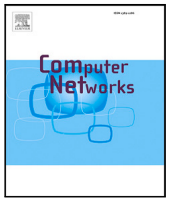
Computer Networks 250 (2024) 110508

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



FTG-Net-E: A hierarchical ensemble graph neural network for DDoS attack detection

Rana Abu Bakar [a,*], Lorenzo De Marinis [a], Filippo Cugini [b], Francesco Paolucci [b]

[a] Scuola Superiore Sant'Anna, PISA, 56124, Italy
[b] CNIT, PISA, 56124, Italy

ARTICLE INFO

Keywords:
DDoS
Network security
Cybersecurity
Deep learning
Ensemble learning
Attack detection
Graph neural networks

ABSTRACT

Distributed Denial-of-Service (DDoS) attacks are a major threat to computer networks. These attacks can be carried out by flooding a network with malicious traffic, overwhelming its resources, and/or making it unavailable to legitimate users. Existing machine learning methods for DDoS attack detection typically use statistical features of network traffic, such as packet sizes and inter-arrival times. However, these methods often fail to capture the complex relationships between different traffic flows. This paper proposes a new DDoS attack detection approach that uses Graph Neural Networks (GNN) ensemble learning. GNN ensemble learning is a type of machine learning that combines multiple GNN models to improve the detection accuracy. We evaluated our approach on the Canadian Institute for Cybersecurity Intrusion Detection Evaluation Dataset (CICIDS2018) and CICIDS2017 datasets, a benchmark dataset for DDoS attack detection. Our work provides two main contributions. First, we extend our DDoS attack detection approach using GNN ensemble learning. Second, we explore the evaluation and fine-tuning of hyperparameter metrics through ensemble learning, significantly enhancing accuracy compared to a single GNN model and achieving an average 3.2% higher F1-score. Additionally, our approach effectively reduces overfitting by incorporating regularization techniques, such as dropout and early stopping. Specifically, we use a hierarchical ensemble of GNN, where each GNN learns the relationships between traffic flows at a different granularity level. We then use bagging and boosting to combine the predictions of the individual GNN, further improving detection accuracy. Results show that our system can achieve 99.67% accuracy, with a F1-score of 99.29%, which is better than state-of-the-art methods, even using single traffic architecture.

1. Introduction

The surge in digital technologies has fueled a rise in cyber-attacks, disrupting operations and causing significant economic, political, military, and privacy-related damages. As per the Federal Bureau of Investigation (FBI) Internet Crime Complaint Center (IC3) report [1], the number of cybercrime complaints in 2021 reached 847,376, marking a 7% increase from 2020 and a staggering 181% jump from 2017. The estimated financial losses due to these attacks exceeded $6.9 billion. Among the most concerning cyber threats is DDoS attacks, which exploit the capacity limits of network resources by orchestrating a massive influx of requests from numerous devices, effectively clogging access to the resource for legitimate users. A prominent example of a DDoS attack driven by political motives was the one launched by the ALtahrea Team against public websites in the UK and Israel during the second quarter of 2022 [2].

DDoS attacks are a prevalent cyber threat characterized by a malicious attempt to overwhelm a target's online resources, rendering them unavailable to legitimate users. These attacks can be categorized into three primary types, each with unique characteristics and implications.

1. Volumetric Attacks: Flooding the Network with Legitimate Traffic - Volumetric DDoS attacks flood the target network with an overwhelming volume of legitimate traffic, typically generated by a botnet, a network of infected devices controlled by an attacker. This massive influx of traffic saturates the network's bandwidth, choking it to a standstill. As a result, legitimate users cannot access the target website, application, or service.
2. Protocol Attacks: Exploiting Layer 3 and 4 Vulnerabilities - Protocol attacks target network protocols operating at Layer 3 (network layer) and 4 (transport layer). These attacks aim to consume the resources of the target server or intermediate communication equipment, such as firewalls and load balancers, rendering them unable to handle legitimate traffic. Common protocol attacks include SYN flood attacks, UDP flood attacks, and ICMP flood attacks.

* Corresponding author.
E-mail address: Rana.abubakar@santannapisa.it (R. Abu Bakar).

https://doi.org/10.1016/j.comnet.2024.110508
Received 20 December 2023; Received in revised form 6 March 2024; Accepted 12 May 2024
Available online 17 May 2024

3. Application Attacks: Exploiting Vulnerabilities at Layer 7 - Application layer attacks target the application layer (Layer 7), the highest layer of the network stack. These attacks utilize legitimate requests to exploit vulnerabilities within the target application, consuming computing resources, such as database queries or read file. Unlike volumetric and protocol attacks, application layer attacks may be more challenging to detect as they often blend in with legitimate traffic.

A comprehensive detection system is essential to detect and mitigate DDoS attacks effectively. This system should be able to analyze both aggregated traffic patterns and specific traffic flows to identify anomalies that may indicate an attack. Analyzing aggregated traffic provides insights into overall network behavior while examining specific traffic flows helps identify suspicious activity between two endpoints. Combining these perspectives allows a flexible detection system to distinguish between legitimate traffic and malicious attacks. The DDoS detection system must reach a trade-off between (i) the delay due to huge receiving traffic under analysis (to deliver an adequate overview to monitor probable malicious patterns) and (ii) the reactivity to implement appropriate mitigation and security rules before consequential damages.

Moreover, DDoS mitigation strategies should also consider the overall network's computational capabilities. Overburdening the network with intensive processing tasks can lead to performance bottlenecks and hinder the ability to respond to other network traffic.

Detection of DDoS attacks using traditional Machine Learning (ML) and Deep Learning (DL) techniques involves utilizing traffic-level or flow-level features that are statistical representations of the data associated with the transmitted packets within a certain flow or time window. These features are significant in identifying potential attacks. Those approaches perform well when trained on famous Intrusion Detection System (IDS) datasets but lack adoption in real-time systems due to their incapacity to generalize and be flexible to different traffic profiles and networks [3]. To tackle this issue, one can shift from statistical aggregation to delving into the structure of traffic flows. This involves examining sequences of exchanged packets between two endpoints and considering aggregated traffic, defined as the collection of flows established among endpoints within a specific time window. This augmented topological knowledge allows the detection of common structural patterns in specific DDoS attack types. A recent breakthrough has emerged with the application of Graph Neural Networks (GNN) to the problem of Distributed Denial of Service (DDoS Detection System (DDS)) [4]. This approach harnesses topological information to enhance both robustness and detection accuracy. However, analyzing the intra- and inter-entity flow level connections is essential for capturing all potential attack patterns.

This paper is the invited extended version of our previous work [5]. In our recent preliminary work [5], we introduced a hierarchical graph structure known as Flow-to-Traffic Graph (FTG), incorporating aggregated traffic-level and flow-level structures in a two-level representation. Additionally, we proposed the FTG-Net model: a GNN designed to process these FTG graphs and effectively classify flows as either legitimate or malicious. This approach adeptly captures and embeds the intricate flow structures between hosts and servers, combining this representation with the overall traffic structure. By doing so, it accommodates various DDoS attacks within the recognized patterns. Significantly, this solution relies solely on traffic topology, eliminating the need for computationally expensive stateful features in real-time scenarios. This divergence avoids potential overfitting to specific training dataset characteristics, as commonly encountered in models incorporating all stateful features [6]. When deploying FTG-Net in real-world scenarios, it is important to integrate specific stateful features into the graph node. These features should be contingent upon the network requirements and functionalities, as they can greatly enrich the representation and ultimately improve the performance. While our approach demonstrated impressive accuracy, we recognized that the battle against DDoS attacks is an ever-evolving challenge. These attacks continuously adapt, seeking to mimic legitimate traffic patterns and evade detection mechanisms.

This paper completes and extends our work [5] by introducing a novel way to detect DDoS attacks using ensemble learning with GNN. Ensemble learning is a method that combines multiple machine learning models to improve the system overall performance [7]. In our case, we use a combination of GNNs to detect DDoS attacks. By learning from the structure of the network traffic, GNNs can identify patterns indicative of DDoS attacks. This is because each GNN model will learn different patterns from the data, and by combining the predictions of the different models, we can get a more accurate overall prediction. Our approach employs a subsampling strategy to reduce the computational complexity of training GNN models on large graphs. This involves dividing the graph into smaller subgraphs and training separate GNN models on each subgraph. This reduces the number of nodes and edges that each GNN model needs to process, significantly improving training efficiency. Instead of using computationally expensive models, we utilize lightweight graph networks that have a simpler architecture and fewer parameters. This further reduces the computational requirements of training and inference. For instance, we employed Graph Convolutional Networks (GCN) with fewer convolutional layers. The GCN can help to make a model from the network topology represented as graphs. They enable the extraction of spatial features by learning from the relationships between nodes and their neighbors. They can extract temporal features, which allow over time to capture changes in network [8]. This lightweight GNN architecture aggregates information from neighboring nodes more efficiently. Ensemble learning involves combining the predictions from multiple models to improve overall accuracy. Our approach trains multiple GNN models on different subgraphs and aggregates their predictions to produce the final output. This approach enhances accuracy and reduces the computational burden compared to training a single large GNN model on the entire graph.

We evaluated our approach using two well-known network traffic datasets on CIC-IDS2017 and CICIDS2018 [9]. The results show that our system can achieve enhanced accuracy and robustness compared to other approaches. Our approach significantly improves the accuracy performance compared to a single GNN model, achieving an average of 3.2% higher F1-score. Additionally, our approach effectively reduces overfitting by incorporating regularization techniques, such as dropout and early stopping.

The contributions of this paper are as follows:

- We transform the traffic data into a new hierarchical graph structure called FTG. It contains both flow and aggregate traffic details.
- We propose an extended FTG-Net [10] with GNN models, which can detect DDoS attacks exploiting the introduced FTG structures.
- We propose an ensemble GNN model, namely FTG-Net-E, to detect DDoS attacks exploiting the introduced FTG structures. It uses a hierarchical approach to capture fine-grained packet interactions and coarse-grained flow relationships. It also uses an ensemble learning approach for enhanced robustness with bagging and boosting techniques.
- We perform experimentation on CIC-IDS2017 and CICIDS2018 [9] to show that the information in the network traffic structure is enough to achieve better results compared to state-of-the-art methods.

The remainder of this paper is organized as follows. Section 2 describes related work. Section 3 describes background studies on GNN, Message Passing Neural Network (MPNN), GCN, Graph Attention Network (GAT), and ensemble learning. Section 4 provides details on graph structures for communication networks. In Section 5 we describe

our ensemble GNN model architecture, data preprocessing, training procedure, and hyperparameter optimization. Section 6 presents the experimental setup and results, including the datasets, evaluation metrics, implementation details, and experimental results. Finally, Section 7 concludes the paper and discusses future research directions.

2. Related work

Previous work to detect DDoS attacks through ML and DL have concentrated on balancing detection performance with introduced latency. This is because these methods typically utilize complex models that require significant computational resources to train and execute, leading to increased latency. Hence, these limitations must be addressed to enable a real-time application of ML/DL based DDoS attack detection. One promising approach is using GNNs, a neural network designed to work with graph data. The graph represents relationships between entities, such as the relationships between nodes in a network. GNNs are effective for various graph data tasks, such as fraud detection and social network analysis [11]. In this section, we evaluate some GNN-based DDoS attack detection techniques, including ensemble-based GNN approaches.

2.1. DDoS attack detection

There are numerous proposed methods and frameworks for detecting DDoS attacks. This subsection emphasizes the ML and DL based approaches. These techniques have proven effective in identifying various DDoS attack types, such as volumetric attacks, protocol attacks, and application-layer attacks.

Previous work on DDoS attack detection has also explored hardware acceleration techniques to improve detection performance and reduce latency. Musumeci et al. [12] introduced a Software Defined Networking (SDN) based DDoS detection system. In this system, the DDoS attack detection module receives stateful traffic information based on a sliding window from the system. They utilized Barefoot Tofino switches enabled with P4, which offload a part of the detection process to the data plane. Moreover, the recent work of De Marinis et al. [13] demonstrated that it is possible to onboard the logic of a DNN inside a programmable switch pipelines made of a cascade of flow tables. The paper evaluated the mapping and the onboarding of a P4-based DDoS detector DNN applied at the packet level [14], achieving a F1-score above 92%. This approach, however, assumes the availability of specialized programmable hardware in the network operator ecosystem.

The LUCID technique, as proposed by Doriguzzi-Corin et al. [15] is employed to detect DDoS attacks, offering lightweight execution with minimal processing overhead and detection time. The distinctive traffic preprocessing mechanism is designed to feed the CNN model with network traffic for online DDoS attack detection. Their evaluation compared LUCID with DeepDefense 3Long Short-Term Memory (LSTM) across multiple datasets, including ISCX2012, CIC2017, CSE-CIC2018, and UNB201X, yielding comparable results. Notably, LUCID outperformed 3LSTM in terms of detection time. Additionally, LUCID was compared with other contemporary methods, including DeepGFL, Multilayer Perceptron (MLP), LSTM, 1D-Convolutional Neural Network (CNN), and 1D-CNN LSTM, with validation on the CIC2017 dataset. The evaluation results demonstrated that LUCID achieves good performance from existing state-of-the-art techniques. The study verified that LUCID effectively learns domain information by calculating kernel activations for each feature. Remarkably, the training time of LUCID on a GPU development board was 40 times faster than the authors' implementation of DeepDefense 3LSTM. However, LUCID has not investigated the case of small flows (i.e., mouse flows with limited bytes exchanged between endpoints) with padding mechanism, which may suffer from scalability issues.

2.2. Graph neural networks based attacks detection

GNNs help to identify potential anomalies by analyzing the flow of each feature in the dataset. [16]. The GNN looks promising in different cyber security fields, such as threat intelligence, vulnerability, and malware detection [17–20]. However, the detection of DDoS needs to be correctly implemented to get the full benefit of GNN nature using hierarchical traffic filtering and tackle adversarial attacks. The previous solutions [19,20] for attack detection used measuring each flow or multiple sketches for independent flows without hierarchical traffic filtering. It is essential to detect elephant flows and aggregate mice flow to improve detection and reduce computational cost [21]. The ensemble-based intrusion detection enhances overall robustness against adversarial attacks since combining GNN models present different sensitivities to attacks [22].

Several recent studies have explored the use of self-supervised Graph Neural Networks (GNNs) for various tasks, including Network Intrusion Detection System (NIDS), Knowledge Tracing (KT), and graph-based clustering. Pujol-Perich et al. [23] highlighted the importance of constructing a graph representation of network flows to unveil meaningful structural patterns for developing robust and accurate NIDS. They articulate network flows and their interconnections within the network using a graph structure. Subsequently, a message-passing function is introduced to effectively glean insights from host connection graphs. The model's efficacy is assessed using the CIC-IDS2017 dataset and is benchmarked against various machine learning classifiers. The evaluation involves artificially modifying flow features relevant to the attack scenarios, such as packet size and inter-arrival times. The results demonstrate that the proposed GNN model maintains baseline accuracy compared to other models that exhibit degraded performance when exposed to altered traffic flows.

The study conducted by Li et al. [24] introduced the GraphDDoS model, designed to detect both low-rate and high-rate DDoS attacks by considering the relationships among packets within a single flow and between different flows. In pursuit of this goal, network packets with the same source and destination IP addresses are grouped, forming the basis for constructing an endpoint graph as the final representation. Message-passing operations are then executed on these graphs using a Graph Isomorphism Network (GIN), and a readout function computes the resulting graph embeddings. The efficacy of the model is evaluated on the CIC-IDS2017 and CIC-DoS2017 datasets.

Song et al. [25] proposed a self-supervised GNN model for KT called Bi-CLKT. Bi-CLKT learns node and graph embeddings using positive and negative pair graph training. The authors showed that Bi-CLKT outperforms state-of-the-art KT models.

Guo et al. [26] introduced GLD-Net, a model adept at integrating topological structure and traffic features. The approach involves segmenting traffic data into time slots and constructing a subgraph for each slot. Topology information is incorporated as node features in these subgraphs, while flow statistics serve as edge features. The subgraphs undergo processing using a GAT, allowing simultaneous analysis of both traffic and topological features. The resulting outputs are then interpreted as a time series, inputting a LSTM network. Yang et al. [27] proposed another self-supervised GNN model for graph-based clustering called Variational Co- embedding Learning Model for Attributed Network Clustering (VCLANC). This model uses node embeddings and attributes to learn local and global mutual affinities between graph elements. The authors showed that VCLANC outperforms other graph-based clustering methods.

Lo et al. [28], proposed E-GraphSAGE, a GNN-based IDS for the Internet of Things (IoT). The authors argue that GNNs are well-suited for IDS because they can learn the complex relationships between the devices and nodes in an IoT network. The proposed E-GraphSAGE model is based on the GraphSAGE algorithm [29], a GNN that can learn node representations in a graph. E-GraphSAGE extends GraphSAGE by incorporating edge features and topological patterns into

learning. This makes E-GraphSAGE more effective at detecting malicious network flows in IoT networks. E-GraphSAGE has been tested on four benchmark IoT NIDS datasets, outperforming other state-of-the-art models on all four datasets. It is the first successful, practical, and extensively evaluated approach to applying GNNs to the problem of network intrusion detection for IoT using flow-based data. Their study observed a significant enhancement in botnet detection performance by utilizing their proposed model. This improvement was particularly evident when comparing the outcomes achieved by logistic regression and the pre-existing botnet detection tool, BotGrep [30]. Alshammari et al. [31] proposed a parameter-free graph reduction method for graph-based clustering. The authors showed their approach can achieve competitive performance with parameter-based graph-based clustering methods without dataset-specific parameter tuning.

Xiao et al. [32] proposed a traditional graph embedding approach to perform network intrusion detection. The first step they took was to convert the network flows into graphs based on the source and destination IP and port pairings. After that, they used traditional graph embedding techniques like DeepWalk (skip-gram) for network intrusion detection. However, there is a significant drawback to this approach: it utilizes conventional transductive graph embedding methods [33] which are not capable of generalizing to unseen node embeddings, such as IP addresses and port numbers that were not included during the training phase. This means that the approach is not suitable for most practical NIDS application scenarios, as it cannot rely on every IP address and port pair appearing in the training data. This means that the model cannot detect new attacks that use IP addresses or port numbers that the model has never seen before. This is a major limitation, as attackers constantly develop novel attacks and techniques.

In contrast, Anomal-E [34] is a self-supervised GNN model that does not require labeled data to train and can be generalized to unseen node embeddings. This makes Anomal-E a more promising approach to network intrusion detection, as it can detect known and unknown attacks. Anomal-E has been developed for network intrusion and anomaly detection. It incorporates and leverages edge features to learn the complex relationships between different devices and nodes in a network. This allows Anomal-E to detect malicious network activity, even if it has never seen that type of attack before. Anomal-E has outperformed baseline methods on two benchmark NIDS datasets regarding accuracy and generalization capability. This demonstrates the potential of Anomal-E to improve the security of networks by more effectively detecting known and unknown attacks. In addition to its performance, it offers several other advantages over traditional NIDS approaches. First, it is a self-supervised model, meaning it does not require labeled data to train, a significant advantage in network intrusion detection, where labeled data is scarce and expensive. Second, Anomal-E can leverages edge features, providing additional information about the relationships between different devices and nodes in a network. This information is essential for detecting sophisticated attacks that exploit vulnerabilities in the network topology.

2.3. Ensemble GNN Learning

GNN models are a powerful tool for learning from graph-structured data. They are used in various applications, including social network analysis, recommendation systems, and computer vision [35]. However, deep learning models can be complex and computationally expensive to train, and they can overfit the training data leading to poor performance on unseen data. Despite these challenges, GNNs have the potential to improve the performance of intrusion detection systems significantly.

Hou et al. [36] proposed a novel approach to aspect-level sentiment classification that leverages the power of GNNs and ensemble learning. Aspect-level Sentiment Classification (ALSC) is a promising solution that uses GNNs to construct multiple dependency trees for an input sentence, each of which captures a different aspect of the sentence's syntactic structure. Some graph networks are then applied to each dependency tree to learn representations of the words and phrases in the sentence. These representations are combined using an ensemble learning technique to predict the aspect-level sentiment. This approach has been shown to outperform state-of-the-art methods on several benchmark ALSC datasets. For example, in the SemEval 2014 Task 4 dataset, the graph ensemble learning approach achieved an F1-score of 88.45%, compared to 85.62% for the best baseline method.

Wei et al. [37] proposed a new GNN-Ensemble approach to address the challenges of complexity, computational expense, and overfitting in intrusion detection systems. This approach reduces the overall complexity and computational cost of training while also improving the robustness of the model to overfitting. It randomly constructs multiple substructures from the input graph and trains a GNN on each substructure. The predictions of the individual models are then combined using a weighted voting scheme. The authors evaluate GNN-Ensemble on several benchmark datasets and show that it outperforms other GNN-based methods in accuracy and generalization. While it is a promising new approach, it has some limitations. First, implementing it is more complex, as it resorts on multiple models. Second, ensemble GNNs may require more training data than single ones as they need to train multiple models, each of which is trained on a different subgraph of the network traffic data. Additionally, ensemble models can be more sensitive to the data quality, so it is vital to ensure that it is well-labeled and representative of the real-world data on which the model will be deployed [38]. Finally, it is still open to research how best to design the substructures and voting scheme for different tasks.

Wang et al. [39] proposed a novel GNN-based NIDS approach called Spatio-Temporal Graph Attention Network (N-STGAT). It leverages a graph attention mechanism to learn representations of network traffic that incorporate the spatial and temporal features of the data. The authors evaluate their approach on the latest flow-based dataset for near-earth remote sensing systems. Their results show it outperforms state-of-the-art GNN-based NIDS approaches by a significant margin. The author's work substantially contributes to the NIDS. Their approach can improve computer network security by making NIDS more effective against novel and zero-day attacks, especially in near-earth remote sensing systems.

Qi et al. [40] proposed a novel Graph Ensemble Network (GENet) for traffic prediction. GENet is a deep learning model that learns to predict traffic flow by leveraging the power of GNNs. A graph-based representation of the traffic network captures the relationships between road segments and intersections. A multi-scale GNN architecture allows GENet to learn representations of traffic at different levels of granularity. In this work, a customized loss function is designed to improve the performance of GENet on traffic prediction tasks.

The above papers introduce ensemble-based GNN use cases for some real-world applications. In the following section, we will address ensemble-based GNN techniques for the detection of network attacks.

2.4. Ensemble-based GNN techniques for attacks detection

Control Area Network (CAN)s are widely used in vehicles and other embedded systems to communicate between components. However, CANs are vulnerable to various attacks, including message injection, suspension, and falsification. These attacks can have serious consequences, such as disrupting vehicle operations or causing accidents. Traditional anomaly detection mechanisms for CANs are either slow or ineffective against some attacks. Additionally, these mechanisms often require a lot of training data, which can take some effort to collect. Zhu et al. [41] proposed a Federated Graph Neural Network (FGNN) for fast anomaly detection in CANs. This novel approach leverages the power of GNNs to learn representations of CAN traffic that can be used to detect anomalies. This technique is federated, meaning it can be trained on a distributed network without compromising privacy. Their approach works by constructing a graph from the CAN traffic data. The nodes of

the graph represent CAN messages, and the edges of the graph represent the relationships between different notes. They trained their model to learn representations of the nodes and, in turn, to detect anomalies in the CAN traffic. The authors compared their model to several state-of-the-art CAN anomaly detection methods, including Support Vector Machine (SVMs), random forests, and other GNN-based methods. The results showed that their approach outperformed all baseline methods regarding accuracy and speed. Specifically, they achieved an accuracy of 99.9% on the UNSW-NB15 dataset and an accuracy of 99.8% on the Kvaser dataset. The F1-score of their model is 99.1% with binary classification and but due to the unbalanced dataset, they achieved a weighted F1-score is 90.7% with multi-classification of attacks. Additionally, FGNN detected anomalies in CAN traffic in as little as three milliseconds. However, more research is needed to evaluate the performance of FGNN on large-scale real-world CAN systems and to investigate how it can be used to detect other types of anomalies.

Esmaeili et al. [42] reported a novel GNN-based adversarial malware detection framework for IoT devices. The proposed framework is designed to be more effective against IoT malware than traditional malware detection methods, as it leverages the power of GNNs to learn representations of IoT devices and their interactions. The proposed framework works by constructing a graph where the nodes represent IoT devices and the edges represent their interactions. The proposed framework then uses a GNN to learn representations of the nodes and edges of the graph and capture the complex relationships between the devices and their interactions. These representations are then used to detect malware on IoT devices. The results showed that the proposed framework could detect malware with an accuracy of 99.18% but they have not reported weighted F1-scores. Their detector achieved a 98.96% detection rate, 1.17% higher than the previous best method. Additionally, their detector had a 5.95% lower False Positive Rate (FPR) than the previous best method. Moreover, their detector achieved an F1 score as good as of 0.9658. The proposed framework also has a fast response time of 3 ms. However, this framework has not been evaluated on a large-scale, real-world IoT system, nor how it can detect other types of anomalies in IoT data, such as malicious node network traffic behavior over the network, and how these malicious nodes are used to conduct DDoS attacks.

Our proposed model, FTG-NET-E, is inspired by these recent studies. However, FTG-NET-E is the first self-supervised GNN model specifically designed for NIDS. Our model leverages edge features to learn the complex relationships between different devices and nodes in a network. This allows FTG-NET-E to detect malicious network activity, even if it has never seen that type of attack before.

3. Background on graph neural networks

Several kind of objects are defined in terms of connections to other things and are thus naturally expressed in a graph form. Molecules, social networks, and citations are just some examples of structures that are naturally described as graphs. While there is an obvious interest in exploiting machine learning on this kind of data, standard neural networks cannot be used to learn on graphs as they are made to work on arrayed-structured data. For this reason, in 2008 Scarselli et al. [43] introduced graph neural networks, a class of deep learning models developed to operate on graph-structured data. GNNs have gained since a rapid interest, and are now successfully applied in several areas such as physics simulations, recommendation systems, antibacterial discovery, and networking [44].

The field of GNNs is vast, and graph networks can differ considerably. As a general consideration, we can define a GNN as a permutation invariant optimizable transformation on graph attributes (nodes, edges, and global parameters) [45]. The permutation invariant attribute refers to the fact that GNNs preserve symmetries. The prediction problems tackled by these networks can be divided in three categories: (i) graph level tasks where predictions try to discover a property of the entire graph (e.g., molecule toxicity), (ii) edge level tasks where predictions concerns the relations between nodes (e.g., recommendation systems), and (iii) node level tasks where predictions try to find the identity or role of each node of the graph.

3.0.1. Message passing neural networks

The first kind of GNN is the MPNN. Let $G = (V, E, U)$ be a graph, where V is the set of vertices (nodes), E is the set of edges, and U is the global features. Consider now a simple problem of node representation learning, where each vertex $v \in V$ has a set of features x_v, represented by a vector of dimension n, namely the hidden state h_v^0. In each iteration t of the process, a message-passing operation occurs, where the hidden states of the nodes are combined with the hidden state of their neighbors (e.g., through addition). This message-passing operation involves two functions: an aggregation function $a(\cdot)$ that consolidates the received hidden states (e.g., through addition), and an update function $u(\cdot)$ that merges the current hidden state of the node with the result of the aggregation (usually employing a differentiable model such as a multilayer perceptron).

At iteration t, the message-passing operation is defined formally as follows:

$$m_{v,w} = m(h_v^t, h_w^t, e_{v,w}) \tag{1}$$

$$M_v^{t+1} = a(\{m_{v,w} | w \in N(v)\}) \tag{2}$$

$$h_v^{t+1} = u(h_v^t, M_v^{t+1}) \tag{3}$$

here hidden state of node v at iteration t is denoted by h_v^t. $e_{v,w}$ is the edge which connects nodes v and w. The message sent from w to v is represented by $m_{v,w}$, whereas the aggregated message received by v is denoted by M_v^t. The neighborhood of v (which comprises all vertices adjacent to v) is represented by $N(v)$.

It is feasible to depict the complete graph as a single vector following the execution of T iterations. This can be accomplished by utilizing a permutation invariant function that operates on all the concealed states of the nodes during a readout phase.

$$y = r(h_v^T | v \in V) \tag{4}$$

where y is the final representation vector; r is the permutation invariant function. Finally, y is used to make the final prediction.

3.0.2. Graph convolutional networks

First proposed in 2017 by Kipf and Welling [46], GCN models are graph networks that rely on graph convolutions to support scalability and fast semi-supervised classification of nodes. GCNs propagate information across the graph through a message-passing mechanism and updates the node features based on the information on its neighborhood only. This process is repeated for several layers, allowing the network to capture increasingly complex relationships between nodes. The message-passing mechanism in GCNs closely resembles the convolution operation in CNNs, and hence the name. In CNNs, the convolution operation updates an element according to the patterns from its neighbors in an arrayed structures. Similarly, in GCNs the message-passing mechanism aggregates information from a node neighbors, effectively capturing local patterns within the graph structure.

The following equations describe the GCN layer update rule:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}) \tag{5}$$

Where $H^{(l)}$ and $H^{(l+1)}$ are the input and output feature matrices of the GCN layer, respectively; $\tilde{D}$ represents the degree matrix of the graph, which is a diagonal matrix where the entries correspond to the number of neighbors for each node; $\tilde{A}$ denotes the normalized adjacency matrix of the graph, which is obtained by adding self-connections to the adjacency matrix A and normalizing it using the square root of the degree matrix $\tilde{D}$; $W^{(l)}$ is the trainable weight matrix of the GCN layer; σ

is a non-linear activation function, such as a sigmoid or Rectified Linear Unit (ReLU). The update rule can be interpreted as a weighted average of the node and of its neighbors.

The node-wise perspective of the GCN update rule is described by the following:

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in N(i)} c_{ij} W^{(l)} h_j^{(l)}\right) \tag{6}$$

Where $h_i^{(l)}$ and $h_i^{(l+1)}$ are the input and output representations of node i or $i+1$ at layer l. $N(i)$ is the set of neighbors of node i. c_{ij} is a normalization factor defined as $c_{ij} = \frac{1}{\sqrt{|N(i)|}\sqrt{|N(j)|}}$. The node-wise update rule can be interpreted as a weighted average of the node's representation and the representations of its neighbors, where the weights are determined by the normalization factor c_{ij}.

3.1. Graph attention network

GATs are a type of GNN that uses an attention mechanism to learn the importance of different neighbors for each node in the graph. GATs were first proposed by Velickovic et al. in 2017 [47], and have since become one of the most popular GNN architectures. These attention networks work by first computing an attention weight for each pair of nodes using a learnable function that takes the node features (and edge features if available) as input. The attention weights are then normalized to sum to 1 for each node, and used to update the node features. As a result, the updated node features are a sum of the features of the node neighborhood, weighted by the attention weights. The following equation describe the GAT layer update rule:

$$\mathbf{h}_i^{(l)} = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W}^{(l)} \mathbf{h}_j^{(l)}\right) \tag{7}$$

Where $\mathbf{h}_i^{(l)}$ is the feature vector of node i at layer l; σ is a non-linear activation function, such as ReLU; $\mathcal{N}_i$ is the set of neighbors of node i; α_{ij} is the attention weight between nodes i and j; $\mathbf{W}^{(l)}$ is the weight matrix at layer l.

3.2. Ensemble learning

Ensemble learning is a machine learning technique that combines multiple models outputs to improve the overall prediction accuracy. It has emerged as a powerful tool for DDoS detection, where attackers are constantly developing new and sophisticated methods. One of the key advantages of ensemble learning is that it can leverage the strengths of different models to capture diverse patterns in the data. This is akin to assembling a team of experts with different backgrounds and perspectives to solve a problem. Just as a team of experts is likely to outperform any individual expert, an ensemble of models will likely outperform any individual model. Another advantage of ensemble learning is that it reduces the risk of overfitting: ensembles are less prone to over-learn the specific features of the training data and better generalize [48].

GNNs based ensemble learning combines the predictions of multiple graph networks to produce a more robust and accurate prediction. This is done by training multiple GNNs on different subsets of the data, and then averaging the predictions of the individual models.

Among the various ways to ensemble neural networks, a common approach is to use a weighted average of the predictions of the individual models [49]. The weights can be assigned based on the performance of the individual network on the training data, or they can be learned using a hyperparameter tuning algorithm. Another approach to ensemble GNNs is to use a stacking technique, where the individual model predictions are used as input to a meta-learner. The meta-learner combines the predictions of multiple models instead of only weighted averages. The meta-learner is then trained to predict the final output.

The following equations describe the ensemble method for a GNN with equal-weights:

$$\hat{y}_i = \frac{1}{M} \sum_{m=1}^{M} g_m(\mathbf{h}_i^{(l)}) \tag{8}$$

Where $\hat{y}_i$ is the predicted output for node i; M is the number of GNNs in the ensemble; g_m is the mth GNN in the ensemble; $\mathbf{h}_i^{(l)}$ is the feature vector of node i at layer l.

4. FTG-Net-E graph structures and GNN models

Network traffic resembles almost naturally message passing between nodes in an undirected graph. It can be modeled as a hierarchy of flow entries, both with fine-grained (pachets) and coarse-grained (endpoint communications). Our proposed basic structure, FTG, is constructed from traffic data divided into time slots of size t_s. The structure is composed of a high-level Traffic graph related to each time slot, with each node corresponding to a low-level Flow Graph. Every level has a GNN model that is processed in FTG-Net-E. The following subsections describe how the graphs are constructed, how the traffic is converted and how the architecture of th traffic and flow GNNs.

4.1. Flow graph

Flow Graphs draw inspiration from the graph structure introduced in GraphDDoS [24], but with a modification: the N nodes limit is replaced by the incorporation of time slots. These graphs are crafted by grouping all packets exchanged between two endpoints within a given time slot, even if they pertain to distinct networking flows. Within each group, packets are arranged in ascending order of time and transformed into nodes, featuring only the packet length as a characteristic. We chose the packet length as a key feature as it aids detecting malicious patterns in network traffic, such as large packet sizes used for DDoS attacks and small packets for port scanning attacks. Moreover, this feature is easily available from network traffic data, allowing for efficient real-time processing and model interpretability.

Assigning positive directional flow, whether upstream (client to server) or downstream (server to client), distinguishes positive values to upstream and negative values to downstream lengths.

Consecutive packets sent by one endpoint constitute a mini-group, with edges linking adjacent nodes representing packets from the same mini-group. Connections are established between the first packet of a mini-group and the first packets of the preceding and succeeding mini-groups, similar to the linkage between the last packets of mini-groups. Fig. 1 shows an example of a Flow Graph.

Given this graph representation of a computer network, A GNNs can be used to classify nodes in the following way:

1. The GNN is initialized with random representations for each node in the network.
2. A message-passing function is applied for multiple time slots t_s: each node sends a message to its neighbors in each slot based on its current representation. The messages are then aggregated at each node, and the node's representation is updated using a function that combines its current representation with the aggregated messages from its neighbors.
3. The final representation of each node is used as the input to a classifier, which predicts the node class.

The Flow Graph structure allows us to observe packet relationships within a single flow, which can be different from legitimate traffic in certain DDoS attacks, and relationships between multiple flows between the same endpoints, which can be used to detect burst information and periodic information.

Burst information corresponds to the scenario in which the attacker sends many packets to establish connections with its target on several ports. Periodic information corresponds to low-peace attacks in which the attacker periodically mimics a normal client to occupy the victim's resources as long as possible.

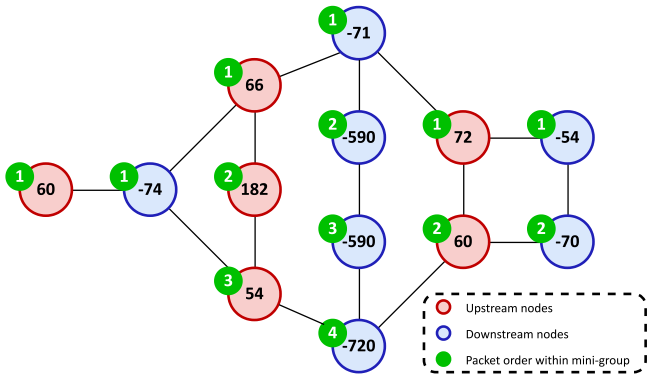


Fig. 1. Flow Graph example: upstream packets are denoted by positive values, while downstream packets are represented by negative values. The sequence of mini-groups is organized from left to right.

4.2. Traffic graph

A Traffic Graph is a graphical representation of traffic flow over time. This graph is constructed for each time slot, which is typically a discrete interval of time, such as a minute, an hour, or a day. The Traffic Graph shows the amount of data that is transmitted across a network during each time slot, providing valuable insights into network performance, usage patterns, and potential issues that need to be addressed. For each time slot, any pair of endpoints that appears in at least one packet from the traffic data is transformed into a graph node. The features of this node are determined by the output vector of the Flow GNN, which includes a Flow Graph at a relatively low-level. If two nodes share the same source or destination IP address, an edge is added in between. According to the real-world implementation requirements and capabilities, the set of features describing the communications between two endpoints can be modified and enriched by concatenating the available statistics.

We have developed a Traffic Graph that incorporates the interactions between different flows, displaying the distribution of devices that send requests to servers and flow-specific behaviors that may contain malicious patterns. The encoding of these flow-level patterns is closely linked to the traffic-level topology since the Flow GNN and the Traffic GNN weights are both optimized using the same gradients during the training phase.

4.3. Traffic converting stage

The traffic data is separated into time slots in this stage. For each time slot, the extraction process produces a Traffic Graph G_t and a set of Flow Graphs $F = G_{f_1}, G_{f_2}, \ldots, G_{f_N}$, where the node indexes of Flow Graphs align with those of the Traffic Graph. The number of Flow Graphs, denoted by N, depends on the time slot. During the supervised learning phase, each time slot data is associated with a corresponding list of labels $Y = y_1, y_2, \ldots, y_N$ consisting of one Flow Graph and one label.

The traffic data is analyzed sequentially in ascending order based on time to extract the graph. An empty Traffic Graph is initialized at the start of each time slot, and an empty sorted list of encountered pairs of endpoints is created. When a new endpoint combination is detected, a new node is added to the Traffic Graph, and a separate Flow Graph is created for the current packet. During this process, the packet direction is preserved as it will be instrumental in categorizing packets into mini-groups. If the combination of endpoints is already present in the list, any new packet that arrives is used to update the corresponding Flow Graph. If the direction of the packet differs from the previously added packet, the mini-group is concluded and interconnected through edges as explained in Section 4.1. As the time slot concludes, edges in the Traffic Graph are added following the guidelines outlined in Section 4.2.

4.4. Flow GNN

The Flow GNN plays a pivotal role in handling flow graphs and constructing embeddings, which will subsequently serve as node features in the traffic graph.

Initially, three GCN Layers are employed on the Flow Graph. These layers disseminate node information throughout of each node with the three-hop neighborhood. Following each GCN Layer, a ReLU activation function is applied.

First, a readout function is utilized to transform the graph into a singular vector. The selected readout function is global mean pooling, which calculates the average of node features across the dimension of a node. The definition of global mean pooling is as follows:

$$h_G = \frac{1}{N}\sum_{n=1}^{N} h_n^{(3)} \tag{9}$$

where in Flow Graph, the number of nodes is represented as N and the feature vector of the nth node after three iterations of GCN is represented by $h_n^{(3)}$. The output vector is obtained from readout and passed to a fully connected layer, producing the final output.

4.5. Traffic GNN

The traffic GNN takes a traffic graph as input and generates a prediction for each node, indicating whether it corresponds to legitimate or malicious traffic. The model consists of three GCN layers followed by a fully connected layer that has a dropout rate of 50%. The dropout is applied to the features of each node individually. The model produces single outputs that are passed through a sigmoid activation function, which provides the final scores ranging from 0 to 1.

4.6. Graph features engineering

Feature engineering is a critical process that helps extract relevant information from raw data. In the context of network traffic analysis, the construction of traffic graphs is a structured representation of network traffic patterns that enables graph-based analysis. In our approach, we extracted critical attributes from network traffic data that serve as the basis for constructing the traffic graph. These attributes include: the URG Flag Count, SYN Flag Count, RST Flag Count, PSH Flag Count, Packet Size Average, Flow Packets Per Second, FIN Flag Count, ECE Flag Count, ACK Flag Count, Destination Port, and protocol information (IPv6 Hop-by-Hop 0, TCP 6, UDP 17) obtained from packet headers and traffic statistics. The selection of these features was not only made by relevance to intrusion detection, distinguishing between normal and malicious traffic, and their ability to capture diverse network behaviors; we have also experimented with different combinations of features to evaluate their impact on model performance. We used feature selection and ablation studies to identify the most relevant features and assess their individual contribution. We scaled the features to ensure they had a consistent range and distribution, which helped prevent features with larger magnitudes from dominating the learning process. We also addressed missing values in the dataset by employing imputation techniques to replace them with appropriate values, ensuring the dataset was complete and ready for analysis.

5. FTG-Net-E architecture and dataset

The architecture of FTG-Net-E is based on FTG-Net, which is a hierarchical GNN designed to process the multi-level graph representation FTG. FTG-Net works in three main steps, as shown in Fig. 2, which are described as follows:

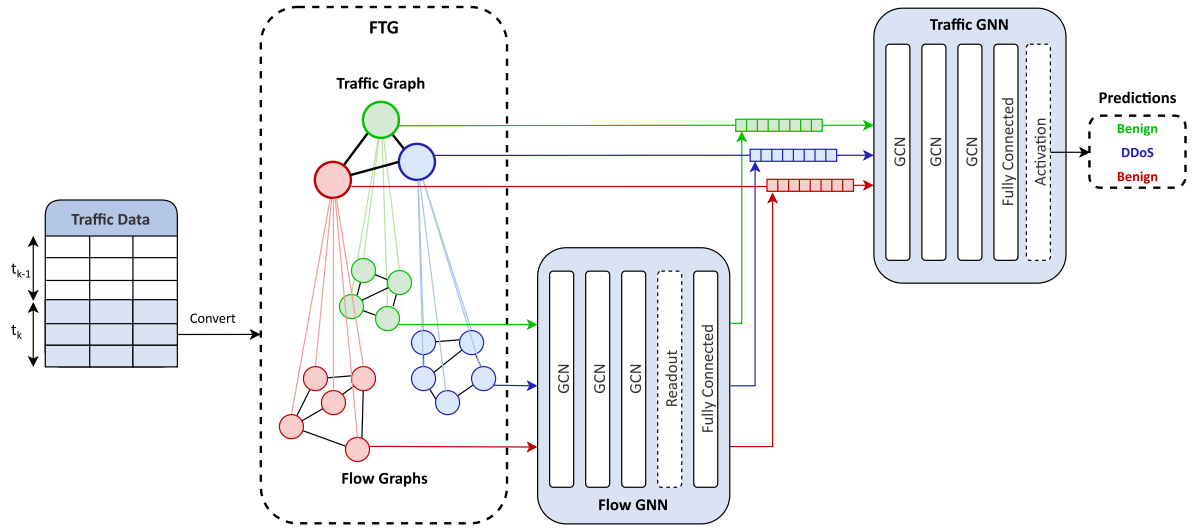


Fig. 2. FTG-Net-E architecture. (1) "Traffic Data Preprocessing": Training on traffic data converted into flow graphs. (2) "Flow GNN" and "Traffic GNN": Learning representations of individual flow graphs and entire traffic graphs, respectively. (3) "Fully Connected Layer", gives the final predictions based on the Traffic graph network outputs as Benign and DDoS.

1. **FTG construction**: The first step is to convert the traffic data into FTG structures. This is accomplished by grouping all packets exchanged between two endpoints within a time slot to create a flow graph. The Flow Graphs are then sorted in chronological order and converted to nodes in the Traffic Graph. The features of each node in the Traffic Graph are given by the Flow GNN output vector, which embeds the corresponding Flow Graph.
2. **Flow GNN Inference**: The second step involves utilizing the Flow GNN to analyze the Flow-level Graphs. This module leverages three GCN layers to extract node representations from individual flow graphs. These layers capture local dependencies and interactions within each flow. A readout layer aggregates information from the entire graph, followed by a fully connected layer for final representation learning. Dropout regularization with a rate of 0.5 is applied after each GCN layer to prevent overfitting. The output of the last GCN layer is passed through a fully connected layer with a sigmoid activation function to generate flow-level predictions. The output vector of the Flow GNN for each Flow Graph is used as the features of the corresponding node in the Traffic Graph.
3. **Traffic GNN Inference**: The third step is to process the traffic-level graphs using the Traffic GNN. This module, consisting of three GCN layers, batch normalization, and dropout regularization with a rate of 0.5 after each GCN layer, captures high-level patterns across the network traffic graph. By processing the aggregated flow representations obtained from the Flow GNN, this module gives a global understanding of the overall network traffic behavior. A final fully connected layer with a ReLU activation generates the final predictions, differentiating between benign and DDoS traffic based on the learned representation. The final prediction is the output of the Traffic GNN for each flow.

FTG-Net-E is designed to exploit the hierarchical structure of FTG to learn more effective representations of the traffic data and to improve the performance of DDoS attack detection. This ensemble model is developed in three main steps: (i) the graph structure is subsampled into subgraphs, and node features into subfeatures, (ii) multiple base GNN models are trained on different subgraphs and subfeatures, such as GCN by [50] and GraphSage by [29]. Each GNN model learns a representation of the nodes in its subgraph, which can then be used to make predictions about the node labels, and (iii) finally the predictions of multiple GNN models are aggregated using a ranking scheme, so to combine the information from different models and make a more accurate prediction.

5.1. DDoS detection datasets

In 2016, a paper listed 11 criteria that must be fulfilled for a dataset to be considered suitable for intrusion detection purposes [51]. Among the various publicly available datasets, we chose the Canadian Institute of Cybersecurity Intrusion Detection System Datasets (CIC-IDS2017, CSE-CIC-IDS-2018) as these are modern flow-based datasets that fulfill all these criteria. The CIC-IDS2017 was developed by Sharafaldin et al. [9] and consists of normal and malicious network traffic. The normal traffic is emulated by utilizing B-profiles, which are obtained from the benign (normal) conduct of 25 individuals (human). The malicious traffic consists of common attacks (DoS, DDoS, Brute Force, XSS, SQL Injection, Infiltration, Port Scan, and Botnet), and is generated by executing existing DDoS attacking tools at specific time windows.

Aside to the CIC-IDS-2017, we have trained FTG-Net-E also on its successor, The CSE-CIC-IDS-2018 dataset. The 2018 dataset was generated using the same tools of the 2017 one, but deployed in AWS rather than on a local university network. The CIC-IDS2018 dataset is labeled with more than 83 traffic features collected using CICFlowMeter [52]. The capturing period for this dataset started at 09:00 on Monday and ended at 17:00 on Friday. The DDoS attacks were performed on Friday afternoon and are the only significant for this work. Traffic data have been converted considering only time slots with at least 20% of labeled data. GNN based preprocessed datasets representing without prediction and with prediction labels. From the 2018 dataset, 127,844 additional infiltration samples were obtained, these samples were kept separately to test the robustness of the zero-day capability of our novel proposed approach.

5.1.1. Ensemble learning model

The ensemble learning model consists of multiple GNN models, each trained with distinct hyperparameters. We created three instances of the GNN model with parameter variations such as learning rate, hidden size, and dropout rate. Despite variations in hyperparameters, the training process remains consistent across all models. After independent training on the same dataset, the predictions from individual GNN models are combined using a soft voting approach. This method involves averaging the predicted probabilities from each model to make final predictions, allowing for more nuanced outcomes compared to simple majority voting.

5.1.2. Dataset preparation

We first loaded the raw dataset to ensure data quality and selected only the relevant attributes for our analysis. After collecting the data, we performed data cleaning procedures that involved eliminating redundant columns, removing rows with missing or infinite values and filtering out any duplicated records. We also handled missing values by forcing the Flow Pkts/s feature to be numeric and dropping rows with missing entries. After preprocessing the dataset, to clean and standardize the features we scaled the data using the StandardScaler implementation from the sci-kit-learn Python library. The data has been then split into training and test sets using a stratified split to ensure that the distribution of classes in each set is representative of the overall dataset. Using one-hot encoding, we encoded categorical variables like the protocol to facilitate classification tasks. The preprocessed data, which included features such as packet size, duration, and source and destination IP addresses, was then saved as CSV files for easy integration into our GNN-based DDoS attack detection model.

The resulting dataset, which consisted of 80% benign and 20% malicious samples, was split into training, validation, and test sets. The down-sampled malicious dataset was divided into 70% for training 15% testing, and 15% for validation. The 20 zero-day samples, representing novel malware attacks that were not previously known to the system, were added to the test set along with the benign samples so to assess the ability of our ensemble graph network to detect novel attacks. The validation set was created by randomly selecting 700 samples from each of the five attack categories, namely UDP flood, TCP SYN attack, and HTTP attack, from the malicious training set. As a result, the validation set consisted of a total of 2,100 samples. The benign samples were split into the train, validation, and test sets to maintain the overall distribution of 80% benign and 20% malicious samples in the final datasets.

5.1.3. Data for the anomaly detector

In the first stage, the anomaly detector was trained using a training set composed of mostly benign traffic. The validation set is instead composed of 80% benign and 20% malicious traffic. This approach was chosen because the anomaly detector primary function is to filter malicious traffic from a stream of predominantly benign traffic. However, it is essential to acknowledge that this method produces a model that is prone to give false positives. One strategy to address this potential issue involves employing a more balanced dataset for training and validation. This entails incorporating a higher proportion of malicious traffic into the validation set, enabling the detector to gain proficiency in distinguishing between benign and malicious traffic. Still, implementing this approach is challenging, as acquiring substantial amounts of malicious traffic data is often tricky. Another approach to enhance the anomaly detector performance involves utilizing a more sophisticated training algorithm. Specific algorithms excel at identifying anomalies and understanding the patterns of normal traffic. Additionally, we fine-tune the parameters of the training algorithm to improve its effectiveness further. Monitoring the anomaly detector's performance is crucial as new forms of malicious traffic emerge. The detector needs to be re-trained or updated periodically to maintain its efficacy.

5.1.4. Classifier dataset

The GNN classifier learns to classify malicious traffic into attack categories. While its validation set is equally distributed of 50% benign and 50% malicious samples, the attack kinds are balanced within the malicious traffic samples. In order to avoid bias towards a specific type of attack, it is crucial to consider the possibility of unknown attacks. Instead, the classifier should be rewarded for outputting a vector with low probabilities for each known attack that it was trained on, and this will result in a higher validation score for benign traffic.

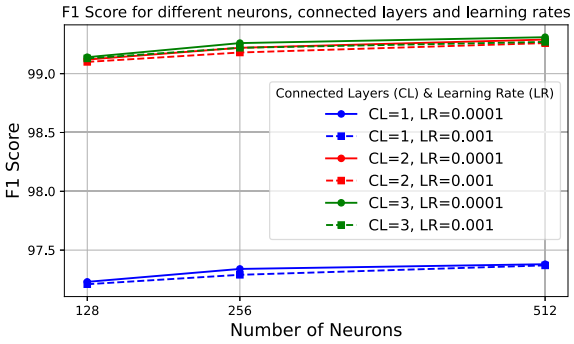


Fig. 3. FTG-Net-E mean F1 scores achieved by FTG-Net-E for all 6 possible combinations of learning rate and hidden size.

6. Experimental results

This section presents the experimental results to assess the performance of the proposed FTG-Net-E model. The experiments were run on a Ubuntu 20.04 LTS workstation with Dell PowerEdge R760Intel® Xeon® Gold 6438M processor and NVIDIA A30X graphics card. The main libraries for implementing FTG-Net-E are TensorFlow, NumPy, PyTorch, Scapy, PyTorch Geometric, and Pandas.

The proposed structure deals with a binary classification task, in which indicators are established through a confusion matrix and in which it is possible to formulate the following metrics:

$$ACC = \frac{x_{correct}}{x_{total}} = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

$$recall = \frac{TP}{TP + FN} \tag{11}$$

$$precision = \frac{TP}{TP + FP} \tag{12}$$

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{13}$$

where ACC stands for accuracy and TP, TN, FP, FN stand for true positive, true negative, false positive, and false negative, respectively.

In classification problems, the malignant class is usually more important, and recall becomes the main metric. But in certain cases, like in DDoS Detection, it is not possible to state that one class is more significant than the other. In such scenarios, combining metrics computed considering each class as positive and averaging the results is a common approach. Weighted F1-Score metric is also used, where F1-Scores are combined by weighted averaging using the number of samples of the positive classes as weights. This helps in achieving a balanced classification that does not block legitimate traffic while detecting malicious packets.

We investigated the impact of two hyperparameters, learning rate and hidden size, on the performance of our ensemble model. We have trained FTG-Net-E with a learning rate of either 0.001 or 0.0001, while we have changed the hidden size of our models between 128, 256, and 512 neurons.

Fig. 3 shows the mean F1 scores achieved by FTG-Net-E on both datasets (CIC-IDS-2017, and CSE-CIC-IDS-2018) when trained with all six possible combinations of learning rate and hidden size. As shown in the table, the ensemble model achieves the highest F1 of 99.29% with a learning rate of 0.0001 and a hidden size of 512, making these hyperparameters the best for our ensemble model. Nonetheless, in all cases the F1 score is above 99%, making it possible to use models that are lighter (lower hidden size) and able to be trained in lower time (higher learning rate), without a significant loss of accuracy. As observed, the model achieves the highest F1 score of 99.29% with a learning rate of 0.0001 and a hidden size of 512 neurons. However, it is important to note that all configurations achieved above 99% F1 scores, indicating

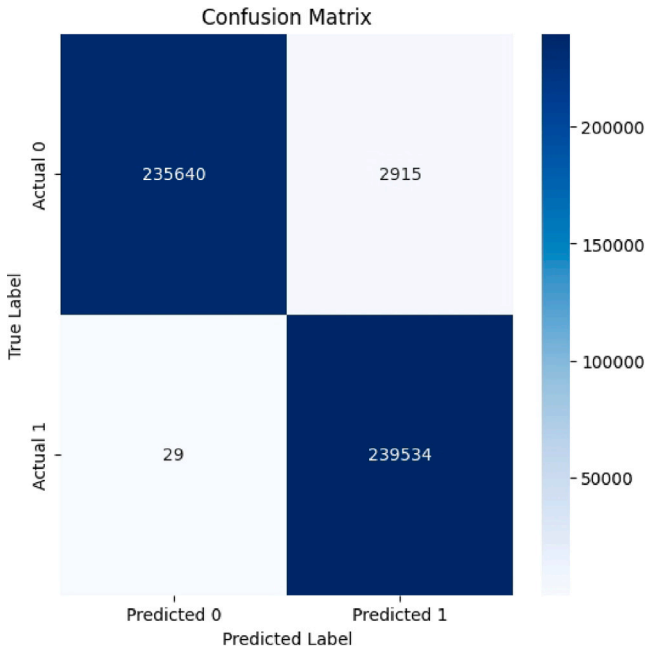


Fig. 4. FTG-Net-E confusion matrix.

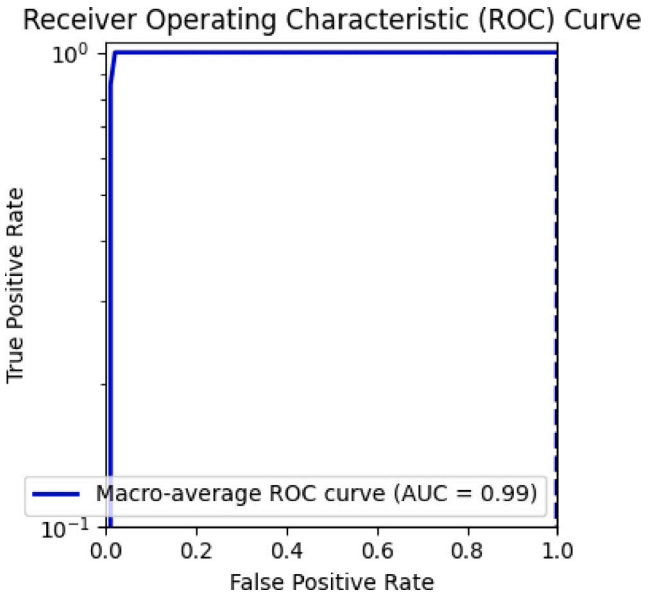


Fig. 6. ROC Curve with different parameters.

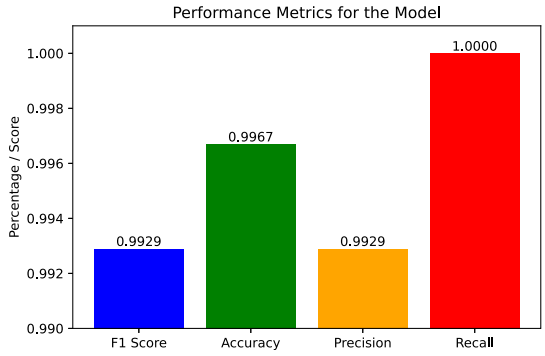


Fig. 5. Performance metrics for the best FTG-Net-E model.

the model's robustness to these specific hyperparameter choices. This allows for flexibility in selecting models based on specific deployment requirements, such as prioritizing faster training times (using a higher learning rate) or lower computational resources (using a smaller hidden size) with minimal impact on accuracy.

Fig. 4 shows the confusion matrix when FTG-Net-E is trained with the best hyperparameter. The ensemble model very high F1 score testifies its ability to accurately classify both positive and negative cases. The model correctly predicted 235640 positives and 239534 negative cases. There were 2915 false positives and 29 false negatives. Our ensemble model correctly predicted 235640 positives and 239534 negative cases. There were 2915 false positives and 29 false negatives. The high F1 score and low false positives and negatives indicate that the ensemble model is a very effective classifier for this task.

Fig. 5 presents all the performance metrics for the best FTG-Net-E. The ensemble model achieved an F1 score of 0.9929, accuracy of 0.9967, precision of 0.9929, and recall of 1.0. These results indicate that the model can accurately predict both positive and negative instances with a high degree of accuracy. The model can also minimize false positives and negatives, which is important for many real-world applications. This high performance is because it combines the predictions of multiple GNN models with different hyperparameters. This averaging process helps reduce the prediction variance and improve the model overall performance.

Fig. 6 presents the Receiver Operating Characteristic (ROC) curve analysis for our GNN models with different hyperparameter settings. The single ROC curve plot showcases the performance of each GNN model across these parameter configurations. Each curve represents the ROC curve for a specific parameter setting, and the area under the curve Accuracy (AUC) is used as a metric to quantify the model's discriminative power. As seen in the plot, the ROC curves demonstrate the trade-off between the true positive rate (sensitivity) and the false positive rate (1-specificity) for various threshold values. The dashed line represents the baseline scenario where the model's predictions are no better than random chance. Our analysis reveals that these ensemble GNN models consistently achieve high AUC values, even with distinct hyperparameter settings, indicating their effectiveness in distinguishing malicious and legitimate traffic. The consistent performance of our models across different hyperparameter configurations underscores the reproducibility of our approach and suggests that our models can generalize well to new data.

6.1. Inference time and model comparison

We evaluated the performance of FTG-Net-E using a 5-fold cross-validation method to determine its effectiveness. The dataset was randomly partitioned into five segments for each fold, with a model trained on four segments and the remaining one serving as the validation set. The configuration details include a 5-second time slot size, an 8-dimensional vector as the output of the Flow GNN, and a 64-hidden channel output for the GCN layers. Initially, the Flow Graph nodes are made up of only one element which corresponds to the length of the packet. In our approach, the size of the time slot needs to be configured carefully. A larger time slot provides a broader view of the traffic topology, which contains more information to detect potential DDoS

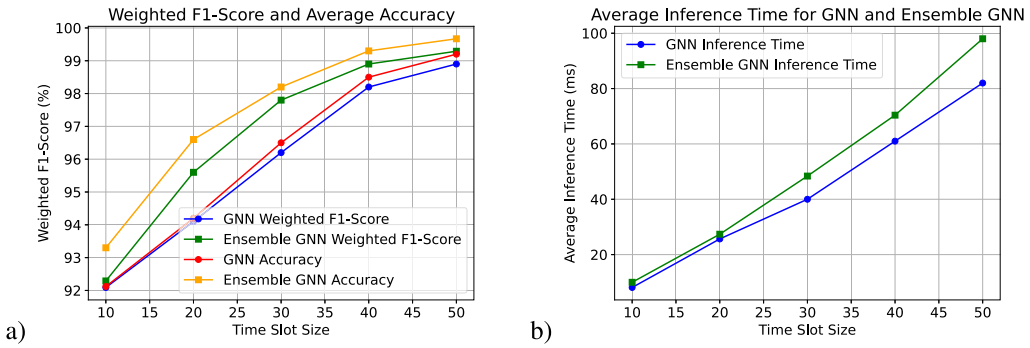


Fig. 7. (a) Weighted F1-Score and accuracy for GNN and Ensemble GNN (b) Average inference time for GNN and Ensemble GNN models results using different time slot sizes.

Table 1
Comparison of results.

Method	Accuracy	F1-score	Precision
LUCID [15]	0.9967	0.9966	
GLD-Net [26]	0.9940	0.9920	
GraphDDoS [24]	0.9959	0.9959	
N-STGAT [39]	0.9788	0.9869	
Anomal-E [34]	0.9412	0.9630	
E-GraphSAG [29]	0.95472	0.9719	
ST-GCN [8]	0.9110	NA	
FTG-Net [5]	0.9914	0.9913	0.9908
FTG-Net-E	0.99675	0.9929	0.99256

attacks. However, the size of the time slot is inversely proportional to the system's responsiveness. A larger time slot may not allow the system to detect attacks in a timely manner, which could result in significant damage. Moreover, with large Traffic Graphs, the inference time increases. It is possible that a short time period may not provide sufficient data to accurately classify traffic. To address this, we tested the performance of four different time slot sizes by evaluating their average inference time and weighted F1-Score.

Fig. 7(a) compares the accuracy and weighted F1-score of the simple GNN and Ensemble GNN models as a function of the time slot size. The simple GNN model achieves an accuracy of 93.5% weighted F1-score of 92.9% for a time slot size of 10 and increases accuracy to 99.14%, the same increase seen in weighted F1-score 99.13% for a time slot size of 50. The Ensemble GNN model achieves a weighted F1-score and accuracy of 93.9% and 92.29%, respectively, for a time slot size of 10 and increases to 99.10% and 99.29% for a time slot size of 50. FTG-Net-E consistently outperforms FTG-Net in the weighted F1-score as expected.

Fig. 7(b) shows the average inference time of the GNN and Ensemble GNN models as a function of the time slot size. The GNN model has an average inference time of 8.12 ms for a time slot size of 10 and increases to 82 ms for a time slot size of 50. The Ensemble GNN model has an average inference time of 10 ms for a time slot size of 10 and increases to 98 ms for a time slot size of 50. Noteworthy, up to a time slot size of 20, FTG-Net-E has an almost negligible increase of inference time, while having a higher F1 score, making it the best choice in this range. In particular, for a time slot of 20, it increases the F1 score of more than 1%.

The results from the best models for each training phase are summarized in Table 1, with a comparison with three models from the literature.

FTG-Net-E shows a higher accuracy in respect to all considered models, and an F1 score that is surpassed by less than 0.5%. Noteworthy, FTG-Net-E achieves this by solely considering the network structure and without relying on numerous stateful features. Hence, our approach significantly reduces computational complexity and system latency, demonstrating the advantage of focusing solely on the network structure rather than incorporating numerous stateful features.

To testify how our approach is significantly more computationally efficient than previous methods consider for example the CICIDS2018 dataset. FTG-Net-E takes 10 min to train a model with 8.3 million parameters, while LUCID [15] takes 1 h to train a model with the same number of parameters. This computational efficiency makes our approach much more practical for real-world applications.

6.2. Limitations and potential challenges

The experimental results showcased the efficacy of FTG-Net-E in mitigating DDoS attacks. However, deploying such a system in real-world scenarios may encounter several challenges and limitations. Scaling FTG-Net-E to large-scale network environments might pose computational and memory challenges. Evaluating scalability issues depends on hardware dependent and requires optimizing model architecture, training procedures, and inference strategies. Real-world deployment relies on the availability and quality of network traffic data. Strategies for collecting and augmenting of network data and representative datasets need also a solid consideration. Network traffic patterns evolve due to changes in network configurations, user behaviors, and emerging threats. We can also consider a factor of implementing FTG-Net-E in resource-constrained environments, such as edge devices or IoT networks, which may necessitate model compression, optimization, or distributed learning approaches to reduce computational and memory overhead. We used to minimize the effect of adversarial attacks but ensuring the robustness and security of FTG-Net-E against adversarial attacks, model poisoning, and data tampering is important to evaluate. Robust training techniques, model validation procedures, and adversarial defense mechanisms need more exploration. Future research endeavors should focus on mitigating these challenges to facilitate the effective deployment of FTG-Net-E in real-world cybersecurity ecosystems.

7. Conclusions and future work

This invited paper presented a novel DDoS detection approach based on ensemble GNNs. This paper extended FTG-Net, a novel approach for robustly detecting DDoS attacks using GNN. FTG-Net-E leverages a hierarchical traffic representation and a hierarchical GNN model, enabling it to capture significant structural patterns embedded in network traffic. Unlike solutions that combine traffic high-level topology with statistical flow features, FTG-Net-E offers a distinct advantage: the Flow GNN and Traffic GNN architectures are intimately connected through a shared training phase. Additionally, FTG-Net-E employs ensemble

GNN techniques to enhance the model's predictive power further. A comprehensive evaluation of FTG-Net-E using the CIC-IDS2017 and CICIDS2018 datasets demonstrates that by focusing on traffic structure only, FTG-Net-E can achieve performance comparable to state-of-the-art approaches while avoiding the need for stateful features. This finding underscores the effectiveness of the FTG-Net-E methodology, including the ensemble GNN component, opening up possibilities for exploring different data representations based on similar principles. FTG-Net-E is well-suited for resource- and time-constrained real-world scenarios, making it an attractive option for distributed and quantized lightweight solutions. Our results demonstrate that the ensemble GNN model can achieve state-of-the-art performance on botnet detection tasks. The model achieved an F1 score of 0.9929, accuracy of 0.9967, precision of 0.9936, and recall of 1.0. These results indicate that this FTG-Net-E with ensemble GNN can accurately predict both positive and negative cases while minimizing false positives and negatives.

In future work, we will extend FTG-Net-E architecture to incorporate additional features, such as historical multi-traffic data or information about the network topology and multi-attack classification. We plan to investigate advanced techniques for efficiently identifying and responding to concept drift, such as online learning algorithms or specialized methods for detecting specific changes in attack patterns. Not only limited to DDoS, we can also explore and evaluate other types of attacks such as SQL injection, DNS poisoning, and Metasploit attack detection.

CRediT authorship contribution statement

Rana Abu Bakar: Conceptualization, Data curation, Investigation, Methodology, Software, Writing – original draft. **Lorenzo De Marinis:** Methodology, Formal analysis, Visualization, Writing – review & editing. **Filippo Cugini:** Funding acquisition, Project administration, Supervision, Writing – review & editing. **Francesco Paolucci:** Project administration, Funding acquisition, Conceptualization, Supervision, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work is supported by the European Union's Horizon RIA research and innovation programme under grant agreement No. 101092908 (SmartEdge) and by the EU under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART"). Work is carried out within the Department of Excellence in AI and Robotics 2023–2027.

References

[1] Internet Crime Complaint Center IC3, FBI internet crime report 2021, 2021.
[2] Kaspersky Lab ZAO, Ddos attacks in Q2 2022, 2022.
[3] I. Ortega-Fernandez, M. Sestelo, J.C. Burguillo, C. Pinon-Blanco, Network intrusion detection system for ddos attacks in ICS using deep autoencoders, Wirel. Netw. (2023) 1–17.
[4] K. Rusek, J. Suárez-Varela, A. Mestres, P. Barlet-Ros, A. Cabellos-Aparicio, Unveiling the potential of graph neural networks for network modeling and optimization in SDN, in: Proceedings of the 2019 ACM Symposium on SDN Research, ACM, 2019, http://dx.doi.org/10.1145/3314148.3314357.
[5] L. Barsellotti, L. De Marinis, F. Cugini, F. Paolucci, FTG-Net: Hierarchical flow-to-traffic graph neural network for ddos attack detection, in: 2023 IEEE 24th International Conference on High Performance Switching and Routing, HPSR, IEEE, 2023, pp. 173–178.
[6] M. Wang, Y. Cui, X. Wang, S. Xiao, J. Jiang, Machine learning for networking: Workflow, advances and opportunities, Ieee Netw. 32 (2) (2017) 92–99.
[7] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016, http://www.deeplearningbook.org.
[8] Y. Cao, H. Jiang, Y. Deng, J. Wu, P. Zhou, W. Luo, Detecting and mitigating ddos attacks in SDN using spatial-temporal graph convolutional network, IEEE Trans. Dependable Secure Comput. 19 (6) (2021) 3855–3872.
[9] I. Sharafaldin, A.H. Lashkari, A.A. Ghorbani, Toward generating a new intrusion detection dataset and intrusion traffic characterization, in: 4th International Conference on Information Systems Security and Privacy, Vol. 1, ICISSP, 2018, pp. 108–116.
[10] L. Barsellotti, F. Alhamed, J.J. Vegas Olmos, F. Paolucci, P. Castoldi, F. Cugini, Introducing data processing units (DPU) at the Edge, in: 2022 International Conference on Computer Communications and Networks, ICCCN, 2022, pp. 1–6, http://dx.doi.org/10.1109/ICCCN54977.2022.9868927.
[11] G. Zhang, Z. Li, J. Huang, J. Wu, C. Zhou, J. Yang, J. Gao, Efraudcom: An e-commerce fraud detection system via competitive graph neural networks, ACM Trans. Inf. Syst. (TOIS) 40 (3) (2022) 1–29.
[12] F. Musumeci, A.C. Fidanci, F. Paolucci, F. Cugini, M. Tornatore, Machine-learning-enabled ddos attacks detection in P4 programmable networks, J. Netw. Syst. Manage. 30 (2022) 1–27.
[13] L. De Marinis, E. Paolini, R. Abu Bakar, F. Cugini, F. Paolucci, Cascaded look up table distillation of P4 deep neural network switches, in: Globecom 2023 - 2023 IEEE Global Communications Conference: Next-Generation Networking and Internet, 2023, pp. 2112–2117.
[14] F. Cugini, D. Scano, A. Giorgetti, A. Sgambelluri, L.D. Marinis, P. Castoldi, F. Paolucci, Telemetry and AI-based security P4 applications for optical networks [invited], J. Opt. Commun. Netw. 15 (1) (2023) A1–A10, http://dx.doi.org/10.1364/JOCN.470118.
[15] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martínez-del Rincón, D. Siracusa, Lucid: A practical, lightweight deep learning solution for ddos attack detection, IEEE Trans. Netw. Serv. Manag. 17 (2) (2020) 876–889, http://dx.doi.org/10.1109/TNSM.2020.2971776.
[16] H. Ko, I. Praca, S.G. Choi, Anomaly detection analysis based on correlation of features in graph neural network, Multimedia Tools Appl. (2023) 1–15.
[17] V.-A. Nguyen, D.Q. Nguyen, V. Nguyen, T. Le, Q.H. Tran, D. Phung, Regvd: Revisiting graph neural networks for vulnerability detection, in: Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings, 2022, pp. 178–182.
[18] C. Lin, Y. Xu, Y. Fang, Z. Liu, VulEye: A novel graph neural network vulnerability detection approach for PHP application, Appl. Sci. 13 (2) (2023) 825.
[19] Y. Zhang, C. Yang, K. Huang, Y. Li, Intrusion detection of industrial internet-of-things based on reconstructed graph neural networks, IEEE Trans. Netw. Sci. Eng. (2022).
[20] C. Liu, B. Li, J. Zhao, Z. Zhen, X. Liu, Q. Zhang, Fewm-hgcl: Few-shot malware variants detection via heterogeneous graph contrastive learning, IEEE Trans. Dependable Secure Comput. (2022).
[21] H. Wang, H. Xu, L. Huang, Y. Zhai, Fast and accurate traffic measurement with hierarchical filtering, IEEE Trans. Parallel Distrib. Syst. 31 (10) (2020) 2360–2374.
[22] S. Günnemann, Graph neural networks: Adversarial robustness, Graph Neural Netw. Found. Front. Appl. (2022) 149–176.
[23] D. Pujol Perich, J.R. Suárez-Varela Maciá, A. Cabellos Aparicio, P. Barlet Ros, Unveiling the potential of graph neural networks for robust intrusion detection, in: 3rd International Workshop on AI in Networks and Distributed Systems, 2021, pp. 1–7.
[24] Y. Li, R. Li, Z. Zhou, J. Guo, W. Yang, M. Du, Q. Liu, GraphDDoS: Effective DDoS Attack Detection Using Graph Neural Networks, in: 2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design, CSCWD, 2022, pp. 1275–1280, http://dx.doi.org/10.1109/CSCWD54268.2022.9776097.
[25] X. Song, J. Li, Q. Lei, W. Zhao, Y. Chen, A. Mian, Bi-CLKT: Bi-graph contrastive learning based knowledge tracing, Knowl.-Based Syst. 241 (2022) 108274.
[26] W. Guo, H. Qiu, Z. Liu, J. Zhu, Q. Wang, GLD-net: Deep learning to detect ddos attack via topological and traffic feature fusion, Comput. Intell. Neurosci. 2022 (2019).
[27] S. Yang, S. Verma, B. Cai, J. Jiang, K. Yu, F. Chen, S. Yu, Variational co-embedding learning for attributed network clustering, Knowl.-Based Syst. 270 (2023) 110530.
[28] W.W. Lo, S. Layeghy, M. Sarhan, M. Gallagher, M. Portmann, E-graphsage: A graph neural network based intrusion detection system for iot, in: NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium, IEEE, 2022, pp. 1–9.
[29] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, Adv. Neural Inf. Process. Syst. 30 (2017).

[30] S. Nagaraja, P. Mittal, C.-Y. Hong, M. Caesar, N. Borisov, {BotGrep}: Finding {p2p} bots with structured graph analysis, in: 19th USENIX Security Symposium, USENIX Security 10, 2010.

[31] M. Alshammari, J. Stavrakakis, M. Takatsuka, A parameter-free graph reduction for spectral clustering and SpectralNet, Array 15 (2022) 100192.

[32] Q. Xiao, J. Liu, Q. Wang, Z. Jiang, X. Wang, Y. Yao, Towards network anomaly detection using graph embedding, in: Computational Science–ICCS 2020: 20th International Conference, Amsterdam, the Netherlands, June 3–5, 2020, Proceedings, Part IV 20, Springer, 2020, pp. 156–169.

[33] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, Adv. Neural Inf. Process. Syst. 30 (2017).

[34] E. Caville, W.W. Lo, S. Layeghy, M. Portmann, Anomal-e: A self-supervised network intrusion detection system based on graph neural networks, Knowl.-Based Syst. 258 (2022) 110030.

[35] S. Wu, F. Sun, W. Zhang, X. Xie, B. Cui, Graph neural networks in recommender systems: a survey, ACM Comput. Surv. 55 (5) (2022) 1–37.

[36] X. Hou, P. Qi, G. Wang, R. Ying, J. Huang, X. He, B. Zhou, Graph ensemble learning over multiple dependency trees for aspect-level sentiment classification, 2021, arXiv preprint arXiv:2103.11794.

[37] W. Wei, M. Qiao, D. Jadav, GNN-ensemble: Towards random decision graph neural networks, 2023, arXiv preprint arXiv:2303.11376.

[38] S. Barai, Y. Reich, Ensemble modelling or selecting the best model: Many could be better than one, Ai Edam 13 (5) (1999) 377–386.

[39] Y. Wang, J. Li, W. Zhao, Z. Han, H. Zhao, L. Wang, X. He, N-STGAT: Spatio-temporal graph neural network based network intrusion detection for near-earth remote sensing, Remote Sens. 15 (14) (2023) http://dx.doi.org/10.3390/rs15143611, URL https://www.mdpi.com/2072-4292/15/14/3611.

[40] Q. Qi, P.H. Kwok, Traffic4cast 2020–graph ensemble net and the importance of feature and loss function design for traffic prediction, 2020, arXiv preprint arXiv:2012.02115.

[41] H. Zhu, J. Lu, Graph-based intrusion detection system using general behavior learning, in: GLOBECOM 2022-2022 IEEE Global Communications Conference, IEEE, 2022, pp. 2621–2626.

[42] B. Esmaeili, A. Azmoodeh, A. Dehghantanha, G. Srivastava, H. Karimipour, J.C.-W. Lin, A GNN-based adversarial internet of things malware detection framework for critical infrastructure: Studying gafgyt, mirai and tsunami campaigns, IEEE Internet Things J. (2023).

[43] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Trans. Neural Netw. 20 (1) (2008) 61–80.

[44] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S.Y. Philip, A comprehensive survey on graph neural networks, IEEE Trans. Neural Netw. Learn. Syst. 32 (1) (2020) 4–24.

[45] I.R. Ward, J. Joyner, C. Lickfold, Y. Guo, M. Bennamoun, A practical tutorial on graph neural networks, ACM Comput. Surv. 54 (10s) (2022) 1–35.

[46] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, 2016, arXiv preprint arXiv:1609.02907.

[47] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, 2017, arXiv preprint arXiv:1710.10903.

[48] A. Mohammed, R. Kora, A comprehensive review on ensemble deep learning: Opportunities and challenges, J. King Saud Univ.-Comput. Inf. Sci. (2023).

[49] L. Von Krannichfeldt, Y. Wang, G. Hug, Online ensemble learning for load forecasting, IEEE Trans. Power Syst. 36 (1) (2020) 545–548.

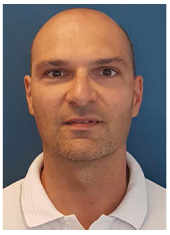
[50] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, 2016, arXiv preprint arXiv:1609.02907.

[51] A. Gharib, I. Sharafaldin, A.H. Lashkari, A.A. Ghorbani, An evaluation framework for intrusion detection dataset, in: 2016 International Conference on Information Science and Security, ICISS, IEEE, 2016, pp. 1–6.

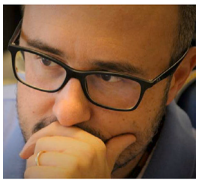
[52] M. Sarhan, S. Layeghy, M. Portmann, Evaluating standard feature sets towards increased generalisability and explainability of ML-based network intrusion detection, Big Data Res. 30 (2022) 100359.

Rana Abubakar received Undergraduate and Master's degrees in Computer Science from COMSATS University and Virtual University in 2015 and 2018. Currently pursuing a Ph.D. degree. From 2015 to October 2022, he was an Assistant Manager of Automation with Century Paper and Board Mills, Pakistan. Since 2019, he has been a visiting lecturer with the Computer Science Department of the University of Sahiwal. He has also served as a research scholar at Chulalongkorn Thailand from 2020 to 2022. His research interests include Software Defined Networking, Datacenter Networking, HighPerformance Computing, Confidentiality Computing, Homomorphic Cryptography, Authentication Protocol development, GPU/DPU Processing, Network Function Virtualization, Distributed systems, Artificial Intelligence and Machine Learning, and Resilient Edge Platforms Network Security.

Lorenzo De Marinis received the B.S. and M.S. degrees (cum laude) in electronic engineering from the University of Pisa, in 2017 and 2019, respectively. He received the Ph.D. (cum Laude) in 2022 from Scuola Superiore Sant'Anna (SSSA), Pisa, Italy. From November 2021 to April 2022, he was a visiting scholar at the Aristotle university of Thessaloniki, Thessaloniki, Greece, at the WinPhos laboratory. He was a research fellow in quantum and neuromorphic photonics from September 2022 to February 2023 at SSSA, where he is currently an Assistant Professor. His main research concerns the conceptualization and design of photonic integrated circuits for quantum and neuromorphic applications, analog computing, photonicelectronic codesign and machine learning for networking scenarios.



Filippo Cugini is currently the Head of Research Sector with CNIT, Pisa, Italy. He is the coauthor of 14 patents and more than 300 international publications. His main research interests include theoretical and experimental studies in the field of communications and networking. In particular, the focus is on flexible optical networks, disaggregated solutions, software-defined networking (SDN), HighPerformance Computing, Photonic Integrated Networks, Edge Computing, and in-network programmability including P4. He currently serves as Project Coordinator of the EU-funded projects BRAINE (Big Data Processing and Artificial Intelligence at the Network Edge), SmartEdge (Semantic Lowcode Programming Tools for Edge Intelligence), and SEASON (Self-managed Sustainable high-capacity Optical Networks).



Francesco Paolucci is Head of Research at CNIT, Pisa, Italy. His main research interests are in the field of network control plane, edge/cloud networking platforms, traffic engineering, network disaggregation, advanced monitoring/telemetry, SDN data plane programmability. He has been involved in many industrial and European research projects on next generation control networking (METROHAUL, B5GOPEN, BRAINE, DESIRE6G, SmartEdge, CLEVER). He is co-author of 3 IETF Internet Drafts, more than 200 publications in international journals, conference proceedings and book chapters, and filed 4 international patents. He is Associate Editor of the IEEE/OSA Journal of Optical Communications and Networking (JOCN) and Executive Editor of the Transactions on Emerging Telecommunications Technologies (ETT).