy	98.28	98.35	98.16	98.32	98.52
	W-F1 Score	98.28	98.35	98.15	98.32	98.51
Dur and OutB	Accuracy	98.35	98.46	98.46	98.56	99.04
	W-F1 Score	98.34	98.45	98.45	98.56	98.03

4.4.2 Robustness to Evasion Attacks

In this section, to further substantiate the efficacy of the proposed model against evasion attacks, we examine the resilience of the GCN-AE under two representative scenarios: adversarial attacks and camouflage attacks.

Robustness to Adversarial Attacks: Adversarial attacks [3] involve the intentional perturbation of input features to evade detection. These perturbations are designed to reduce the model’s confidence or shift its decision boundary while preserving the overall semantics of the input data.

To assess this scenario, we consider two settings: a single-feature attack using Dur, and a joint-feature attack using Dur and OutB, both identified by the KS test as highly discriminative features (Figure 2). Specifically, we apply noise levels of 0%, 0.25%, 0.5%, 0.75%, and 1% of the feature range and evaluate the resulting classification performance.

The results in Table 7 indicate that, although slight variations are observed across different noise levels, the proposed GCN-AE model maintains its separability capability, as the W-F1 score does not decrease significantly. These findings, consistent with earlier studies [8, ?], suggest that graph-based approaches are inherently robust to adversarial feature manipulation.

Robustness to Camouflage Attacks: Camouflage attacks [28] represent a more sophisticated evasion technique designed to make malicious flows resemble benign traffic. To simulate this scenario, we consider two settings: camouflage based on Dur only, and camouflage based on the joint features Dur and OutB. For a feature Feat, we adjust its value toward the benign mean using:

$$\text{Feat}_{\text{camouflaged}} = \text{Feat}_{\text{attack}} + \alpha \cdot (\mu_{\text{benign}} - \text{Feat}_{\text{attack}}) \quad (13)$$

where $\alpha \in \{0, 0.25, 0.5, 0.75, 1\}$ and μ_{benign} denotes the mean value of the corresponding benign feature.

In the joint-feature setting, this adjustment is applied separately to both Dur and OutB, moving attack flow features closer to the benign mean to simulate camouflage attempts. The results in Table 8 show that the proposed GCN-AE model remains stable even when attack flows are gradually shifted toward benign behavior. Across all tested values of α , the model consistently maintains strong accuracy and W-F1 scores.

5. CONCLUSION AND FUTURE WORKS

We presented GCN-AE, a one-class intrusion detection model that integrates Graph Convolutional Network with an Autoencoder framework to model normative network behavior without relying on labeled attack data. Through a non-standard message-passing mechanism, the model captures structural dependencies within network flows and effectively identifies anomalous deviations. Experimental evaluation on the NF-UQ-NIDS-v1 dataset demonstrates that GCN-AE outperforms recent approaches. The model’s effectiveness in detecting zero-day, adversarial, and camouflage attacks can be largely attributed to its ability to capture underlying structural patterns of malicious flows, which remain relatively invariant despite perturbations in individual flow features. As future work, we aim to extend the model to support multi-class intrusion detection by learning discriminative latent representations for different attack categories. In addition, we plan to investigate scalability to larger and more complex graph structures, optimize the model for real-time deployment, and further evaluate its robustness under more realistic zero-day and adversarial conditions.

REFERENCES

- [1] S. Morgan, “Cybercrime facts and statistics—report: Cyber warfare in the c-suite,” <https://cybersecurityventures.com/wp-content/>

- uploads/2021/01/Cyberwarfare-2021-Report.pdf, Mar. 2021.
- [2] I. Corona, G. Giacinto, and F. Roli, "Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues," *Information Sciences*, vol. 239, 2013.
 - [3] S. Shah, P. S. Muhuri, X. Yuan, K. Roy, P. Chatterjee, K. Rahman, and E. Gamess, "Implementing a network intrusion detection system using semi-supervised support vector machine and random forest," in *ACM Southeast Conference (ACM SE '21)*, USA, 2021, pp. 180–184.
 - [4] T. Le, H. Kim, H. Kang, and H. Kim, "Classification and explanation for intrusion detection system based on ensemble trees and shap method," *Sensors*, vol. 22, no. 3, p. 1154, 2022.
 - [5] B. S. Bhati, G. Chugh, F. Al-Turjman, and N. S. Bhati, "An improved ensemble based intrusion detection technique using xgboost," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 6, 2021.
 - [6] C. K. Joshi, C. Bodnar, S. V. Mathis, T. Cohen, and P. Liò, "On the expressive power of geometric graph neural networks," arXiv preprint arXiv:2301.09308, 2023.
 - [7] W. W. Lo, S. Layeghy, M. Sarhan, M. Gallagher, and M. Portmann, "E-graphsage: A graph neural network based intrusion detection system for iot," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2022.
 - [8] Z. Sun, A. M. H. Teixeira, and S. Toor, "Gnn-ids: Graph neural network based intrusion detection system," in *Proceedings of the 19th International Conference on Availability, Reliability and Security (ARES '24)*, 2024, pp. 1–12.
 - [9] J. Busch, A. Kocheturov, V. Tresp, and T. Seidl, "Nf-gnn: Network flow graph neural networks for malware detection and classification," in *Proceedings of the ACM Workshop*, 2021, pp. 121–132.
 - [10] E. Caville, W. W. Lo, S. Layeghy, and M. Portmann, "Anomale: A self-supervised network intrusion detection system based on graph neural networks," *Knowledge-Based Systems*, vol. 258, p. 110030, 2022.
 - [11] H. Nguyen and R. Kashef, "Ts-ids: Traffic-aware self supervised learning for iot network intrusion detection," *Knowledge-Based Systems*, vol. 279, 2023.
 - [12] J. Zhao, X. Liu, Q. Yan, B. Li, M. Shao, and H. Peng, "Multi attributed heterogeneous graph convolutional network for bot detection," *Information Sciences*, vol. 537, pp. 380–393, 2023.
 - [13] Z. Song, X. Yang, Z. Xu, and I. King, "Graph-based semi supervised learning: A comprehensive review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 11, pp. 8174–8194, 2023.
 - [14] H. Choi, M. Kim, G. Lee, and W. Kim, "Unsupervised learning approach for network intrusion detection system using autoencoders," *The Journal of Supercomputing*, vol. 75, no. 9, pp. 5597–5621, 2019.
 - [15] J. Ghadermazi, S. Hore, A. Shah, and N. D. Bastian, "Gtae-ids: Graph transformer-based autoencoder framework for realtime network intrusion detection," *IEEE Transactions on Information Forensics and Security*, vol. 20, pp. 4026–4041, 2025.
 - [16] M. Zhong, M. Lin, C. Zhang, and Z. Xu, "A survey on graph neural networks for intrusion detection systems: Methods, trends and challenges," *Computers & Security*, vol. 141, p. 103821, 2024.
 - [17] Premkumar, M. Schneider, C. Spivey, J. V. Pavlik, and N. D. Bastian, "Graph representation learning for context-aware network intrusion detection," in *Proc. SPIE*, vol. 12538, 2023, pp. 82–92.
 - [18] R. Xu, G. Wu, W. Wang, X. Gao, A. He, and Z. Zhang, "Applying self-supervised learning to network intrusion detection for network flows with graph neural network," *Computer Networks*, vol. 248, p. 110495, 2024.
 - [19] D. Pujol-Perich, J. Suárez-Varela, A. Cabellos-Aparicio, and P. Barlet-Ros, "Unveiling the potential of graph neural networks for robust intrusion detection," *ACM SIGMETRICS Performance Evaluation Review*, vol. 49, no. 4, pp. 111–117, 2024.
 - [20] P. K. Mvula, P. Branco, G. V. Jourdan, and H. L. Viktor, "A survey on the applications of semi-supervised learning to cyber-security," *ACM Computing Surveys*, vol. 56, no. 10, Jun. 2024.
 - [21] R. Zhao, Y. Wang, Z. Xue, T. Ohtsuki, B. Adebisi, and G. Gui, "Semi-supervised federated learning based intrusion detection method for internet of things," *IEEE Internet of Things Journal*, vol. 10, no. 10, pp. 8645–8657, 2023.
 - [22] Y. Zhang and J. Yan, "Semi-supervised domain-adversarial training for intrusion detection against false data injection in the smart grid," in *Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–7.
 - [23] P. Sarantos, J. Violos, and A. Leivadreas, "Enabling semisupervised learning in intrusion detection systems," *Journal of Parallel and Distributed Computing*, vol. 196, p. 105010, 2025.
 - [24] M. Sarhan, S. Layeghy, N. Moustafa, M. Portmann, Z. Deze, H. X. Huang, R. Hou, S. Rho, and N. Chilamkurti, "Netflow datasets for machine learning-based network intrusion detection systems," in *Big Data Technologies and Applications (BDTA)*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 371, Springer, Cham, 2021.
 - [25] M. A. Salahuddin, V. Pourahmadi, H. A. Alameddine, M. F. Bari, and R. Boutaba, "Chronos: Ddos attack detection using time-based autoencoder," *IEEE Transactions on Network and Service Management*, vol. 19, no. 1, pp. 627–641, 2022.
 - [26] F. J. Massey, "The kolmogorov-smirnov test for goodness of fit," *Journal of the American Statistical*

Association, vol. 46, no. 253, 1951.

- [27] X. Zhou, W. Liang, W. Li, K. Yan, S. Shimizu, and K. I. K. Wang, "Hierarchical adversarial attacks against graphneural-network-based iot network intrusion detection system," *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 9310–9319, 2022.
- [28] Z. Lu, G. Liu, L. Lai, and W. Xu, "Camouflage adversarial attacks on multiple agent systems," arXiv preprint arXiv:2401.17405, 2024.



Zahra Eskandari received the B.S. degree in Computer Engineering from Kharazmi University, Tehran, Iran, in 2006, and the M.S. and Ph.D. degrees in Computer Engineering from Ferdowsi University of Mashhad, Mashhad, Iran, in 2008 and 2020, respectively. From July 2016 to March 2017, she was a visiting researcher with the Cybersecurity Section, DTU Compute, Technical University of Denmark. She is currently a full-time Assistant Professor in the Department of Computer

Engineering, Faculty of Electrical and Computer Engineering, Quchan University of Technology, Quchan, Iran. Her research interests include computer network and IoT security, and leveraging machine learning to advance the field.