Towards an Efficient Network Intrusion Detection System for IoT Networks Leveraging Graph Neural Networks

by **Tanzeela Altaf**

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

under the supervision of Dr. Xu Wang, Prof. Ren Ping Liu, Prof. Robin Braun

Thesis by Compilation

School of Electrical and Data Engineering

Faculty of Engineering and IT

University of Technology Sydney

April 29, 2024

Certificate of Authorship / Originality

I, Tanzeela Altaf, declare that this thesis is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the School of Electrical and Data Engineering at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis. This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Date: April 29, 2024

Abstract

Existing deep learning approaches are barely effective for identify new attacks in IoT traffic because they treat network flows independently. Graph Neural Networks (GNNs) have emerged as a promising alternative having the ability to capture the underlying network topology. However, existing approaches focus solely on either node or edge features, limiting their capacity to fully understand the complexities of network data. This limitation impacts the performance of NIDS in detecting new attacks, as they fail to utilize the contextual information provided by both node and edge features. To address this, our research explores GNN mechanisms and graph structures tailored to IoT traffic.

We introduce NE-GConv, a Directed Graph model that incorporates both node and edge features, and a Multi-graph capable of representing comprehensive communication between IoT nodes. NE-GConv improves upon existing methods with enhancements in algorithm input, message aggregation, update functions, and output. Our approach enables deep inspection by incorporating flow and packet content-related features. Additionally, our Multi-edged model accommodates multiple edges and features, utilizing modified message-passing layers and aggregation functions. We introduce novel equations to integrate multi-edge considerations into the GNN framework. Extensive experiments, evaluated using metrics, validate the effectiveness of our proposed models compared to state-of-the-art GNN approaches.

Acknowledgements

First of all, I would like to thank God for His countless blessings, and for giving me the wisdom and strength to accomplish this research. I extend my deepest appreciation to my perfect parents for their unconditional love and support, for believing in me. Their profound impact on my life cannot be overstated, and I am forever grateful for their presence and dedication.

I would like to express my deep gratitude to my supervisors; Dr. Xu Wang, for allowing me to undertake this research and for his unceasing guidance and invaluable support, Prof. Wei Ni, for his beneficial advice, interesting and challenging conversations that enriched my knowledge, and research over the past few years, Prof. Ren Ping for believing in me and for his unlimited support, and Prof. Robin Braun for his continuous encouragement and motivating me all through the PhD. A very special thanks goes out to my co-author Guangsheng Yu for his guidance and help in improving the paper writing.

The completion of this academic endeavor would not have been possible without the unwavering support and encouragement of my husband and two beloved children. I extend my heartfelt gratitude to my siblings, whose served as a source of inspiration throughout this journey. Special thanks is due to my brother, who graciously endured my mock presentations, attentively providing feedback and constructive insights. Their collective contributions have been invaluable in shaping the outcome of this work, and for that, I am profoundly grateful. My friends have always been there for me, without their support, I would not have come that far. Thank you to all my friends and family for looking up to me.

Tanzeela Altaf

April 29, 2024

Sydney, Australia

Contents

List of Figures

List of Tables

Published Journal Papers

J-1. **T. Altaf**, X. Wang, W. Ni, R. P. Liu, R. Braun, "NE-GConv: A lightweight node edge graph convolutional network for intrusion detection," *Elsevier: Computers Security*, 2023, (Chapter 3).

J-2. **T. Altaf**, X. Wang, W. Ni, G. Yu, R. P. Liu, R. Braun, "A new concatenated Multigraph Neural Network for IoT intrusion detection," *Elsevier: Internet of Things*, 2023, (Chapter 4).

Published Conference Papers

C-1. **T. Altaf**, R. Braun, "Roadmap to Smart Homes Security Aided SDN and M," *IEEE Conference on Cloud and Internet of Things (CIoT)*, 2022, (Chapter 2).

C-2. G. Yu, Q. Wang, **T. Altaf**, X. Wang, X. Xu and S. Chen, "Predicting NFT Classification with GNN: A Recommender System for Web3 Assets," *IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2023.

Chapter 1

Introduction

The Internet of Things (IoT) is a leading technology that has taken markets, industries and, business goals to a next level by introducing the idea of smart applications in every aspect of life. It contains a vast pool of internet-enabled sensors some of which have limited capability and operate in a specific manner only. Many manufacturers are launching devices at an ever-increasing rate and security requirements are often an afterthought [1]. Due to the heterogeneity and operational complexity in the design, these devices pose vulnerabilities within the network and may lead to zero-day attacks. Therefore, maintaining the security of the IoT network is fundamental to the successful deployment of resource-constraint devices. Existing studies in the past decade have led to profound improvements in the development of NIDS using Artificial Intelligence (AI), and in particular, DL [2]–[8]. However, there is need to explores the suitability of using new and non conventional DL techniques, e.g. GNN, to build the next generation of IDS. This thesis investigates the limitations of conventional and deep learning techniques used in IDS and identifies the specific challenges they face in capturing network topology information.

1.1 Background

A smart home, as an integral applications of Internet of Things (IoT), allows users to conveniently control home devices remotely [9], [10]. It involves computing techniques to interact with smart appliances, heating, and lighting systems, as well as recognizing user activities through various means like voice or gesture recognition, computer vision, and human behavior analysis [11]–[13]. With smart home utilities, users can remotely access and manage their homes, controlling temperature sensors, lighting systems, and security cameras even from a distance.

This revolution in smart homes aims to minimize human intervention in daily tasks and provide control of the home environment. According to an analysis by Gartner, there has been significant growth in the sales of smart home devices [14], but the full potential is yet to be realized due to a gap between consumer demands and manufacturer expectations. The devices are often produced by different manufacturers and may have interdependencies within the home network [15]. However, IoT-based smart homes are susceptible to cybercrimes and security exploitation, posing a major concern as sensitive user information is transmitted through the network. Many communications between devices lack encryption [16], and recent attacks incidents highlight the exploitation of vulnerabilities and demonstrate the evolving sophistication of cyber threats.

1. **Mirai Botnet Attack (2016)**: This attack exploited insecure IoT devices to launch massive Distributed Denial of Service (DDoS) attacks, causing widespread internet disruptions. The Mirai botnet infected cameras, routers, and other IoT devices by exploiting default passwords [17].

2. **BrickerBot Attack (2017)**: BrickerBot targeted vulnerable IoT devices with the intent to permanently disable them. The attack exploited security weaknesses in devices such as cameras and routers, rendering them inoperable [18].

3. **VPNFilter Malware (2018)**: This malware infected hundreds of thousands of routers and IoT devices worldwide. It allowed attackers to spy on network traffic, steal sensitive data, and even launch destructive attacks [19].

4. **Trickbot IoT Variant (2020)**: Trickbot, originally a banking trojan, expanded its capabilities to target IoT devices. It aimed to create a botnet for launching DDoS attacks, stealing sensitive information, and spreading malware across networks [20].

5. **Ripple20 Vulnerabilities (2020)**: A series of 19 zero-day vulnerabilities discovered in a popular TCP/IP stack used in many IoT devices. These vulnerabilities could allow attackers to remotely execute code, steal data, or disrupt device functionality [21].

Once a hacker gains access to one device, they can exploit the entire network through techniques like spoofing, eavesdropping, or flooding attacks.

The heterogeneous nature of smart home sensors, ranging from simple light bulbs to complex smartphones, adds to the complexity, with no uniform standards among devices. The varying connectivity modes and protocols of different devices present challenges for communication and security. It is not possible to apply the same defense mechanisms to attacks in IoT as are applied in the conventional network [1]. There is a need to devise automated security solutions that can eliminate these evolving attacks.

ML approaches in securing IoT systems have been pivotal in enhancing intrusion detection systems and safeguarding against cyber threats. ML algorithms, such as Support Vector Machines (SVMs), Random Forests, and k-Nearest Neighbors (k-NN), have been instrumental in identifying patterns and anomalies in network traffic, contributing to the development of effective security frameworks. These techniques analyze historical data to classify network activity as normal or malicious, aiding in the timely detection and prevention of potential intrusions. Deep Learning (DL) approaches, however, have emerged as the forefront of security solutions for IoT-based smart homes, surpassing traditional ML methods in adaptability and efficacy.

This thesis aims to address these challenges and devise robust security solutions for IoT-based applications. The focus will be on developing automated and adaptive detection mechanisms to counter emerging cyber threats and enhance the security and privacy of IoT networks.

Conventional DL techniques often face limitations in effectively detecting intrusions in IoT environments due to their inability to adapt to the dynamic and complex nature of network traffic. These techniques may struggle to capture the subtle patterns and anomalies indicative of cyber threats, leading to higher false-positive rates and reduced accuracy in intrusion detection. Additionally, traditional DL models may lack the flexibility to handle the diverse and evolving attack vectors encountered in IoT networks, resulting in suboptimal performance in real-world scenarios. By leveraging advanced ML and DL techniques, we aim to build a comprehensive and effective security framework that can proactively detect and prevent potential intrusions by considering the complexities of underlying network topology and intricate traffic patterns, safeguarding users' data and ensuring the smooth functioning of IoT systems. Through this research, we aim to contribute to the advancement of IoT security, bridging the gap between consumer expectations and manufacturer innovations in this rapidly evolving field.

1.1.1 Smart Home and Services

The purpose of this section is to explain the smart home in detail and also highlight the level of maturity of present challenges in different services. We provide an overview of some of the services that a smart home offers to its users by enabling remote monitoring and discuss some of the research challenges in each sub-section. This, in turn, will give a direction to future researchers in defining the scope of research in these areas. Security and privacy is one of the primary issues in all of these sub-areas.

Health Monitoring and Elderly Care

Healthcare is one of the most valuable part of IoT-based smart homes which is gaining relevance with the improvement of IT. IoT has enabled a world of possibilities: a remote health monitoring with the aid of smart health gadgets leading to a run-time care [22], providing physicians with online data to monitor the ongoing symptoms and conditions [23], and thus excelling the medical processes. It is in-turn turning into a compelling answer to the requirements of physically disable and elderly people and enabling more independent and self-controlling lives.

Different countries have carried out studies to make a smart home fully equipped with the needs of elderly and physically hindered individuals. Remote health monitoring systems play one of the vital role including keeping a record of blood pressure, blood glucose, heart rate, body temperature [22], weight, pulmonary peak flow meters and other psychological information.

Apart from improving the health sector with the help of technology another key research area in this domain is privacy and security of patients data [23]. Author insists that despite huge claims by previous studies security and privacy is significantly overlooked and recent work is not enough for catering this issue [24]. Medical data may be exposed to participatin entities for the purpose of maintenance and storage i.e. third party cloud service providers which is creating more privacy concerns as the dependencies on other parties are nearly impossible to eliminate [25].

Smart home Energy Management

By definition, this serves to optimize energy consumption with in a smart home by allowing its users to track, control and monitor their gas and electricity usage with the aid of energy-management devices. Smart energy management devices may include a smart thermostat, gateways, smart meter, smart switches, plugs and sensors connected to the internet.

A lot of research is going on in this domain where some of the key issue is to schedule electrical task scheduling to manage cost while considering consumer's preferences. one such algorithm is presented specifying a time slot for a task to improve user comfort and manage electricity usage [26]. Smart plugs are used to develop a facility which can turn off the home plugs from central server when not in use [27], in addition consumer behaviour is learned by analysing the data obtained from smart plugs.

Home Automation

The idea of home automation was first proposed in 1975 with the release of X10 technology which allowed users to send signals over electric wiring to the home switches and eventually to the devices telling them when to turn on. Since, signals diminished over the wires of different polarities, it was not an ideal and fully reliable communication protocol. With the evolution of smart home and its devices, a company named Insteon came up with technology that connected together wireless signals and electric wiring. Over time other technologies emerged e.g. Z-wave, Zig-Bee, Insteon, Bluetooth, Ethernet,Wifi, RS232, RS485, C-bus, UPB, KNX, EnOcean and Thread [28] which can counter problems occurred in X10 and each comes with its own pros and cons.

Consequently, in smart home and services, security and privacy concerns pervade across various use cases. In health monitoring and elderly care, the challenge lies in safeguarding sensitive patient data, as medical information transmitted through IoT devices faces risks of exposure to third-party entities. Similarly, smart home energy management confronts issues of data privacy, particularly concerning the collection and analysis of energy consumption patterns. Home automation, while offering convenience, raises concerns regarding unauthorized access to connected devices and potential breaches of personal privacy. These challenges underscore the critical need for robust security measures to mitigate risks and ensure the integrity of IoT-based smart home systems.

1.1.2 Machine Learning and its Categories

Machine learning is a type of Artificial Intelligence that empowers a framework to learn from previous data rather than through explicit programming. However, machine learning is not a straightforward procedure. It consists of a variety of algorithms which act as building blocks to the whole system. Their job is to create a logic by learning from the training data and predict outcomes close to reality.The more representative training data an algorithm ingests, the more accurate the model is. Also, choosing a right algorithm is very important and can be achieved by trial-and-error in most cases. Machine learning can be grouped into three main categories [29].

Supervised Learning

Supervised learning is used in a case when a certain problem has an established set of data and a significant information as to how that data is classified. It tends to figure out patterns in the incoming data that can be applied to a category that define the meaning of data.

Unsupervised Learning

Unsupervised learning is applied in a case when a problem has a massive amount of unlabeled data. This category includes a set of algorithms that can help classify this massive amount of data based on patterns or features into groups or clusters. So, it is a process to discover the relationship among data points.

Reinforcement Learning

Reinforcement learning is a behavioral learning model and is applied to the problems where future actions are based on the output of present responses and next actions are required to be determined. In order to get a precise outcome, feedback is received by algorithm after analysing the data. As the system learns through trial and error, a sequence of successful decisions will result in the process being “reinforced” because it best tackles the current issue.

Deep Learning

A neural network consists of set of algorithms that aspires to find links in the incoming data in a much similar way as a human brain may operate. it contains three or more layers: an input layer, one or many hidden layers, and an output layer. Data is fed through the input layer and modified depending on defined weights on the nodes of hidden layer and output layers. A neural network may comprise of hundreds or even millions of nodes connected to each other. The typical neural network may consist of thousands or even millions of simple processing nodes that are densely interconnected in one layer. The term deep learning is implied when a neural network consists of more than one hidden layers.

1.2 Topic and Thesis

"Enhancing Intrusion Detection Systems using Graph Neural Networks: A Novel Approach for Incorporating Network Topology in Deep Learning"

Problem Statement: IDS in the IoT face significant challenges due to the dynamic and heterogeneous nature of IoT environments. Traditional IDS approaches often struggle to adapt to the complexities of IoT networks, leading to limitations in accuracy, scalability, and robustness. Furthermore, the rapid proliferation of IoT devices increases the challenge of effectively monitoring and securing network traffic against emerging cyber threats. Therefore, there is a pressing need to enhance IDS capabilities to address these challenges and ensure the security and integrity of IoT systems.

Hypothesis: By leveraging GNNs, a novel and promising deep learning technique, it is hypothesized that IDS in IoT can be significantly improved. GNNs offer unique advantages in capturing and analyzing complex network topologies inherent in IoT environments, allowing for more accurate and comprehensive intrusion detection.

This thesis explores and demonstrates the potential of Graph Neural Networks (GNNs) as a non-conventional deep learning technique for addressing the limitations of traditional machine learning and deep learning approaches in Intrusion Detection Systems (IDS). By leveraging the inherent graph structure designed to represent the nature of intrusion detection problems, the research will develop and evaluate novel GNN-based architectures that effectively capture network topology information, leading to more accurate and robust intrusion detection capabilities. The findings of this study will contribute to the advancement of IDS technology, offering insights into the suitability and practical deployment of GNNs for building the next generation of intrusion detection solutions.

1.3 Aims, Objectives and Significance

The primary aim of this thesis is to investigate and assess the suitability of GNNs as a non-conventional deep learning technique for enhancing IDS. The research aims to address the limitations of conventional machine learning and deep learning approaches in capturing network topology information, by leveraging the graph structure designed to represent the nature of intrusion detection problems. The thesis will develop and evaluate novel GNN-based architectures specifically tailored for IDS, with a focus on their ability to improve the accuracy and robustness of intrusion detection in IoT network environments. Investigating and assessing the suitability of Graph Neural Networks (GNNs) for enhancing Intrusion Detection Systems (IDS) is crucial for several reasons. Firstly, traditional machine learning and deep learning approaches often struggle to effectively capture the intricate network topology information inherent in intrusion detection problems, especially in the context of IoT network environments. By leveraging GNNs, which are adept at understanding and processing graph-structured data, it becomes possible to address these limitations and develop more accurate and robust intrusion detection mechanisms.

Moreover, the adoption of GNNs in IDS has the potential to solve several critical problems in cybersecurity. These include improving the detection of sophisticated and evolving cyber threats, enhancing the resilience of IoT networks against malicious activities, and minimizing the occurrence of false positives and false negatives in intrusion detection. Additionally, GNNs offer scalability and adaptability, making them well-suited for the dynamic and heterogeneous nature of IoT environments. By developing novel GNN-based architectures tailored for IDS, it becomes possible to achieve higher levels of accuracy and efficiency in identifying and mitigating security breaches in IoT networks.

In terms of security goals, this PhD thesis aims to contribute towards enhancing the overall security posture of IoT networks. By improving the accuracy and robustness of IDS through GNNs, the thesis seeks to fortify IoT systems against a wide range of cyber threats, thereby safeguarding sensitive data, preserving user privacy, and ensuring the uninterrupted operation of IoT applications and services. Furthermore, by developing advanced intrusion detection techniques, the thesis aims to foster trust and confidence in IoT technologies among users, regulators, and industry stakeholders.

Beyond technical impacts, the research conducted in this thesis can have significant social, economic, and global implications. From a social perspective, enhancing the security of IoT networks contributes to protecting individuals' privacy and personal information, thereby fostering trust in digital technologies and promoting digital inclusion. Economically, mitigating the risks associated with cyber threats in IoT networks can prevent financial losses for individuals and organizations, as well as safeguard critical infrastructure and services. Additionally, on a global scale, strengthening the security of IoT networks can contribute to cybersecurity efforts worldwide, helping to mitigate the growing threats posed by cybercriminals and state-sponsored actors. Overall, investigating and enhancing GNNs for IDS in IoT networks holds the potential to have wide-ranging positive impacts on society, the economy, and global cybersecurity.

The main objectives of this thesis can be summarised in the following research questions:

1. How do conventional machine learning and deep learning techniques currently used in Intrusion Detection Systems (IDS) fall short in effectively capturing network topology information?

2. What are the fundamental principles and capabilities of GNNs as a non-conventional deep learning technique, and how can they be harnessed to incorporate network topology information into IDS models?

3. How can novel GNN-based architectures be designed and implemented to cater specifically to the requirements of IDS, and how do their performance compare against traditional machine learning and deep learning approaches on benchmark datasets?

4. What are the implications of using GNN-based IDS models in large-scale network environments, in terms of their scalability and computational efficiency?

5. How do the proposed GNN-based IDS models perform in comparison to conventional techniques when applied to IoT network data and diverse intrusion patterns?

Relocating the last paragraph of previously 1.0.3 into section 1.3 as suggested by the reviewer.

This thesis examines the capabilities of machine learning, deep learning and in particular GNNs as a non-conventional deep learning technique for effectively incorporating network topology information into IDS models. Different GNN structures are modelled based on IoT network behaviour, and model architectures are designed and implemented to tailor the specific requirements of IDS. This thesis proposes a series of models to capture the interaction between IoT devices, normal and abnormal exchange of information by employing nodes and edges in the graph topology, use of spectral and convolutional methods, and the use of unique aggregation and propagation mechanism in GNN learning. Results demostrate and analyze the strengths and weaknesses of GNN-based IDS models in comparison to conventional techniques, with a focus on their ability to handle network topology and adapt to different types of intrusion patterns.

This thesis holds significant importance in the field of cybersecurity and intrusion detection. By exploring the utilization of Graph Neural Networks, a novel deep learning technique specifically tailored for handling graph-structured data, the research aims to bridge the gap between conventional machine learning approaches and the complexities of network topology information. The findings of this study have the potential to revolutionize intrusion detection capabilities by providing more accurate and efficient models capable of detecting sophisticated cyber threats. Moreover, the investigation into the interpretability and robustness of GNN-based IDS models will contribute to the development of more trustworthy and resilient security systems. Ultimately, this research's significance lies in advancing the state-of-the-art in IDS technology and providing valuable insights for practitioners and researchers seeking to build the next generation of effective and adaptive intrusion detection solutions.

1.4 Thesis Structure

The thesis embarks upon an exhaustive exploration of the intersection between Internet of Things (IoT) applications and deep learning methodologies, elucidating the landscape of intrusion detection within these domains. The thesis begins with an Introduction (Chapter 1), providing background information IoT applications, and machine / deep learning categories, followed by a delineation of the topic and thesis, as well as the aims, objectives, and significance of the research. Chapter 2 delves into the literature review, covering topics such as detection of multiple attack vectors in IoT traffic, latest developments in counteracting attacks, deep learning models for IDS, and GNN modeling in cyberspace. In Chapter 3, the novelty lies in the design of a graph structure that integrates node and edge features synergistically to identify anomalies within IoT networks. This approach innovatively combines IP addresses and port numbers to form nodes, while the edges encapsulate communication patterns between these nodes. By jointly considering both node and edge features, the method effectively discerns node anomalies, contributing a novel perspective to intrusion detection in IoT environments. Chapter 4 introduces a pioneering approach by constructing a multigraph framework where IP addresses serve as nodes, representing individual IoT devices. The distinguishing feature of this methodology lies in its utilization of multi-edged communication channels to capture the complexities inherent in IoT node traffic. Additionally, the integration of spectral and spatial GNN layers, such as Graph Convolutional Networks and Graph Attention Networks, facilitates efficient learning from the intricate graph structure, thereby enhancing the detection of anomalies within IoT networks. In Chapter 5, a novel methodology is presented wherein a multiedges time series graph is formulated, with IP addresses acting as nodes and multi-edged sequential communication serving as the edge features. The innovation in this approach lies in the utilization of sequential Graph Neural Networks, particularly Gated Graph Convolutional Networks, to detect anomalies within IoT network traffic. By leveraging the temporal dynamics of communication patterns, this method offers a sophisticated means of identifying anomalous behavior, thereby contributing to the advancement of intrusion detection techniques in IoT environments.The Conclusion (Chapter 6) summarizes the findings and contributions of the research.

Chapter 2

Literature Review

In the past, many AI-based models have been developed to enhance security in the IoT. IDS-based methods are considered an important tool for network security and information systems. However, with emerging technologies and market trends, there is a need to update IDS to adapt to new attacks [30]. Network Intrusion Detection Systems (NIDS) [31],[32] are a traditional way of protecting networks from malicious activities. In general, NIDS can be classified into two types based on detection strategy: signature-based IDS [33] and anomaly-based IDS [34]. The former analyzes network traffic looking out for specific network patterns or known attacks and is unable to cater for unknown attacks whereas the latter is capable to detect zero-day attacks by flagging out any network deviation from the known normal behavior, thus, effective in determining known and unknown attacks. The downside of anomaly-based IDS is that it suffers from a high false alarm rate. The issue is aggravated in an IoT setting where, hundreds of devices, typically generated by different vendors and with limited operational capacity are connected together making the overall network prone to multiple and novel attacks. Hence, conventional intrusion detection techniques are not effective in controlling emerging cyber-attacks. There is a need to modify IDS in contemplation of new network requirements. IoT systems face various types of attacks that require a set of information exchange by hackers to compromise or attack a device or network. Some of these attacks include Distributed Denial of Service (DDoS), Man-in-The-Middle (MiTM), injection, spoofing, Sybil, buffer overflow, and botnet attacks [35]. These attacks have patterns of network traffic that are characteristics of targeted activity, e.g., a large number of devices communicating with single IP, repeated connections to a specific IP address, unusual DNS queries or responses, and/or repeated connections to different ports on the same device. These attacks can have severe consequences, including the disruption of critical infrastructure, the compromise of sensitive data, and the loss of life. It is therefore important to develop effective detection and mitigation strategies to protect against these threats.

2.1 Detection of Multiple Attack Vectors in IoT Traffic

As the IoT continues to proliferate, securing IoT networks against diverse attack vectors becomes increasingly critical. This section explores the detection methods relevant to IoT environments, focusing on the identification of multiple attack vectors that threaten the integrity and security of IoT devices and networks [35]–[37].

2.1.1 Denial of Service (DoS) Attacks

DoS attacks aim to disrupt the availability of IoT devices and services by overwhelming them with a flood of traffic or requests. Detection techniques for DoS attacks in IoT traffic often involve monitoring network traffic patterns, identifying sudden spikes in traffic volume, and implementing rate limiting or traffic filtering mechanisms to mitigate the impact of such attacks.

2.1.2 Man-in-The-Middle (MiTM) Attacks

MiTM attacks involve intercepting and manipulating communication between IoT devices and their intended destinations. Detecting MiTM attacks in IoT traffic typically involves analyzing network traffic for anomalies, such as unauthorized changes in packet headers or unexpected routing paths. Advanced cryptographic protocols and secure communication mechanisms can also be deployed to detect and prevent MiTM attacks.

2.1.3 Injection Attacks

Injection attacks, such as code injection or command injection, target vulnerabilities in IoT devices or applications to execute unauthorized commands or access sensitive data. Detection methods for injection attacks in IoT traffic often involve analyzing incoming data packets for suspicious payloads or unexpected command sequences. Intrusion detection systems (IDSs) can also be deployed to monitor device behavior and identify potential signs of compromise.

2.1.4 Spoofing Attacks

Spoofing attacks involve impersonating legitimate IoT devices or users to gain unauthorized access to network resources or services. Detecting spoofing attacks in IoT traffic typically involves implementing authentication mechanisms, such as digital signatures or certificate-based authentication, to verify the identity of devices and users. Network traffic analysis can also be used to detect anomalous behavior indicative of spoofing attempts.

2.1.5 Botnet Attacks

Botnet attacks involve compromising multiple IoT devices to create a network of bots under the control of a malicious actor. Detecting botnet attacks in IoT traffic often involves analyzing network behavior for patterns indicative of botnet activity, such as coordinated communication between devices, unusual spikes in traffic, or repetitive connection attempts to known command and control servers. Machine learning algorithms can also be employed to detect botnet behavior based on learned patterns from historical data.

In summary, detecting multiple attack vectors in IoT traffic requires a combination of techniques, including network traffic analysis, anomaly detection, cryptographic protocols, and machine learning algorithms.

2.2 Latest Developments in Counteracting Attacks

Recent developments in countering major types of attacks targeting IoT systems have also focused on integrating advanced ML and DL techniques into security solutions. ML and DL algorithms are being employed to enhance the detection and mitigation capabilities of IDS against various attacks, including DDoS, MiTM, injection, spoofing, Sybil, buffer overflow, and botnet attacks. These algorithms enable IDS to continuously learn and adapt to new attack patterns, improving their accuracy in identifying and mitigating threats. Furthermore, ML and DL techniques are being applied to anomaly detection algorithms to reduce false positive rates and improve the overall performance of intrusion detection in IoT environments.

In addition to leveraging ML and DL, researchers are exploring innovative approaches to mitigate specific types of attacks. For example, for DDoS attacks, techniques such as flow-based anomaly detection and traffic filtering are being developed to identify and block malicious traffic in real-time [38]. Similarly, for MiTM attacks, advanced cryptographic protocols and secure communication mechanisms are being deployed to prevent unauthorized interception and tampering of data [39].

The security and privacy aspects of IoT-based smart homes have been a subject of extensive research, particularly focusing on the application of machine learning techniques. In a paper [40], author addressed the privacy drawbacks of smart homes by the means of traffic analysis which can be done by intercepting traffic from gateways or by finding out profile resident‘s behaviour through digital traces. This research work designed a utility optimal differential privacy mechanism to mystify adversaries from the source of traffic flows. In addition a multi-hop routing scheme has been developed in order to preserve privacy in a hostile wireless environment. This work takes in account network energy consumption and resource constraint IoT environment. In this [41] emphasized the need of IoT based automation in homes and compared the existing technologies in terms of cost and compatibility for the awareness in clients. In turn, a nodeMCU based automation system has been implemented in which electronic switches can be controlled by a user-friendly web through Wi-Fi technology after a simple authentication.

Authors in [42] presented a survey on conservation of energy in the automated homes based on presence or absence of residents in the home. One of the most commonly deployed idea in literature is based on machine learning where data from sensors is collected to predict the mood of resident and controller makes the decision after user intervention which is making the whole output more accurate. Whereas, authors suggested that automated control advancements and energy conservation should go hand in hand and automated control based on human presence/absence is merely insufficient for conserving energy. Authors in [43] developed a framework based on machine learning which suggests a suitable action in order to conserve energy. They also implied a layer of security by using MQTTS server. Another paper [44] performed behavioural modelling of a home user embedded with authentication based inatural behaviour using Recurrent Neural Network (RNN). Goals are to preserve privacy and reduce cost and infrastructure without human inference. In another paper[45], authors explained the work done towards data annotation for activity recognition where semi-automated approach is focused for gathering up and manipulating data from sounds in smart homes and propose the use of an app (ISSA) embedded with smart microphones to help distinguish sounds.

A research paper [46] suggested incorporating deep learning and forensics techniques into the IoT architecture to expose botnets, emphasizing the applicability of network forensics and highlighting the challenges in this area. Another study [47] developed an SVM-based classifier for detecting network attacks in smart home environments, effectively differentiating between mutation and regular codes. The proposed system demonstrates reduced complexity and timely response in identifying network attack information. A study [48] generated real-time data from a smart home testbed, utilizing Hidden Markov Model (HMM) for detecting abnormal behaviors in sensors, achieving 97% accuracy in behavioral modeling. However, this approach focused on limited and specific devices, overlooking the presence of multiple residents. [49] proposed behavioral device templates through the fusion of statistical and machine learning techniques to detect anomalies in smart home networks. The system identifies anomalies based on deviations from the center of cluster developed from statistical metrics, but further work is needed to optimize thresholds and threat scores for improved performance.

In the quest to identify the behavior of smart devices within smart environments, [50] explored a machine learning framework utilizing network traffic characteristics. While this work does not directly address device security, it lays the foundation for achieving security through behavioral modeling of devices. Additionally, [51] investigated the security and privacy of smart home speakers, focusing on privacy leakage and voice command fingerprinting attacks, which involve traffic analysis. However, further research is required in this domain to enhance security measures. The IOTFLA architecture, proposed by [52], leveraged federated learning in IoT-based smart homes to enhance data security and privacy. The architecture's merits and demerits have been discussed in terms of implementation within the IoT ecosystem.

By analyzing these recent studies in Table 2.1, we aim to establish a foundation, which seeks to explore innovative approaches integratingIoT and Machine Learning (ML) technologies to enhance the security of IoT-based smart homes. Despite their promise, ML and DL techniques also present certain downsides when applied to smart home security. One major concern is the potential for adversarial attacks, where malicious actors can manipulate ML/DL models by injecting specially crafted inputs to evade detection or cause false alarms. Additionally, ML/DL models require large amounts of labeled training data to achieve high accuracy, which may be challenging to obtain for IoT environments due to privacy and data scarcity concerns. Moreover, ML/DL algorithms often require significant computational resources and energy consumption, which may be impractical for resource-constrained IoT devices in smart home

Table 2.1: List of studies in the literature that has used machine learning methods to address security issues in IoT.

Ref	**Methodology**	**Attack Type**
[46]	Review on Forensics and deep learning mechanisms	Bots
[47]	SVM Method	Network attacks (mutation code detection)
[48]	Hidden Markov Model (HMM)	Network anomaly detection
[49]	Fusion of statistical and machine learning techniques	Network anomaly detection
[50]	Multi-stage machine learning algorithm	IoT device type detection
[51]	ML algorithms	Voice command fingerprinting attack
[52]	Federated learning and data aggregation	Anomaly detection
[53]	Supervised ML, ANN, KNN	Device security and N/w security
[54]	ML algorithms and DL	Authentication, DoS, IDS, Malware

environments. Therefore, while ML and DL hold great potential for enhancing security in IoT systems, careful consideration of these limitations is necessary to ensure their effective and sustainable deployment in smart home environments.

2.3 Deep Learning Models for Intrusion Detection Systems

DL has emerged as a promising approach for enhancing the capabilities of intrusion detection systems (IDS) in detecting and mitigating various types of cyber-attacks. DL-based IDS leverage neural network architectures to automatically learn and extract features from raw data, such as network traffic or system logs, without the need for manual feature engineering. This enables DL models to capture complex patterns and anomalies associated with different types of attacks with high accuracy. DL-based IDS offer several advantages over conventional ML-based IDS. Firstly, DL models can handle high-dimensional and unstructured data more effectively, allowing them to capture subtle patterns and variations in network traffic or system behavior that may indicate malicious activity. Additionally, DL models have demonstrated superior performance in detecting unknown or zero-day attacks compared to traditional ML algorithms. Furthermore, DL-based IDS can adapt and learn from new data in real-time, enabling them to continuously improve their detection capabilities over time without manual intervention.

Despite their advantages, DL techniques often suffer from a high false positive rate and a low detection rate. The two aspects are important and need attention when designing an IDS. A study [55] put forth the design of a DAS-Collaborative Intrusion Detection Systems (CIDS) leveraging semi-supervised algorithms in aspects of intrusion detection and false alarm reduction. The results show effectiveness in detecting intrusion with good accuracy. Another work [56] entailed a thorough survey focusing on deep neural network approaches and explaining their contributions and capabilities. In [57] authors used advanced deep learning techniques for the detection of IoT attacks while network traffic is monitored at Modbus protocol. Packets are extracted and trained on an ensemble of LSTM (Long Short Term Memory) models and output is further aggregated and labeled by a decision tree. In [58], deep random neural networks and multi-layer perceptrons were combined for Industrial Internet of Things (IIoT) attack detection. The proposed scheme showed good accuracy on the DS20S and UNSW-NB15 datasets.

Table 2.2: Summary of related Studies on deep learning models for IDS in IoT.

Ref	**Methodology**	**Contributions**
[55]	DAS-Collaborative IDS leveraging semi-supervised algorithms for intrusion detection and false alarm reduction.	Effective in detecting intrusion with good accuracy.
[56]	Survey focusing on deep neural network approaches for IDS.	Provides insights into contributions and capabilities of deep learning techniques.
[57]	Uses advanced deep learning techniques for IoT attack detection using Modbus protocol.	Utilizes ensemble of LSTM models and decision tree for detection.
[58]	Combines deep random neural networks and multi-layer perceptrons for IIoT attack detection.	Shows good accuracy on DS20S and UNSW-NB15 datasets.

Continued on next page

Table 2.2 – *Continued from previous page*

Ref	**Methodology**	**Contributions**
[59]	Proposes an autonomous anomaly detection model using Gated Recurrent Unit (GRU) for compromised IoT devices.	Achieves a detection rate of 95.6% with no false alarms.
[60]	Comprehensive review of machine learning approaches for IoT network security.	Highlights challenges in applying machine learning to resource-constrained IoT systems.
[61], [62]	Addresses computational complexity in deep learning models for IDS.	Identifies the need for efficient IDS models considering computational overhead.
[63]	Investigates SVM-based intrusion detection system challenges in handling online detection.	Points out the trade-off between IDS accuracy and training time in hybrid models.
[64]	Proposes SAE-SVM hybrid model for IDS combining machine learning and deep learning benefits.	Utilizes deep learning for feature selection and machine learning for fast and accurate detection.
[65]	Develops anomaly detection for edge devices using attention-CNN and LSTM.	Implements top-k gradient selection criteria to reduce communication overhead.

IoT has emerged as a prominent technology that encompasses a wide range of internet-connected sensors/devices that are geographically dispersed and operate independently. These devices collect information from the physical environment and transmit it to other devices or servers for analysis and processing. This distributed nature enables many entry points to attacks and vulnerabilities within the system. Previously, AI-based models have been developed with the purpose of enhancing security in the IoT. Among these models, IDS-based techniques have been recognized as a critical tool for ensuring network security and safeguarding information systems. Nevertheless, given the rapid advancements in technology and the changes in market trends, there exists a need to update IDS to effectively counter new and emerging cyber threats [30]. To address this, the authors in [59] proposed an autonomous anomaly detection model that utilizes Gated Recurrent Unit (GRU) to identify compromised IoT devices. The proposed method involves training the GRU model at various security gateways, each containing specific device types, using a distributed learning approach. Under this approach, the GRU models are trained locally, and their weights are updated using an IoT security service for aggregation. The proposed model achieves a detection rate of 95.6% with no reported false alarms. Additionally, the communication overhead is managed by training models locally several times prior to sending updates. The current deep learning techniques used for IDS are often characterized by a high rate of false positives and a low rate of detection. In a study [60], the authors emphasized the significance of emerging attacks in IoT network security and provide a comprehensive review of various machine learning approaches. However, the application of machine learning algorithms to IoT systems poses significant challenges due to their resource constraints, which produce a large volume of data. The incorporation of machine or deep learning in IoT systems may result in computational complexity, as highlighted in the works by [61], [62]. Therefore, investigating the computational overhead is critical in designing effective IDS models. Incorporating machine learning with IoT systems leads to more computational complexity. Another review paper [63] stated that even though a lot of efforts are made in developing a Support Vector Machine (SVM)-based intrusion detection system, these schemes can not handle an online intrusion detection system that requires periodic retraining. With the improvement in IDS accuracy and reduction of false positives, SVM schemes require long training and testing time. Considering the shortcomings of IDS and their types, hybrid models can be developed which benefit from different intrusion detection methods. However, a trade-off in training time must be considered while establishing hybrid models.

A hybrid scheme SAE (Sparse Auto Encoder)-SVM is proposed in [64] which combines the benefits of machine learning and deep learning approaches. To improve processing time and cost, deep learning methods were used for feature selection. On the other hand, to get fast and accurate detection of intrusion in the network traffic, machine learning approaches in compatibility with big data processing engines, e.g., Apache Spark, have been used. Time, accuracy, detection, and cost are the utmost important metrics. The work in [65] developed anomaly detection for edge devices in a distributed manner. Authors used the attention mechanism-CNN (Convolutional Neural Networks) for feature selection and LSTM for the timely detection of anomalies in an accurate manner. Moreover, the top-k gradient selection criteria are proposed and implemented to reduce communication overhead to 50% with good model accuracy.

2.4 GNN Modelling in Cyber-space

Table 2.3: Summary of GNN-based NIDS in literature review

Ref	**Graph Technique Used**	**Impact in NIDS**
[66]	Modified GraphSage with edge aggregate function for NIDS using network flow graph.	Outperforms non-GNN approaches in four out of six datasets.
[67]	GNN with message-passing function for learning from host connection graphs.	Maintains baseline accuracy on CIC-IDS2017 dataset even with altered traffic flows.
[68]	Graph-based distributed anomaly detection scheme with graph structure built from flow-level information.	Enables monitoring of the entire Multi-Agent System (MAS) framework.
[59]	Combination of PSI graph and CNN classifier for IoT botnet detection.	Effective with accuracy up to 92% in detecting IoT botnets.
[69]	Investigates the vulnerability of PSI graph structures to adversarial attacks.	Calls for robust defensive models with effective algorithms.
[70]	GNN framework for P2P botnet detection using node and topological features.	Needs improvement in reducing false positive rate (FPR).

In all the above articles, conventional machine learning and deep learning techniques are used to solve different issues of IDS. These techniques lack a network topology. Understanding network topology benefits intrusion detection by providing valuable contextual information about the structure and relationships within the network. Network topology refers to the arrangement of devices and connections in a network, including information about nodes (such as routers, switches, and hosts) and the links between them. By incorporating network topology into intrusion detection, analysts can gain insights into the layout of the network, the flow of traffic between devices, and the normal behavior patterns exhibited by network components. GNN is a new and popular deep learning sub-field due to its ability to learn on the graph structure designed on the nature of the problem. Authors in [66] proposed a deep learning-based NIDS using a newly emerging graph neural network. A network flow graph is generated in which information is passed on the edges, the original GraphSage is modified by adding an edge aggregate function, and Softmax is applied to predict output for tuning the detection model in the backward propagation phase. Other model parameters include the ReLU activation function for non-linear transformation, cross-entropy loss, and ADAM optimizer with a learning rate of 0.001. The experimental results show that their GraphSage-based NIDS performs better than the non-GNN approaches in four out of six datasets. In [67], authors emphasized designing graph representation of network flow to display meaningful structural flow patterns for the development of robust and accurate NIDS. They represented network flows and their relationships within the network in the form of a graph structure. Then, a message-passing function was proposed to efficiently learn from host connection graphs. The model is evaluated on the CIC-IDS2017 dataset and compared against other machine learning classifiers by artificially changing flow features (packet size and inter-arrival times) relevant to the attack. Results show that the proposed GNN model maintains the baseline accuracy as compared to other models that show degraded performance on altered traffic flows. In [68], a graph-based distributed anomaly detection scheme was proposed to monitor the entire Multi-Agent System (MAS) framework and graph structure is built by reconstructing flow-level information into node-level. Another work in [59] presented a combination of the Population Stability Index (PSI) graph and CNN classifier for the detection of IoT botnets. Experimental results show effectiveness with accuracy up to 92%. Whereas, the study in [69] showed that PSI graph structures are prone to adversarial attacks and need a defensive model designed with robust and effective algorithms. In [70], authors proposed a framework for Peer-To-Peer (P2P) botnet detection utilizing node features and topological features. Experimental results show that FPR is high as 60% and needs improvement. Table 2.3 shows the summary of all the GNN based

NIDS in IoT networks.

Graph-structured data is beneficial for developing an effective NIDS because it allows for a more comprehensive representation of the relationships and dependencies within a network. GNNs have a benefit over traditional Deep Learning (DL) models when dealing with graph-structured data, nevertheless, the GNN algorithms can benefit from the graph topology as well as the features within. Even though GNN algorithms have shown promising results in the detection of attacks, there is a lack of research on the joint representation of edge and node features in network graphs with the aim to capture more network traffic relations. Existing research has used network features at either edge or node level. However, these features have structural relevance with the overlooked nodes or edges. This aspect of graph data representation has not been explored fully and there is still room for improving the detection accuracy. Moreover, the deployment of GNN-based NIDS in IoT networks has not been sufficiently addressed by the existing research.

2.4.1 Overview of GNN Methods

GNNs have emerged as a powerful class of ML models for analyzing graph-structured data, including network traffic data in IDS. In this section, we provide a general overview of different GNN methods and discuss their key features.

Graph Convolutional Networks (GCNs)

GCNs are one of the most widely used GNN architectures. GCNs operate by iteratively aggregating information from neighboring nodes in the graph to update node representations. This aggregation process is performed using graph convolution operations, which allow GCNs to capture structural information and learn representations that incorporate both local and global graph properties [71].

Graph Attention Networks (GATs)

GATs are another popular variant of GNNs that incorporate attention mechanisms to selectively attend to different parts of the graph during message passing. GATs assign attention weights to neighboring nodes based on their importance, allowing them to focus on relevant information while aggregating node features. This attention mechanism enables GATs to capture complex relationships and dependencies within the graph more effectively [72].

GraphSAGE

GraphSAGE is a GNN architecture that operates by sampling and aggregating information from neighboring nodes in the graph. Unlike traditional GCNs, which operate on the entire graph structure, GraphSAGE uses a sampling strategy to select a subset of neighboring nodes for aggregation, making it more scalable and efficient for large graphs. GraphSAGE also allows for the incorporation of different aggregation functions, such as mean or max pooling, to aggregate node features [73].

Graph Neural Networks with Recurrent Units (GRUs)

GRUs extend traditional recurrent neural networks (RNNs) to operate on graph-structured data. GRUs leverage recurrent units to capture temporal dependencies and dynamic changes in the graph topology over time. By incorporating recurrent units, GRUs can effectively model sequential data and capture long-range dependencies within the graph.

Overall, these GNN methods offer powerful tools for analyzing and modeling graph-structured data, including network traffic data in IDS. Each method has its unique features and advantages, making them suitable for different applications and scenarios. In the subsequent sections, we will review the application of GNN-based IDS and discuss how these methods are utilized to improve the detection and mitigation of security threats in network environments.

2.4.2 GNN - Use of Spectral and Spatial Convolutional Methods

GNN-based NIDS has taken a research boom due to the ability to with-hold network topology and its characteristics in the learning process. However, most of the proposed models rely either on spatial or spectral graph convolution methods. In contrast, joining both convolution methods in GNN frameworks can play a crucial role in learning the hidden structural representation of IoT network traffic.

To briefly state the difference, the spectral-based approaches apply filtering on eigen decomposition of the graph Laplacian matrix, enabling the Fourier transform on the graph structure. However, this generalization assumes a fixed graph and each node's features aggregate from its direct neighborhood [74], [75]. The spatial-based approaches use attribute aggregation and compute node representation directly by propagating information to the neighboring nodes. Node features are transformed and aggregated by permutation invariant functions, such as mean, sum, and max functions.

Table 2.4: Summary of Graph Neural Network (GNN) approaches for intrusion detection.

Reference	**Convolution Type**	**Results**
[76]	Spatial	Combination of IP and port number to represent IoT nodes and network traffic as edges, using two GraphSAGE layers. 1%, 3%, and 4% improvement in F1 score on different datasets.
[77]	Spatial	Extension of [76] using modified Deep Graph Infomax (DGI) and traditional machine learning algorithms in the Anomal-E model. Improvement in attack detection by combining machine learning approaches with GraphSAGE model.
[73]	Spatial	Extension of E-GraphSAGE with Residual learning and Graph Attention Network (GAT). Improvement of 0.2% and 0.11% over baseline models for detecting malicious traffic.

Continued on next page

Table 2.4 – continued from previous page

Ref	**Convolution Type**	**Results**
[67]	Spatial	Host-connection directed graph structure with GNN model based on Message Passing Neural Networks (MPNN) for NIDS. Improvements in attack detection compared to baseline ML approaches.
[78]	Spectral (GCN)	Knowledge graph with GCN model for automatic botnet detection and mitigation. Effectiveness seen on datasets with larger botnet communities.
[79]	Spectral (GCN)	ST-GCN based DoS attack mitigation and detection in SDN. Model improved accuracy up to nearly 10% compared to classical models.
[80]	Spectral (CGCN)	CGCN used for cyberattack detection in smart grids. Improvement of approximately 7% and 9% in detection rate and false alarm rate compared to other ML models.

Spatial convolution-based graph learning approaches [81] performed well in intrusion detection frameworks in the recent past. The convolution is simpler to implement and have produced state-of-the-art results on edge-level detection. For example, in [76], a combination of IP and port number was used to represent IoT nodes and network traffic to represent edges, whereas the graph model consisted of two GraphSAGE layers. Results were generated on four of the benchmark datasets, and F1 score was compared with state-of-the-art Machine Learning (ML) approaches. Results showed 1%, 3% and 4% improvement in performance on different datasets. In [77], authors extended the model in [76] to include modified Deep Graph Infomax (DGI) and traditional machine learning algorithms comprising of PCA-based anomaly detection (principal component analysis), IF (Isolation Forest), clustering-based local outlier factor (CBLOF), and histogram-based outlier score (HBOS) to design the Anomal-E model. The graph structure was made by turning IP-Port# combination into nodes and flows into edges. Results were produced on two datasets, and a comparison was presented, entailing the importance of combining machine learning approaches with the GNN model for attack detection. Results showed improvement when machine learning algorithms were used in addition to the GraphSAGE model. Another extension of E-GraphSAGE was proposed in [73] that explored more alternatives of GNN. Residual learning was combined with E-GraphSAGE and Graph Attention Network (GAT), and two separate models were proposed for detecting malicious traffic. IP address and port numbers were combined to identify nodes, and communication was identified as edges. Experiments were performed on four datasets, and the results of the modified E-GraphSAGE algorithm showed 0.2% improvement, and ResGAT showed 0.11% improvement over the baseline models. The work in [67] presented a host-connection directed graph structure to exhibit the attack patterns and employs the GNN model based on standard Message Passing Neural Networks (MPNN) to develop an NIDS. The experiment involved testing on one of the benchmark datasets of NIDS and showed improvements in detecting attacks as compared to baseline ML approaches in NIDS.

Graph Convolutional Neural Networks (GCN) [71] is the most popular spectral graph convolution method and has been widely used in the detection of attacks. The article [78] explored its potential for automatic botnet detection and mitigation. The Denial of Service (DoS) attack detection framework consisted of a knowledge graph with nodes representing botnets, a 12-layered GCN model, and node classification. The knowledge graph construction from the datasets has not been described in this paper, whereas results showed that this approach is more data-driven as effectiveness was seen on datasets with larger botnet communities only. A Spatial-Temporal Graph Convolutional Network (ST-GCN) based DoS attack mitigation, and detection in Software-Defined Networking (SDN) strategy was proposed in [79] that depicted SDN switch topology and captured spatial and temporal characteristics. SDN switches were turned into nodes with temporal features such as states of the switches, and communication between switches was turned into edges. The GCN model was used to perform node classification. Results showed that the model improved the accuracy up to nearly 10%, as compared to classical models. Another graph embedding approach in [80] was used for cyberattack detection in smart grids using CGCN (Chebyshev Graph Convolutional Network). A smart grid graph was constructed by turning bus voltages into nodes and transformers into edges in an undirected and weighted graph. The graph model consisted of three CGCN layers and one deep layer. Results were generated on a synthetic dataset and showed approximately 7% and 9% improvement in detection rate and false alarm rate, compared to other ML models.

While the methods reviewed in Table 2.4 showcase promising advancements in intrusion detection using Graph Neural Network (GNN) approaches, there are several downsides worth considering. Firstly, many of these methods rely heavily on the availability of labeled training data, which can be challenging to obtain for real-world network environments. Additionally, the effectiveness of these methods may be limited by the quality and representativeness of the training data, leading to potential biases or inaccuracies in the learned models. Moreover, some GNN-based IDS approaches may suffer from scalability issues when applied to large-scale networks, as the computational and memory requirements of GNN algorithms can increase significantly with the size of the network graph. Furthermore, the interpretability of GNN-based IDS models may be limited, making it difficult for security analysts to understand and interpret the decisions made by the model. Finally, the deployment and operationalization of GNN-based IDS approaches in practical network environments may pose logistical challenges, including integration with existing security infrastructure and compliance with regulatory requirements. Overall, while GNN-based IDS approaches offer promising capabilities for enhancing network security, addressing these downsides will be essential for their successful adoption and deployment in real-world scenarios.

2.4.3 GNN - Sequential Network Analysis

Table 2.5: Summary of GNN-based Approaches for Intrusion Detection and Botnet Detection

Ref	**Methodology**	**Purpose**
[70]	Peer-To-Peer (P2P) Botnet Detection - GNN with node and topological features	High False Positive Rate (FPR) indicates a need for improved performance.
[82]	Sequential Network Traffic Analysis - Pre-trained CNN and deep autoencoder	Used for feature extraction and capturing temporal changes.
[83]	Sequential Network Traffic Analysis - LSTM and FCN	Designed for multi-classification of malicious connections in intrusion datasets.
[84]	Sequential Network Traffic Analysis - LSTM-CNN-based method	Introduced for insider threat detection with fixed-sized feature matrices.
[85]	Sequential Network Traffic Analysis - Gated Recurrent Unit (GRU-BWFA)	Designed for effective detection of Distributed Denial of Service (DDoS) attacks.

Ref	**Methodology**	**Purpose**
[78]	Botnet Detection - End-to-end data-driven GNN approach	Utilized for detecting Peer-To-Peer (P2P) botnets with diverse communication patterns.
[86]	Botnet Detection - BD-GNNExplainer	Proposed for evaluating the trustworthiness of GNN-based botnet detection models.
[87]	Botnet Detection - GNN based model with GNNExplainer	Overcame over-smoothing issue and enhanced network forensics for botnet detection.

In recent past, various Graph Neural Network (GNN) architectures have been employed for network-related tasks, particularly in the context of Network Intrusion Detection (IDS) and anomaly detection. In [70], the authors propose a Peer-To-Peer (P2P) botnet detection framework utilizing node and topological features. However, the experimental results show a high False Positive Rate (FPR), indicating a need for improvement in the model's performance. Overall, these studies demonstrate the potential of GNN-based approaches in network Traffic analysis for intrusion detection. While they show promising results, challenges related to sequential attacks, multi classification need further exploration.

The existing literature consists of machine and deep learning approaches for the sequential network traffic analysis. In [82], a pre-trained CNN model is employed to extract features from processed data streams. An optimized deep autoencoder (DAE) is used to capture temporal changes in the surveillance stream's actions. They then trained their model using a quadratic SVM to classify human activities. In another study [83], Recurrent neural networks (RNN) is used to learn from the previous time-steps in the datasets. A deep learning model based on LSTM (Long Short-Term Memory) and FCN (Fully Connected Network) is designed for a multi-classification of malicious connections in intrusion datasets.

In [88], Gray Wolves Optimization (GWO) algorithm in conjunction with a wrapper feature selection technique is used to optimize the binary feature space. They introduced time-variant transfer function that adapts GWO for enhanced botnet detection accuracy in IoT environments. A novel LSTM-CNN-based method is introduced in [84] for insider threat detection. The method involves initial extraction of temporal behavioral features by feeding single-day user action sequences into the LSTM model. Subsequently, the extracted features are transformed into fixed-sized feature matrices, which are then used in a CNN-based classification model for the final detection of insider threats. In [85], a Gated Recurrent Unit based on Bidirectional Weighted Feature Averaging (GRU-BWFA) classifier is designed for effectively detecting Distributed Denial of Service (DDoS) attacks and capturing the time series events.

Recent papers have explored the integration of GNNs in botnet detection tasks due to their ability to capture botnet topology. In this study [78], an end-to-end data-driven approach is utilizing GNNs to detect Peer-To-Peer (P2P) botnets. To comprehensively evaluate the automated detection method, synthetic or real botnet topologies are overlaid with diverse communication patterns on large-scale real background traffic graphs, generating datasets for analysis. In [86], a method BD-GNNExplainer (GNN-based botnet detection) is proposed to evaluate the trustworthiness of GNN-based botnet detection models. BD-GNNExplainer extracts the most contributing data to the GNN's decision, quantifying a score that expresses interpretability, ultimately guiding model optimization and providing a guideline to enhance the understandability of the botnet detection methodology. In another paper [87], a GNN based botnet detection model is proposed to overcome over-smoothing issue and enhance network forensics. Furthermore, GNNExplainer and saliency maps are integrated to identify suspicious network flows and botnet nodes, enhancing the transparency and interpretability of the detection process for automatic network forensics – features that are lacking in existing botnet detection literature.

The related work in botnet detection using GNNs has demonstrated notable advancements in capturing graph patterns and improving detection accuracy. However, a significant gap in the existing literature lies in the lack of exploration regarding the utilization of Gated Graph Convolution neural network for detecting botnets in network traffic. It is a powerful variant of GNN that excels in capturing the sequential nature of botnet activities, which is essential for detecting dynamic and evolving botnet behaviors in real-time network traffic. This specific gap in the research indicates an unexplored avenue to leverage the sequential information inherent in botnet activities through Gated GNN based models. By modelling a Gated GNN based framework for the detection of botnet attacks leads to a more robust and efficient methods for combating evolving botnet threats in the dynamic landscape of IoT networks. The methods reviewed in this section for intrusion detection and botnet detection using GNN approaches exhibit several potential downsides. Firstly, some of these methods may suffer from a high false positive rate (FPR), as observed in the Peer-To-Peer (P2P) botnet detection approach [70]. This indicates a need for improved performance to reduce false alarms and enhance the accuracy of detection. Secondly, scalability may be a concern for some methods, particularly those that rely on computationally intensive GNN architectures or large-scale network datasets. As the size and complexity of network environments increase, scalability becomes increasingly important to ensure efficient and effective detection of security threats.

2.5 Advancing Intrusion Detection through GNNs

Traditionally, NIDSs using ML lack topological information critical for attack determination. Recent studies applying GNNs in NIDSs offer basic graph structures capturing partial node communication. In contrast, our proposed GNN-based NIDS improves upon existing approaches by processing multi-edges with multi-dimensional features in the graph structure. Our Multigraph GNN model combines spectral and spatial convolution methods to address multiple edges and features, enhancing network representation.

Moreover, we present a comprehensive GNN-based NIDS model that is capable of capturing relations in the network graph and combines both the node and edge features to pick out abnormal behavior in the traffic. Our approach offers to minimize multiple attack vectors by including node as well as edge information to capture the complex relationship and nuances of the IoT network traffic. We use IP address and Port number combination to represent the IoT sessions as nodes, and network flows to represent the exchange of communication as edges. The node features represent the application-level information specific to packet content, whereas, edge features represent the network-level information. In this way, complete information is defined on the graph structure. As a result, our proposed NE-GConv model combats and mitigates multiple attack vectors and is sensitive to changes in the application and network layer as compared to the existing research. Moreover, we also investigate the computational requirement of our NIDS and its performance in the resource constraint environment. To keep the balance between the model performance and the computational needs, we keep the model design simple containing the least number of hidden units and selected features. In this way, our research develops an effective and novel method by combining node and edge features while suppressing the design complexity for the detection of anomalous behavior in IoT networks.

2.6 Chapter Conclusion

This chapter reviewed literature related to this thesis and laid a solid foundation for the following chapters. To be specific, this chapter summarized and compared the latest research progress about DL models, i.e., attack defense models in network traffic, their overall performance while focusing computational overhead and false alarm rates, GNN modelling in cyberspace, exploration of graph structures implied and type of convolutional methods used in the graph models. This chapter also identified the gap between research targets and existing researches and pointed out key research points of this thesis.

Chapter 3

NE-GConv: A Lightweight Node Edge Graph Convolutional Network for Intrusion Detection

3.1 Introduction

With the hype in Artificial Intelligence (AI), a lot of research analysis has come forth in recent years aiming to improve the detection performance of the machine and deep learning models in new attack types [3], [89]. However, fewer studies have focused on the deployment challenges of IDS in IoT systems. Deep learning models, in general, are complex in nature and require significant resources in terms of computational cost and memory. IoT systems offer a limit to these characteristics. Until recently and to the best of our knowledge, no clear model has been laid out to determine the optimal threshold and trade-off of accuracy and implementation needs of IDS for resource constraint devices.

In this chapter, we propose a novel GNN-based NIDS framework that offers a larger attack surface management and a lightweight model. In view of large attack surface management, we incorporate a graph structure that is equipped with both the node and edge features, thereby, making the graph model sensitive to the changes at both levels. On the other hand, we focus on keeping the overall framework lightweight by reducing dimensionality in the feature selection unit and designing a shallow two-layer graph convolutional model, thereby, reducing the overall computational complexity and paving way for the deployment in resource-constraint networks.

Our framework is capable of improving the detection of attacks by exploiting the structural characteristics of attacks with high accuracy and reduced complexity. The key contributions of this research are as follows:

- We present a new graph structure to capture network traffic relations in a comprehensive way, where nodes are referred as hosts and edges as the communication between nodes. The network data features have a structural association with either the nodes or the edges. We refer to network packet content/application level features as node features and network flows as edge features, respectively. This captures the traffic characteristics of hosts and flow connections within the network graph structure.
- We propose a two-layer Node Edge-Graph Convolutional (NE-GConv) model which is sensitive to the intrusions in the node and edge features. A new aggregation function is defined to learn, aggregate, and generalize from node and edge features (payload and flows) in the message-passing phase of NE-GConv.
- We develop a lightweight IDS framework for meeting the deployment needs of resource-constraint devices. The IDS framework employs the Recursive Feature Elimination (RFE) algorithm to select the minimum set of input features for the NE-GConv and implements an efficient two-layer architecture in the NE-GConv.

In the end, we give a detailed analysis of our proposed model by comparing it with several other state-of-the-art GNN models on the datasets evaluating accuracy, precision, recall, F1, AUROC, False Positive Rate (FPR), model size, and running time. Until recently and to the best of our knowledge, this is the first work that investigates the computing resources required for GNN-based NIDS by testing the implementation in the Raspberry Pi device. The experimental results demonstrate the effectiveness of the proposed approach.

The research questions answered in this chapter are as follows.

- How does incorporating both node and edge features in a graph structure improve the performance of IDS in IoT systems?
- What impact does reducing dimensionality and employing a shallow two-layer graph convolutional model have on the computational complexity and deployment feasibility of IDS in resource-constrained networks?
- How does the proposed NE-GConv model compare with existing GNN models in terms

of detection accuracy, precision, recall, F1-score, AUROC, FPR, model size, and running time?

The rest of the chapter is organized as follows. Section 5.3 illustrates the design of our proposed framework and implementation of the layered architecture of the proposed model. Section 4.3 presents experimental results and a comparative analysis of the model.

3.2 The Proposed IDS Framework.

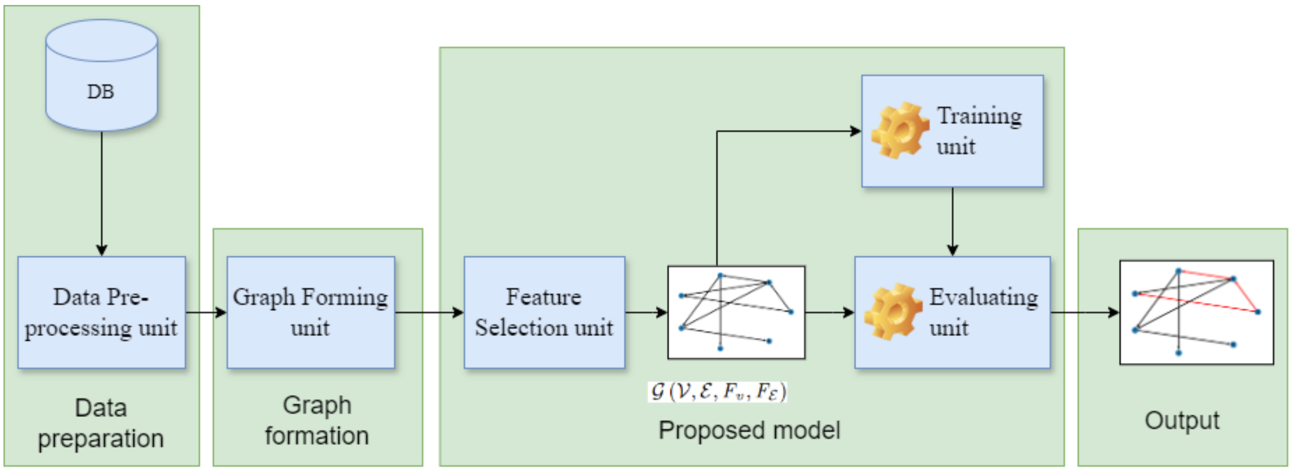


Figure 3.1: An overview of the proposed IDS framework that consists of three main components, i.e., data preparation, graph formation, and the proposed model.

In this section, we present the design of our graph-powered IDS that consists of three main components: Data preparation, Graph formation, and Proposed model as given in Fig. 3.1. In addition, we introduce the layered architecture of the proposed NE-GConv model in detail. Table. 5.1 summarizes the notations used in the rest of the chapter.

3.2.1 Overview of the Graph-powered IDS

Graph-powered IDS combines NIDS and DPI (Deep Packet Inspection) on the NE-GConv algorithm. It monitors the host- and flow- behavior and classifies the network flows into normal or malicious. Our framework consists of the following components:

1. Data pre-processing unit: It is responsible for pre-processing the raw dataset before it is sent to the graph-forming unit. This unit performs label encoding and standardization of categorical and numerical features to better adapt to training. It is further divided

Notation	**Definition**
$\mathcal{A}_\epsilon$	Featured adjacency vector of edge
$\mathcal{E}$	Set of edges
$\lvert\mathcal{E}\rvert$	Number of edges
ϵ_{ij}	The edge between nodes v_i and v_j
F_v	The node feature matrix
$F_\mathcal{E}$	The edge feature matrix
f_{v_i}	Feature vector of node v_i
$f_{\epsilon_{ij}}$	Feature vector of the edge ϵ_{ij}
$\mathcal{G}$	The graph
$h(\cdot)$	Input to output transformation function
K	Number of classes
$\mathcal{L}$	Cross-entropy loss
$N_\mathcal{G}(v_i)$	Neighborhood of node v_i
$\lvert\mathcal{V}\rvert$	Number of nodes
$\mathcal{V}$	Set of nodes
W^{GCN}	The hidden weight matrix
$\mathcal{X}$	Model input set of data
$\mathcal{X}^{F_\mathcal{E}}$	The edge feature matrix in train/test set
$\mathcal{X}^{F_v}$	The node feature matrix in the train/test set
$\mathcal{Y}$	The ground truth label
$\tilde{\mathcal{Y}}$	The predicted output

Table 3.1: Notations used in the chapter and their definitions.

into three steps. Step 1: Data cleaning: Data is cleaned by removing nulls, nans, and replacing missing values. Step 2: Categorical feature mapping: The categorical features such as protocol, service, and state are encoded using label encoding. Step 3: Numerical feature mapping: All features are then passed through standard scalar for normalization. This preserves the relative importance of each feature, and the algorithm is less likely to overfit and be biased towards features with significant values.

2. Graph forming unit: This unit is responsible for the formation of a graph structure $\mathcal{G}$ that can carry node features F_v and flow features $F_{\mathcal{E}}$. This enables the integration of host and network IDS in the model design.

3. Feature selection unit: This unit performs the feature selection task by filtering out the best features from the raw dataset using the RFE technique [90].

4. Training unit: This unit trains the proposed model based on the graph neural network into identifying anomalies. Considering that there are m number of observations $(x^1, y^1), \cdots, (x^m, y^m)$ for training a network intrusion detection model. The input set is referred as $\mathcal{X}$ that includes network node features $\mathcal{X}^{F_v}$ and the network edge features $\mathcal{X}^{F_{\mathcal{E}}}$. The corresponding labels to input train set $\mathcal{X}^{F_{\mathcal{E}}}$ for edge classification are: $\mathcal{Y} \in \mathbb{R}^{m,1}$. Consider an input sample $x^i \in \mathbb{R}^b$ from train set $\mathcal{X}^{F_{\mathcal{E}}}$ where i is i-th input observation and b is the dimension of edge features, then the label is termed as $y^i \in \mathbb{R}^1$. For K number of classes, the model is optimized by reducing the loss function $\mathcal{L}$.

$$\mathcal{L} = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\mathcal{Y}_{i,k} \log \tilde{\mathcal{Y}}_{i,k}, \tag{3.1}$$

   where, m is the total number of observations, $\mathcal{Y}_{i,k}$ is 1 if observation i is in class k and 0 otherwise, $\tilde{\mathcal{Y}}_{i,k}$ is the predicted probability that observation i is in class k and $\mathcal{L}$ is cross entropy loss. We use the supervised learning method to train the model and label each input as normal or malicious. $\mathcal{Y}_k$ and $\tilde{\mathcal{Y}}_k$ are closely monitored for optimization during the training phase and loss is reduced.

5. Evaluating unit: This unit is responsible for testing and evaluation of the proposed trained model. For n test examples, model predictions $\tilde{\mathcal{Y}}$ are given by

$$\begin{aligned} \tilde{\mathcal{Y}} &= h(\mathcal{G}), \\ \tilde{\mathcal{Y}} &= h(\mathcal{X}^{F_v}, \mathcal{A}_\epsilon), \end{aligned} \tag{3.2}$$

   where $h(\cdot)$ is the model transformation function that maps the input $\mathcal{X}^{F_v}$ and $\mathcal{A}_\epsilon$ to $\tilde{\mathcal{Y}}$, $\mathcal{X}^{F_v}$ is node feature matrix and $\mathcal{A}_\epsilon$ is featured adjacency matrix in the test set.

3.2.2 Graph Formation

We propose a hybrid model that integrates host features and flow features into the IDS framework by defining a graph structure capable of handling both node and edge characteristics.

Network dataset is mapped into a directed graph $\mathcal{G}$ consisting of the set of nodes $\mathcal{V}$, the set of edges $\mathcal{E}$, node features F_v and edge features $F_{\mathcal{E}}$. Graph nodes $\mathcal{V}$ are the communication endpoints/hosts referring to the application layer processes, and node features F_v are the host-specific observations, e.g., payload fields. Edges $\mathcal{E}$ are the links between two endpoints directed from the source to the target node. The edge features $F_{\mathcal{E}}$ are selected from flow fields, e.g., duration, packet count, and arrival time. The graph structure $\mathcal{G}$ is crucial to the analysis of network and host intrusions. The closer the graph structure is to the real network, the better will be the model's ability to identify intrusions. Since an attack may take place at the host and network level, it is critical to include both edge and node features for the identification of anomalies.

Given a network traffic dataset, a graph can be described as $\mathcal{G}\left(\mathcal{V}, \mathcal{E}, F_v, F_{\mathcal{E}}\right)$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges in the graph $\mathcal{G}$. Let v_i denote the i-th node in the graph. F_v is the $|\mathcal{V}| \times a$-dimensional node feature matrix; $F_v \in \mathbb{R}^{|\mathcal{V}|,a}$, where $|\cdot|$ denotes cardinality and a is the number of features of a node. Then, the i-th row of F_v is denoted by f_{v_i} which is an a-dimensional feature vector of node v_i; $f_{v_i} = [f_1^{v_i}, \cdots, f_a^{v_i}]$.

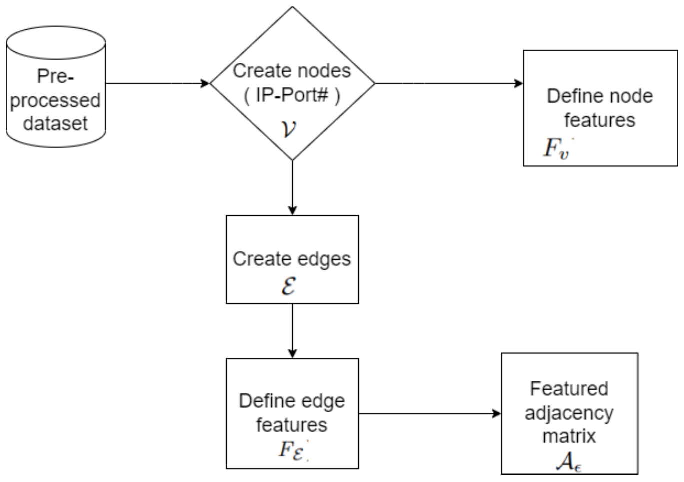

Pre-processed dataset
Create nodes (IP-Port#) $\mathcal{V}$
Define node features F_v
Create edges $\mathcal{E}$
Define edge features $F_{\mathcal{E}}$
Featured adjacency matrix $\mathcal{A}_\epsilon$

Figure 3.2: Step by step graph structure formation.

The edge from node v_i to node v_j is denoted by ϵ_{ij} such that $\epsilon_{ij} \in \mathcal{E}$. Each edge has a b-dimensional vector that defines the nature of the link. The feature vector of edge ϵ_{ij} is denoted by $f_{\epsilon_{ij}} = [f_1^{\epsilon_{ij}}, \cdots, f_b^{\epsilon_{ij}}]$. $F_{\mathcal{E}} \in \mathbb{R}^{|\mathcal{E}|,\mathrm{b}}$ is the edge feature matrix where $|\mathcal{E}|$ is the number of edges in the graph and b is the number of features of an edge. The featured adjacency matrix of graph $\mathcal{G}$ is a $|\mathcal{V}| \times |\mathcal{V}|$-dimensional matrix termed as $\mathcal{A}_\epsilon$. $\mathcal{A}_{\epsilon_{ij}} = \mathbf{0}$, if there is no edge between v_i and v_j; $\epsilon_{ij} \notin \mathcal{E}$ and $\mathbf{0}$ is a b-dimensional vector of all zeros whereas $\mathcal{A}_{\epsilon_{ij}} = f_{\epsilon_{ij}}$, if $\epsilon_{ij} \in \mathcal{E}$. For any node v_i, its neighborhood can be denoted by $N_{\mathcal{G}}(v_i) = \{v_j | v_j \in \mathcal{V} \text{ and } \epsilon_{ij} \in \mathcal{E}\}$. The edge and node features are extracted from the dataset and $\mathcal{A}_\epsilon$ is defined in the graph formation step as given in Fig. 3.2. In the graph forming unit, graph nodes are built by pairing IP addresses and Port numbers to represent hosts. Edges are the connections from source to target nodes. Next, graph adjacency matrix $\mathcal{A}_\epsilon$ is built based on the link of nodes with each other. $\mathcal{A}_\epsilon$ is a square matrix in which rows and columns refer to the nodes in the graph and the matrix values represent the nature of connection in form of edge feature, e.g.,$f_{\epsilon_{ij}} \forall \epsilon_{ij} \in \mathcal{E}$. The first column of the adjacency matrix is the source node and the second is the target node to which an edge is present, e.g., each row contains **(Source-node, Target-node, [Edge-features])**. We define F_v as payload-related fields and $F_{\mathcal{E}}$ as flow-related fields from the dataset.

3.2.3 The NE-GConv Model

The proposed model (NE-GConv) is an improvement of the GConv model [71] that takes the host and network features in a two-layer GConv model and classifies edges into normal and malicious. The graph knowledge is built in the graph formation step and is sent into the NE-GConv model. The layered architecture of the proposed NE-GConv model is given in Fig. 3.3.

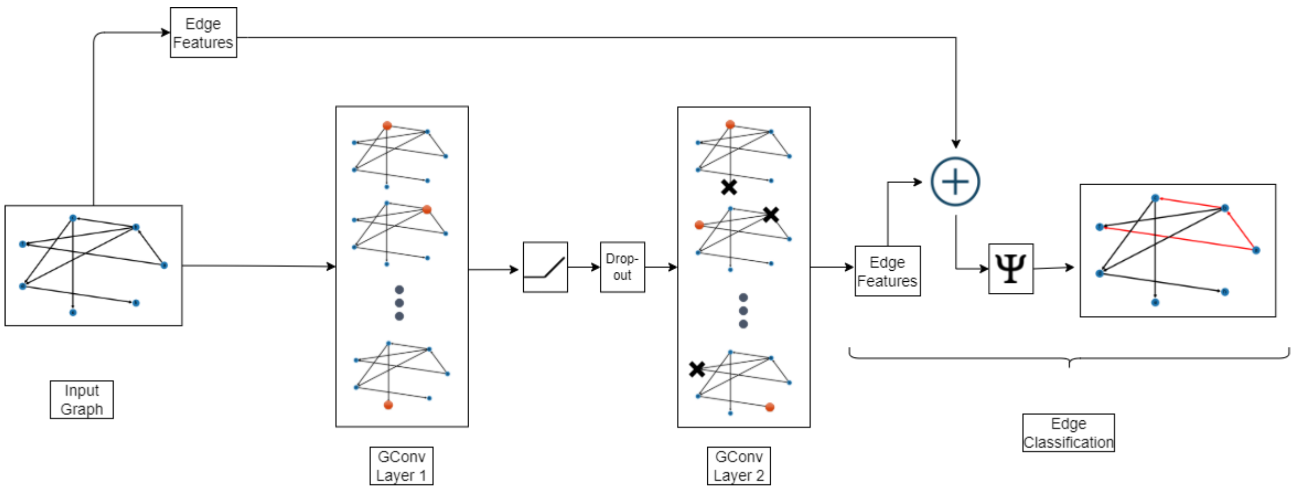


Figure 3.3: The layered architecture of the model

The layered architecture takes the form:

$$h(F_v, \mathcal{A}_\epsilon) = (\mathcal{A}_\epsilon \; ReLU(\mathcal{D}(\mathcal{A}_\epsilon \; F_v W^{GCN(1)}))W^{GCN(2)}), \tag{3.3}$$

where $W^{GCN(1)}$ and $W^{GCN(2)}$ are the weight matrices of hidden layers 1 and 2. $\mathcal{D}$ is the dropout function [91] and $ReLU$ is the activation function [92] in the model. $W^{GCN(1)} \in \mathbb{R}^{a,H}$ where a is the dimension of node features and H is the dimension of the first hidden layer. Similarly, $W^{GCN(2)} \in \mathbb{R}^{H,O}$ where O is the dimension of the second hidden layer which is also the output layer in the proposed model. (3.3) summarises a two-layer NE-GConv model where F_v and $\mathcal{A}_\epsilon$ are fed as model inputs. Consider $x = \mathcal{A}_\epsilon \, F_v W^{GCN(1)}$, then the dropout function $\mathcal{D}$ can be expressed as $d \circ x$ such that $d \in \{0,1\}^{a,H}$, and $\circ$ is element-wise product. $d \circ x$ will drop x with a probability p in such a way that its dimensions are retained. The $ReLU$ function on any matrix x can be defined as below:

$$\begin{aligned} ReLU(x) &= \max(0, x) \\ \text{i.e.,} & \\ ReLU(x_{ij}) &= \max(0, x_{ij}) = \begin{cases} x_{ij}, & \text{if } x_{ij} > 0 \\ 0, & \text{if } x_{ij} < 0. \end{cases} \end{aligned} \tag{3.4}$$

The input to NE-GConv, as shown in Algo. 1, consists of a graph $\mathcal{G}$ with $|\mathcal{V}|$ number of nodes, $|\mathcal{E}|$ number of edges, the feature vectors $F_v, F_{\mathcal{E}}$. The feature vectors are passed to initialize embeddings as given in step 1 and 2 of Algo. 1. Essentially, a graph neural network operates on the message-propagation/passing mechanism. Typically, the message-passing function only considers node features for aggregation. It is essential to modify the standard GConv model to include edge features to achieve a co-effect of node and edge features in the model. The improvement in the message-passing function is shown in step 4. F_{v_i} is the node feature matrix of central node v_i, F_{v_j} is the neighboring node feature matrix and $F_{\epsilon_{ij}}$ is edge features between node v_i and v_j, then, the aggregation function AGG is given as:

$$AGG(F_{v_i}, F_{v_j}, F_{\epsilon_{ij}}) = \sum_{v_j \in N_{\mathcal{G}}(v_i) \bigcup (v_i)} F_{\epsilon_{ij}} \circ F_{v_j}, \tag{3.5}$$

where, $\circ$ is element-wise multiplication. A key difference is treating self-node features, neighboring node features and the corresponding edge features together for getting a co-effect of all the features in the detection of attacks. With ${h_{v_i}}^0$ denoting node features of node v_i, ${h_{v_j}}^0$ denoting neighboring node features and ${z_{ij}}^0$ denoting edge features from node v_i to node v_j, aggregated embeddings created are described in step 4. In step 5, node embeddings $\tilde{h^1_{v_i}}$ of node v_i at layer 1 is calculated by applying the model's trainable parameters W^1 and b^1 on the aggregated message. The drop-out technique is applied to regularise and make the overall model

Algorithm 1: NE-GConv Algorithm

```
Input:
    Graph $\mathcal{G}\,(\mathcal{V}, \mathcal{E}, F_v, F_{\mathcal{E}})$
    Node features $F_v$
    Edge features $F_{\mathcal{E}}$
Output:
    Edge embedding $z_{ij}\ \forall \epsilon_{ij} \in \mathcal{E}$,

▷Initialization
1  $h^0_{v_i} \leftarrow F_{v_i} : \forall v_j \in N_{\mathcal{G}}(v_i)$,
2  $z^0_{ij} \leftarrow F_{\epsilon_{ij}} : \forall \epsilon_{ij} \in \mathcal{E}$,
▷Node Embedding in Layer 1
3  for $\forall v_i \in \mathcal{V}$ do
4      $\tilde{h^1_{v_i}} \leftarrow AGG\big(h^0_{v_i}, h^0_{v_j}, z^0_{ij} : \forall v_j \in N_{\mathcal{G}}(v_i)\big)$,
5      $\tilde{h^1_{v_i}} \leftarrow \mathcal{D}(W^1 \times \tilde{h^1_{v_i}} + b^1)$,
6      $h^1_{v_i} \leftarrow ReLU(\tilde{h^1_{v_i}})$,
7      $z^1_i \leftarrow h^1_{v_i}$,
▷Node Embedding in Layer 2
8  for $\forall v_i \in \mathcal{V}$ do
9      $\tilde{h^2_{v_i}} \leftarrow AGG\big(h^1_{v_i}, h^1_{v_j}, z^1_{ij} : \forall v_j \in N_{\mathcal{G}}(v_i)\big)$,
10     $h^2_{v_i} \leftarrow ReLU(W^2 \times \tilde{h^2_{v_i}} + b^2)$,
11     $z^2_i \leftarrow h^2_{v_i}$,
▷Edge Embedding
12 for $\epsilon_{ij} \in \mathcal{E}$ do
13     $z_{ij} = \psi\left(concat(z^1_i, z^1_j) + concat(z^2_i, z^2_j)\right)$
return $z_{ij}$
```

less likely to overfit training data. The node embeddings are updated by passing through the *ReLU* function, as shown in step 6 and are saved in a separate variable in step 7. We repeat the same steps for layer 2 as given in 8, 9, 10, and 11, except for the use of the drop-out function.

To extract edge features from node embedding, we perform column-wise concatenation of z^1_i and z^2_i. To optimize the model and deal with over-smoothing issue [93] in graph operations, initial node embedding z^1_i is added in the final embedding z^2_i as given in steps 12 and 13. Our aim is to keep computational complexity low. Therefore, the final output of the algorithm is in the shape of edge classification rather a packet-level classification. Edge classification is performed by passing the sum of initial and final embedding through ψ layer as given in (3.6) and steps

12 and 13 of the algorithm. We represent w_ψ as weight, $concat(z_i^1, z_j^1) + concat({z_i}^2, {z_j}^2)$ as the input and b_ψ as the bias, then the linear feed forward layer ψ becomes

$$\begin{gathered}\psi(concat(z_i^1, z_j^1) + concat({z_i}^2, {z_j}^2)) = \\ w_\psi \times (concat(z_i^1, z_j^1) + concat({z_i}^2, {z_j}^2)) + b_\psi,\end{gathered} \tag{3.6}$$

Key differences to the original GConv [91] algorithm are in regards to the algorithm input, message aggregation, update functions, and output. Our algorithm offers the ability to include flow features and packet content-related features for deep inspection. The algorithm has 2 hidden layers. At each layer, node embeddings are updated by aggregating features. Finally, the edge information is predicted by applying $\psi-$ transformation on each updated edge. For supervised binary classification, we then evaluate the cross-entropy error over all the labeled examples.

3.3 Experimental Results

3.3.1 Dataset

The UNSW-NB15 [94] is an IDS dataset generated in the Cyber Range Lab of the Australian Centre for Cyber Security in 2015 that overcomes the shortcomings in the IDS benchmark datasets by including emerging attacks and IoT traffic. The dataset has modern attacks and new patterns of normal traffic and is quite suitable for NIDS evaluation settings [95]. The total number of records in the dataset are given in Table 3.2 and are stored in four CSV files, namely, UNSW-NB15_1.csv, UNSW-NB15_2.csv, UNSW-NB15_3.csv, and UNSW-NB15_4.csv. It contains 49 attributes, including a label category to identify traffic as normal and malicious. The dataset includes a variety of packet-based features and flow-based features. The attributes are further categorized into five main categories: Flow features (1-5), Basic features (6-18), Content features (19-26), Time features (27-35), Additional features (36-47), and labeled features (48-49). In this experiment, we use flow features (1-5) to generate a graph structure, Basic features (6-18) to represent edge features, and Content features (19-26) to represent node features. This, in turn, increases the ability of the proposed model to manage attack detection from several attack vectors at once. The packet-based features assist in the examination of the payload, whereas flow-based features assist in the examination of packet flow and make the classification analysis computationally less expensive.

Dataset	Total Data	Normal Data	Malicious Data
UNSW-NB15 [94]	2539431	2218211	321220

Table 3.2: Details of the UNSW-NB15 dataset in terms of the dataset size, the number of malicious and normal data entries.

3.3.2 The Experimental Setup

To test and verify the effectiveness of our experiment on the joint use of node and edge features in NIDS we use supervised classification. In all of these experiments, nodes correspond to application layer processes/hosts and edges to (directed) network flow. Node features are the elements of packet content that contribute to understanding the malicious possibilities associated with payload, and edge features correspond to the elements of network connection attributes. All the experiments and data acquisition steps are performed on Google Colab Pro with PyTorch 1.13.1 and Intel(R) Xeon(R) CPU @ 2.20GHz. Python libraries, e.g., Pandas, cikit-learn, imblearn, NetworkX, and PyTorch-Geometric (PyG), are used. For the implementation of different models i.e., GConv [71], GraphSAGE [96], GatedGraphConv [97] and EGConv [98], PyG's in-built functions are used.

Datasets	Train dataset	Test dataset	Training Nodes	Training Edges	Testing Nodes	Testing Edges	Ratio
UNSW-NB15 [94]	1777601	761830	519049	601787	519425	602224	7:3

Table 3.3: Details of training and testing datasets for a 70:30 ratio.

3.3.3 Experiment Steps and Metrics

Several typical experiments are conducted to evaluate the performance and complexity of GNN in the development of NIDS using the UNSW-NB15 dataset. We perform the following test steps:

- Perform feature selection using RFE on the datasets and find out the best possible minimum set of features aiming to capture maximum relevance from the dataset in feature selection.
- Form an IP: Port number paired graph structure for network traffic scenario, design a two-layered NE-GConv model as explained in Section 3.3.

- Pass in the adjacency matrix, node and edge features to NE-GConv model, and train and evaluate on the test dataset.
- Design node classifiers from state-of-the-art GNN models e.g., GraphSAGE, EGConv and GatedGraphConv keeping the same graph structure, layer architecture, and hyper-parameters, train and evaluate as is done for NE-GConv.

Subsequently, we analyze the comparison of the other GNN models with the proposed NE-GConv model. The following metrics are considered for the performance evaluation of the proposed approach:

1. **Accuracy (ACC):** This is a measure of correctly classified observations out of all data observations and is given by $\frac{T_p+T_n}{T_p+F_p+T_n+F_n}$. Here, T_p is true positive, T_n is true negative, F_p is false positive, and F_n is false negative.
2. **Precision:** This is a measure of how precise a model is in correctly predicting positives (T_p) out of all the predicted positives ($T_p + F_p$), and is given by $\frac{T_p}{T_p+F_p}$.
3. **F1-Score:** This is a function of precision and recall and is used where there is imbalanced data. It is given by $(2 \times \frac{Precision \times Recall}{Precision+Recall})$.
4. **Recall:** This is a measure of how many true positives a model predicts out of all the actual positives, and is given by $\frac{T_p}{T_p+F_n}$.
5. **False Alarm Rate (FAR):** This is a measure of how many normal data points are classified as malicious and is given by $\frac{F_p}{T_n+F_p}$.
6. **Area Under the Receiver Operating Characteristics (AUROC):** This is a measure of the rank correlation between predictions and actual labels **roc**. The higher the score, the better the model's ability to classify positive and negative class points. It summarizes the precision-recall curve.
7. **Training Time:** The training time is a measure of how quickly the model is capable to distinguish classes in time. The lesser the training time, the faster the model learns.
8. **Testing Time:** The lesser the testing time, the faster the model picks out the anomalies.
9. **Memory:** This is the measure of memory consumption of a model in bytes.
10. **FPR:** This is a measure of false alarms/positives given by the model.

3.3.4 Results

The experiments are carried out by identifying the best set of edge features. The dataset has 8 node features and 13 edge features. The node features consist of payload related fields such as: **Source TCP window advertisement (swin):** This feature indicates the window size advertised by the source TCP during the communication session. While not directly related to payload content, the window size can impact the flow and transmission of payload data by influencing the rate at which data is sent and acknowledged. **Destination TCP window advertisement (dwin):** Similar to swin, this feature represents the window size advertised by the destination TCP. It affects the flow and transmission of payload data by determining the amount of data that can be sent without acknowledgment from the destination. **Source TCP sequence number (stcpb):** This feature denotes the sequence number assigned to the packets transmitted by the source TCP. While not directly encoding payload content, sequence numbers are crucial for reconstructing the order of payload data at the receiver end. **Destination TCP sequence number (dtcpb):** Analogous to stcpb, dtcpb represents the sequence number assigned to packets by the destination TCP. It assists in reconstructing the payload data and ensuring its integrity during transmission. **Mean of the flow packet size transmitted by the source (smeansz):** This feature calculates the average packet size transmitted by the source node within the flow. While not directly encoding payload content, variations in packet size can provide insights into the characteristics and nature of the payload data being transmitted. **Mean of the flow packet size transmitted by the destination (dmeansz):** Similar to smeansz, dmeansz calculates the average packet size transmitted by the destination node within the flow. It helps in understanding the characteristics of the payload data received by the destination node. **Depth into the connection of HTTP request/response transaction (trans_depth):** This feature measures the depth or level of interaction within an HTTP request/response transaction. While not directly encoding payload content, the transaction depth can indicate the complexity and nature of the payload data being exchanged.**Content size of the data transferred from the server's HTTP service (res_bdy_len):** This feature quantifies the size of the content transferred from the server's HTTP service in the payload. It directly relates to the payload content and provides insights into the volume and nature of the data being exchanged between the client and server. These features collectively provide valuable information about the characteristics, behavior, and content of the payload data exchanged between network nodes during communication sessions. While some features directly encode payload content, others offer insights into the flow, structure, and context of the payload data, contributing to the overall understanding and analysis of network traffic.

To match the dimensions of node features and for the best results, the 8 best features are selected from the edge features. The experiments are run to analyze the performance metrics of the proposed model on the selected features. The best features according to RFE have a ranking of one as shown in Fig. 3.4.

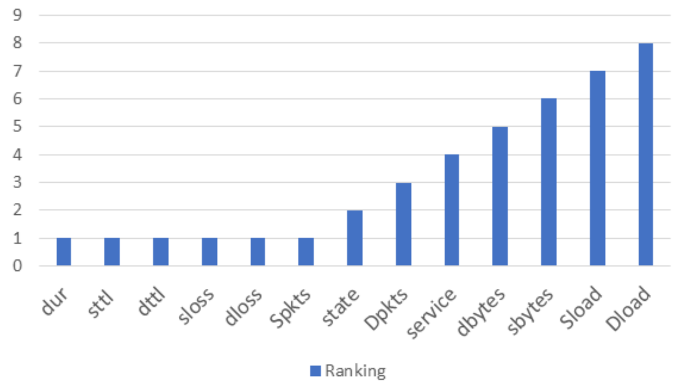


Figure 3.4: RFE Feature Ranking.

A Comparison Study with Varying Train:Test Ratios

All graph models are tested against varying train:test ratios to investigate the stability and generalizability of the proposed model. The GNN models are tested as node classifiers and are compared with the proposed NE-GConv model. From the results in Table 3.4, it is clear that all graph models perform well under varying train:test ratios. The best results are achieved at the 70:30 train:test ratio across most of the metrics, at which, FPR reduces to 2.36% and accuracy reaches 97.64%. The proposed NE-GConv improves performance up to $2\% - 4\%$ than GraphSAGE in accuracy, precision, recall and F1 scores, whereas, FPR is reduced to almost half at varying train:test ratios. The high performance of NE-GConv as given in Table 3.4 on the test data in all the metrics confirms that our model is making accurate predictions while suppressing false positives and false negatives. High F1 and AUROC scores across varying ratios confirm the model's ability to classify positives from negatives well.

We conducted an in-depth analysis of the computational complexity of various graph models, focusing on training time, testing time, and model size, as detailed in Table 3.5. The results demonstrate that NE-GConv exhibits superior performance metrics despite its slightly longer training and testing times compared to GraphSAGE and EGConv. Notably, NE-GConv achieves this while maintaining the smallest model size among the architectures evaluated. The increased training and testing times observed with NE-GConv can be attributed to its unique capability to learn from both edge and node features during the training process, whereas other graph models exclusively utilize node features. This additional computational overhead associated with NE-GConv is expected given its broader feature set. However, it is important to emphasize that despite these modest increases in time, NE-GConv remains significantly faster than GatedGraphConv while still delivering robust performance in anomaly detection tasks. These findings underscore the efficiency and effectiveness of NE-GConv as a promising model for various applications, particularly those requiring compact model sizes and competitive computational performance.

Graph Model	**Ratio**	**Training ACC%**	**Testing ACC%**	**Precision%**	**Recall%**	**F1%**	**AUROC**	**FPR**
NE-GConv	60:40	98.55	96.70	97.30	96.70	96.92	0.91	0.0330
	70:30	**98.75**	**97.64**	**97.93**	**97.64**	**97.76**	**0.88**	**0.0236**
	80:20	98.63	96.52	96.36	96.52	96.42	0.84	0.0348
GraphSAGE	60:40	93.61	95.34	95.33	95.34	95.32	0.94	0.0466
	70:30	94.40	95.17	95.23	95.18	95.19	0.95	0.0482
	80:20	93.31	94.51	94.52	94.52	94.51	0.94	0.0548
EGConv	60:40	89.28	85.44	87.95	85.45	84.27	0.79	0.1455
	70:30	88.19	91.83	92.68	91.84	91.61	0.89	0.0861
	80:20	83.03	62.17	72.00	62.17	52.59	0.56	0.3783
GatedGraphConv	60:40	97.09	93.40	93.59	93.40	93.28	0.91	0.0660
	70:30	96.98	95.37	95.51	95.38	95.40	0.96	0.0462
	80:20	97.31	96.01	96.09	96.02	96.03	0.96	0.0398

Table 3.4: The performance of NE-GConv and other baseline GNN models on the UNSW-NB15 dataset with different train:test ratios.

NE-GConv Performance with Varying Attack Ratios

The attack ratio in UNSW-NB15 dataset is 7:1, where, 7 refers to the proportion of normal data entries and 1 to the malicious data entries. To check the influence of attack ratio on the model, another experiment is performed in which the UNSW-NB15 training dataset is adjusted using sampling-strategy **imblearn** to have different attack ratios, i.e., 4:1, 2:1, and 1:1, in such a way that the number of malicious entries in the training dataset remain fixed to 224854,

Graph Model	**Ratio**	**Model Size (B)**	**Training Time (s)**	**Testing Time (s)**
NE-GConv	60:40	**4096**	1585.27	3.70
	70:30		1586.95	2.97
	80:20		1538.59	2.26
GraphSAGE	60:40	5120	785.56	2.31
	70:30		**785.08**	1.85
	80:20		796.94	**1.56**
EGConv	60:40	7168	1358.06	2.22
	70:30		1412.05	1.95
	80:20		1338.66	**1.56**
GatedGraphConv	60:40	10240	8898.01	5.54
	70:30		8069.77	4.69
	80:20		8131.62	4.49

Table 3.5: Evaluating computational complexity of NE-GConv and other baseline GNN models on the UNSW-NB15 dataset with different train:test ratios.

whereas normal entries are under-sampled. The performance of NE-GConv for different attack ratios is presented in Table 3.6. Results demonstrate that our model is not overly influenced by the changes in the attack ratio. There is slight fluctuation in the training accuracy when the attack ratio is changed from 7:1 to 4:1 to 2:1 and 1:1. As can be seen from the table that the training accuracy reduces with the size of the training dataset. The testing dataset is kept fixed to 30% of the original dataset that contains 665464 normal entries and 96366 malicious entries. The testing accuracy is found stable between 97-98%, even though the model is trained on different attack ratios. High performance achieved on the testing dataset under varying attack ratios further validates that our model is able to learn the distinguishing features of malicious traffic effectively, and low FPR, high F1 and AUROC scores determine that model is not biased towards attack prevalence in the training dataset.

Evaluation on the ToN_IoT Dataset

To show the model's ability to generalize on different datasets, we run the experiment on the ToN_IoT dataset [99]. The dataset contains 461043 total entries, out of which 300000 are normal, and 161043 are malicious. The dataset is passed through pre-processing and graph

Attack Ratio	**# Normal in Training**	**Training ACC%**	**Testing ACC%**	**Precision%**	**Recall%**	**F1%**	**AUROC**	**FPR**
7:1	1552747	**98.75**	97.64	97.93	97.64	97.76	0.88	0.0236
4:1	899416	98.01	**98.21**	**98.38**	**98.21**	**98.28**	0.91	**0.0179**
2:1	449708	97.14	97.73	98.15	97.73	97.88	0.91	0.0227
1:1	224854	96.87	97.36	98.23	97.36	97.65	**0.95**	0.0264

Table 3.6: Performance of NE-GConv on different attack ratios on UNSW-NB15 dataset.

formation steps before training and testing. The training dataset contains 322730 entries, and the testing dataset contains 138311 entries following the 70:30 split ratio. Model evaluation is done on the testing dataset and is presented in Tables 3.7 and 3.8. Table 3.7 shows the performance of graph models, confirming the superiority of NE-GConv on the ToN_IoT dataset. The NE-GConv performs well in all metrics as compared to other models, with the best accuracy of 88%.

A high accuracy on the ToN_IoT and UNSW-NB15 datasets indicates that the model is able to generalize well by making correct predictions most of the time. Table 3.8 shows that NE-GConv offers a smaller model size, and the training time and the testing time of the proposed NE-GConv are less than EGConv and much less than the GatedGraphConv.

Graph Model	**Training ACC%**	**Testing ACC%**	**Precision%**	**Recall%**	**F1%**	**AUROC**	**FPR**
NE-GConv	**93.26**	**88.38**	**91.75**	**88.38**	**88.88**	**0.92**	**0.1162**
GraphSAGE	86.68	61.30	63.70	61.30	62.06	0.60	0.3870
EGConv	90.49	51.45	57.29	51.46	52.51	0.53	0.4854
GatedGraphConv	77.65	69.05	78.96	69.05	59.70	0.56	0.3095

Table 3.7: Performance evaluation of NE-GConv with other baseline graph models on the ToN_IoT dataset.

Testing on IoT Devices

To investigate the computing resources of NE-GConv, it is evaluated on Raspberry Pi 4 with 8 GB RAM that further uses python 3.9 and PyTorch version 1.13.1 to test the model. After setting up the python environment with the required dependencies of PyTorch, the trained

Graph Model	Model Size (B)	Training Time (s)	Testing Time (s)
NE-GConv	**4096**	**262.72**	**0.76**
GraphSAGE	5120	140.76	0.66
EGConv	7168	337.73	0.71
GatedGraphConv	12288	6719.15	3.26

Table 3.8: Computational overhead of NE-GConv with other baseline graph models on the ToN_IoT dataset.

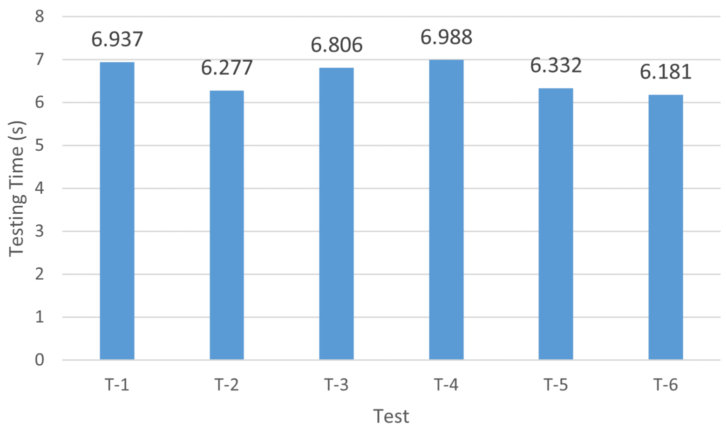


Figure 3.5: Test on Raspberry Pi 4 to investigate Testing Time (s)

proposed model is deployed on the Raspberry Pi device and evaluated on the UNSW-NB15 testing dataset for the 70:30 train:test ratio. An experiment consisting six independent tests is carried out on the testing dataset containing 519425 nodes and 602224 edges as given in Fig. 3.5. Results show that the NE-GConv model can process 309.3 MB of data while consuming a peak memory usage of 2.46 GB and occupying one out of the four CPU cores in 6.18 - 6.98 seconds and achieves the same performance in each test as given in Table 3.4 for the 70:30 ratio. This test confirms the credibility of NE-GConv that it is not only efficient for high computing resources but also for resource constraint platforms.

3.4 Conclusion

In this chapter, we presented a novel and lightweight NIDS exploiting the potential of graph neural networks. The findings can be summarized as below.

- Incorporating both node and edge features in a graph structure enhances the performance of IDS in IoT systems. This finding suggests that capturing both node and edge characteristics allows for a more comprehensive representation of network traffic relations, leading to improved detection accuracy and sensitivity to network intrusions.
- Reducing dimensionality and employing a shallow two-layer graph convolutional model significantly reduces the computational complexity and facilitates deployment in resource-constrained networks. This finding indicates that implementing techniques such as Recursive Feature Elimination for feature selection and designing a lightweight architecture in the NE-GConv model can effectively address the challenges of deploying IDS in resource-constrained environments without sacrificing detection performance.
- Comparative analysis of the proposed NE-GConv model with state-of-the-art GNN models demonstrates its effectiveness in various performance metrics. This finding highlights the superiority of the NE-GConv model in terms of detection accuracy, precision, recall, F1-score, AUROC, and False Positive Rate, validating its potential as a robust solution for intrusion detection in IoT systems.

Chapter 4

A New Concatenated Multigraph Neural Network for IoT Intrusion Detection

4.1 Introduction

Few existing studies [100] in other areas than NIDS and cyber-security refer to the problem of predicting multiple types of edges between nodes in their GNN model design. These researches demonstrate the importance and potential of multi-edge classification in various domains and the development of novel and effective methods to solve this problem. In contrast to these approaches, we propose a novel GNN framework capable of considering a variable number of multi-edges between any pair of nodes implying multi-dimensional edge features. Our model is not limited to a constant / fixed number of edges to do edge classification, bringing an advancement not only in NIDS but also in GNN edge classification in general.

Recent studies on GNN-based NIDS considered a basic graph structure consisting of an IP-Port number pair as a node and a network flow as an edge [73], [76], [77]. The existing graph structures are limited to directional or non-directional types that can not capture full communication between a pair of nodes and may release some network flows. In contrast, We propose a multi-edge-based graph structure that seizes full communication between a pair of nodes. There is a lack of research on the design of the GNN model handling multi-edges in their architecture. We propose new functions and equations to update the existing GNN model to include multi-edges and multi-dimensional edge features in the proposed algorithm. These steps have been updated from the original message passing layers by taking into account the total number of edges, and the edge feature between pair of nodes in the functions, e.g., aggregation functions and attention function. Moreover, the existing GNN models in NIDS have used either spectral convolution or spatial convolution methods only. We propose a concatenated GNN model that is capable of learning underlying spectral and spatial information in the graph structure. Results demonstrate the effectiveness of our model by showing performance improvement of up to 2-5% across all metrics.

In this chapter, we propose a new framework for GNN-based NIDS, which includes all the topological patterns of the knowledge graph and combines both spectral and spatial convolution methods to learn effectively the characteristics of attacks. The framework addresses the over-smoothing issue by treating each layer with different key operations in the node updating step. Moreover, we present a more natural graph structure by enabling multiple edges between a pair of nodes formed by source and destination IP addresses. The rest of the communication is captured in multiple edges. As a result, full traffic gets sealed in the graph structure without any leaky information. We demonstrate that this solution is effective in the detection of malicious behavior in the traffic. With an aim to overcome the shortcomings of recent studies in GNN-based NIDS, the key contributions of this research are summarized as follows:

- We propose an improved network graph structure capable of processing multi-edges with multi-dimensional edge features in the graph structure, thereby allowing to capture of the complete exchange of information between any pair of nodes. Nodes are represented by source and destination IP addresses, and communication between devices is represented as multiple adjacent edges and the multi-dimensional edge feature matrix.
- We develop a new concatenated GNN framework that allows the incorporation of spectral and spatial convolution methods to learn all the spectral and spatial characteristics of underlying graph geometry. We propose new functions and equations to include multi-edges in the proposed model design. Our approach is especially helpful in learning graph geometry and traffic patterns in complex networks.

We evaluate our proposed model on well-known public benchmark datasets and compare results using several performance metrics. Our joint spectral and spatial convolution approach using the multi-edge graph is tested against spectral-only and spatial-only graph models, implying the

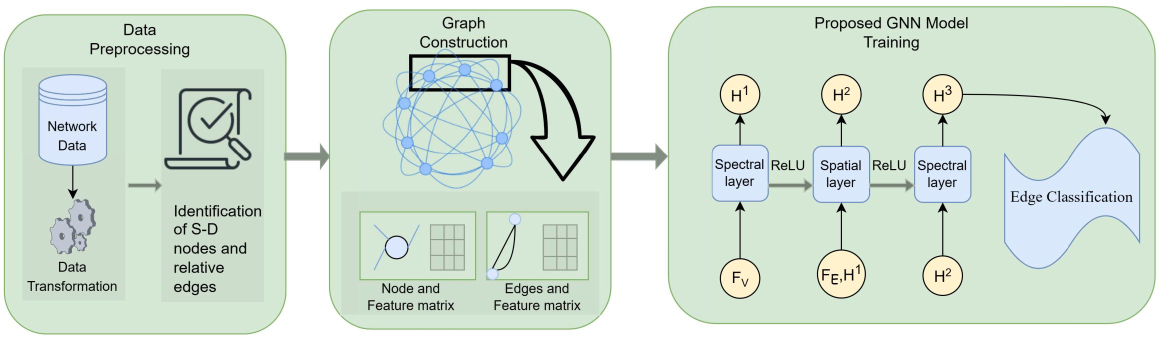


Figure 4.1: Step-by-step illustration of the proposed GNN-based NIDS Framework: Raw dataset is transformed into the processed dataset in Data Preprocessing block as shown on the left-hand side, then the graph is constructed from S(ource)-D(estination) IP nodes and relative edges (obtained from the left block) in Graph Construction block (middle block). The proposed model learns the patterns of the constructed graph in the proposed GNN Model Learning block, as shown on the right-hand side.

same multi-edge graph at the input. Results demonstrate that our proposed model outperforms the other graph models primarily due to the added capability of the proposed model to address both the spectral and spatial geometrical aspects of the complex multi-edge knowledge graph. The research questions explored are as follows.

- How does the proposed GNN framework for NIDS improve upon existing approaches by considering multi-edge classification and multi-dimensional edge features?

- What impact does incorporating both spectral and spatial convolution methods have on the effectiveness of the GNN-based NIDS framework in learning the characteristics of attacks?

- How does the improved network graph structure, capable of processing multi-edges with multi-dimensional edge features, enhance the detection capabilities of the proposed GNN-based NIDS framework?

The rest of the chapter is organized as follows. Section 4.2 illustrates the design of our proposed framework and its layered architecture. Section 4.3 presents detailed experimental results, comparison and evaluation of the model.

4.2 The Proposed Framework

In this section, we describe the stages of the proposed GNN-based NIDS that integrates spectral and spatial domains into a coherent framework to benefit from the proposed graph structure fully. The overall framework consists of three stages: Data Pre-processing, Graph Construction and the Proposed GNN Model for the detection of abnormal behaviors in the network traffic as illustrated in Fig. 4.1. Table 4.1 summarizes the notations and their definitions used in the chapter.

Notation	**Definition**
$\mathcal{A}$	The adjacency matrix
$\hat{\mathcal{A}}$	The normalized Laplacian adjacency matrix
$\tilde{D}$	The diagonal degree matrix
$\mathcal{E}$	The set of edges
$\lvert\mathcal{E}\rvert$	The number of edges
e_{ijr}	The r-th edge from v_i to v_j
$F_{\mathcal{V}}$	The node feature matrix
$F_{\mathcal{E}}$	The edge feature matrix
$g(\cdot)$	Input to output transformation function
$\mathcal{G}_m$	The multigraph
$\mathcal{N}_{v_i}$	The multi-edge neighborhood of node v_i
R_{ij}	Total number of edges from v_i to v_j.
$\mathcal{U}$	Non-linear activation function
$\mathcal{V}$	The set of nodes
$\lvert\mathcal{V}\rvert$	The number of nodes
W^g	The GCN weight matrix
W^{att}	The GAT weight matrix

Table 4.1: Notations used in the chapter and their definitions

4.2.1 Data Preprocessing

The initial phase of the proposed framework involves pre-processing the data, which entails various tasks such as data transformation, data cleaning, categorical feature encoding, and data normalization. The pre-processing unit begins by transforming the source and destination IP addresses into small integer values, which replaces each unique IP address with a unique integer. This conversion not only reduces the computational load resulting from long IP addresses, but also ensures that the learning process is independent of the fixed IP addresses. Subsequently, the data is thoroughly cleansed by removing any Null and Nan values to avoid ambiguity caused by unknown values. The next step involves encoding the categorical features using a categorical encoder, and normalizing the numerical features using a standard scalar, thereby rendering the dataset ready for the next phase of the proposed framework.

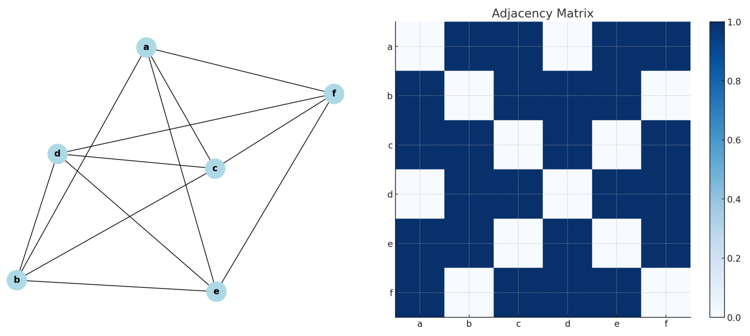


Figure 4.2: Multi-edge graph structure depicting a small IoT network: The left panel illustrates the graph, complete with nodes and interconnecting edges, while the right panel features the adjacency matrix as a heatmap. The heatmap delineates the node connections, with varying shades of blue indicating the existence of edges between node pairs.

4.2.2 Graph Construction

To model a knowledge graph close to the nature of a real IoT network, we propose to construct a multigraph structure denoted by $\mathcal{G}_m = (\mathcal{V}, \mathcal{E}, F_\mathcal{V}, F_\mathcal{E})$ where $\mathcal{V} = (v_1, \cdots, v_N)$ is a set of N IoT nodes identified by their IP addresses, and $\mathcal{E} = \{e_{ijr}, \forall i, j, r\}$ collects edges signifying the network flows between pairs of IoT nodes as shown in Fig. 4.2. Let R_{ij} denote the number of edges from v_i to v_j, and e_{ijr} $(1 \leq r \leq R_{ij})$ denotes the r-th edge feature. The nodes and edges have their feature matrices to exhibit their nature and properties in the knowledge graph and are denoted by $F_\mathcal{V}$ and $F_\mathcal{E}$, respectively. $F_\mathcal{V}$ is an $N \times K$-dimensional feature matrix $(F_\mathcal{V} \in \mathbb{R}^{N,K})$ where K is the number of node features. The i-th element of $F_\mathcal{V}$, i.e., F_{v_i}, collects the node features of node v_i. $F_\mathcal{E}$ is a $|\mathcal{E}| \times K$-dimensional feature matrix $(F_\mathcal{E} \in \mathbb{R}^{|\mathcal{E}|,K})$ where $|\cdot|$ is the cardinality and K is the number of features of each edge. The edge features of edge e_{ijr} are collected in $F_{e_{ijr}} \in \mathbb{R}^{1,K}$. The edge features are the flow features of the IoT network dataset, and node features are denoted as all ones with similar dimensions as edge features as [76]. This approach is used to avoid errors such as mismatched dimensions and to keep edge feature values (traffic flows) significant during the graph embedding process.

The adjacency matrix $\mathcal{A}$ of a multigraph $\mathcal{G}_m$ is an $N \times N$-dimensional matrix where matrix elements correspond to the multiplicity of the existing edges. The element at the i-th row and j-column $a_{ij} = 0$, when there is no edge from v_i to v_j; $e_{ijr} \notin \mathcal{E}$. $a_{ij} = R_{ij}$, when R_{ij} edges exist from v_i to v_j. The multi edge neighborhood of node v_i denoted by $\mathcal{N}_{v_i}$, is a set of all the nodes connected to v_i, $\mathcal{N}_{v_i} = \{v_j | e_{ijr} \in \mathcal{E}\}, 1 \leq r \leq R_{ij}$.

In the constructed graph, there could be more than one edge between a pair of nodes in the case of multiple network flows between the same pair of IoT hosts having different source-destination port#. Our IP-as-node multigraph design can link network flows related to the same host (IP address) rather than losing this connection in the flows of the same IP as in the other models, e.g., [76]. Our design can also capture all the network flows without lossy aggregation of network flows between a pair of hosts. Nevertheless, the multigraph design makes the overall knowledge graph complex in nature and requires a novel combination of GNN algorithms.

4.2.3 Proposed GNN Model

The key feature of the proposed intrusion detection model is its ability to cater to multiple edges and their features, and its concatenated architecture design with modified message-passing layers and aggregation functions. We propose new equations to include multi-edges in the proposed GNN model as given in the Algo. 2 The proposed model is based on a cross-domain approach to connect the spectral and spatial graph convolution methods to optimize the learning of geometric patterns of the multi-edged graph structure posing as an IoT network. It is a common practice to combine several graph layers in one model to investigate the node's depth. After knowledge graph construction, it is passed through the spectral and spatial graph convolution learning methods to learn from the multi-dimensional multi-edge features and graph dynamics implicitly and explicitly. We propose a spectral-spatial-spectral graph model in which GCN [71] represents the spectral and GAT [72] represents the spatial layer. The proposed model consists of three layers with a GAT layer stacked in between two GCN layers to learn inherent graph characteristics in depth.

The embeddings learn the graph topology and its properties using GCN in the first-hop neigh-

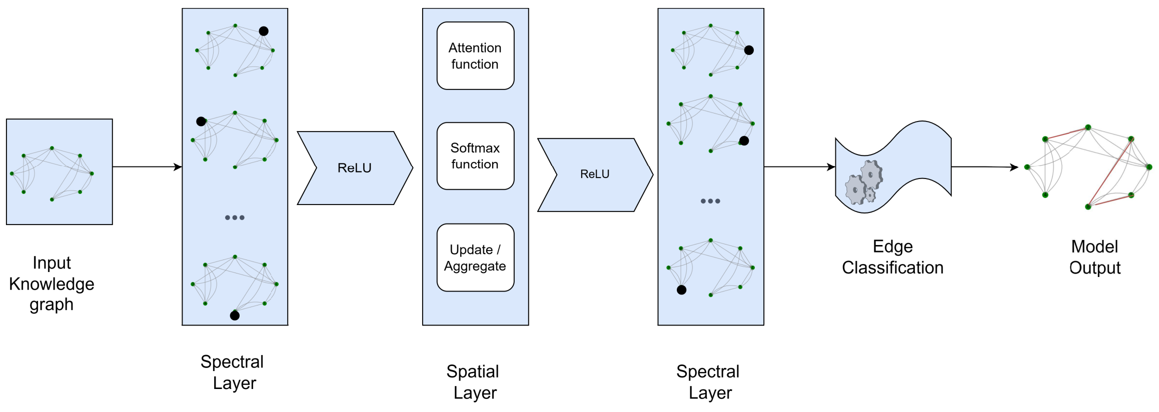


Figure 4.3: Our proposed model architecture consists of 3 layers: Spectral layer (GCN), Spatial layer (GAT), and Spectral layer (GCN). A multi-edge knowledge graph is fed as model input, and link prediction is performed to detect malicious flows in the output.

bors. The embeddings formed are then weighed using edge features and updated through the attention mechanism around the second-hop neighborhood. Lastly, embeddings are updated by learning topological properties around the third-hop neighborhood in the third layer. This method offers improved performance compared to the standard GNN models consisting of the same graph layers and computing the same graph convolution and filters in each hop. In addition, it prevents over-smoothing of edge features as we include edge features in the attention mechanism only. The rest of the two layers update node embeddings from multi-edge graph topology.

The layered architecture of the proposed GNN model is illustrated in Fig. 4.3 and is expressed as:

$$g(F_v, F_{\mathcal{E}}, \mathcal{A}) = \hat{\mathcal{A}}\mathcal{U}\left(\mathcal{A}F_{\mathcal{E}}\mathcal{U}\left(\hat{\mathcal{A}}F_{\mathcal{V}}W_1^g\right)W_2^{att}\right)W_3^g, \tag{4.1}$$

where, W_1^g, W_2^{att} and W_3^g are the weight matrices of the hidden layers 1, 2, 3, and are trained using the gradient descent. $W_1^g \in \mathbb{R}^{K,H_1}$, where, K is the dimension of node and edge features and H_1 is the dimension of the first hidden layer. Similarly, $W_2^{att} \in \mathbb{R}^{H_1,H_2}$, where, H_2 is the dimension of second hidden layer, and $W_3^g \in \mathbb{R}^{H_2,O_3}$, where, O_3 is the dimension of the third layer. $\hat{\mathcal{A}}$ is the normalized Laplacian representation of adjacency matrix $\mathcal{A}$ and can be given by

$$\hat{\mathcal{A}} = \tilde{D}^{-1/2}\tilde{\mathcal{A}}\tilde{D}^{-1/2}. \tag{4.2}$$

Here, $\tilde{\mathcal{A}} = \mathcal{A} + I_N$, and $I_N \in \mathbb{R}^{N,N}$ is the identity matrix, $\tilde{D}_{ii} = \sum_{v_j} \tilde{\mathcal{A}}$ is the diagonal node degree matrix of the adjacency matrix, and $\tilde{D}_{ii}^{-1/2}$ is the inverse of the square root of

$\tilde{D}$. $\mathcal{U}$ denotes a non-linear activation function, i.e., ReLU [92] in this chapter. Equation (4.1) summarizes the three-layered GNN propagation that requires $F_{\mathcal{V}}$, $\mathcal{A}$ and $F_{\mathcal{E}}$ as the model inputs. In the first layer, the proposed model learns from the underlying graph structure by eigen decomposition of the graph Laplacian matrix and graph nodes are updated using the mean aggregation of edge weight and initial node features divided by node degrees. The aggregation operation is followed by a non-linear activation function $\mathcal{U}$. The second layer applies an attention mechanism to learn the importance of each node v_j for the central node v_i w.r.t edge features between them, unlike treating all nodes equally as is in GCN. The attention coefficients are computed from the updated node representations from layer 1 and initial edge features, using the addition aggregation function and non-linear activation $\mathcal{U}$. Layer 3 of the proposed model applies GCN to recompute features in the third hop neighborhood by learning about the eigen decomposition of the graph structure again and is updated in the same way as layer 1. The proposed model layers generate node embeddings using the multi-edge adjacency matrix and the edge features.

Algorithm 2: Multigraph Neural Network Algorithm

```
Input:
    Graph $\mathcal{G}_m$
    Multi edge Adjacency matrix $\mathcal{A}$
    Node features matrix $F_{\mathcal{V}}$
    Edge features $F_{\mathcal{E}}$
Output:
    Edge embedding $z_{ijr}, \forall e_{ijr} \in \mathcal{E}$
▷Initialization
1  $h^0_{v_i} \leftarrow F_{v_i}, \forall v_i \in \mathcal{V}$,
2  $\tilde{z}^0_{ijr} \leftarrow F_{e_{ijr}}, \forall e_{ijr} \in \mathcal{E}$,
▷Node Embedding in Layer 1
3  for $\forall v_i \in \mathcal{V}$, do
4      $\tilde{h}^1_{v_i} \leftarrow \sum_{v_j \in \mathcal{N}_{v_i} \bigcup v_i} \sum_{r \in [1, R_{ji}]} \frac{h^0_{v_j}}{\sqrt{\hat{d}_i \hat{d}_j}}$,
5      $h^1_{v_i} \leftarrow \text{ReLU}(W^g_1 \times \tilde{h}^1_{v_i} + b^g_1)$,
▷Node Embedding in Layer 2
6  for $\forall v_i \in \mathcal{V}$, do
7      for $\forall v_j \in \mathcal{N}_{v_i} \bigcup v_i$ and $r \in [1, R_{ji}]$ do
8          $\epsilon_{jir} \leftarrow \text{ATT}(h^1_{v_i}, h^1_{v_j}, \tilde{z}^0_{jir})$,
9      $\tilde{h}^2_{v_i} \leftarrow \sum_{v_j \in \mathcal{N}_{v_i} \bigcup v_i} \sum_{r \in [1, R_{ji}]} (h^1_{v_j} \circ \epsilon_{jir})$,
10     $h^2_{v_i} \leftarrow \text{ReLU}(W^{att}_2 \times \tilde{h}^2_{v_i} + b^{att}_2)$,
▷Node Embedding in Layer 3
11 for $\forall v_i \in \mathcal{V}$, do
12     $\tilde{h}^3_{v_i} \leftarrow \sum_{v_j \in \mathcal{N}_{v_i} \bigcup v_i} \sum_{r \in [1, R_{ji}]} \frac{h^2_{v_j}}{\sqrt{\hat{d}_i \hat{d}_j}}$,
13     $h^3_{v_i} \leftarrow W^g_3 \times \tilde{h}^3_{v_i} + b^g_3$,
▷Edge Embedding
14 for $\forall v_i \in \mathcal{V}$, do
15     for $r \in [1, R_{ji}]$ do
16         $z_{ijr} = \text{MLP}(h^3_{v_i} || h^3_{v_j})$,
return $z_{ijr}$
```

The input of Algo. 2 consists of a multi-edge graph $\mathcal{G}_m$ with node features $F_{\mathcal{V}}$ set to all ones $\{1, \ldots, 1\}$ for each node and edge features $F_{\mathcal{E}}$ to R_{ij} parallel edge features. The node and edge features are initialized with the pre-processed knowledge graph values.

Layer 1 spans steps $3, 4$, and 5. In the first layer, the input adjacency matrix is transformed into normalized Laplacian matrix $\hat{\mathcal{A}}$ as given in (4.2). The spectral convolution is achieved by the multiplication of neighboring node features, edge weight, and the central node features in the spectral neighborhood $\tilde{\mathcal{N}}_{v_i}$, and their aggregation around the neighborhood. Step 4 shows the first aggregated node embedding, where, $h^0_{v_i}$ is the initial node feature matrix of node v_i, $h^0_{v_j}$ is neighboring node feature matrix, and $\hat{d}_i = 1 + \sum_{v_j \in \mathcal{N}_{v_i}} e_{ijr}$. In step 5, node embedding $\tilde{h}^1_{v_i}$ of node v_i is calculated by applying the trainable parameters W^g_1 and b^g_1 on the aggregated message and updated using the ReLU function. The aggregation function AGG is given as:

$$AGG(h^l_{v_i}, h^l_{v_j}, R_{ij}) = \sum_{v_j \in \mathcal{N}_{v_i} \bigcup v_i} \sum_{r \leq R_{ji}} \frac{h^l_{v_j}}{\sqrt{\hat{d}_i \hat{d}_j}}, \tag{4.3}$$

where, l is the layer number, $\hat{d}_i = 1 + \sum_{v_j \in \mathcal{N}_{v_i}} R_{ij}$. In step 5, node embedding $\tilde{h}^1_{v_i}$ of node v_i is calculated by applying model's trainable parameters W^g_1 and b^g_1 on the aggregated message and updating using $ReLU$ function.

Layer 2 starts by applying the attention mechanism ATT on the node and edge features that consists of a single layer feed-forward linear neural network and is expressed in (4.4), where the attention function α is subject to the restriction $\alpha : \mathbb{R}^K \times \mathbb{R}^{K'} \rightarrow \mathbb{R}$, where $\mathbb{R}^K$ is the dimension of input node features and $\mathbb{R}^{K'}$ is the dimension of parameterized weight matrix of the attention function $W^{att}_2 \in \mathbb{R}^{K'}$.

$$\begin{aligned} &\mathrm{ATT}(h_{v_i}, h_{v_j}, \tilde{z}_{ijr}) = \\ &\qquad \frac{\exp(\mathrm{LeakyReLU}(\alpha(W^{att}_2 h^1_{v_i} || W^{att}_2 h^1_{v_j} || W^{att}_2 \tilde{z}^0_{ijr}))}{\sum_{\forall v_m \in \mathcal{N}_{v_i}} \sum_{q \leq R_{mi}} \exp(\mathrm{LeakyReLU}(W^{att}_2 h^1_{v_i} || W^{att}_2 h^1_{v_m} || W^{att}_2 \tilde{z}^0_{miq}))}. \end{aligned} \tag{4.4}$$

Learnable attention coefficients ϵ_{ijr} are computed by passing $h^1_{v_i}$, $h^1_{v_j}$ and $\tilde{z}^0_{ijr}$ through the attention function α in the step 8. It determines the importance of node v_j features to node v_i by considering the source and destination node embeddings and the multi-edge features between them. We upgrade Steps 7 and 8 to include edge features and adjusted it for multiple edges. An attention score depends on the multiple edge features to obtain the structural information computed via attention score in second-hop neighbors. In addition, Leaky-ReLU [101] (with negative input slope = 0.2) nonlinearity function is applied, followed by a softmax function to have a common scaling and normalized coefficients across the entire neighborhood. The node information is aggregated with the attention coefficients using the addition aggregation in step 8, where $\circ$ is element-wise multiplication. The result is passed through the ReLU function to update layer 2 in step 9.

Layer 3 recomputes features by learning the graph structure in the same way as layer 1 in steps $11, 12$ and 13. The multi edge embedding is computed by concatenating layer 3 features $h^3_{v_i}$

and $h^3_{v_j}$ and passing through an MLP function [102] in steps 14 and 15.

$$\text{MLP}(h^3_{v_i}||h^3_{v_j}) = W_{mlp} \times (h^3_{v_i}||h^3_{v_j}) + b_{mlp}, \tag{4.5}$$

where $||$ is the concatenation operation, W_{mlp} is the weight, and b_{mlp} is the bias.

GNNs have gained popularity in various domains due to their ability to learn from graph-structured data, e.g., social network analysis, molecular and chemical analysis, cyber-security, and traffic and transportation networks. The proposed model has the potential to be generalized and presents a new viewpoint for a wide range of tasks involving the same characteristics and graph structure, aiming to predict edge properties with appropriate modifications and tuning. For example, in social network graph, individuals are represented as nodes and the relation between them is represented as edges. These relations can be of various types, e.g., friendships, family ties, work relationships. Often, individuals in a social network can have multiple types of relationships with each other, resulting in a graph with multiple edges between nodes. Edge classification in social network graph can involve predicting the type of relationship between two individuals connected by the edges. Our proposed model can be applied to such scenarios where the graph structure consists of multi-edges and edge classification is a requirement. The node and edge feature matrix dimensions need to be matched to avoid uncertainty during the training process. Pre-processing steps and model hyperparameters, e.g., the number of hidden units, optimizer, learning rate, and activation functions, may need adjustments for better results on other tasks.

The proposed model is inspired by the approach in prior work [76] that considers a simple IoT graph and runs it through the E-GraphSAGE model with some adjustments on the inclusion of edge features. The node features are considered as all ones and edge features are the representation of network flows. Edge classification is performed to classify the network flows as normal or malicious. We take this approach a step further and build a multi-edge-based edge classifier with modified equations and functions, e.g., steps $4, 7, 8, 9, 12, 15$, and 16 in Algo. 3. These steps have been updated from the original message passing layers by taking into account R_{ij} and $e_{ijr} : (r \in [1, R_{ij}])$, which are the total number of edges, and $e_{ijr} : (r \in [1, R_{ij}])$ denotes the r-th edge feature from v_i to v_j in all the functions, e.g., aggregation functions and attention function. The proposed model takes in multi-edged graph data, which is a novel IoT network design. Then, spectral and spatial convolution operations are combined in the form of GCN and GAT layers. The message-passing layers are adjusted to include multiple edges with relevant multi-dimensional features. This improves the GNN model's ability to capture more complex dependencies and traffic patterns of IoT network graphs as compared to the existing studies in GNN.

4.3 Experiment and Results

This section describes the benchmark datasets used and the experimental evaluation of the datasets. A comparison study is carried out to show the efficiency of the proposed framework and the baseline models in this domain.

4.3.1 Performance Evaluation Metrics

The proposed model addresses a classification task. For the evaluation of the proposed model in the detection of attacks, the following classification metrics are considered:

- **Accuracy:** This is a measure of correctly classified observations out of all data observations, and is given by $\frac{T_p+T_n}{T_p+F_p+T_n+F_n}$. Here, T_p is True Positive, T_n is True Negative, F_p is False Positive, and F_n is False Negative.

- **Precision:** This is a measure of how precise a model is in correctly predicting positives (T_p) out of all the predicted positives ($T_p + F_p$), and is given by $\frac{T_p}{T_p+F_p}$.

- **F1-Score:** This is a function of Precision and Recall, and is used where there is imbalanced data, and is given by $(2 \times \frac{Precision \times Recall}{Precision + Recall})$.

- **Recall:** This is a measure of how many True Positives a model predicts out of all the actual positives, and is given by $\frac{T_p}{T_p+F_n}$.

- **False Alarm Rate (FAR):** This is a measure of how many normal data points are classified as malicious and is given by $\frac{F_p}{T_n+F_p}$.

- **Receiver Operating Characteristic - Area Under the Curve (ROC-AUC) Score:** This is a measure of the rank correlation between predictions and actual labels **roc**. The higher the score, the better the model's ability to classify positive and negative class points. It summarizes the Precision-Recall curve.

4.3.2 Datasets and Experimental Setup

The datasets used to evaluate the proposed model are Ton IoT [99], BoT IoT [103], NF-Ton IoT [104], and NF-BoT IoT [104]. These are the most widely used datasets for the detection of attacks in IoT network traffic. We purposely include the recently developed variants of UNSW datasets [104] that are the NetFlow versions of already existing datasets (NF-Ton IoT and NF-BoT IoT). The datasets are preprocessed and transformed into knowledge graphs. The knowledge graph is equipped with nodes, edges and features, and the details of each knowledge graph, along with the dataset sizes before knowledge graph construction, are given in Table 4.2. The knowledge graphs are mapped to split 70% of edges for training and 30% for testing for the evaluation process. All the experiments are carried out on Google Colab with server specs as Intel(R) Xeon(R) CPU @ 2.20GHz running Python v3.7, Pytorch Geometric v1.12.1 and CUDA 113.

Datasets	**Total Data**	**Normal Data**	**Malicious Data**	**No. of Nodes**	**No. of Edges**	**Ratio**
Ton IoT [99]	461043	300000	161043	11536	461043	7:3
BoT IoT [103]	659363	430	658933	84	659363	7:3
NF-Ton IoT [104]	1379274	270279	1108995	1478	1379274	7:3
NF-BoT IoT [104]	600100	13859	586241	129	600100	7:3

Table 4.2: Details of datasets in terms of total data entries, number of malicious and normal data entries, and the knowledge graph of each dataset.

Datasets	**Accuracy %**	**Precision**	**F1-Score**	**Recall %**	**ROC-AUC**	**FAR**
Ton IoT	99.55	0.9950	0.9955	99.50	**1.00**	0.0144
BoT IoT	**99.96**	**0.9996**	**0.9995**	**99.96**	0.79	**0.0004**
NF-Ton IoT	99.28	0.9929	0.9928	99.29	0.98	0.0071
NF-BoT IoT	98.91	0.9886	0.9874	98.91	0.75	0.0109

Table 4.3: Performance evaluation of the proposed Multigraph Neural Network model.

4.3.3 Experimental Analysis

The experimental analysis is based on the binary classification results of the proposed model on all four datasets. The model hyper-parameters are set to 32 hidden units, Adam optimizer [105] with 0.01 learning rate, and Cross Entropy Loss function [106] in the training process. Table 4.3

presents comprehensive results of our intrusion detection model on each of the benchmark datasets. Results show that the proposed model performs exceptionally well across all the performance metrics in all the datasets. The high F1 and ROC-AUC scores determine the strength and stability of our proposed model across imbalanced datasets. The potential factors contributing to the variation in ROCAUC are disparities in data quality and class distribution. To address the specific challenge of data imbalance in GNNs, it's essential to acknowledge that traditional oversampling and undersampling techniques may not be suitable due to their potential to introduce multiple edges with the same features, which can disrupt the learning process. Therefore, it's imperative to devise tailored strategies to tackle this issue effectively.

Different evaluation approaches are considered to distinguish the significance of the proposed model. Firstly, an experimental setting is designed to compare the effectiveness of hybrid convolution in our model with other spectral-only and spatial-only convolution models. The comparison is presented in Figs. 4.4 to 4.9. A three-layered GCN model represents the spectral convolution, and a three-layered GAT model represents the spatial convolution method. We purposefully choose GCN and GAT models as they also form the layer combination in our proposed model. This comparison with these models is recorded under a similar experimental setup as the proposed model. All the datasets are tested on three-layered spectral or spatial models with a multi-edge graph structure at the input. In addition, we also present a comparison with other state-of-the-art GNN studies such as [76] by listing down their results in the table as mentioned in their papers.

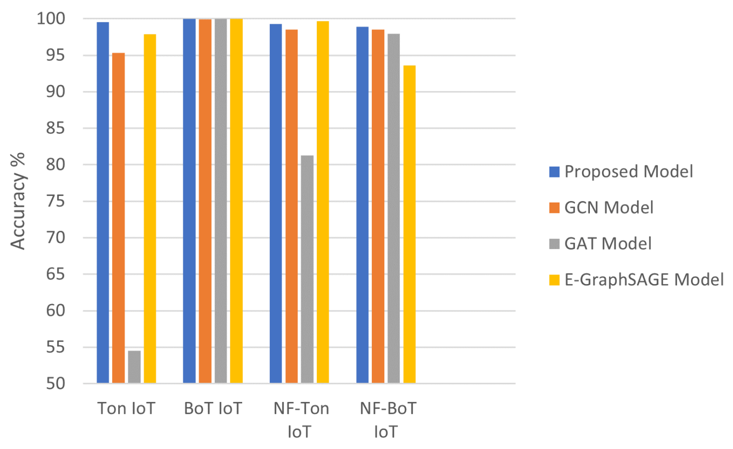


Figure 4.4: Accuracy score comparison of all the graph models on the datasets.

Fig. 4.4 plots the accuracy of the proposed GNN model and its counterparts on each of four datasets: Ton IoT, BoT IoT, NF-Ton IoT, and NF-BoT IoT at x-axis and the accuracy score at y-axis. Fig . 4.4 demonstrates the superior performance of the proposed Multigraph Neural Network model, achieving the best accuracy across all four datasets: 99.55% on Ton IoT, 99.96% on BoT IoT, 99.28% on NF-Ton IoT, and 98.91% on NF-BoT IoT. While E-GraphSAGE exhibits commendable performance, however, it falls short of the proposed model's superior performance. Despite achieving similar accuracy to the proposed model on the BoT IoT dataset, almost all evaluated models reach 99.9% accuracy. The GCN model displays consistent performance with over 95% accuracy across all datasets, whereas the GAT model excels only on the BoT IoT (99.96% accuracy) and NF-BoT IoT (97.96% accuracy) datasets but underperforms on Ton IoT (54.48% accuracy) and NF-Ton IoT (81.3% accuracy). This discrepancy may be attributed to the relative balance of the latter two datasets compared to the others. The results indicate that the unique design of the proposed Multigraph Neural Network model effectively combines the advantages of GCN and GAT, ultimately enhancing accuracy. Fig. 4.5

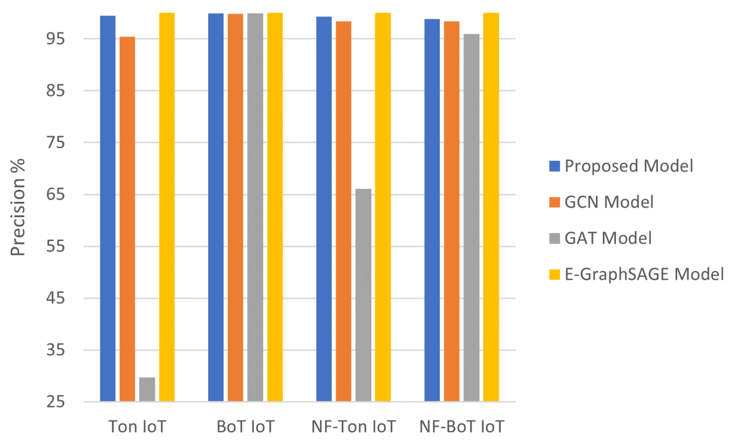


Figure 4.5: Precision score comparison of all the graph models on the datasets.

plots each GNN model on the datasets used on the x-axis vs precision score (%) on the y-axis. The comparison shows that the proposed model achieves high precision across all four datasets: 99.50% on Ton IoT, 99.96% on BoT IoT, 99.29% on NF-Ton IoT, and 98.86% on NF-BoT IoT, which confirms proposed model's ability to detect false positives effectively. Although the E-GraphSAGE model achieves 100% performance on all four datasets, it is important to consider that it has significantly lower accuracy than proposed model especially on Ton IoT and NF-BoT IoT datasets, which suggests that E-GraphSAGE may be making many false negative predictions. Hence, E-Graph may not be able to generalize well to new data and might be overfitting the data. The GCN model shows consistent performance across all datasets, with its best performance on the BoT IoT dataset. However, it is outperformed by the proposed model on all datasets. The GAT model performs well on the BoT IoT dataset but poorly on the Ton IoT dataset, suggesting model's relatively poor ability to generalize well and distinguish false positives.

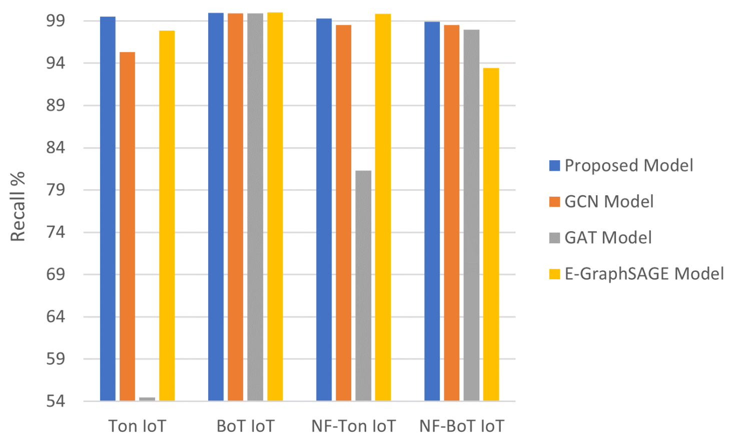


Figure 4.6: Recall score comparison of all the graph models on the datasets.

Fig. 4.6 plots GNN models and the datasets on the x-axis vs the Recall score on the y-axis. The comparison shows that our model surpasses all the other GNN models on recall score across all datasets with 99.50% on Ton IoT, 99.96% on BoT IoT, 99.29% on NF-Ton IoT, and 98.91% on NF-BoT IoT, indicating its effectiveness on these cases. The E-GraphSAGE model performs low on the Ton IoT and NF-BoT IoT datasets with 97.86% and 93.43%, which indicates that the E-Graph is failing to identify a significant portion of the minority class (malicious traffic), which can be critical on the determination of few attacks. The GCN model exhibits consistent performance across all datasets. It performs particularly well on the BoT IoT and NF-Ton IoT datasets. However, it is outperformed by the proposed model on most of the datasets. The GAT model shows variable performance but performs relatively poorly on the Ton IoT dataset with 54.48% indicating its limited capability to generalize across various datasets.

Fig. 4.7 shows another plot depicting performance evaluation in terms of F1 score. The proposed model outshines other graph models on all the datasets with 99.00%, 99.95%, 99.28%, and 98.74% on Ton IoT, BoT IoT, NF-Ton IoT, and NF-BoT IoT, respectively, indicating its effectiveness in these cases. The E-GraphSAGE model does not perform well on the NF-Bot IoT datasets, this could be an indication of overfitting or a flaw in the evaluation process that the E-GraphSAGE cannot effectively capture the patterns in the minority class as good as the proposed model. The GCN model exhibits consistent performance across all datasets. It performs particularly well on the BoT IoT dataset, with only marginal differences on the Ton IoT dataset. However, it cannot compete wiht the proposed model on all datasets. The GAT model achieves good F1 scores on the BoT IoT dataset but performs relatively poorly on the Ton IoT dataset showing that it has inconsistent performance for different datasets and can not generalize well to new data.

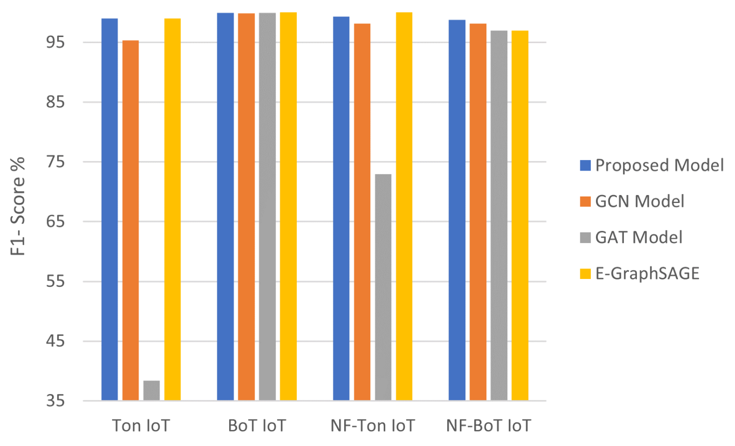


Figure 4.7: F1-Score comparison of all the graph models on the datasets.

The ROC-AUC results are also presented in Fig. 4.8, showing graph models and different datasets on the x-axis and the score on the y-axis. Results demonstrate that our proposed model has high ROC-AUC score than all the other models across all the datasets, validating our model's distinguished performance among others in discriminating between the positive and negative samples. The GCN model exhibits varying performance across the datasets. It performs well on the Ton IoT dataset, although not as well as the proposed model. On the NF-Ton IoT and NF-BoT IoT datasets, GCN model performance is moderate, and it is outperformed by the proposed model. Its performance on the BoT IoT dataset is relatively weak, with an ROC-AUC of 0.5, indicating random chance. The GAT model shows relatively poor performance across all datasets. It has the lowest ROC-AUC on the Ton IoT and NF-Ton IoT datasets, with a value of 0.5, indicating that its predictions are no better than random chance. Its performance on the BoT IoT and NF-BoT IoT datasets is slightly better but still considerably lower than the proposed model. This demonstrates proposed model's ability to efficiently learn from the graph topology and structure in addition to the features than the rest. This metric is missing in E-GraphSAGE [18] results and discussion.

Fig. 4.9 shows that the proposed model demonstrates low FAR across all four datasets: Ton

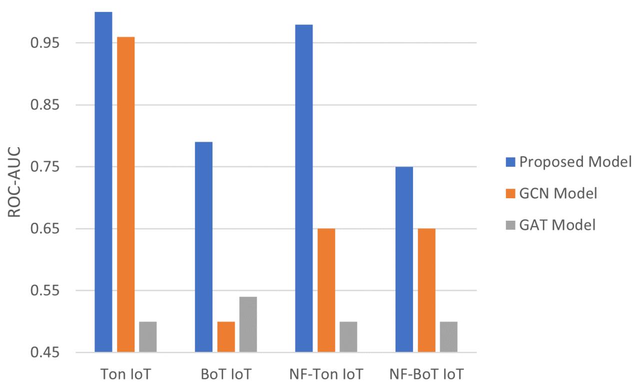


Figure 4.8: ROC-AUC score comparison of all the graph models on the datasets.

IoT, NF-Ton IoT, BoT IoT, and NF-BoT IoT with values 0.0144, 0.0071, 0.00041, and 0.0109, indicating its effectiveness in reducing false alarms. All models achieve the highest FAR scores on the Ton IoT dataset, which is the most balanced among the four, containing 300,000 normal data entries and 161,043 malicious data entries. This demonstrates that reducing the FAR on this dataset poses a significant challenge. Nevertheless, the proposed model attains the lowest FAR of 0.0144, while the FAR scores of other models, including E-GraphSAGE, are approximately 0.02. The E-GraphSAGE model performs well on other datasets. The GCN model exhibits varying FAR across the datasets. It performs better than the GAT model on all datasets, but its FAR is higher in most cases than the proposed models. The GAT model has the highest FAR on the Ton IoT and NF-Ton IoT datasets, indicating poor performance. On the BoT IoT and NF-BoT IoT datasets, its FAR is lower but still not as low as the proposed model. As confirmed by results in Fig. 4.9, our model is capable to limit false alarms on intrusion detection.

Results show the effectiveness of the hybrid convolution approach for intrusion detection in all the benchmark datasets as compared to spatial-only and spectral-only approaches. The proposed model is shown to have lower precision than E-GraphSAGE [76] in three datasets, recall score in one out of four datasets. The FAR of the proposed model is moderately higher in two out of four datasets as compared to one of the other graph models (E-GraphSAGE), whereas, the proposed model closely follows [76] in performance despite its complex graph geometry. This difference is the bare minimum in comparison, considering the fact that the graph structure and the proposed graph model designed in our research are highly complex in nature, whereas the knowledge graphs in otherwise the proposed studies are unable to capture full information.

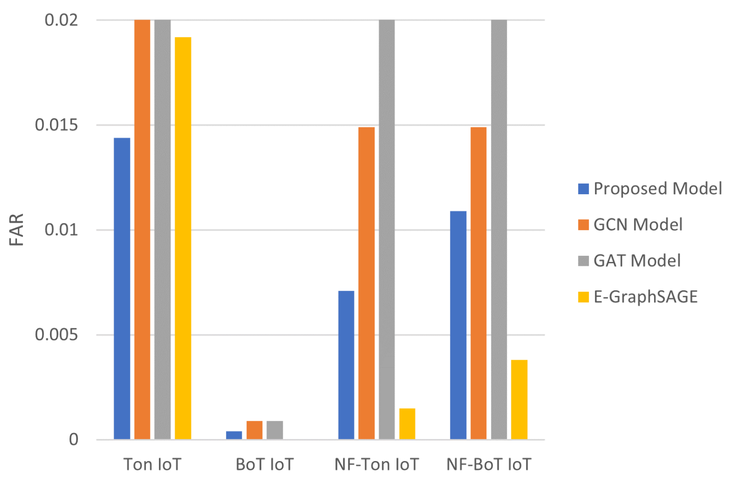


Figure 4.9: FAR score comparison of all the graph models on the datasets.

Overall, the proposed model achieves high accuracy on all the datasets, ranging from 98.91% to 99.96% on the four benchmark datasets. Precision and Recall values are also quite high, with precision ranging from 98.86% to 99.96% and recall ranging from 98.91% to 99.96%. This suggests the model has a low false positive rate and a high true positive rate and is accurately identifying positive samples while avoiding false positives. The F1-score, which is a measure of the model's accuracy, is also high, ranging from 98.74% to 99.95%. The high F1 scores suggest that the model has a good balance between precision and recall, which is crucial for NIDS, where false positives or false negatives can have significant consequences. The False Alarm Rate (FAR) is relatively low, ranging from 0.00041 to 0.0144, indicating that the model is effective at minimizing false alarms. The ROC-AUC values show that the models have a high level of discrimination between positive and negative classes, with values ranging from 0.75 to 1.00, making it a reliable tool for identifying malicious traffic. The results indicate that our proposed model performs exceptionally well on all the datasets and is a valuable advancement in the field of NIDS.

The proposed model is also evaluated on the UNSW NB-15 dataset [107], which is a benchmark public NIDS dataset capturing IoT traffic characteristics. The proposed model achieves a test accuracy of 98.88%, precision of 98.89%, recall of 98.88%, F1 score of 98.88%, AUC-ROC score of 0.91, and FPR of 0.0112. The result proves that the proposed model can perform exceptionally well across all metrics on other datasets possessing similar characteristics as multi-edges structure and that its performance is not restricted to the characteristics of a particular dataset.

Datasets	Metrics	Proposed Model	GCN Model [71]	GAT Model [72]
Ton IoT	Train Time (s)	0.9673	**0.9147**	1.1576
	Model Size (B)	24576	**23552**	26624
BoT IoT	Train Time (s)	1.7804	**1.4639**	2.2738
	Model Size (B)	25088	**24064**	27136
NF-Ton IoT	Train Time (s)	2.8994	**2.4131**	3.442
	Model Size (B)	22016	**20992**	24064
NF-BoT IoT	Train Time (s)	1.1607	**1.1341**	1.5317
	Model Size (B)	22016	**20992**	26624

Table 4.4: The comparison of training time and model size between our method and other GNN models on the benchmark datasets.

In Table 4.4, we compare Training-time/Epoch in seconds and model size in bytes for all the graph models on the datasets. Specifically, all codes run on Google Colab in Pytorch Geometric with the same experimental settings (see Section 4.3). The Epoch range varies across datasets but is the same across all the GNN models on each dataset such as Epoch is 4500 for Ton IoT, 1500 for NF-Ton IoT, 2000 for NF-BoT IoT and 300 for BoT IoT. The python time module, *time.time*(), is called at the beginning and end of the training process to evaluate the training time. The python module *torch.cuda.memory_allocated*() is called to determine the GNN model sizes in bytes.

The proposed model demonstrates competitive training times across all four datasets. In the Ton IoT and NF-BoT IoT datasets, it takes slightly longer to train than the GCN model, but its training time is shorter than the GAT model. The GCN model generally has the shortest training time among the three models, with the exception of the NF-Ton IoT dataset, where the proposed model trains faster. The GAT model takes the longest time to train across all datasets.

The proposed Model has a moderate model size across all datasets. In all cases, its model size is larger than the GCN Model but smaller than the GAT model. The GCN model has the smallest model size among the three models for all datasets. The GAT model has the largest model size in all datasets.

In conclusion, the proposed model demonstrates competitive performance in terms of training time and model size across all datasets. While the GCN model has the advantage of shorter training times and smaller model sizes, the proposed model offers a good balance between these factors and the performance metrics discussed in previous observations. The GAT model, on the other hand, has longer training times and larger model sizes, which may be a disadvantage when considering resource constraints and deployment scenarios. The high performance of the

Dataset	Train:Test Ratio	Test Accuracy	Precision	Recall	F1	AUC-ROC	FPR
Ton IoT	60:40	96.66	96.83	96.66	96.67	0.97	0.0334
	70:30	**99.55**	**99.50**	**99.00**	**99.50**	**1**	**0.0144**
	80:20	97.27	97.39	97.27	97.28	0.98	0.0273
BoT IoT	60:40	99.95	99.95	99.95	99.94	0.78	0.0005
	70:30	**99.96**	**99.96**	**99.95**	**99.96**	**0.79**	**0.00041**
	80:20	99.96	99.96	99.96	99.96	0.79	0.0004
NF-Ton IoT	60:40	99.85	99.85	99.85	99.85	1	0.0015
	70:30	**99.28**	**99.29**	**99.28**	**99.29**	**0.981**	**0.0071**
	80:20	99.37	99.37	99.37	99.37	0.98	0.0063
NF-BoT IoT	60:40	98.43	98.32	98.43	98.07	0.66	0.0157
	70:30	**98.91**	**98.86**	**98.74**	**98.91**	**0.75**	**0.0109**
	80:20	97.74	97.79	97.74	96.63	0.5	0.0226

Table 4.5: Performance of proposed model on various train : test ratios on all the benchmark datasets.

proposed model on all four datasets for the varying train : test ratio across performance metrics, e.g., accuracy, precision, recall and F1, suggests that our proposed model is overall stable to these changes and is capable enough to generalize well as long as train dataset is representative of the population and the distribution is similar across normal and abnormal subsets of the data. This can be confirmed by relatively poor AUC-ROC scores in BoT IoT and NF-BoT IoT datasets, which are highly imbalanced datasets as can be seen in Table 4.5. The BoT IoT dataset contains only 430 normal data points out of a total of 659363 data points, whereas NF-BoT IoT contains 13859 normal data points out of 600100 total data points. This limits the model's capability to differentiate positives from negatives. It is also observed in the Ton IoT dataset that when the training set is smaller (60:40), the model does not have enough data to learn the patterns of each class in the training set, resulting in lower generalization performance on the test set (96%). Overall, these results suggest that the proposed GNN NIDS model is an effective framework for detecting network intrusions. The proposed model is stable to varying train : test ratios having good generalizability.

4.4 Conclusion

In this chapter, we proposed a novel framework for NIDSs that has implied GNN to learn from the multi-edge features as well as the hidden patterns of the graph structure for the detection of attacks.

- The proposed GNN framework for NIDS advances existing approaches by addressing the challenge of multi-edge classification and incorporating multi-dimensional edge features. This allows for a more comprehensive representation of network traffic relations, enabling the model to capture the complete exchange of information between any pair of nodes.
- Incorporating both spectral and spatial convolution methods in the GNN-based NIDS framework facilitates effective learning of the spectral and spatial characteristics of underlying graph geometry. This approach mitigates the over-smoothing issue and improves the model's ability to detect anomalous behavior in network traffic.
- The improved network graph structure significantly enhances the detection capabilities of the proposed GNN-based NIDS framework by capturing the full communication between nodes without any leaky information. Experimental results demonstrate the effectiveness of the proposed approach, showing performance improvement across all metrics compared to existing models.

Chapter 5

GNN-based Sequential Network Traffic Analysis for Botnet Detection

5.1 Introduction

With the expansion of Internet of Things (IoT) ecosystem, it is foreseen that a substantial portion—approximately 52%—of these devices will comprise low-cost and low-maintenance massive IoT devices, which can perform simple tasks and lack the ability to run complex real-time algorithms for attack detection and prevention [108]. As a consequence, IoT devices are frequently vulnerable to attackers [109]–[111], with Denial of Service (DoS) attacks being a common type, accounting for approximately 20% of all attacks targeting the IoT [112]. With the rapid growth of IoT systems, the threat landscape has witnessed a surge in emerging botnet attacks, posing significant challenges to Intrusion Detection Systems (IDS). Conventional deep learning and machine learning techniques, while effective in certain domains, have struggled to effectively capture the intricate patterns and dynamics of IoT botnet attacks due to the absence of inherent graph-like structures in their methodologies. However, Graph Neural Networks (GNNs) [113] have emerged as a promising solution for structured data analysis in various domains, such as social media and recommendation systems, enabling the exploitation of underlying graph structures in data representations. Given that network traffic inherently possesses underlying topology, it can be naturally expressed as nodes and edges, leveraging network flow information for improved botnet detection. GNNs are particularly well-suited for this task, as they excel in capturing and learning from graph-structured data, allowing for a more comprehensive representation of complex and interconnected network behavior.

One of the key advantages of GNNs in botnet detection lies in their ability to handle the sequential nature of botnet attacks. Botnets often exhibit dynamic and evolving behaviors over time, which conventional methods struggle to model effectively. GNNs, however, can inherently capture temporal dependencies by aggregating information from neighboring nodes in sequential order. This sequential nature of botnet attacks can be depicted in the architecture of GNNs, facilitating the detection of sophisticated and evasive botnet activities in real-time network traffic. The incorporation of GNNs into the domain of botnet detection opens up promising strategies for enhanced accuracy and adaptability, which otherwise are crucial in countering ever-evolving cyber threats in IoT networks.

In this chapter, we propose a novel sequential Gated Graph Convolutional Neural Network (GGCN) framework specifically designed for botnet detection in IoT network environments that takes sequential network flows as input and learns the relation between them using Gated mechanism. By harnessing the capabilities of GNNs in representing network traffic as graph structures and handling the sequential dynamics of botnet attacks, we aim to improve the accuracy and efficiency of attack detection in emerging IoT botnet threats.

The proposed GGCN architecture allows the model to iteratively update and refine node embeddings based on information propagated from neighboring nodes and edge attributes representing network traffic, enabling a deeper understanding and analysis of sequential network behavior. The proposed intrusion detection model presents the key contributions aimed at enhancing botnet detection in IoT network environments as below:

- We propose a time-stamped multi-edge graph structure representing network traffic flows provided by edge attributes, enabling to discern the temporal dynamics of bot network. This unique feature enables the model to uncover hidden patterns and dependencies in the data, leading to more accurate and timely detection of sophisticated botnet activities.
- We propose a sequential graph learning framework that comprises of time-sequenced edges and features in a multi-edged graph structure, and a two-layered gated graph model with modified message-passing layers and aggregation functions to effectively handle time-series traffic characteristics. This novel design provides the model with a deeper understanding of relation of edges in time underlying network topology. By efficiently capturing complex dependencies and patterns in the graph data, our model can better distinguish different

types of attacks and make accurate predictions.

We demonstrate the efficacy of our approach in capturing the intricacies of botnet behavior and its superiority over conventional deep learning and machine learning techniques for IoT botnet detection through extensive experimental evaluations on diverse datasets. Our work contributes to the growing body of research on applying GNNs to cybersecurity and reinforces the significance of topology-aware approaches in combating the evolving landscape of cyber attacks.

This study diverges from our earlier research endeavors, where we introduced various graph structures for NIDS in IoT networks. For instance, in [114], we integrated both node and edge attributes within a directed graph structure. Each node represented an application layer process through the combination of IP:Port# pairs, while edge attributes conveyed network layer characteristics and node attributes captured application layer attributes. In [115], we escalated the intricacy of the graph structure to encompass multi-edges with multi-dimensional edge features. Here, IoT devices were portrayed as nodes, and inter-node communication was depicted through multiple edges, each possessing corresponding features. In this contribution, we propose a temporal sequencing of multiple edges to capture the sequential dynamics of botnet attacks. Additionally, we extend the original GGCN layers to encompass temporally sequenced multiple edge features within the message passing and aggregation mechanisms.

The research questions are as follows.

- How does the sequential nature of attacks manifest in graph structures, and what are the key temporal patterns and dependencies that characterize these attacks?

- What methodologies can be employed to represent the sequential dynamics of attacks in graph structures, considering both node and edge attributes, to create a sequential graph?

- How do different graph-based models, such as GGCNs, perform in capturing the sequential nature of attacks and detecting anomalous behavior in network traffic represented as sequential graphs?

5.2 Related Work

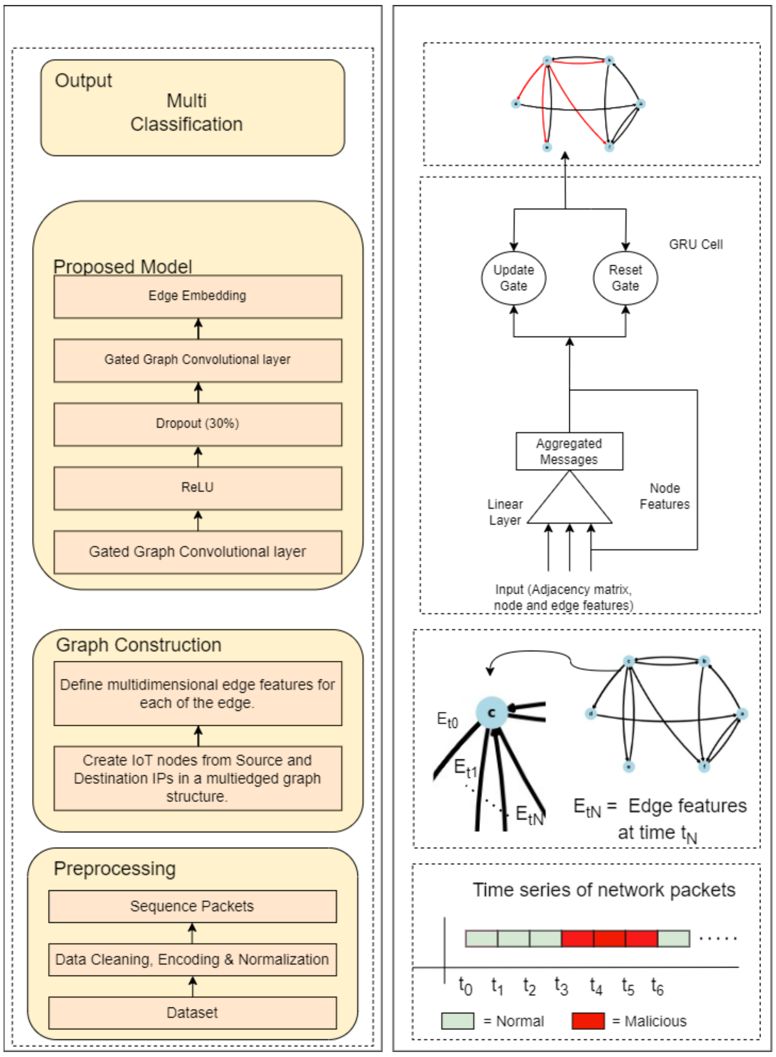
In recent times, various GNN architectures have been employed for network-related tasks, particularly in the context of IDS and anomaly detection. In [66], the authors propose a deep learning-based NIDS using a newly emerging graph neural network. They generate a network flow graph where information is passed on the edges, modifying the original GraphSage by incorporating an edge aggregate function and using Softmax for prediction during backward propagation. The results demonstrate that their GraphSage-based NIDS outperforms non-GNN approaches in four out of six datasets. Similarly, in [67], the focus is on designing graph representations of network flows to capture meaningful structural flow patterns, aiming to develop robust and accurate NIDS. They propose a message-passing function for efficient learning from host connection graphs. Evaluation on the CIC-IDS2017 dataset reveals that the proposed GNN model maintains baseline accuracy even when subjected to artificially altered traffic flows, whereas other models show degraded performance. In [68], the authors propose a graph-based distributed anomaly detection scheme for monitoring the entire Multi-Agent System (MAS) framework. They construct a graph structure by transforming flow-level information into node-level representation. Finally, in [70], the authors propose a Peer-To-Peer (P2P) botnet detection framework utilizing node and topological features. However, the experimental results show a high False Positive Rate (FPR), indicating a need for improvement in the model's performance. Overall, these studies demonstrate the potential of GNN-based approaches in network Traffic analysis for intrusion detection. While they show promising results, challenges related to sequential attacks, multi classification need further exploration.

The existing literature consists of machine and deep learning approaches for the sequential network traffic analysis. In [82], a pre-trained CNN model is employed to extract features from processed data streams. An optimized deep autoencoder (DAE) is used to capture temporal changes in the surveillance stream's actions. They then trained their model using a quadratic SVM to classify human activities. In another study [83], Recurrent neural networks (RNN) is used to learn from the previous time-steps in the datasets. A deep learning model based on LSTM (Long Short-Term Memory) and FCN (Fully Connected Network) is designed for a multi-classification of malicious connections in intrusion datasets. in [88], Gray Wolves Optimization (GWO) algorithm in conjunction with a wrapper feature selection technique is used to optimize the binary feature space. They introduced time-variant transfer function that adapts GWO for enhanced botnet detection accuracy in IoT environments. A novel LSTM-CNN-based method is introduced in [84] for insider threat detection. The method involves initial extraction of temporal behavioral features by feeding single-day user action sequences into the LSTM model. Subsequently, the extracted features are transformed into fixed-sized feature matrices, which are then used in a CNN-based classification model for the final detection of insider threats. In [85], a Gated Recurrent Unit based on Bidirectional Weighted Feature Averaging (GRU-BWFA) classifier is designed for effectively detecting Distributed Denial of Service (DDoS) attacks and capturing the time series events.

Recent papers have explored the integration of GNNs in botnet detection tasks due to their ability to capture botnet topology. In this study [78], an end-to-end data-driven approach is utilizing GNNs to detect Peer-To-Peer (P2P) botnets. To comprehensively evaluate the automated detection method, synthetic or real botnet topologies are overlaid with diverse communication patterns on large-scale real background traffic graphs, generating datasets for analysis. In [86], a method BD-GNNExplainer (GNN-based botnet detection) is proposed to evaluate the trustworthiness of GNN-based botnet detection models. BD-GNNExplainer extracts the most contributing data to the GNN's decision, quantifying a score that expresses interpretability, ultimately guiding model optimization and providing a guideline to enhance the understandability of the botnet detection methodology. In another paper [87], a GNN based botnet detection model is proposed to overcome over-smoothing issue and enhance network forensics. Furthermore, GNNExplainer and saliency maps are integrated to identify suspicious network flows and botnet nodes, enhancing the transparency and interpretability of the detection process for automatic network forensics – features that are lacking in existing botnet detection literature.

The related work in botnet detection using GNNs has demonstrated notable advancements in capturing graph patterns and improving detection accuracy. However, a significant gap in the existing literature lies in the lack of exploration regarding the utilization of Gated Graph Convolution neural network for detecting botnets in network traffic. It is a powerful variant of GNN that excels in capturing the sequential nature of botnet activities, which is essential for detecting dynamic and evolving botnet behaviors in real-time network traffic. This specific gap in the research indicates an unexplored avenue to leverage the sequential information inherent in botnet activities through Gated GNN based models. By introducing into the botnet detection domain, the interoperability and effectiveness of the botnet detection process can improve, leading to more robust and efficient methods for combating evolving botnet threats in dynamic

IoT network environments.



Output
Multi Classification
Proposed Model
Edge Embedding
Gated Graph Convolutional layer
Dropout (30%)
ReLU
Gated Graph Convolutional layer
Graph Construction
Define multidimensional edge features for each of the edge.
Create IoT nodes from Source and Destination IPs in a multiedged graph structure.
Preprocessing
Sequence Packets
Data Cleaning, Encoding & Normalization
Dataset
GRU Cell
Update Gate
Reset Gate
Aggregated Messages
Linear Layer
Node Features
Input (Adjacency matrix, node and edge features)
E_{t0}
E_{t1}
E_{tN}
E_{tN} = Edge features at time t_N
Time series of network packets
t_0 t_1 t_2 t_3 t_4 t_5 t_6
🟩 = Normal
🟥 = Malicious

Figure 5.1: Proposed GGCN Model: Textual representation (left) and visual representation (right). The GGCN integrates graph convolutional layers with gated graph attention mechanisms to capture local and global graph structures effectively. Directional arrows illustrate the data flow through the framework's stages.

5.3 The Proposed Framework

This section presents a detailed description of the proposed GNN-based botnet detection. Our proposed system incorporates a time-series multi-edged-graph structure and a GGNN model for learning the complex patterns in time and underlying topology, thus paving ways for enhanced botnet detection. The overall framework comprises three stages, namely, Data Preprocessing, Graph Construction, Proposed Model and the Output, as illustrated in Fig.5.1. Table 5.1

summarizes the notations and their definitions used in the chapter.

Notation	Definition
$\mathcal{A}$	The adjacency matrix
$\mathcal{E}$	The set of edges
$\|\mathcal{E}\|$	The number of edges
e_{ijr}	The r-th edge from v_i to v_j
$F_{\mathcal{V}}$	The node feature matrix
$F_{\mathcal{E}}$	The edge feature matrix
$\mathcal{G}_m$	The multigraph
$\mathcal{N}_{v_i}$	The multi-edge neighborhood of node v_i
R_{ij}	Total number of edges from v_i to v_j.
$\mathcal{U}$	Non-linear activation function
$\mathcal{V}$	The set of nodes
$\|\mathcal{V}\|$	The number of nodes
W^{gru}	The GRU weight matrix

Table 5.1: Notations used in the chapter and their definitions

5.3.1 Data Preprocessing

The data pre-processing utilizes existing knowledge and techniques to streamline the proposed research process and build on previous work [76], [114], [115]. Additionally, the datasets are converted into time series Firstly, the source and destination IP addresses are transformed into small integer values replacing each unique IP address with a unique integer to save the computational load due to long IP addresses originally present in the dataset and to make the learning process independent of the fixed IP addresses. Data is then cleansed from Null and Nan values to avoid ambiguity due to unknown values. The feature encoding is done using a categorical encoder for the categorical features, and a standard scalar for the numerical features in order to normalize them. The dataset is now ready to be moved into stage 2 of the proposed framework.

5.3.2 Graph Construction

To model an IoT communications graph containing botnet traffic, we construct a multi-graph $\mathcal{G}_m$ to represent the topology of an IoT network. The graph comprises nodes $\mathcal{V}$ representing IoT devices and edges $\mathcal{E}_T$ representing device communication at time $T = [t_0, t_1, \dots, t_T]$. Formally, we define $\mathcal{G}_m = (\mathcal{V}, \mathcal{E}_T, F_\mathcal{V}, F_{\mathcal{E}_T})$, where $\mathcal{V} = (v_1, \dots, v_N)$ is a set of N IoT nodes identified by their IP addresses, and $\mathcal{E}_T = \{e_{ijr}\}$ collects $r \in [1, R_{ji}]^T$ number of edges representing network flows between pairs of IoT nodes over time, and R_{ji} is the maximum number of edge. The network flows in $\mathcal{E}_T$ are time-series, capturing the temporal dynamics of communications between devices.

Let $F_\mathcal{V}$ and $F_\mathcal{E}$ denote the feature matrices associated with nodes and edges, respectively, to characterize their properties within the constructed graph. The feature matrix $F_\mathcal{V}$ is $N \times K$ dimensional ($F_\mathcal{V} \in \mathbb{R}^{N \times K}$), where K represents K-dimensional node features. The feature matrix $F_{\mathcal{E}_T}$ represents the edge features of the sequential network flows and has dimensions of $|\mathcal{E}_T| \times K$ ($F_{\mathcal{E}_T} \in \mathbb{R}^{|\mathcal{E}_T|, K}$). Here, $| \cdot |$ denotes the cardinality of the multi-edge set $\mathcal{E}_T$, and K represents the number of features associated with each edge. Specifically, the features of edge e_{ijr} are collected in the matrix $F_{e_{ijr}} \in \mathbb{R}^{1,K}$. In contrast, the node features are represented by matrices of all ones, matching the dimensions of the edge features, which ensures consistent dimensions and preserves the significance of edge feature values, such as traffic flows, during the graph embedding process. In this representation, the graph $\mathcal{G}_m$ captures the evolving nature of IoT device communication by incorporating time series network flows, while the feature matrix $F_{\mathcal{E}_T}$ provides additional information about the characteristics and properties of time-series edges in the IoT network.

The adjacency matrix $\mathcal{A}$ of a multigraph $\mathcal{G}m$ is an $N \times N$-dimensional matrix. The matrix elements correspond to the multiplicity of existing edges over time. For the element at the i-th row and j-th column, a_{ij}, the following conditions apply: If there is no edge from node v_i to v_j at any time step, i.e., $e_{ijr} \notin \mathcal{E}_T$ for all $r \in [1, R_{ji}]^T$, then $a_{ij} = 0$. If there are R_{ij} edges from node v_i to v_j over time, then $a_{ij} = R_{ij}$. Furthermore, the multi-edge neighborhood of node v_i, denoted by $\mathcal{N} v_i$, is defined as the set of all nodes connected to v_i through any of its edges and can be expressed as $\mathcal{N}_{v_i} = \{v_j | e_{ijr} \in \mathcal{E}_T\}$, where $r \in [1, R_{ji}]^T$.

This graph considers the time series nature of the edge features in the adjacency matrix $\mathcal{A}$, capturing the evolving connectivity patterns of the multigraph $\mathcal{G}_m$ over multiple time steps.

5.3.3 Proposed GNN Model

The key feature of the proposed botnet detection model is its ability to cater for time-series traffic characteristics in its architecture with modified message-passing layers and aggregation functions. We propose a graph structure that depicts a time series of network flows and a novel GGCN architecture with improved message passing mechanism to include sequential multi-edged features in the proposed GNN model as given in Algo. 3. Our model effectively exploits the underlying topology of network traffic, represented as nodes and edges, to address the unique challenges of botnet detection in IoT environments. This sequential analysis aligns well with the inherent nature of botnet activities in network traffic.

The proposed botnet detection model comprises two GGCN layers to learn the meaningful representations from the underlying topology, and the sequential relationships in edges by considering the corresponding edge features. The layered architecture of the proposed GGCN model is illustrated in Fig. 5.1 and is expressed as:

$$\mathcal{G}_m(F_v, F_{\mathcal{E}_T}, \mathcal{A}) = \mathcal{A}F_{\mathcal{V}}F_{\mathcal{E}_T}\mathcal{U}\left(\mathcal{A}F_{\mathcal{V}}F_{\mathcal{E}_T}W_1^{gru}\right)W_2^{gru}, \tag{5.1}$$

where, W_1^{gru} and W_2^{gru} are the weight matrices of the hidden layers 1 and 2, respectively, and are trained using the gradient descent. $W_1^{gru} \in \mathbb{R}^{K,H_1}$, and, K is the dimension of node and edge features and H_1 is the dimension of the first hidden layer. Similarly, $W_2^{gru} \in \mathbb{R}^{H_1,O_2}$, and, O_3 is the dimension of the third layer. $\mathcal{A}$ is the sparse adjacency matrix and can be represented as $\mathcal{A} \in \mathbb{R}^{KN\times 2KN}$. $\mathcal{A}$ belongs to the set of real numbers ($\mathbb{R}$) and has a shape of $KN \times 2KN$. $\mathcal{U}$ denotes a non-linear activation function, i.e., ReLU [92]. For a matrix $\mathbf{H}$, ReLU can be given by

$$\begin{aligned} &\text{ReLU}(\mathbf{H}) = \max(0, \mathbf{H}) \\ &\text{i.e.,} \\ &\text{ReLU}(H_{ij}) = \max(0, H_{ij}) = \begin{cases} H_{ij}, & \text{if } H_{ij} > 0 \\ 0, & \text{if } H_{ij} < 0. \end{cases} \end{aligned} \tag{5.2}$$

Equation (5.1) summarizes the forward pass of two-layered GGNN propagation that takes as input the node features $F_{\mathcal{V}}$, adjacency matrix $\mathcal{A}$ and, the time dependent edge features $F_{\mathcal{E}_T}$. In the first layer, the proposed model learns from the underlying graph structure and network traffic information by concatenating the features of neighboring nodes and the corresponding edge features, and passes them through a linear layer. The aggregated messages are combined with the current node embeddings using a Gated Recurrent Unit (GRU) cell. This process iterates until the desired number of layers is reached. The number of layers in the GGCN determines the depth and complexity of the model's architecture. Each layer allows the network to iteratively update and refine the node embeddings based on the information propagated from neighboring nodes and the edge attributes representing network traffic. The activation function $\mathcal{U}$ (e.g., ReLU) applied in the first layer helps introduce non-linearity and capture complex relationships in the data. In the second layer, both the updated node features from the first layer and the edge features continue to contribute to the overall graph embedding. The edge features contain information about the relationships and attributes associated with the connections between nodes, such as network traffic patterns or communication characteristics. These edge features provide additional context and fine-grained details to refine the node embeddings. The node features, on the other hand, are set as all ones. They may act as a bias or constant term during the graph convolution operations. Combining the updated node features and edge features in the second layer allows the GGCN to capture and integrate both the global graph structure and the local edge-level information. Since the edge attributes contain time-dependent information (e.g., time-stamped network traffic flows), the model can capture the temporal dynamics of the network. Overall, the model can learn representations that capture the complex interplay between nodes and their connections, enhancing its ability to understand and analyze sequential network behavior.

Algorithm 3: Sequential GGCN Algorithm

```
Input: Graph $\mathcal{G}_m$
Multi-edge adjacency matrix $\mathcal{A}$
Node features matrix $F_{\mathcal{V}}$
Edge features $F_{\mathcal{E}_T}$
Output: Edge embedding $z_{ijr}, \forall e_{ijr} \in \mathcal{E}_T$

▷Initialization
1 $\tilde{h}_{v_i} \leftarrow F_{v_i}, \forall v_i \in \mathcal{V}$
2 $\tilde{z}_{ijr} \leftarrow F_{e_{ijr}}, \forall e_{ijr} \in \mathcal{E}_T$
▷Node Embedding
3 for $\forall v_i \in \mathcal{V}$ do
4     $\tilde{h}_{v_i} \leftarrow \tilde{h}_{v_i} \,\|\, \mathbf{0}$
5     for $l \in num.layers$ do
6         for $\forall v_j \in \mathcal{N}_{v_i} \bigcup v_i$ and $r \in [1, R_{ji}]^T$ do
7             $\mathbf{m}_{v_i}^{(l)} \leftarrow \phi(\tilde{z}_{ijr}^{(l)} \| \tilde{h}_{v_j}^{(l)})$
8         $\tilde{h}_{v_i}^{(l)} \leftarrow \mathrm{GRU}(\mathbf{m}_{v_i}^{(l)}, \tilde{h}_{v_i}^{(l)})$
9 Apply dropout: $\tilde{h}_{v_i}^{(l)} \leftarrow \mathrm{Dropout}(\tilde{h}_{v_i}^{(l)}, 0.3)$
▷Edge Embedding
for $\forall v_i \in \mathcal{V}$ do
    for $r \in [1, R_{ji}]^T$ do
        $\tilde{z}_{ijr} \leftarrow \mathrm{MLP}(\tilde{h}_{v_i} \,\|\, \tilde{h}_{v_j})$
return $z_{ijr}$
```

The input of Algo. 3 consists of a multi-edge graph $\mathcal{G}_m$ with node features $F_{\mathcal{V}}$ set to all ones $\{1, \ldots, 1\}$ for each node and edge features $F_{\mathcal{E}_T}$ to R_{ij} parallel edge features. Steps 1 and 2 are the initialization steps and copy node and edge features into the first hidden states.

The node embedding spans Step 3 through to Step 8. In Step 4, the node features are combined with a zero vector to initialize the hidden states for all the nodes in the graph. Then, in Steps 7 and 8, the proposed model iterates over the specified number of layers to create messages $\mathbf{m}_{v_i}^{(l)}$ for all the nodes and multiple edges in the graph. In each layer, the node features are transformed using a weight matrix, followed by message passing and aggregation operations. The message passing in Step 7 concatenates the features of neighboring nodes and the corresponding edge features, and passes them through a linear layer. The aggregated messages are combined with the current node embeddings using a GRU cell as shown in Step 8. The hidden states are updated using the GRU operation, incorporating both the aggregated messages and the previous hidden states in Step 9. This process iterates until the desired number of layers ($num.layers$) is reached to capture complex dependencies and patterns in the graph data. Step 9 applies dropout technique [91] between two GGCN layers in order to remove overfitting during the learning process.

The edge embeddings are retrieved from the concatenation of neighbouring node embeddings as shown in Steps 11 and 12, that returns all edges between them and, MLP [102] is applied to learn more expressive and informative representations of edges. This transformed representation of the edge features is utilized for edge classification tasks. The MLP captures higher-level features and interactions between the connected nodes, which provides valuable information for distinguishing different types of edges or making predictions about their properties or labels.

5.3.4 Performance Evaluation Metrics

The multi-classification task at hand is effectively tackled by the proposed model. To assess the model's performance in detecting attacks, the evaluation comprises of the following classification metrics:

- **Accuracy:** This metric quantifies the proportion of accurately classified observations among all the data instances.
- **Precision:** This metric quantifies the precision of the model in accurately predicted positive instances out of all the instances predicted as positive.
- **Recall:** This metric quantifies the ratio of correctly predicted True Positives by a model among all the actual positive instances.
- **F1-Score:** The F1-Score is a metric that combines Precision and Recall, commonly utilized in scenarios involving imbalanced data.
- **False Alarm Rate (FAR):** The False Alarm Rate is a metric that evaluates the number of normal data points that are incorrectly classified as malicious.

5.3.5 Datasets

The evaluation of the proposed model involves the utilization of two prominent datasets, namely BoT IoT [103] and Mirai [116]. These datasets hold significant prominence in the domain of botnet detection considering IoT network traffic.

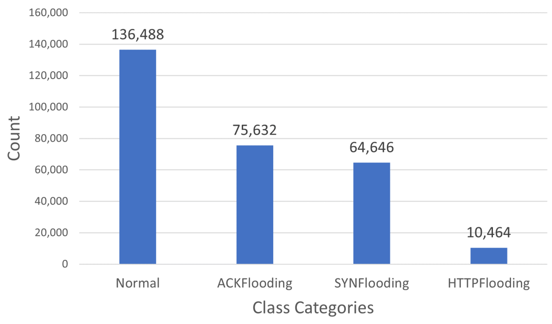


Figure 5.2: A graph showing categories of network traffic instances and their count in Mirai dataset.

The Mirai [116] dataset comprises multiple classes of network traffic instances, with Normal traffic representing the majority at 136,488 instances. It also includes instances of ACKFlooding (75,632), SYNFlooding (64,646), and HTTPFlooding (10,464), each showcasing distinct patterns of network activity as shown in Fig. 5.2. This distribution offers an opportunity to explore and analyze various types of IoT network flooding attacks, along with a substantial representation of normal traffic, facilitating the development and evaluation of effective intrusion detection strategies.

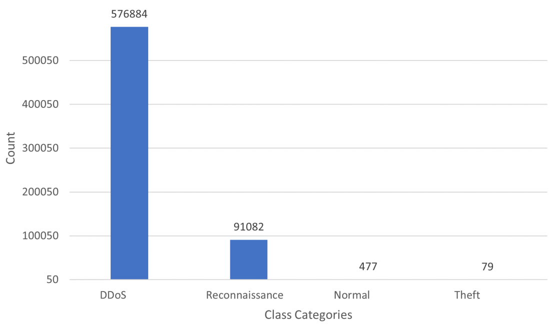


Figure 5.3: A graph showing categories of network traffic instances and their count in BoT IoT dataset.

The BoT IoT [103] dataset comprises diverse classes of network traffic instances, including 576,884 instances of DDoS attacks, 91,082 instances of Reconnaissance activities, 477 instances classified as Normal traffic, and 79 instances categorized as Theft events as shown in 5.3. This distribution showcases the prevalence of DDoS attacks and Reconnaissance activities, while highlighting the relatively limited occurrence of Normal and Theft classes. This dataset provides a valuable opportunity to explore the detection and classification of botnet attacks in

IoT traffic, with a focus on modeling sequential patterns and capturing the nuanced behavior of different attack types.

5.3.6 Experimental Setup

Prior to model evaluation, the datasets undergo preprocessing and transformation into multi-edged graphs. Each graph is enriched with nodes, edges, and associated features. Comprehensive details pertaining to each knowledge graph, including dataset sizes prior to knowledge graph construction, are tabulated in Table 5.2. To facilitate the evaluation process, the knowledge graphs are mapped to split the edges, allocating 70% for training and the remaining 30% for testing. The experiments are conducted on Google Colab, employing server specifications en-

Datasets	Total Data	No. of Nodes	No. of Edges	Ratio
Mirai [116]	201061	11536	201061	7:3
BoT IoT [103]	659363	84	659363	7:3

Table 5.2: Dataset description in terms of total data entries, number of malicious and normal data entries, and the no. of nodes and edges in each graph.

compassing an Intel(R) Xeon(R) CPU @ 2.00GHz, Python version 3.10.12, Pytorch Geometric version 2.4.0, and CUDA 118.

5.3.7 Results and Discussion

We begin by presenting the results of our analysis focused on binary classification. This involves distinguishing between normal attacks and malicious network activity. Next, we extend the experiment derive results from our experiments involving multiclass classification. Here, the focus shifts to identifying various specific types of attacks associated with each network flow.

Binary Classification results

Table 5.3 showcases the performance achieved by our proposed GGCN model as binary classifier by comprehensively displaying the results of each metric for the two benchmark datasets under consideration. The GGCN model demonstrates remarkable performance on the Mirai dataset, achieving an accuracy of 99.99%. This underscores its competence in distinguishing between

Datasets	Accuracy%	Precision	Recall%	F1-Score	FPR	AUC ROC
Mirai	99.99	0.9999	99.99	0.9999	0.0001	1.00
BoT IoT	99.25	0.9993	99.25	0.9956	0.0075	1.00

Table 5.3: Performance evaluation of the proposed GGCN as a binary classifier.

benign and malicious network activities. The model achieves high precision and recall rates of 0.9999 and 99.99%, respectively, along with F1-Score of 0.9999 with an FPR of 0.0001. The results indicate model's robustness in minimizing false alarms. The AUC ROC score of 1.00 substantiates its effective class discrimination. Similarly, when applied to the BoT IoT dataset, the GGCN model maintains its efficacy as a binary classifier, achieving an accuracy of 99.25%. It showcases precision and recall rates of 0.9993 and 99.25%, respectively, underlining its skill in identifying true positives and capturing a substantial proportion of actual positive instances. Although the FPR is relatively higher at 0.0075 compared to the Mirai dataset, indicating a slightly elevated false positive tendency, the AUC ROC score of 1.00 still attests to the model's robust discriminatory capacity.

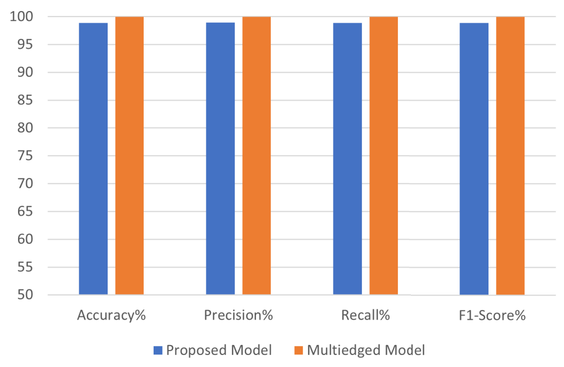


Figure 5.4: Performance evaluation of the proposed GGCN with other state-of-the-art GNN models on BoT IoT dataset.

Fig. 5.4 presents results from a binary classification task comparing the proposed model and the Multiedged model [115] on the BoT IoT dataset. The dataset, characterized by a skewed distribution of instances across attack types. The proposed model achieves a notable accuracy of 98.88%, showcasing its proficiency in distinguishing between normal and malicious traffic. This accuracy underscores the model's effectiveness in distinguishing between normal and malicious traffic instances. The proposed model achieves a notable accuracy of 98.88%, showcasing its proficiency in distinguishing between normal and malicious traffic. Precision, recall, and F1-Score values for the proposed model are consistently high, standing at 98.97%, 98.88%, and 98.90%, respectively. While the Multiedged model outperforms in terms of accuracy, it is essential to recognize that the proposed model's performance metrics are strong and competitive. Further investigation and potential fine-tuning may reveal insights into enhancing the proposed model's accuracy, aiming to close the performance gap of 1.08% with the Multiedged model in the binary classification.

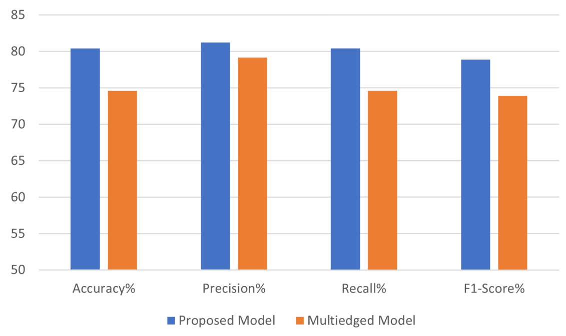


Figure 5.5: Performance evaluation of the proposed GGCN with other state-of-the-art GNN models on Mirai dataset.

Fig. 5.5 presents the comparative analysis on the Mirai dataset. The proposed model demonstrates a superior overall performance. With an accuracy of 80.40%, outperforming the Multiedged model, which achieves an accuracy of 74.58%. The precision of the proposed model (81.21%) is also higher than that of the Multiedged model (79.17%), indicating a better ability to correctly classify instances, while the recall of the proposed model (80.40%) surpasses that of the Multiedged model (74.58%), highlighting a stronger capability to capture instances of interest.

Furthermore, the F1-Score of the proposed model (78.87%) underscores its balanced precision and recall, surpassing the Multiedged model's F1-Score of 73.88%.

In the context of capturing the sequential nature of attacks, the proposed model exhibits enhanced performance metrics, indicating its effectiveness in discerning nuanced patterns within the Mirai dataset. This comparative analysis supports the proposition that the proposed model, with its numerical superiority, is better suited for capturing the sequential attack nature in the context of IoT network traffic, outperforming the Multiedged model in this specific task.

Multiclass Classification results

Datasets	Accuracy %	Precision	F1-Score	Recall %	FAR
BoT IoT	98.86	0.9895	0.9888	98.86	0.0114
Mirai	76.70	0.9002	0.8170	76.70	0.2330

Table 5.4: Performance evaluation of the proposed GGCN as a multiclassifier.

Table 5.4 shows that our botnet detection model on shows promising performance on different datasets. For the BoT IoT dataset, the GNN model achieves an accuracy of 98.86%, indicating that it can accurately classify normal and different attack types in the network traffic data. The precision of 0.9895 indicates that the model has a high ability to correctly identify true positive instances among the predicted positive cases. The F1-score of 0.9888 suggests a good balance between precision and recall. The recall of 98.86% indicates that the model effectively captures true positive instances. However, the False Alarm Rate (FAR) of 0.0114 shows that there is still a small proportion of false positive predictions.

Our model achieves an accuracy of 76.70% on Mirai dataset. The precision of 0.9002 suggests that the model identifies a considerable number of true positive instances. The F1-score of 0.8170 indicates a trade-off between precision and recall. The recall of 76.70% indicates that the model captures a significant portion of true positive instances. The relatively high FAR of 0.2330 indicates a proportion of false positives in the predictions.

5.4 Conclusion

In this chapter, we have proposed a novel intrusion detection model to enhance botnet detection in IoT network environments. The key contributions of our research lie in the following aspects:

- We introduced a sequential architecture with modified message-passing layers and aggregation functions, enabling effective time-series traffic analysis. This capability allows our model to capture and analyze the temporal dynamics of network flow data, leading to real-time detection of dynamic and evolving botnet behaviors.

- We developed a GGCN architecture with multi-edge features, providing the model with a deeper understanding of the underlying multi-graph structure of the network topology. By efficiently capturing complex dependencies and patterns in the graph data, our model can better distinguish different types of edges and make accurate predictions.
- Our model effectively leverages GGCN's superior capability in handling graph-structured data, specifically addressing the unique challenges of botnet detection in IoT environments. The sequential analysis aligns well with the inherent nature of botnet activities in network traffic.
- We incorporated temporal dependency modeling, utilizing time-stamped network traffic flows from edge attributes to discern the temporal dynamics of the network. This unique feature enables our model to uncover hidden patterns and dependencies in the data, leading to more accurate and timely detection of sophisticated botnet activities.
- Extensive evaluations demonstrated that our GGCN-based intrusion detection model outperforms traditional deep learning and machine learning approaches, achieving higher accuracy and precision in detecting botnet attacks. Overall, our proposed model presents a robust and efficient solution for sequential botnet detection in IoT networks, contributing to enhancing the security and resilience of IoT environments.

Chapter 6

Conclusion and Future Works

6.1 Conclusion

In conclusion, this thesis set out to explore the potential of GNNs as a non-conventional deep learning technique for enhancing IDS. The research objectives were achieved through a comprehensive investigation into the limitations of conventional machine learning and deep learning techniques in capturing network topology information and the evaluation of GNNs' capabilities to address these challenges.

The findings of this study reveal that conventional techniques often fall short in effectively capturing the intricate network topology crucial for robust intrusion detection. In contrast, GNNs demonstrated their suitability by leveraging graph structures designed to represent the nature of intrusion detection problems. The developed GNN-based architectures showed promising performance improvements in accurately detecting intrusions and adapting to diverse intrusion patterns on benchmark datasets. The comparative analysis against conventional techniques demonstrated the superiority of GNNs in handling network topology and addressing the intricacies of intrusion detection.

Overall, the significance of this research lies in its contribution to advancing the field of intrusion detection, offering valuable insights into the potential of Graph Neural Networks as the next generation of IDS models. The demonstrated effectiveness, scalability, and interpretability of GNN-based solutions provide practitioners and researchers with a promising framework for developing more accurate and adaptive intrusion detection systems, ultimately strengthening cybersecurity measures in an increasingly interconnected and vulnerable digital landscape.

6.2 Future Works

There are several avenues for future work and research that can further enhance and extend the findings of this thesis.

1. Enhanced Graph Representation: Investigate advanced techniques to improve the representation of network data as graphs. This could involve incorporating temporal information, handling dynamic and evolving networks, and exploring graph augmentation strategies to handle missing or noisy data effectively.

2. Heterogeneous Graphs: Explore the application of GNNs on heterogeneous graphs, where nodes and edges may have different types and attributes. Develop techniques to handle the complexity of diverse network entities and relationships in intrusion detection scenarios.

3. Transfer Learning and Few-shot Learning: Investigate techniques for transferring knowledge from well-labeled datasets or adapting GNN-based NIDS to new, unseen network environments with limited labeled data. Few-shot learning approaches could be explored to train the model with minimal samples from new attack types.

4. Adversarial Robustness: Evaluate the robustness of GNN-based NIDS against adversarial attacks, developing defense mechanisms to mitigate the impact of adversarial perturbations and ensuring the reliability of the system in real-world settings.

5. Federated Learning for Distributed NIDS: Explore the application of federated learning to train GNN-based NIDS across multiple distributed network devices or organizations, enabling collaborative intrusion detection while maintaining data privacy.

6. Combination with Traditional Techniques: Investigate the integration of GNN-based NIDS with traditional signature-based or anomaly-based detection methods to create hybrid systems that leverage the strengths of both approaches.

Bibliography

[1] E. Bertino, K. K. R. Choo, D. Georgakopolous, and S. Nepal, "Internet of things (IoT): Smart and secure service delivery," *ACM Transactions on Internet Technology*, vol. 16, no. 4, pp. 1–8, 2016, ISSN: 15576051. DOI: 10.1145/3013520.

[2] H. Ahmetoglu and R. Das, "A comprehensive review on detection of cyber-attacks: Data sets, methods, challenges, and future research directions," *Internet of Things*, vol. 20, p. 100 615, 2022, ISSN: 2542-6605. DOI: https://doi.org/10.1016/j.iot.2022.100615.

[3] A. Thakkar and R. Lohiya, *A Review on Machine Learning and Deep Learning Perspectives of IDS for IoT: Recent Updates, Security Issues, and Challenges.* Springer Netherlands, 2021, vol. 28, pp. 3211–3243, ISBN: 0123456789. DOI: 10.1007/s11831-020-09496-0.

[4] T. Altaf and R. Braun, "A roadmap to smart homes security aided sdn and ml," in *2022 5th Conference on Cloud and Internet of Things (CIoT)*, 2022, pp. 129–136. DOI: 10.1109/CIoT53061.2022.9766525.

[5] X. Wang, K. Zheng, X. Niu, B. Wu, and C. Wu, "Detection of command and control in advanced persistent threat based on independent access," in *2016 IEEE International Conference on Communications (ICC)*, 2016, pp. 1–6. DOI: 10.1109/ICC.2016.7511197.

[6] C. Wu, K. Zheng, X. Wang, X. Niu, and T. Lu, "Computing adaptive feature weights with pso to improve android malware detection," in *Security and Communication Networks*, 2017, p. 14. DOI: 10.1155/2017/3284080.

[7] H. Yixun, K. Zheng, X. Wang, and Y. Yang, "Worm-hunter: A worm guard system using software-defined networking," in *KSII Transactions on Internet and Information Systems*, vol. 11, 2016. DOI: 10.1155/2017/3284080.

[8] X. Wang, K. Zheng, X. Niu, B. Wu, and C. Wu, "Detection of command and control in advanced persistent threat based on independent access," in *2016 IEEE International Conference on Communications (ICC)*, 2016, pp. 1–6. DOI: 10.1109/ICC.2016.7511197.

[9] W. Li, T. Logenthiran, V. T. Phan, and W. L. Woo, "Implemented IoT-based self-learning home management system (SHMS) for Singapore," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2212–2219, 2018, ISSN: 23274662. DOI: 10.1109/JIOT.2018.2828144.

[10] T. Liang, B. Zeng, J. Liu, L. Ye, and C. Zou, "An unsupervised user behavior prediction algorithm based on machine learning and neural network for smart home," *IEEE Access*, vol. 6, pp. 49 237–49 247, 2018, ISSN: 21693536. DOI: 10.1109/ACCESS.2018.2868984.

[11] H. D. Mehr, "Human Activity Recognition in Smart Home With Deep Learning Approach," *2019 7th International Istanbul Smart Grids and Cities Congress and Fair (ICSG)*, pp. 149–153,

[12] F. D. Jivani, M. Malvankar, and R. Shankarmani, "A Voice Controlled Smart Home Solution with a Centralized Management Framework Implemented Using AI and NLP," *Proceedings of the 2018 International Conference on Current Trends towards Converging Technologies, ICCTCT 2018*, pp. 1–5, 2018. DOI: 10.1109/ICCTCT.2018.8550972.

[13] Suraj, I. Kool, D. Kumar, and S. Barma, "Visual Machine Intelligence for Home Automation," *Proceedings - 2018 3rd International Conference On Internet of Things: Smart Innovation and Usages, IoT-SIU 2018*, pp. 1–6, 2018. DOI: 10.1109/IoT-SIU.2018.8519915.

[14] F. Elizalde, "Hype Cycle for the Connected Home , 2018," Gartner, Tech. Rep. 30 July, 2018.

[15] P. K. Sharma, J. H. Park, Y. S. Jeong, and J. H. Park, "SHSec: SDN based Secure Smart Home Network Architecture for Internet of Things," *Mobile Networks and Applications*, vol. 24, no. 3, pp. 913–924, Jun. 2019, Publisher: Springer New York LLC, ISSN: 15728153. DOI: 10.1007/s11036-018-1147-3.

[16] M. Serror, M. Henze, S. Hack, M. Schuba, and K. Wehrle, "Towards in-network security for smart homes," *ACM International Conference Proceeding Series*, 2018. DOI: 10.1145/3230833.3232802.

[17] B. Krebs. "Mirai botnet." Accessed: [Insert Date]. ().

[18] Radware. "Brickerbot: The iot kill switch that ensures your devices stay off." Accessed: [Insert Date]. ().

[19] "Alert (ta18-145a) - russian state-sponsored cyber actors targeting network infrastructure devices." Accessed: [Insert Date], US-CERT. ().

[20] "Trickbot targets iot devices." Accessed: [Insert Date], Dark Reading. ().

[21] "Ripple20." Accessed: [Insert Date], JSOF. ().

[22] A. Alelaiwi, M. M. Hassan, and M. Z. A. Bhuiyan, "A secure and dependable connected smart home system for elderly," *Proceedings - 2017 IEEE 15th International Conference on Dependable, Autonomic and Secure Computing, 2017 IEEE 15th International Conference on Pervasive Intelligence and Computing, 2017 IEEE 3rd International Conference on Big Data Intelligence and Compu*, vol. 2018-Janua, pp. 722–727, 2018. DOI: 10.1109/DASC-PICom-DataCom-CyberSciTec.2017.126.

[23] A. Algarni, "A Survey and Classification of Security and Privacy Research in Smart Healthcare Systems," *IEEE Access*, vol. 7, pp. 101 879–101 894, 2019, ISSN: 2169-3536. DOI: 10.1109/access.2019.2930962.

[24] M. Talal, A. A. Zaidan, B. B. Zaidan, *et al.*, "Smart Home-based IoT for Real-time and Secure Remote Health Monitoring of Triage and Priority System using Body Sensors: Multi-driven Systematic Review," *Journal of Medical Systems*, vol. 43, no. 3, 2019, ISSN: 1573689X. DOI: 10.1007/s10916-019-1158-z.

[25] U. Mustafa, E. Pflugel, and N. Philip, "A Novel Privacy Framework for Secure M-Health Applications: The Case of the GDPR," *Proceedings of 12th International Conference on Global Security, Safety and Sustainability, ICGS3 2019*, pp. 1–9, 2019. DOI: 10.1109/ICGS3.2019.8688019.

[26] Y. Hamouda, "Optimal home energy management for smart home using random bit climbing," 2019.

[27] A. Srinivasan, K. Baskaran, and G. Yann, "IoT Based Smart Plug-Load Energy Conservation and Management System," *2019 IEEE 2nd International Conference on Power and Energy Applications (ICPEA)*, pp. 155–158, 2019. DOI: 10.1109/icpea.2019.8818534.

[28] H. Lin and N. W. Bergmann, "IoT privacy and security challenges for smart home environments," *Information (Switzerland)*, vol. 7, no. 3, 2016, ISSN: 20782489. DOI: 10.3390/info7030044.

[29] P. Langley and J. G. Carbonell, *Approaches to machine learning.* 1984, vol. 35, pp. 306–316, ISBN: 9781119454953. DOI: 10.1002/asi.4630350509.

[30] H. Huang, Z. Fang, X. Wang, Y. Miao, and H. Jin, "Motif-preserving temporal network embedding," *IJCAI International Joint Conference on Artificial Intelligence*, vol. 2021-Janua, pp. 1237–1243, 2020, ISSN: 10450823. DOI: 10.24963/ijcai.2020/172.

[31] M. K. Asif, T. A. Khan, T. A. Taj, U. Naeem, and S. Yakoob, "Network Intrusion Detection and its strategic importance," *BEIAC 2013 - 2013 IEEE Business Engineering and Industrial Applications Colloquium*, pp. 140–144, 2013.

[32] D. E. Denning, "An intrusion-detection model," in *IEEE Symposium on Security and Privacy*, 1986, pp. 118–118. DOI: 10.1109/SP.1986.10010.

[33] H. Gascon, A. Orfila, and J. Blasco, "Analysis of update delays in signature-based network intrusion detection systems," *Computers and Security*, vol. 30, no. 8, pp. 613–624, 2011, ISSN: 01674048. DOI: 10.1016/j.cose.2011.08.010.

[34] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Computers and Security*, vol. 28, no. 1-2, pp. 18–28, 2009, ISSN: 01674048. DOI: 10.1016/j.cose.2008.08.003.

[35] B. K. Mohanta, D. Jena, U. Satapathy, and S. Patnaik, "Survey on iot security: Challenges and solution using machine learning, artificial intelligence and blockchain technology," *Internet of Things*, vol. 11, p. 100 227, 2020, ISSN: 2542-6605.

[36] K. Kaur and J. Ayoade, "Analysis of ddos attacks on iot architecture," in *2023 10th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, 2023, pp. 332–337. DOI: 10.1109/EECSI59885.2023.10295766.

[37] H. Suo, J. Wan, C. Zou, and J. Liu, "Security in the internet of things: A review," in *2012 International Conference on Computer Science and Electronics Engineering*, vol. 3, 2012, pp. 648–651. DOI: 10.1109/ICCSEE.2012.373.

[38] M. S. E. Sayed, N.-A. Le-Khac, M. A. Azer, and A. D. Jurcut, "A flow-based anomaly detection approach with feature selection method against ddos attacks in sdns," *IEEE Transactions on Cognitive Communications and Networking*, vol. 8, no. 4, pp. 1862–1880, 2022. DOI: 10.1109/TCCN.2022.3186331.

[39] Z. Cekerevac, Z. Dvorak, L. Prigoda, and P. Čekerevac, “Internet of things and the man-in-the-middle attacks – security and economic risks,” *MEST Journal*, vol. 5, pp. 15–5, Jul. 2017. DOI: 10.12709/mest.05.05.02.03.

[40] J. Liu, C. Zhang, and Y. Fang, “EPIC: A Differential Privacy Framework to Defend Smart Homes Against Internet Traffic Analysis,” *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1206–1217, 2018, ISSN: 23274662. DOI: 10.1109/JIOT.2018.2799820.

[41] H. K. Singh, S. Verma, S. Pal, and K. Pandey, “A step towards Home Automation using IOT,” *2019 Twelfth International Conference on Contemporary Computing (IC3)*, pp. 1–5, 2019. DOI: 10.1109/ic3.2019.8844945.

[42] J. J. Padmini, A. A. Selva Jothi, and R. Harikrishnan, “Advancement Of Home Appliances For Home Automation Using Human Detection,” *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*, pp. 368–371, 2019. DOI: 10.1109/iceca.2019.8822140.

[43] A. Basu, A. Nandy, M. Biswas, R. N. Sharma, and R. Biswas, “Home Automation Using Energy Usage,” *2018 6th Edition of International Conference on Wireless Networks & Embedded Systems (WECON)*, pp. 73–78, 2019. DOI: 10.1109/wecon.2018.8782049.

[44] Sonia, “Recognizing Humans from Their Behavioral Patterns,” *2019 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2019*, pp. 718–721, 2019. DOI: 10.1109/PERCOMW.2019.8730570.

[45] M. Garcia-Constantino, J. Beltran-Marquez, D. Cruz-Sandoval, *et al.*, “Semi-Automated Annotation of Audible Home Activities,” *2019 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2019*, pp. 40–45, 2019. DOI: 10.1109/PERCOMW.2019.8730729.

[46] N. Koroniotis, N. Moustafa, and E. Sitnikova, “Forensics and Deep Learning Mechanisms for Botnets in Internet of Things: A Survey of Challenges and Solutions,” *IEEE Access*, vol. 7, pp. 61 764–61 785, 2019, ISSN: 21693536. DOI: 10.1109/ACCESS.2019.2916717.

[47] S. Hou and X. Huang, “Use of Machine Learning in Detecting Network Security of Edge Computing System,” *2019 4th IEEE International Conference on Big Data Analytics, ICBDA 2019*, pp. 252–256, 2019. DOI: 10.1109/ICBDA.2019.8713237.

[48] S. Ramapatruni, S. N. Narayanan, S. Mittal, A. Joshi, and K. Joshi, “Anomaly Detection Models for Smart Home Security,” *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart*

Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS), pp. 19–24, 2019. DOI: 10.1109/bigdatasecurity-hpsc-ids.2019.00015.

[49] G. Spanos, K. M. Giannoutakis, K. Votis, and D. Tzovaras, "Combining Statistical and Machine Learning Techniques in IoT Anomaly Detection for Smart Homes," *2019 IEEE 24th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pp. 1–6, 2019, ISSN: 23784873. DOI: 10.1109/camad.2019.8858490.

[50] A. Sivanathan, H. H. Gharakheili, F. Loi, *et al.*, "Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics," *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1745–1759, 2019, ISSN: 15580660. DOI: 10.1109/TMC.2018.2866249.

[51] S. Kennedy, H. Li, C. Wang, H. Liu, B. Wang, and W. Sun, "I Can Hear Your Alexa: Voice Command Fingerprinting on Smart Home Speakers," *2019 IEEE Conference on Communications and Network Security (CNS)*, pp. 232–240, 2019. DOI: 10.1109/cns.2019.8802686.

[52] U. M. Aivodji, S. Gambs, and A. Martin, "IOTFLA : A Secured and Privacy-Preserving Smart Home Architecture Implementing Federated Learning," pp. 175–180, 2019. DOI: 10.1109/spw.2019.00041.

[53] L. Cui, S. Yang, F. Chen, Z. Ming, N. Lu, and J. Qin, "A survey on application of machine learning for Internet of Things," *International Journal of Machine Learning and Cybernetics*, vol. 9, no. 8, pp. 1399–1417, 2018, ISSN: 1868808X. DOI: 10.1007/s13042-018-0834-5.

[54] F. Hussain, R. Hussain, S. A. Hassan, and E. Hossain, "Machine Learning in IoT Security: Current Solutions and Future Challenges," pp. 1–23, 2019. arXiv: 1904.05735.

[55] W. Li, W. Meng, and M. H. Au, "Enhancing collaborative intrusion detection via disagreement-based semi-supervised learning in IoT environments," *Journal of Network and Computer Applications*, vol. 161, no. February, 2020, ISSN: 10958592. DOI: 10.1016/j.jnca.2020.102631.

[56] S.-W. Lee, H. Mohammed sidqi, M. Mohammadi, *et al.*, "Towards secure intrusion detection systems using deep learning techniques: Comprehensive analysis and review," *Journal of Network and Computer Applications*, vol. 187, p. 103 111, 2021, ISSN: 10848045. DOI: 10.1016/j.jnca.2021.103111.

[57] M. Saharkhizan, A. Azmoodeh, A. Dehghantanha, K. K. R. Choo, and R. M. Parizi, “An Ensemble of Deep Recurrent Neural Networks for Detecting IoT Cyber Attacks Using Network Traffic,” *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8852–8859, 2020, ISSN: 23274662. DOI: `10.1109/JIOT.2020.2996425`.

[58] Z. E. Huma, S. Latif, J. Ahmad, *et al.*, “A Hybrid Deep Random Neural Network for Cyberattack Detection in the Industrial Internet of Things,” *IEEE Access*, vol. 9, pp. 55 595–55 605, 2021, ISSN: 21693536. DOI: `10.1109/access.2021.3071766`.

[59] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, “Dïot: A federated self-learning anomaly detection system for iot,” *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 756–767, 2018.

[60] S. M. Tahsien, H. Karimipour, and P. Spachos, “Machine learning based solutions for security of Internet of Things (IoT): A survey,” *Journal of Network and Computer Applications*, vol. 161, no. February, 2020, ISSN: 10958592. DOI: `10.1016/j.jnca.2020.102630`. arXiv: `2004.05289`.

[61] R. Doshi, N. Apthorpe, and N. Feamster, “Machine learning ddos detection for consumer internet of things devices,” in *2018 IEEE Security and Privacy Workshops (SPW)*, 2018, pp. 29–35.

[62] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, “The computational limits of deep learning,” 2020.

[63] M. Mohammadi, T. A. Rashid, S. H. Karim, *et al.*, “A comprehensive survey and taxonomy of the SVM-based intrusion detection systems,” *Journal of Network and Computer Applications*, vol. 178, no. January, p. 102 983, 2021, ISSN: 10958592. DOI: `10.1016/j.jnca.2021.102983`.

[64] S. N. Mighan and M. Kahani, “A novel scalable intrusion detection system based on deep learning,” *International Journal of Information Security*, no. 0123456789, 2020, ISSN: 16155270. DOI: `10.1007/s10207-020-00508-5`.

[65] Y. Liu, S. Garg, J. Nie, *et al.*, “Deep anomaly detection for time-series data in industrial iot: A communication-efficient on-device federated learning approach,” *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6348–6358, 2021. DOI: `10.1109/JIOT.2020.3011726`.

[66] W. W. Lo, S. Layeghy, M. Sarhan, M. Gallagher, and M. Portmann, “E-graphsage: A graph neural network based intrusion detection system for iot,” in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, IEEE, 2022, pp. 1–9.

[67] D. Pujol-Perich, J. Suarez-Varela, A. Cabellos-Aparicio, and P. Barlet-Ros, “Unveiling the potential of graph neural networks for robust intrusion detection,” *SIGMETRICS Perform. Eval. Rev.*, vol. 49, no. 4, pp. 111–117, Jun. 2022, ISSN: 0163-5999.

[68] A. Protogerou, S. Papadopoulos, A. Drosou, D. Tzovaras, and I. Refanidis, “A graph neural network method for distributed anomaly detection in IoT,” *Evolving Systems*, vol. 12, no. 1, pp. 19–36, 2021, ISSN: 18686486. DOI: `10.1007/s12530-020-09347-0`.

[69] “Adversarial Attack and Defense on Graph-based IoT Botnet Detection Approach,” *3rd International Conference on Electrical, Communication and Computer Engineering, ICECCE 2021*, no. June, pp. 12–13, 2021. DOI: `10.1109/ICECCE52056.2021.9514255`.

[70] Y. Yang and L. Wang, “LGANet: Local Graph Attention Network for Peer-to-Peer Botnet Detection,” *Proceedings - 2021 3rd International Conference on Advances in Computer Technology, Information Science and Communication, CTISC 2021*, pp. 31–36, 2021. DOI: `10.1109/CTISC52352.2021.00013`.

[71] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *International Conference on Learning Representations (ICLR)*, 2017.

[72] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph attention networks,” in *International Conference on Learning Representations*, 2018.

[73] L. Chang and P. Branco, “Graph-based Solutions with Residuals for Intrusion Detection: The Modified E-GraphSAGE and E-ResGAT Algorithms,” Nov. 2021, arXiv: 2111.13597.

[74] D. K. Hammond, P. Vandergheynst, and R. Gribonval, “Wavelets on graphs via spectral graph theory,” *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, 2011, ISSN: 1063-5203.

[75] M. Defferrard, X. Bresson, and P. Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering,” in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS’16, Barcelona, Spain: Curran Associates Inc., 2016, pp. 3844–3852, ISBN: 9781510838819.

[76] W. W. Lo, S. Layeghy, M. Sarhan, M. Gallagher, and M. Portmann, “E-graphsage: A graph neural network based intrusion detection system for iot,” in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, 2022, pp. 1–9. DOI: `10.1109/NOMS54207.2022.9789878`.

[77] E. Caville, W. W. Lo, S. Layeghy, and M. Portmann, "Anomal-e: A self-supervised network intrusion detection system based on graph neural networks," *Knowledge-Based Systems*, vol. 258, p. 110 030, 2022, ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2022.110030.

[78] J. Zhou, Z. Xu, A. M. Rush, and M. Yu, "Automating botnet detection with graph neural networks," *AutoML for Networking and Systems Workshop of MLSys 2020 Conference*, 2020.

[79] Y. Cao, H. Jiang, Y. Deng, J. Wu, P. Zhou, and W. Luo, "Detecting and mitigating DDoS attacks in SDN using spatial-temporal graph convolutional network," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2021. DOI: 10.1109/TDSC.2021.3108782.

[80] O. Boyaci, M. R. Narimani, K. Davis, and E. Serpedin, "Cyberattack detection in large-scale smart grids using chebyshev graph convolutional networks," in *2022 9th International Conference on Electrical and Electronics Engineering (ICEEE)*, 2022, pp. 217–221. DOI: 10.1109/ICEEE55327.2022.9772523.

[81] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Neural Information Processing Systems (NIPS)*, 2017.

[82] M. Amin, T. A. Tanveer, M. Tehseen, M. Khan, F. A. Khan, and S. Anwar, "Static malware detection and attribution in android byte-code through an end-to-end deep system," *Future Generation Computer Systems*, vol. 102, pp. 112–126, 2020, ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2019.07.070.

[83] S. K. Sahu, D. P. Mohapatra, J. K. Rout, K. S. Sahoo, Q.-V. Pham, and N.-N. Dao, "A lstm-fcnn based multi-class intrusion detection using scalable framework," *Computers and Electrical Engineering*, vol. 99, p. 107 720, 2022, ISSN: 0045-7906. DOI: https://doi.org/10.1016/j.compeleceng.2022.107720.

[84] F. Yuan, Y. Cao, Y. Shang, Y. Liu, J. Tan, and B. Fang, "Insider threat detection with deep neural network," in *Computational Science – ICCS 2018*, Y. Shi, H. Fu, Y. Tian, *et al.*, Eds., Cham: Springer International Publishing, 2018, pp. 43–54, ISBN: 978-3-319-93698-7.

[85] P. Rajasekaran and V. Magudeeswaran, "Malicious attacks detection using gru-bwfa classifier in pervasive computing," *Biomedical Signal Processing and Control*, vol. 79,

p. 104 219, 2023, ISSN: 1746-8094. DOI: https://doi.org/10.1016/j.bspc.2022.104219.

[86] X. Zhu, Y. Zhang, Z. Zhang, D. Guo, Q. Li, and Z. Li, "Interpretability evaluation of botnet detection model based on graph neural network," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2022, pp. 1–6. DOI: 10.1109/INFOCOMWKSHPS54753.2022.9798287.

[87] W. W. Lo, G. Kulatilleke, M. Sarhan, S. Layeghy, and M. Portmann, "Xg-bot: An explainable deep graph neural network for botnet detection and forensics," *Internet of Things*, vol. 22, p. 100 747, 2023, ISSN: 2542-6605. DOI: https://doi.org/10.1016/j.iot.2023.100747.

[88] M. Alazab, "A discrete time-varying greywolf iot botnet detection system," *Computer Communications*, vol. 192, pp. 405–416, 2022, ISSN: 0140-3664. DOI: https://doi.org/10.1016/j.comcom.2022.06.016.

[89] M. Macas and C. Wu, "Review: Deep learning methods for cybersecurity and intrusion detection systems," in *2020 IEEE Latin-American Conference on Communications (LATINCOM)*, 2020, pp. 1–6. DOI: 10.1109/LATINCOM50620.2020.9282324.

[90] I. GUYON, S. JASON WESTON, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using Support Vector Machines," *Machine Learning*, vol. 46, pp. 389–422, 2002, ISSN: 16113349. DOI: 10.1007/978-3-540-88192-6_8.

[91] N. Srivastava, G. Hinton, I. Sutskever, A. Krizhevsky, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, 2014, ISSN: 03702693. DOI: 10.1016/0370-2693(93)90272-J.

[92] W. Shang, K. Sohn, D. Almeida, and H. Lee, "Understanding and improving convolutional neural networks via concatenated rectified linear units," *33rd International Conference on Machine Learning, ICML 2016*, vol. 5, pp. 3276–3284, 2016. arXiv: 1603.05201.

[93] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 770–778, 2016, ISSN: 10636919.

[94] "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," *2015 Military Communications and Information Systems Conference, MilCIS 2015 - Proceedings*, pp. 1–6, 2015.

[95] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers and Security*, vol. 86, pp. 147–167, 2019, ISSN: 01674048. DOI: 10.1016/j.cose.2019.06.005. arXiv: 1903.02460.

[96] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in Neural Information Processing Systems*, vol. 2017-Decem, no. Nips, pp. 1025–1035, 2017, ISSN: 10495258. arXiv: 1706.02216.

[97] Y. Li, R. Zemel, M. Brockschmidt, and D. Tarlow, "Gated graph sequence neural networks," *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, no. 1, pp. 1–20, 2016.

[98] S. A. Tailor, F. L. Opolka, P. Liò, and N. D. Lane, "Adaptive Filters and Aggregator Fusion for Efficient Graph Convolutions," 2021.

[99] N. Moustafa, "A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets," *Sustainable Cities and Society*, vol. 72, p. 102 994, 2021, ISSN: 2210-6707.

[100] X. Fu, J. Zhang, Z. Meng, and I. King, "Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding," in *WWW*, 2020.

[101] A. L. Maas, "Rectifier nonlinearities improve neural network acoustic models," in *Proceedings of the 30th International Conference on Machine Learning*, 2013.

[102] R. Collobert and S. Bengio, "Links between perceptrons, MLPs and SVMs," ser. ICML '04, Banff, Alberta, Canada: Association for Computing Machinery, 2004, p. 23, ISBN: 1581138385. DOI: 10.1145/1015330.1015415.

[103] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. P. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset," *Future Gener. Comput. Syst.*, vol. 100, pp. 779–796, 2019.

[104] M. Sarhan, S. Layeghy, N. Moustafa, and M. Portmann, "Netflow datasets for machine learning-based network intrusion detection systems," in *Big Data Technologies and Applications*, Z. Deze, H. Huang, R. Hou, S. Rho, and N. Chilamkurti, Eds., Cham: Springer International Publishing, 2021, pp. 117–135, ISBN: 978-3-030-72802-1.

[105] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," in *International Conference on Learning Representations*, 2018.

[106] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 20, no. 2, pp. 215–232, 1958. DOI: https://doi.org/10.1111/j.2517-6161.1958.tb00292.x.

[107] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *2015 Military Communications and Information Systems Conference, MilCIS 2015 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., Dec. 2015, ISBN: 978-1-4673-7008-0. DOI: 10.1109/MilCIS.2015.7348942.

[108] Cisco, *Cisco Annual Internet Report (2018–2023)*, https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.htm, [Online], Mar. 2020.

[109] M. Frustaci, P. Pace, G. Aloi, and G. Fortino, "Evaluating critical security issues of the iot world: Present and future challenges," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2483–2495, 2018.

[110] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1125–1142, 2017.

[111] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, "A survey on security and privacy issues in internet-of-things," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1250–1258, 2017.

[112] S. Benzarti, B. Triki, and O. Korbaa, "A survey on attacks in internet of things based networks," in *2017 International conference on engineering & MIS (ICEMIS)*, IEEE, 2017, pp. 1–7.

[113] J. Zhou, G. Cui, S. Hu, *et al.*, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020, ISSN: 2666-6510. DOI: https://doi.org/10.1016/j.aiopen.2021.01.001.

[114] T. Altaf, X. Wang, W. Ni, R. P. Liu, and R. Braun, "Ne-gconv: A lightweight node edge graph convolutional network for intrusion detection," *Computers Security*, vol. 130, p. 103 285, 2023, ISSN: 0167-4048.

[115] T. Altaf, X. Wang, W. Ni, G. Yu, R. P. Liu, and R. Braun, "A new concatenated multigraph neural network for iot intrusion detection," *Internet of Things*, vol. 22, p. 100 818, 2023, ISSN: 2542-6605.

[116] K. S. Kalupahana Liyanage, D. M. Divakaran, R. P. Singh, and M. Gurusamy, *Nss mirai dataset*, 2020. DOI: `10.21227/t970-nd64`. [Online]. Available: `https://dx.doi.org/10.21227/t970-nd64`.