Graph Neural Networks in Intrusion Detection

Nuno Ferreira
Mestrado em Ciências de Computadores
Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto
2024

Graph Neural Networks in Intrusion Detection

Nuno Ferreira

Dissertação realizada no âmbito do Mestrado em Ciências de Computadores

Departamento de Ciência de Computadores

2024

Orientador
Doutora Ivone Amorim, Faculdade de Ciências da Universidade do Porto

Coorientador
Doutora Eva Maia, Instituto Superior de Engenharia do Porto

UNIVERSIDADE DO PORTO

MASTERS THESIS

Graph Neural Networks in Intrusion Detection

Author:
Nuno FERREIRA

Supervisor:
Ivone AMORIM

Co-supervisor:
Eva MAIA

A thesis submitted in fulfilment of the requirements
for the degree of MSc. Computer Science

at the

Faculdade de Ciências da Universidade do Porto
Departamento de Ciência e Computadores

July 25, 2024

" I am and always will be the optimist, the hoper of far-flung hopes and the dreamer of improbable dreams "

Matt Smith as *The Doctor*, written by Matthew Graham

Acknowledgements

I would like to thank my supervisors, Ivone Amorim and Eva Maia, for the opportunity of working on this topic, for their guidance and patience.

I would also like to thank Edgardo Almeida, Daniel Barroso, João Dias and Carolina Rodrigues for their unknown support and encouragement.

UNIVERSIDADE DO PORTO

Abstract

Faculdade de Ciências da Universidade do Porto
Departamento de Ciência e Computadores

MSc. Computer Science

Graph Neural Networks in Intrusion Detection

by Nuno FERREIRA

The increasing complexity and frequency of cyberattacks demands advanced methods for network intrusion detection. Traditional intrusion detection systems often fall short in identifying complex attack patterns due to their inability to capture relationships within network data. Graph Neural Networks (GNNs) offer a promising solution by leveraging graph structures to model these relationships more effectively.

This thesis explores the application of GNNs in the domain of Network Intrusion Detection Systems (NIDSs). The primary goals include providing an in-depth review of recent research in GNN-based NIDSs, conduct experiments to evaluate the effectiveness of various GNN models, optimize these models using multiple datasets and parameters, and demonstrate the advantages of graph structures in intrusion detection.

Key findings revealed that our GAT model excelled at detecting intrusions by using their attention mechanisms to differentiate between normal and anomalous data. By incorporating edge features into the model, our E-GraphSAGE model provided a different representation of network interactions, improving the detection of intrusions. This study demonstrates that graph structures have the potential to significantly improve network intrusion detection by leveraging relationships within networks. As demonstrated, centrality measures hold significant importance, offering valuable network insights that could potentially improve the model's results.

UNIVERSIDADE DO PORTO

Resumo

Faculdade de Ciências da Universidade do Porto
Departamento de Ciência e Computadores

Mestrado Integrado em Engenharia Física

Titulo da Tese em Portugês

por Nuno FERREIRA

A crescente complexidade e frequência dos ciberataques exige métodos avançados para a deteção de intrusões de rede. Os sistemas tradicionais de deteção de intrusões muitas vezes não conseguem identificar padrões de ataque complexos devido à sua incapacidade de capturar relações dentro dos dados da rede. As Graph Neural Networks (GNNs) oferecem uma solução promissora ao aproveitar as estruturas de grafos para modelar estas relações de forma mais eficaz.

Esta tese explora a aplicação das GNNs no domínio dos Network Intrusion Detection Systems (NIDSs). Os objetivos principais incluem fornecer uma revisão aprofundada da investigação recente em NIDSs baseados em GNN, conduzir experiências para avaliar a eficácia de vários modelos GNN, otimizar estes modelos utilizando vários *datasets* e parâmetros e demonstrar as vantagens das estruturas gráficas na deteção de intrusões.

As principais descobertas revelaram que o nosso modelo GAT se destacou na deteção de intrusões utilizando os seus mecanismos de atenção para diferenciar entre dados normais e anómalos. Ao incorporar características das arestas no modelo, o nosso modelo E-GraphSAGE forneceu uma representação diferente das interações de rede, melhorando a deteção de intrusões. Este estudo demonstra que as estruturas gráficas têm o potencial de melhorar significativamente a deteção de intrusões na rede, aproveitando as relações dentro das redes. Como demonstrado, as medidas de centralidade têm uma importância significativa, oferecendo informações valiosas sobre a rede, que poderiam potencialmente melhorar os resultados do modelo.

Contents

List of Figures

List of Tables

Symbols

V	Set of nodes
E	Set of edges
v	Node v such that $v \in V$
e	Edge e such that $e \in E$
$N(v)$	Neighbors of node v
X_v	Mapping functions that map node v to a list of attributes
X_e	Mapping functions that map edge e to a list of attributes
V_t	Node structure at timestamp t
E_t	Edge structure at timestamp t
$X_{v,t}$	Mapping functions that map node v to a list of attributes at a the t timestamp
$X_{e,t}$	Mapping functions that map edge e to a list of attributes at a the t timestamp
γ_v	Mapping function that maps node v to a type
γ_e	Mapping function that maps edge e to a type
k	k-th layer
K	Number of layers
V	Set of nodes where $v \in V$
ρ_{st}	number of shortest paths from node s to node t
$\rho_{st}(v)$	number of shortest paths from node s to node t that pass through v
a	Aggregation function
m_v^k	Messages collected at layer k of node v
h_v^k	Embeddings of node v at layer k

U	Update function (e.g. Neural Network)
$\hat{y}$	Feature vector
R	Readout function
σ	Activation function (e.g. ReLu)
A	Adjacency matrix
I	Identity matrix
D	Degree matrix
W	Learnable parameter
c	Weight vector
C	Weight matrix
c_{Φ}^{t}	Node-level attention for that meta-path at timestamp t
N_{i}^{Φ}	Meta-path-based neighbours of node i
$h_{i}^{'}$	Embedding of the node i
q	Semantic-level attention vector
Y	Weight parameters
b	Bias parameters
$Z_{\Phi i}$	Semantic-specific node embeddings from meta-path Φi
P	Total number of meta-paths
s	Number of samples to select from the full neighborhood
$N_s(u)$	Uniformly sampled neighborhood of node u
e_{uv}^{t-1}	Embedding of edge (u, v) at layer $t-1$

Chapter 1

Introduction

Cyberattacks are a major issue in today's world [1], targeting companies and critical infrastructure like power grids [2], hospitals [3], and financial institutions [4]. These attacks can steal sensitive information, disrupt essential services, and cause significant financial losses [5]. Critical infrastructure attacks can lead to extensive disruptions, affecting many people's daily lives. Detecting these attacks early is crucial to prevent extensive damage and protect data, infrastructure and ultimately people. To combat cyberattacks, the cybersecurity field has developed tools like rule-based detection systems and Intrusion Detection Systems (IDS) [6] that rely on known patterns of malicious activity to identify threats. By analyzing these patterns, such as specific code sequences or behavior signatures, they can quickly flag suspicious activities. While these methods are great at detecting known patterns of attack, they often fail in detecting new patterns. A slight change in the attack vectors and it goes undetected.

To counter this problem, Machine Learning (ML) methods have been used in the defense against cyberattacks [7–9]. These methods efficiently detect new patterns of attack using a small portion of data along with feature engineering, which corresponds to the process of selecting and transforming raw data into features. However, feature engineering requires domain experts to go through the data and modify it, which can be subject to errors. Deep Learning (DL) models, including Convolutional Neural Networks (CNNs) [10] and Recurrent Neural Networks (RNNs) [11], improve on these issues by learning hierarchical features from the model. However, these models are typically trained using sub-optimal data structures such as fixed-size vectors, limiting their ability to represent complex relationships within network traffic. Consequently, both ML and

DL approaches may fail to detect certain types of attack that require the understanding of such relationships.

GNNs provide a promising solution by leveraging network data's graph structure. Examples of its usage include social networking analysis [12], biologic research [13], traffic flow predictions [14], recommendation systems [15] and intrusion detection [16]. Networks with hosts, devices, and their interactions, can be naturally modeled as graphs where nodes represent entities and edges represent communications between the entities. GNNs excel at learning from such graph-structured data, capturing dependencies and interactions more effectively than traditional ML and DL models. They dynamically update their understanding as new data arrives, making them adaptable to changes in network traffic patterns and attack strategies. Additionally, GNNs improve the interpretability of the data by providing insights into the relationships and patterns that contribute to the detection of intrusions.

1.1 Motivation

GNNs are increasingly being adopted for IDSs due to their ability to effectively capture and analyze the structural relationships in network data, which is crucial for identifying complex attack patterns that traditional ML models often miss. For instance, Pujol-Perich et al. [17] GNN-based NIDS demonstrates state-of-the-art performance on the CIC-IDS2017 dataset, maintaining high accuracy even under adversarial conditions that degrade the performance of other models by up to 50%. This robustness is largely attributed by the authors to the GNN's ability to learn and recognize structural patterns in network traffic, providing a significant advantage over other approaches. On the other hand, recent advancements in self-supervised and unsupervised learning further improve the capabilities of GNN-based IDSs. Anomal-E [18], a self-supervised network intrusion detection system, uses autoencoders to identify anomalies without requiring labeled data, achieving good results in detecting intrusions. Additionally, the E-GraphSAGE [19] model leverages both topological and edge features from network flow data, significantly improving the detection of individual attack flows compared to traditional methods. However, several gaps and areas for improvement remain [16]. Certain datasets, like the CIC-IDS2017, ToN-IoT and NF-BoT-IoT datasets face a problem of class imbalance [20, 21]. This is explained by the fact that attacks typically do not happen

frequently in a network, resulting in a dataset that mostly contains benign traffic. Additionally, newer and larger datasets that cover more attack types and represent real-life situations are necessary to improve model generalization [16]. While GNNs have shown robustness to some adversarial attacks, there is still a need to improve their resilience even further [22]. Attackers can alter the graph structure or node features, leading to incorrect predictions, and GNN models must be prepared for that.

1.2 Main Objectives

The main objectives of this thesis are:

1. Perform a comprehensive literature review of the recent research developed in the field of GNN-based NIDSs and identify any potential gaps for further exploration;
2. Identify the most relevant datasets used in this area and improve their quality using adequate pre-processing methods;
3. Develop different GNN models, and assert theirs ability to detect intrusions, using the identified datasets. Fine-tune the models.
4. Present a comparative analysis of those models with the ones found in the existing literature;
5. Illustrate some of the advantages of GNNs and the significance of a graph structure.

1.3 Main Contributions

The following contributions were made in this thesis:

1. A comprehensive review on the recent research of GNN-based NIDS was created. Based on this we found that the CIC-IDS2017 dataset deserved more exploration and that the NF-BoT-IoT and ToN-IoT datasets are one of the most used datasets. We observed that E-GraphSAGE and GAT models also required additional exploration;
2. A detailed description of the datasets is was created, highlighting their importance in the context of intrusion detection. Regarding their problems, we detail that they show low feature quality and a substantial imbalance issue;

3. The development of a base GNN model, ideal for intrusion detection, was built showing impressive results with a weighted F1 value of 0.97 for binary classification and 0.96 for multi-class classification;

4. More advanced GNN models were also constructed. Overall, all models show high results in detection of intrusions. The GRU model has a weighted F1 score of 0.95 on the CIC-IDS2017 dataset. Our E-GraphSAGE model demonstrates good results in edge classification tasks, with a weighted F1 value of 0.97 for binary classification in the NF-BoT-IoT dataset. GATs shows good performance on binary classification task, with a weighted F1 score of 0.95 in the ToN-IoT dataset;

5. The comparison of our models against the ones in the literature was done. Overall, we obtained lower results when testing our models with the CIC-IDS2017 dataset and competitive results on ToN-IoT and NF-BoT-IoT datasets, sometimes even surpassing the existing models;

6. Finally, we showcased a new perspective on the advantages of using GNNs by demonstrating the importance of centrality measures like the betweenness centrality and how can they improve the model's performance.

1.4 Thesis Organization

The thesis is structured as follows: Chapter 2 introduces fundamental concepts and provides a review of the most recent research and its contributions to the field. In Chapter 3 the datasets used, as well as the pre-processing steps performed on each dataset, are detailed. Next, we propose and discuss the developed models, their hyperparameters, applicability and reason of use. Chapter 4 presents the experiments made, discusses the results, highlights the improvements achieved and compares the obtained results against the literature. Finally, Chapter 5 draws conclusions and discusses future work on this area.

Chapter 2

State of the art

The increasing complexity of cyber threats has led to a need for advanced intrusion detection methods. ML and DL methods have been applied to improve the performance of IDSs. In this Chapter, we explore the theoretical foundations, presented in Section 2.1, and advances of these technologies in enhancing IDSs. In Section 2.2 we dive into the use of ML and DL models that are shown to improve the accuracy and efficiency of detecting those attacks. Furthermore, in Section 2.2.2 we investigate the relatively new application of GNNs in IDSs, highlighting their ability to model complex relationships within network data. By examining different types of graphs, we illustrate the different approaches to capturing complex network behaviors and identifying potential intrusions. From Section 2.2.2.1 onwards, a comprehensive review of research methodologies and key studies is presented, provided a holistic understanding of how GNNs are transforming intrusion detection and bolstering cybersecurity defenses.

2.1 Theoretical Background

Nowadays, there are three main types of intrusion detection systems, each serving a distinct purpose within a network [23]:

- **Network Intrusion Detection System:** a NIDS is deployed across a network to monitor and analyze network traffic for potential cyberattacks. It focuses on network-level activities and can handle large volumes of data. However, because it deals with so much information, it sometimes lacks precision, meaning it might miss detecting an attack or fail to identify issues within encrypted communications;

- **Network Node Intrusion Detection System (NNIDS):** applied to individual hosts, NNIDS is similar to NIDS in that it checks each node for threats;
- **Host Intrusion Detection System (HIDS):** installed on individual network devices, a HIDS monitors the entire system's file set. Its main objective is to identify significant differences and notify administrators of altered settings or files. It provides a second line of defense for malicious packets that NIDS might have missed.

In addition, NIDSs can be categorized into two groups [24]:

- **Signature-based IDSs:** employ a predefined set of rules, metrics, or methods to categorize and handle network traffic. When the network traffic corresponds with any of the signatures stored in the database, the system activates an alert. Despite performing very well on known attacks, it performs poorly on new or unseen behavior.
- **Anomaly-based IDSs:** respond to the limitations of the previous group by making use of the known data to create ML models that can promptly identify anomalies or deviations. Despite their effectiveness, these models are far from perfect, as they may mistakenly identify previously unknown legitimate behavior as anomalous since they rely on the quality and quantity of data available.

ML is a sub-field of Artificial Intelligence (AI) that enables computers to enhance their performance on a particular job by using experience and knowledge. The roots of ML can be traced back to the 1940s and 1950s, when pioneers like Alan Turing [25] and Arthur Samuel [26] laid the groundwork for this area of work. ML concentrates on creating algorithms and statistical models that empower computers to learn and make predictions or decisions without explicit programming. It involves the study of patterns and inference from data to improve the performance of tasks over time [27]. There are four fundamental categories of ML [27]:

- **Supervised Learning** involves creating algorithms that are taught using labeled datasets, where each input is associated with a matching output. By utilizing this annotated dataset, the model acquires the ability to establish a correlation between the given inputs and their corresponding outputs. Consequently, it becomes proficient in generating predictions for novel and unobserved data;
- **Unsupervised learning** involves working with data that does not have labels, with the goal of uncovering inherent patterns or structures in the input;

- **Semi-supervised learning** is a ML approach that combines supervised and unsupervised learning. It uses a small set of labeled data combined with a larger set of unlabeled data. The model acquires knowledge from the labeled data while using the unlabeled data to enhance its comprehension of the fundamental framework;

- **Reinforcement Learning (RL)** is a learning approach where agents acquire knowledge by actively engaging with an environment. Agents are autonomous entities or systems that interact with an environment to learn and make decisions. The agent is provided with feedback in the form of rewards or penalties, which are determined by its behavior. This enables the agent to acquire the best tactics by engaging in a trial and error process.

DL refers to a category of ML techniques that utilize many layers to gradually extract more complex information from the original input. The neural network is the fundamental basis of DL [28]. The design of this computational model draws inspiration from the structure and function of the human brain. The system is composed of linked nodes, also known as neurons, which are organized in layers. We assign a weight to each connection between nodes. Neural networks use weights and activation functions to analyze input data across several layers, enabling them to alter the data and acquire knowledge of patterns.

A neural network takes an input vector of p variables $X = (X_1, X_2, ..., X_p)$ and builds a nonlinear function $f(X)$ to predict the response Y. The hidden layer computes activations $A_k = h_k(X)$ that are nonlinear transformations of linear combinations of the inputs $X_1, X_2, ..., X_p$. A single layer neural network, as displayed in Figure 2.1, will contain a single hidden layer, while a multi layer neural network will contain more than one hidden layer. The flow of information is unidirectional, from the input nodes to the output nodes, without any loops, hence the term *feed-forward*.

The origins of DL can be traced back to the 1940s, when McCulloch et al. [29] introduced the first artificial neuron. However, it was not until the 1980s and 1990s that significant advancements were made, thanks to many distinguished researchers. Rumelhart et al. [30] introduced the backpropagation algorithm, a fundamental technique for training artificial neural networks. LeCun et al. [31] applied gradient-based learning to document recognition. Bengio et al. [32] introduced a neural probabilistic language model that laid the groundwork for using neural networks in natural language processing tasks.

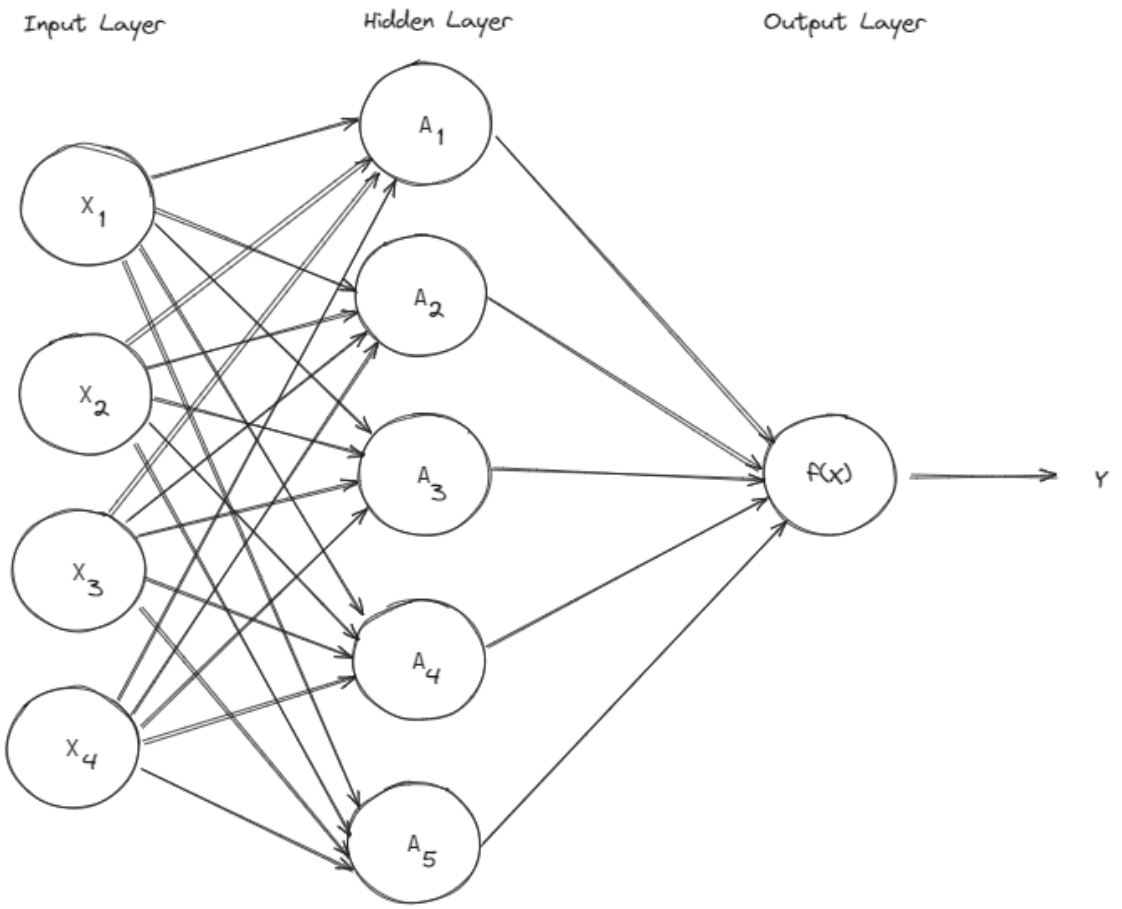

FIGURE 2.1: A simple feed-forward neural network

A CNN [31] is a form of neural network that is capable of learning feature engineering independently through the optimization of filters (also known as kernels). They have applications in image and video recognition, recommender systems, and others, including IDSs [33–35]. CNNs imitate the human process of image classification by identifying distinct characteristics in the pictures that are unique to a certain item category. It identifies low-level features in the input images (small edges, colors,etc.). These low features are then combined to form high-level features like the ears of a dog or the eyes of a cat. The presence or absence of these features contributes to the response.

RNN [36] is a kind of artificial neural network that is best suited for jobs where the input and output occur sequentially. Sequential data processing is its primary purpose. RNNs are different from standard feedforward neural networks in that their connections form directed cycles, which enables them to keep track of past inputs in a hidden state

throughout computations. RNNs, with their sequential learning capabilities, are well-suited for capturing temporal dependencies in network traffic. This is crucial in identifying subtle and complex attack patterns that unfold over time [37, 38].

A family of neural network models called GNN [39] is specifically made to operate on data that is organized into graphs. A graph represents data as nodes and edges, where nodes are typically entities and edges are the relationships between them. Designed to learn and comprehend the intricate relationships and patterns found within graphs, GNNs are ideal for jobs requiring relational data. The main difference between graph data and other types of data is that graph data is structured (we define the relation between data points that gives us a structure).

A graph is represented as

$$G = (V, E) \tag{2.1}$$

Varying circumstances require distinct methodologies and diverse forms of graphical representation. Considering an intrusion detection scenario, variables such as time, location and attributes are of the utmost importance as they provide context for analyzing normal behavior, detecting anomalies, correlating events, assessing risk, and facilitating effective incident response. But using those parameters is not possible according to the aforementioned definition. As we are going to see in the next sections, many authors use models using various kinds of graphs to address the issue of intrusion detection.

An attributed graph [16] is a kind of graph where attributes are attached to both nodes and/or edges, providing additional information. They are officially defined as

$$G = (V, E, X_v) \tag{2.2}$$

for a node-based attributed graph and

$$G = (V, E, X_e) \tag{2.3}$$

A spatio-temporal graph [16] combines geographic and temporal dimensions to represent relationships and interactions among entities throughout space and time. The structure of the graph is maintained over time. Formally, they can be defined as

$$G_t = (V, E, X_{v,t}, X_{e,t}) \tag{2.4}$$

A dynamic graph [16] changes over time (in space and/or time), with nodes and edges being added or removed as the underlying relationships evolve. They can be subdivided

into many different representations. However, the majority uses Discrete-Time Dynamic Graphs (DTDGs) where the graph is modeled by a sequence of snapshots at a certain timestamp. Formally, they are defined as

$$G_t = (V_t, E_t, X_{v,t}, X_{e,t}) \tag{2.5}$$

Heterogeneous graphs [16] contain multiple types or categories of nodes and edges, each with unique features. They are well suited for scenarios that contain diverse relationships and connections between entities in a complex network or system. Formally, they can be defined as

$$G = (V, E, \gamma_v, \gamma_e) \tag{2.6}$$

In the context of graph theory, centrality is a measure used to assess a node's importance or influence within a graph. Centrality measures help identify the most significant nodes, which can be crucial for understanding the structure and dynamics of the network. Betweenness centrality is one example. It measures the extent to which a node is on the shortest paths between other nodes. For a node v the betweenness centrality $C_B(v)$ is given by

$$C_B(v) = \sum_{s \neq v \neq t} \frac{\rho_{st}(v)}{\rho_{st}} \tag{2.7}$$

Nodes with high betweenness centrality act as bridges or critical connectors in the network, facilitating communication between different parts. If an attacker gains control over these nodes, they can potentially intercept or influence a large amount of traffic. Also, a sudden increase in the betweenness centrality of a node could indicate malicious activity, such as a node being used to route a large amount of traffic as part of an attack (e.g., a man-in-the-middle attack).

Moreover, there are two main methods for learning from the previously described graphs [16]:

- **Inductive Learning:** involves training the model on a portion of the graph and subsequently using it to generate predictions on unfamiliar or novel data, such as nodes or graphs. The objective is to apply the acquired patterns from the training data to make predictions on new instances without the need to retrain the model;

- **Transductive learning:** entails training the model on the full graph, which includes both the training and test data. During training, the model generates predictions for

all nodes/edges in the graph, including the ones that will later be used for testing or validation.

Intrusion detection identifies intrusions based on a series of suspicious and benign interactions between different entities. One can use Graph Representation Learning (GRL) models, like GNNs, to learn interactions by representing them using one of the previously mentioned graph types. GRL comprises the extraction of informative characteristics, referred to as embeddings, from a graph. Embeddings are representations of items, concepts, or entities in a space with fewer dimensions. They encapsulate the fundamental characteristics and connections among objects in a manner that maintains significant data [16]. As we will see in the following sections, some approaches compute the node embeddings so that similar nodes are embedded close together in the embedding space, and other approaches involve the computation of edge embeddings and graph embeddings.

Several authors proposed general frameworks to encapsulate common GNN models that were used in the papers presented in the following section [40, 41]. It is then essential to understand them and their origins. The fundamental concept behind GNNs is to systematically update the node representations by merging the representations of their neighboring nodes with their own representations. Xu et al. [40] introduces a general framework of GNNs where in each layer, two functions are applied:

- AGGREGATE - function that aggregates the information from the neighbors of each node;
- COMBINE - update the node representations by combining the aggregated information from neighbors with the current node representations.

Formally, the general framework of GNNs can be defined as follows:

$$a_v^k = AGGREGATE^k \left\{ H_u^{k-1} : u \in N(v) \right\} \tag{2.8}$$

$$H_v^k = COMBINE^k \left\{ H_v^{k-1}, a_v^k \right\} \tag{2.9}$$

The choice of AGGREGATE and COMBINE functions is of the utmost importance.

Gilmer et al. [41] also proposed a general framework for GNNs named Message-Passing Neural Network (MPNN) that involves passing messages between nodes in a graph, typically in multiple steps or iterations. MPNNs essentially consist of two phases:

a message passing phase and a readout phase. In the first phase, a message for each node is computed based on its previous embeddings and the neighbor's embeddings.

$$m_v^{k+1} = \sum_{w \in N(v)} M^k(h_v^k, h_w^k, e_{v,w}) \tag{2.10}$$

Then, the representation of a node $h_v^{k+1}$is updated using the output of the previous function

$$h_v^{k+1} = U_k(h_v^k, m_v^{k+1}) \tag{2.11}$$

In the readout step, a feature vector $\hat{y}$ is computed for the whole graph using a readout function R in the following way

$$\hat{y} = R(\left\{ h_v^K | v \in G \right\}) \tag{2.12}$$

We can see that the MPNN framework is very similar to the previous framework. The AGGREGATE function defined here is simply a sum of all the messages from the neighbors. The COMBINE function is the same as the node U function.

The Graph Convolutional Network (GCN) presented by Kipf et al. [42] is a popular GNN architecture due to its simplicity and effectiveness in a variety of tasks and applications. It is a generalization of convolutions on graphs. The node representations in each layer is updated according to

$$H^{k+1} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^k W^k) \tag{2.13}$$

where $\tilde{A} = A + I$ is the adjacency matrix of the given graph G with self-loops and $\tilde{D}$ is the degree matrix of $\tilde{A}$. Together $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ represents an adjacency matrix with self-loops that has been normalized with respect to the degree of the matrix. Normalization is needed to avoid exploding/vanishing gradients. The AGGREGATE function is defined as the weighted average of the representations of neighboring nodes. The COMBINE function is the sum of the aggregated messages and the normalized node representation, where the normalization is based on the node's degree. With this in mind, several authors proposed instantiations of these frameworks. Before delving into their applicability to IDSs, it is critical to understand them. In GCNs, the significance of a neighbour j for a given node i is defined by the weight of their edge A_{ij}, which is then normalized by the degree matrix D. This can lead to complications because, in reality, the input graph may contain noise. As a result, the edge weights may not accurately represent the actual

strength between two nodes, which is not ideal in some situations.

To address these restrictions, Graph Attention Network (GAT) [43] was offered as a solution. GATs employ an attention mechanism, which has been suggested in several natural language processing tasks [44], to acquire a weight that signifies the relevance of each neighboring node for a given node. GATs enable the capture of more intricate interactions between nodes, resulting in a more valuable collection of information. The attention coefficients between two nodes u and v are given by

$$\alpha_{uv} = \frac{exp(\sigma(c[WH_u^{k-1}, WH_v^{k-1}]))}{\sum_{k \in N(u)} exp(\sigma(c[WH_u^{k-1}, WH_k^{k-1}]))} \tag{2.14}$$

The new node representation H_i^k is then given by a linear combination of the neighboring node representations with the weights determined by the attention coefficients.

$$H_i^k = \sigma(\sum_{j \in N(i)} \alpha_{ij} W H_j^{k-1}) \tag{2.15}$$

Additionally, the author also suggests that we can have more than one single attention mechanism named multi-head attention. For each attention head, we can independently obtain a new node representation. The final node representation will be a concatenation of the node representations learned by different attention heads

$$H_i^k = \prod_{t=1}^{T} \sigma(\sum_{j \in N(i)} \alpha_{ij}^t W^t H_j^{k-1}) \tag{2.16}$$

The variety and complexity of real-world data, together with their vast range of entities and interactions, may make typical attributed graphs inadequate. In order to tackle this issue, researchers are progressively utilizing Heterogeneous Graph Attention Networks (HAN) to identify attacks. Meta-paths are just one example. They are ordered sequences of node and edge types commonly employed in HANs because of their diverse composition and effective semantic extraction. According to Wang et al. [45], HAN uses an attention mechanism similar to the original GAT to gather information from meta-paths. Specifically, the model learns two attention mechanisms: one that focuses on individual nodes and another that focuses on the overall semantic meaning. The former involves understanding the significance of a node and its neighbors based on meta-paths. The latter involves acquiring knowledge about the significance of assigning weights to

each meta-path. The importance of a node j for a node i can be calculated by:

$$\alpha_{ij}^{\Phi} = \frac{exp(\sigma(c_{\Phi}^{t}.[h_i^{'}, h_j^{'}]))}{\sum_{k \in N_i^{\Phi}} exp(\sigma(c_{\Phi}^{t}.[h_i^{'}, h_k^{'}]))} \tag{2.17}$$

The meta-path-based embedding of a node i can then be aggregated using the attention coefficients:

$$z_i^{\Phi} = \sigma(\sum_{j \in N_i^{\Phi}} \alpha_{ij}^{\Phi}.h_j^{'}) \tag{2.18}$$

Next, the model is trained to assign appropriate significance to various meta-paths using semantic-level attention. The normalized significance coefficient for each meta-path i is given by:

$$\beta_{\Phi i} = softmax(\frac{1}{|V|} \sum_{i \in V} q^T.tanh(Y.z_i^{\Phi} + b)) \tag{2.19}$$

We derive the final embeddings by aggregating the calculated attention coefficients, so that

$$Z = \sum_{i=1}^{P} \beta_{\Phi i}.Z_{\Phi i} \tag{2.20}$$

Traditional ML assumes that data records are statistically independent, allowing for large dataset training using mini-batches. However, the connected nature of graph nodes prevents them from being considered independent, which makes scaling GNNs on large graphs with billions of nodes or edges more challenging. Models like GCN and GAT require memory storage for the entire adjacency matrix, making them unusable on large graphs. GraphSAGE [46] was the first model to address this by sampling a fixed number of nodes s from the neighborhood of a given node u, rather than using all neighbors. The uniformly sampled neighborhood of node u is given by

$$N_s(u) = SAMPLE(N(u), s) \tag{2.21}$$

To compute node embeddings, GraphSAGE employs two operations. The first aggregates neighbors' embeddings, and the aggregated neighborhood $h_{N_s(u)}^{t}$ is given by

$$h_{N_s(u)}^{t} = a_t(\left\{h_v^{t-1}, \forall v \in N_s(u)\right\}) \tag{2.22}$$

Then, a second operation consists of updating the embedding of u using its previous embeddings and the aggregation of the neighbors' embeddings

$$h_u^t = \sigma(C^t.\left[h_u^{t-1}, h_{N_s(u)}^{t}\right]) \tag{2.23}$$

E-GraphSAGE [19] changes the previous model to work with edge-level features:

$$h^t_{N_s(u)} = a_t(\left\{e^{t-1}_{uv}, \forall v \in N_s(u), uv \in E\right\}) \tag{2.24}$$

$$h^t_u = \sigma(C^t . \left[h^{t-1}_u, h^t_{N_s(u)}\right]) \tag{2.25}$$

2.2 Machine Learning for Intrusion Detection

Before the integration of ML, rule-based detection systems were used to detect intrusions [16, 47]. While effective against known threats, these approaches struggled with new or evolving attacks and often generated false positives. A few simple modifications from the attacker would be enough to bypass a detection system. There was a clear need for a different kind of system. With the emergence of ML, IDSs were able to identify complex patterns and anomalies in network traffic more effectively, thereby enhancing their overall capability to detect threats. Some of the most studied ML algorithms [48] are Decision Trees (DT) [49–51], Support Vector Machines (SVM) [51–53] and clustering algorithms like K-nearest neighbors (KNN) and k-means clustering [54, 55] are among the most researched ML algorithms.

Al-Omari et al. [56] suggested a new Decision Tree Split (DTS) algorithm based on a common decision tree to predict and detect cyberattacks. Particularly, they rank and select features in order to choose the most important ones by using the Gini Index to measure the impurity of the security features. Their new approach determines the split value by computing the mean of the values within a certain property's range at each node. The classifier assigns equal weight to all values in the domain, ensuring that it is impartial towards the most common values of an attribute. Testing was performed on UNSW-NB 15 dataset [57], and the results indicated that the algorithm outperforms similar approaches when applied to networks with a smaller number of characteristics. The constructed model also takes less time to build, which increases efficiency.

Horng et al. [58] proposed an SVM-based IDS that made use of clustering techniques and feature selection to detect intrusions. It uses the BIRCH hierarchical clustering algorithm to preprocess the KDD Cup 1999 dataset [59] before SVM training. It does that due to the fact that SVMs are not favorable in large dataset scenarios as the training complexity is dependent on the amount of data. The clustering algorithm provides a reduced

dataset but still represents all data points in the original dataset. Feature selection is done by using the "leave-one-out" technique: the importance of a feature is assessed by evaluating the system's accuracy and the occurrence of false positives, both with and without the feature. Finally, the SVM classifiers for the IDS are trained and tested. The proposed system displayed a high accuracy and a low false positive rate compared to other NIDSs that also applied the same dataset.

Lin et al. [60] focused on the problem of extracting more representative features for normal connections and effective detection of attacks, as they stated that, at the time, few studies existed on that problem. They created a novel method to detect intrusions by combining cluster centers and nearest neighbors, which they named CANN. The initial stage in this methodology is employing the k-means clustering algorithm to extract the central points of clusters. Furthermore, each individual data point in the provided dataset is recognized together with its closest neighbor inside the same cluster. The second stage involves calculating and adding two distances: the distance between all data points in the supplied dataset and the cluster centers, and the distance between each data point and its closest neighbor within the same cluster. Afterwards, a KNN classifier employs the distance-based characteristic to represent each data instance for intrusion detection purposes. Results show an improvement over the KNN and SVM classic classifiers, which proves advantageous as it requires less computational effort than both classifiers.

2.2.1 Enhancing Intrusion Detection with Deep Learning

Despite providing acceptable results, the aforementioned ML methods rely heavily on the quality of the data, necessitating thorough data examination and analysis by domain experts. This procedure is not only time-consuming and costly, but the effectiveness of the extracted features directly impacts the performance of the models and, by inheritance, the systems.

DL techniques such as CNNs, RNNs and GNNs have the ability to autonomously acquire significant characteristics from unprocessed data, thereby minimizing the necessity for manual feature engineering. In addition, they effectively manage complex data, including graphics and text, and exhibit excellent scalability with extensive datasets and computing resources. Despite this, they also require a large amount of data and significant processing resources. As such, they have drawn significant interest for their potential

use in a wide variety of research and practical implementations of IDSs [35, 61–65]. Indeed, CNNs are effective in capturing spatial dependencies in network traffic or system logs, while RNNs excel at modeling temporal sequences. DL-based IDS approaches provide superior accuracy and reduced false positive rates compared to previous methods. Furthermore, the model learns the features itself. This allows them to effectively detect and analyze more complex attack patterns, eliminating the need for repetitive feature engineering, which saves time and money.

Vinayakumar et al. [10] encompassed the previously mentioned models and aims to evaluate the efficacy of CNNs, and the combination of convolution and sequential data modeling approaches such as RNN, Long Short-term Memory (LSTM), and Gated Recurrent Unit (GRU), in analyzing and classifying normal and malicious network connections on the famous KDDCup 99 challenge dataset [59]. The authors assert that CNNs have the capacity to extract high-level feature representations that capture the abstract form of low-level feature sets of network traffic links. The results show that the CNN algorithms proposed outperformed other models that used the same dataset. The results also show that introducing a model capable of capturing temporal patterns proves advantageous in categorizing attacks into their corresponding categories, where a 3-layer CNN followed by a LSTM shows the best results.

Yin et al. [11] proposed a DL approach for intrusion detection using recurrent neural networks (RNN-IDS). The performance of the model, constructed based on the classical mechanisms of Forward Propagation and Back Propagation, is then studied in a binary classification setting and in a multi-class classification setting. The performance of the model is also compared against classic ML approaches to IDSs like Random Forests, SVMs, Naive Bayes and others, in both settings. The experimental findings demonstrated that RNN-IDS is very appropriate for intrusion detection. It outperforms the conventional classification techniques on the NSL-KDD dataset [66] in both binary and multi-class classification. Particularly, it shows a higher accuracy rate and detection rate with a low false positive rate than its counterparts, hence offering a novel research approach for intrusion detection.

2.2.2 Graph Neural Networks

The previously mentioned methods treat each data record independently, missing important relationships between network events, and often overlook correlations between

them. They also typically lack a holistic view of the network. This proves disadvantageous as these methods are limited in their ability to detect complex attacks on systems such as smart power grids or any IoT network [16]. Furthermore, intrusions and malicious behaviors in a network, such as DDoS attacks or cross-site scripting (XSS), exhibit certain spatial and temporal patterns. For instance, a DDoS attack on a host likely exposes other hosts in the same local network to similar attacks. A DDoS attack or an intrusion attempt can take hours, if not days. Therefore, it is of the utmost importance to gain knowledge of the network topology and understand the correlation between concurrent data points [67].

GNNs appear to be ideal for IDSs because they can represent complex relationships and interactions within network data and capture spatio-temporal characteristics. They are highly proficient at capturing the inherent patterns and changes in graph data, enabling them to accurately detect abnormal behavior or potential security risks. GNNs improve the system's capacity to identify elaborate and linked patterns of malicious behavior by using the underlying graph structure. Considering the importance displayed by GNNs and their graph structure, we have conducted a comprehensive analysis to assess their strength and suitability in NIDSs. The analysis done is detailed in the next sections.

2.2.2.1 Research Methodology

A review of the latest research articles pertaining to GNN-based NIDSs was carried out to bring together the results of different studies and find any possible areas that needed more research, according to the PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) [68] framework. It consists of a collection of instructions created to enhance the reporting of systematic reviews and meta-analyses. The process has three main stages: Identification, Screening and Inclusion. A visual representation of this process is presented in Figure 2.2.

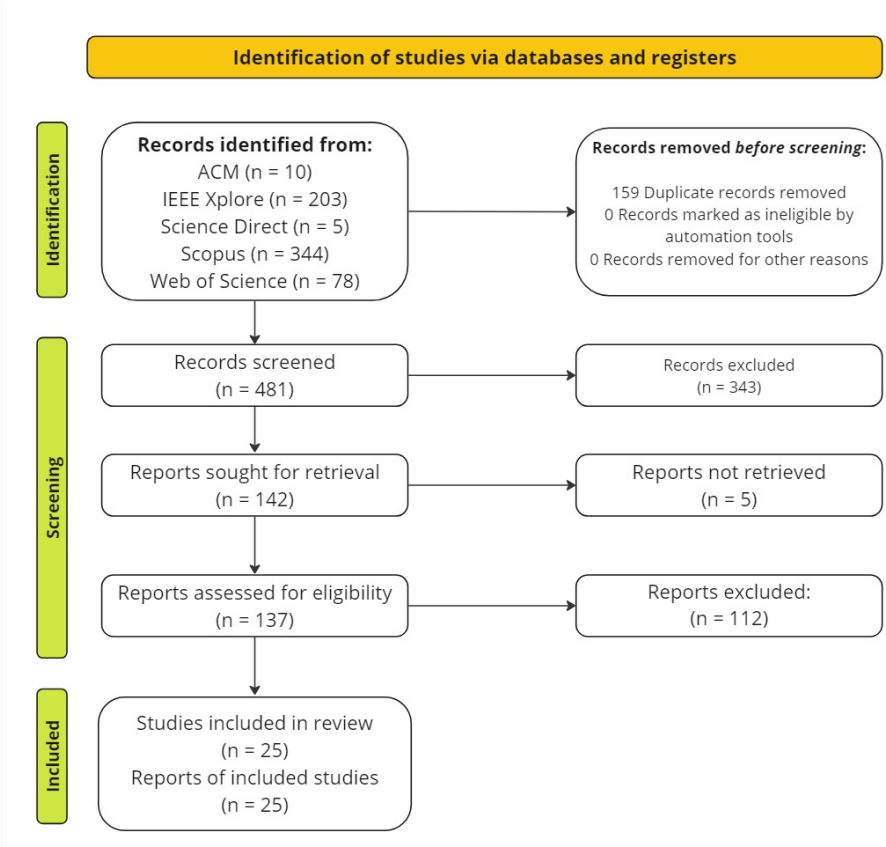

FIGURE 2.2: Systematic Review

In the Identification stage, five different search engines were chosen in order to search for relevant studies for analysis: Association for Computing Machinery (ACM) Digital Library [69], Institute of Electrical and Electronics Engineers (IEEE) Xplore [70], Science Direct [71], Scopus [72] and Web Of Science [73].

Query
”Graph Neural Networks” AND (”Machine Learning” OR ”Deep Learning” OR ”Artificial intelligence”) AND (”Intrusion” OR ”network Security” OR ”cybersecurity” OR ”*IDS”

TABLE 2.1: Search query performed

The search query presented in Table 2.1 was designed to retrieve academic papers, articles and research documents that focus on the use of GNNs within the fields of ML, DL, or AI and their application to intrusion detection, network security, cybersecurity, or related IDSs.

The query, searched on all search engines, produced a grand total of 640 records across the five previously stated databases. We identified 159 of the total entries as duplicates, leaving 481 distinct records. These unique records are presented in Table 2.2.

Search Engine	Records
ACM	1
IEEE Xplore	132
Science Direct	0
Scopus	343
Web of Science	5

TABLE 2.2: Query results

During the Screening stage, exclusion criteria were established in order to narrow down the search and identify the most pertinent publications for the research:

- **Topics not related to the field**: publications that were using GNNs in other fields (e.g. Chemistry, Mathematics, Industry);
- **Topics not related to intrusion detection**: publications that were using GNNs in the context of other topics such as Image Recognition, Network Traffic Analysis, Code/Software Analysis, GNN improvements, Anomaly Detection and others;
- **Publication not written in English**: publications written in other languages were not considered in this research.

We conducted the review of each record's title, keywords, and abstract based on these precise criteria. By using them, we deemed 343 records irrelevant and discarded them. Consequently, we sought to retrieve 142 reports that remained. We excluded 5 of those 142 reports due to their inaccessibility, leaving a total of 137 reports for eligibility evaluation. A full read of the publications determined that 112 reports met the exclusion criteria. As a result, 25 studies were deemed relevant to the research.

Based on this analysis, the following sections present a selection of articles, based on these 25 studies, and their respective contributions to the subject. As we are analyzing graph-based models, articles were grouped according to their graph type.

2.2.2.2 Heterogeneous Graphs

Pujol-Perich et al. [17] states that existing ML-based NIDS treat and classify network flows independently and that they are not as robust as they should be against adversarial attacks, considering their applicability to real networks. As a solution, they propose a new approach to detect intrusions with GRUs. It is built based on network flows, and for each flow, three nodes are created: the source host node, the destination host node and the flow node, where two undirected edges exist between the source host and flow nodes and between the flow and destination host nodes. By creating a separate flow node, the authors explain that it fits the way GNN models operate, and they will naturally better learn the embeddings from flows. The authors have implemented a non-standard MPNN to handle heterogeneity between host and flow nodes, aggregating messages using two distinct functions: one for host-to-flow edges and another for flow-to-host edges. New node embeddings are computed by learning an update function and by concatenating the previous node embedding with the message. A final MPNN is used at node-level to classify among multiple attack classes using the softmax activation function along with the categorical cross-entropy loss for training. Experiments with the CIC-IDS2017 [74] dataset show that their graph representation accurately shows the flows' properties and how they connect to each other. They also discovered that the GNN model they created can withstand two common adversarial attacks. These attacks change key flow features (like packet size and inter-arrival times) on purpose to avoid being caught. While machine learning-based NIDS experience a decrease in accuracy to 50%, the suggested model remains robust against such attacks.

According to MLTracer [75], malicious logins pose a serious threat to companies, and rule-based detection systems, as previously explained, struggle to detect new attack patterns. In addition, they require security experts to create and adapt those rules. The authors propose a GNN-based system that utilizes logs to represent login behavior as a Heterogeneous Information Network (HIN), a specialized type of heterogeneous graph, and extracts contextual semantic information using meta-paths. Each meta-path is processed using a CNN and co-attention mechanisms to learn vector representations. A fully-connected layer then predicts harmful edges by taking as input the concatenation of the embeddings of both destination and source nodes. By running different experiments using the LANL dataset [76], the authors conclude that the new model identifies malicious

logins with high accuracy and is capable of adapting to new scenarios and vectors of attack without human oversight.

APT-KGL [77] is suggested to identify Advanced Persistent Threat (APT) attacks. At the time, solutions for APT attack prevention based on provenance graphs (built with provenance data from the Operating System audit logs) attempted to detect APTs by manually designing a variety of rules based on threat knowledge, which, as seen before, was disadvantageous. The authors propose APT-KGL, an intelligent system that utilizes GNNs to detect APT attacks and operates on heterogeneous provenance graphs. The process involves analyzing a subgraph that encompasses a recently inserted node in order to gather the embedding information that was previously calculated by a GNN in previous iterations. This technique allows for the inference of novel system entities within a reasonable time frame by solely examining a subgraph. The process of combining a newly arrived node is accomplished by applying the propagation rule of the relational graph convolutional network (R-GCN) to its local subgraph. The authors tackle the task of detecting real-world APT scenarios by creating a module that produces synthetic examples of attack graphs. The graphs are generated by transforming threat reports into visual depictions. These visual depictions are subsequently combined with authentic, benign provenance graphs to enhance the reliability of the data. Multiple evaluation approaches are analyzed utilizing both proprietary data and the publicly available DARPA TC dataset [78]. APT-KGL demonstrates superior performance compared to other ML and DL-based models and performs better than rule-based APT detection systems.

2.2.2.3 Attributed Graphs

E-GraphSAGE [19] is another method of intrusion detection in IoT networks. It was the first, at the time, to use network flow data in GNN-based NIDS for these networks. Unlike node-based methods, it works by extracting edge features, where the flow of communication between two nodes is located, as well as using topological patterns from the network. For that, they changed the original, node-based, GraphSAGE model, so that it works on edge features. The original GraphSAGE algorithm [46] is modified in its input, message passing/aggregator function, and output, transitioning from a node-based algorithm to an edge-based algorithm. The aggregation and update functions are similar to the ones used on the original model. However, the new aggregator embeds sampled neighbor

edges instead of nodes. The model employs two E-GraphSAGE layers, with the aggregation function being the mean. The output edge embeddings are subjected to a softmax layer, and the output is used for supervised edge classification. Results, based on tests on the BoT-IoT [79], ToN-IoT [80], NF-ToT-IoT [81] and NF-BoT-IoT [81] datasets, show that edge-based models for intrusion detection are extremely valid and, as such, should be considered in real-word scenarios. It shows similar performance, and, in some scenarios, even better performance than the compared methods like XGBoost, Extra Tree Classifier and Ensemble.

Using a self-supervised approach, Anomal-E [18] introduces a GNN-based method that leverages edge-based features and the graph topological structure. The authors state that, at the time, most GNN-based NIDS implementations relied on labeled network traffic, restricting the NIDS's ability to detect unseen attacks, and most implementations rely on node features, which may reduce the efficiency of the systems as important edge information is left out. The system comprises two primary components: GraphSAGE and Deep Graph Infomax (DGI). DGI is used to generate positive and negative graph samples. As this is an edge-based approach, the algorithm was adapted to learn edge embeddings instead. E-GraphSAGE was chosen as the encoder to compute positive and negative embeddings and a graph summary. A discriminator function then computes two scores by comparing positive and negative embeddings with the graph summary. These scores are then used to train the model, using the data from the xx dataset. After the completion of training, the obtained graph embeddings are utilized to train four anomaly detection algorithms: PCA, IF, CBLOF and HBOS. The model was applied to two datasets, NF-UNSW-NB15-v2 [81] and NF-CSE-CIC-IDS2018-v2 [81]. The results are compared against baseline methods (e.g. Anomal-E PCA vs GraphSAGE PCA) and it clearly shows that the model performs better. The second advantage of this method is the fact that it doesn't require any labeling, which makes it more applicable to real-world scenarios.

TPE-NIDS [82] develops a novel approach for identifying individuals using a new algorithm named TPE-GraphSAGE, which uses a directed attributed graph to model network traffic to generate edge embeddings from selected nodes. According to the authors, edge embedding algorithms like this one can overcome the problems of strong randomness and ignoring important node information, among others. They design an algorithm that splits into two parts: sampling and aggregation. In the sampling stage, the algorithm uses a degree-based Top-K split-hop sampling method to obtain the node index of

the top k. That allows the algorithm to extract information from the selected nodes. A gate operation is applied, and a sampling node matrix is obtained. In the aggregation stage, the edge features are aggregated layer by layer according to the sampling results, and the updated nodes are embedded in series to generate edge embedding. It uses a maxpooling aggregator, and the result is concatenated with the target node in series, and then the embedded representation of the final target node is obtained through a layer of nonlinear transformation. A softmax layer will convert the output into specific category probabilities. Finally, the TPE-GraphSAGE method is utilized for NIDS using the data in the UNSW-NB15 dataset [57], thus converting the issue of network intrusion detection into the task of edge categorization. Given that this algorithm relies on sampling and does not account for all graph information, the results demonstrate relatively high values for the F1-score metric in both binary and multi-classification.

2.2.2.4 Spatio-temporal Graphs

Euler [83] presents the first usage, at the time, of temporal graph link prediction for intrusion detection, in an unsupervised manner, capable of detecting network lateral movements. It is a versatile framework specifically developed for detecting lateral movements using GNNs in temporal graphs. It utilizes the temporal relationships within the graphs to predict abnormal links. The authors argue that the temporal structure is a significant characteristic that enhances the accuracy of APT predictions. They suggest implementing a distributed architecture using the leader/worker model to simultaneously carry out the time-consuming message-passing procedures. The leader distributes graph snapshots to many workers, who are tasked with loading the graphs into memory and calculating the embeddings using a specified GNN encoder (e.g. GAT, GCN and others). The snapshot embeddings are subsequently returned to the leader for temporal dimension capture using an RNN block (e.g. GRU and LSTM). The outcomes, based on experiments ran on the LANL dataset [76], are then distributed to the workers for decoding and anomaly detection. The paper shows that EULER can create highly precise IDSs through temporal graphs, performing very well or even better against similar models.

FTG-NET [67] brings a new approach to DDoS detection. At the time, most papers and ML methods were based on flow and traffic information and ignored topological connections. The authors introduced a new graph structure called Flow-to-Traffic Graph (FTG), which combines flow-level and aggregated traffic-level structures in a two-level

hierarchical representation. Additionally, the authors propose a GNN model, FTG-Net, to analyze these detailed-to-general graphs and classify flows as either legitimate or malicious. The conversion of traffic data creates a unique hierarchical graph structure known as FTG. This structure provides detailed information on flows and aggregated traffic. The implemented FTG structures enable FTG-Net to identify DDoS attacks. The FTG graph structure is constructed using time-divided traffic data and comprises a high-level Traffic Graph for each time slot, with each node having a corresponding low-level Flow Graph. Each level in the FTG-Net is associated with a distinct GNN model for processing. The FTG-NET framework is comprised of three distinct steps: initially, the traffic data from the CIC-IDS2017 [74] dataset undergoes a conversion process to transform it into FTG structures. During the second stage, the Flow-level Graphs undergo processing by the Flow GNN, resulting in the generation of an embedded representation vector for each graph. The vectors serve as node attributes in the Traffic-level Graphs, which are then analyzed by the Traffic GNN in the third stage. The Traffic GNN generates the final predictions for each flow. Both GNNs utilize three GCN layers. Results show that the proposed traffic structure is enough to obtain results comparable with state of-the-art approaches.

2.2.2.5 Dynamic Graphs

Paudel et al. [84] addresses the problem of the usage of static snapshots to learn from dynamic networks. Static snapshots ignore temporal information and lead to the loss of important information. For that, the authors propose PIKACHU, which takes into account the temporal feature of graphs and uses that to capture short and long temporal dependencies. The authors represent the network as a sequence of graphs, utilizing time to combine several graph snapshots, each potentially exhibiting distinct topologies. Both random walks and the Skip-Gram model are utilized to build embeddings that capture the spatial and temporal relationships in the network. The authors categorize an edge as anomalous when it has a low likelihood of existing between two nodes at a specific timestamp. Based on this definition, a anomaly score is calculated for each edge. The acquired embeddings are shown to be efficient in acquiring the chronological sequence of edges, which is crucial for detecting network abnormalities such as lateral movement and APT. Experiments on two datasets, LANL [76] and OpTC [85], show that the model achieves a significant reduction of False Positives and an improvement in the Area under

the ROC Curve (AUC) metric, in comparison to other models, showing the importance of using temporal information in intrusion detection scenarios.

DynAD, introduced by D. Zhu et al. [86], aims to identify anomalous edges in dynamic networks. At the time, the author states that most of the research focused on learning the node embeddings, but, as it is a dynamic network, the node-based methods are sensitive to frequent changes of nodes. DynAD is a method designed to detect intrusions based on anomalous edge detection. It uses GCN to capture both the network topology and node properties of the current snapshot, and combines them into a single representation. To transmit the historical network status to the current representation, the authors incorporate the output and weight matrix of GCN into a GRU, similar to the method presented by Pareja et al. [87]. A weight matrix is obtained at a new timestamp, which includes the previous node state. Additionally, an attention mechanism is employed to emphasize the disparities between the weight matrices in short-term snapshots. Ultimately, the GCN output node embedding will be employed to compute the anomalous probability of each edge in the related snapshot. The results, following experiments ran on three datasets, UCI messages [88], arXivhep-th [89] and Digg [90], show a significant improvement over the state-of-the-art baseline methods in anomaly detection.

2.2.2.6 Summary of studies

Table 2.3 summarizes the studies that were analyzed. In this table, *Graph Type* indicates the type of graph used, *Classification* refers to the downstream classification task, *Type of learning* represents the learning method, *Model* indicates the types of models used in the study, *Year* refers to the publication year, and *Datasets* provides the datasets used in the study.

Paper	Graph Type	Classification	Type of Learning	Model	Datasets	Year
D. Zhu et al. [86]	Dynamic Graph	Edge	Unsupervised	GCN GRU Attention mechanism	UCI Messages arXiv hep-th Digg	2021
W. Yang et al. [91]	Spatio-temporal Graph	Edge	Supervised	SCGN Encoder-Decoder GAT	LANL CERT	2022
H. Zhang et al. [92]	Attributed Graph	Graph	Unsupervised	GNN	CAN Signal Extraction and Translation Dataset	2023
H. Zhu and J. Lu [93]	Attributed Graph	Edge	Supervised	GNN Line-GraphSAGE	ToN IoT BoT-IoT UNSW-NB15	2022
Y. Zhang et al. [82]	Attributed Graph	Edge	Supervised	TPE-GraphSAGE	UNSW-NB15	2022
Caville et al. [18]	Attributed Graph	Edge	Self-Supervised	E-GraphSAGE	NF-UNSW-NB15-v2 NF-CSE-CIC-IDS2018-v2	2023
W. W. Lo et al. [19]	Attributed Graph	Edge	Supervised	E-GraphSAGE	BoT-IoT ToN-IoT NF-ToT-IoT NF-BoT-IoT	2022
Pujol-Perich et al. [17]	Heterogeneous Host-Connection Graph	Node	Supervised	GNN GRU	CIC-IDS2017	2022
Kapoor et al. [94]	Heterogeneous Provenance Graph	Node	Supervised	R-CGN HAN		2022
Nguyen and Kashef [24]	Attributed Graph	Edge	Self-Supervised	GNN GAT	NF-BoT-IoT NF-ToN-IoT NF-CSE-CIC-IDS2018-v2 NF-UNSW-NB15-v2	2023
King and Huang [83]	Temporal Graphs	Edge	Unsupervised	GAT GCN GraphSAGE	LANL 2015	2023
Li et al. [95]	Heterogeneous Graph	Node	Supervised	GCN	ToN-IoT	2023
Dang and Nguyen [96]	Attributed Graph	Edge	Supervised	GNN	UNSW-NB15	2023
Gorricho-Segura et al. [97]	Attributed Graph	Edge	Supervised	E-GraphSAGE	MedBIoT BETH	2023
Himmelhuber et al. [98]	Attributed Graph	Edge	Unsupervised	GraphSAGE	OPC-UA server data	2022
Zhang et al. [99]	Heterogeneous Graph	Node	Supervised	ACGAN-GNN	CIC-IDS2017	2022
Venturi et al. [100]	Attributed Graph	Node	Semi-supervised	ARGA Autoencoder GAE GCN	CTU-13 ToN_IoT	2023
Duan et al. [101]	Spatio-temporal Graph	Edge	Semi-supervised	DGNN Line Graph	NF-BoT-IoT NF-ToN-IoT NF-CSE-CIC-IDS2018 NF-BoT-IoT-V2 NF-ToN-IoT-V2 NF-CSE-CIC-IDS2018-V2	2023
Barselloti et al. [67]	Temporal Graph	Node	Supervised	GCN	CIC-IDS2017	2023
Day et al. [102]				MLP	CIC-IDS2017 Lawrence Berkeley National Laboratory FTP	2023
Lin et al. [103]	Attributed Graph	Node	Supervised	GNN GCN	CIC-AndMal2017	2023
Zhou et al. [104]				GCN JK-NET	UNSW-SOSR2019	2022
Purnama et al. [105]	Attributed Graph	Edge	Supervised	E-GraphSAGE	ToN-IoT	2022
Jia and Zheng [106]	Attributed Graph	Node	Supervised	GCN	UNSW-NB15	2023
Lakha et al. [107]	Attributed Graph	Node	Unsupervised	NWR-GAE	BETH	2022

TABLE 2.3: State of the art papers on GNN-based NIDSs

The Figure 2.3 presents the number of NIDS related papers that made use of GNNs throughout the years.

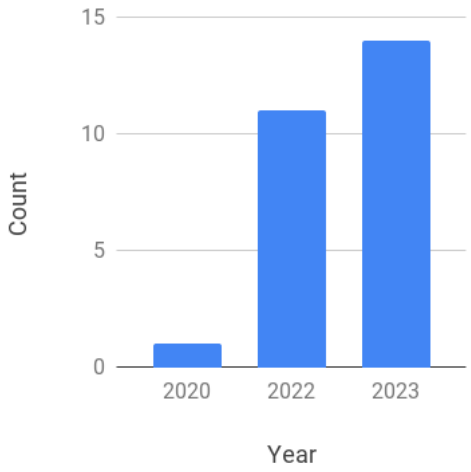

FIGURE 2.3: Yearly distribution of articles on GNN-based NIDSs

Looking at the papers presented in Table 2.3, there are some interesting questions that deserve a deeper analysis. As it is observable, GNNs application in NIDSs systems is a novel topic, as all the papers are very recent, with the oldest paper being published in 2020. It is equally noticed the uptick in publications on the topic, with the majority of the papers being published in 2023.

Another notable concern, however, not readily apparent in Table 2.3, is the limited quantity of publicly accessible datasets and their lack of variety. Most of them concentrate on either network-based or host-based data, but not both simultaneously. The collected data snapshots are taken at very brief time intervals (e.g. the CIC-IDS2017 dataset only contains data collected over a span of five days). The long-lasting nature of attacks [108] and the low frequency of intrusions in data can lead to class imbalances, affecting models' ability to identify intrusions. Additionally, the lack of diversity in datasets, which often include information from a uniform network environment, can make them obsolete and make it difficult to keep up with new attacks, causing them to become obsolete.

We can also observe that authors tend to use attributed graphs and create models that classify nodes. This rather simplistic approach makes sense: an IP address and a port identify nodes when building a graph, and an attributed graph can store all that information. Others (e.g. E-GraphSAGE [19]) choose to view the problem as an edge classification task. Edges store and represent the flow of communication between two nodes, and, as such, contain vital information. Some of that information is:

- **Type of communication protocol**: different protocols have different characteristics and security implications;

- **Timestamp or duration**: useful for identifying temporal patterns and anomalies, such as unusually long or short communication periods;
- **Flow volume**: significant deviations in volume can indicate potential attacks such as data exfiltration or denial-of-service attacks;
- **Flow hashes or signatures**: unique identifiers for flows based on certain characteristics. It is useful for quickly matching against known malicious patterns or signatures.

Finally, a closer look into the model usage brings to light some questions that deserve further exploration, as displayed in Figure 2.4.

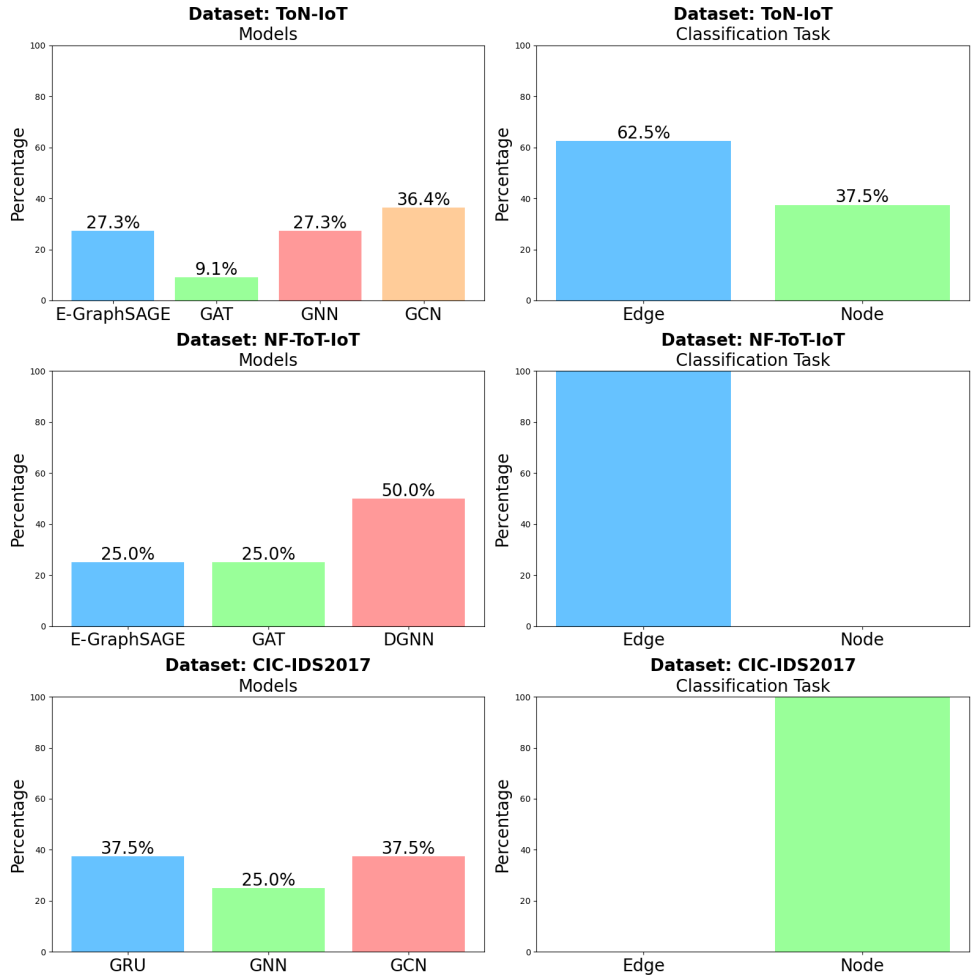

FIGURE 2.4: Distribution of Models and Classification Tasks

According to information presented, edge classification task is not applied, to the best of our knowledge and according to the research, to the CIC-IDS2017 dataset. Moreover, it is also observable in Figure 2.4 that papers that used this dataset did not use E-GraphSAGE or GAT-based models.

Based on these findings, we will address these gaps in the literature. As will be seen throughout this document, we will first explore the datasets available to us and the information that they provide, select a subset and submit them to pre-processing steps. Next, we will create, develop and enhance our models with the goal of obtaining the best results possible. Finally, we will present and discuss the results, highlighting some of the advantages of using a graph-based structure.

Chapter 3

Data Processing and GNN Models Development

Building on the identified gaps in the current literature, this chapter focuses on the important steps of dataset selection and pre-processing of the data, a crucial process in ML. Subsequently, we move on to the process of defining and developing models for intrusion detection. We begin by carefully selecting and pre-processing the datasets, described in Section 3.1, ensuring they are well-suited for our analysis. This involves cleaning, normalizing, and transforming the data to facilitate the training and evaluation of our models. Next, we define and create the models, indicated in Section 3.2, emphasizing the implementation of advanced GNN architectures. This chapter details the methodologies and technical considerations involved in these processes, in preparation for the subsequent analysis and evaluation of our proposed models.

3.1 Datasets and Pre-Processing

The choice of any model in the context of intrusion detection is fundamentally influenced by the selection and pre-processing of the datasets. Choosing relevant and comprehensive datasets is critical as it provides the necessary variety and depth of network traffic patterns essential for training accurate GNN models. Data pre-processing, which includes cleaning, normalizing, and transforming the raw data, is equally crucial as it ensures the data quality and consistency required for effective model training. This process facilitates the construction of meaningful graphs by defining appropriate nodes, edges, and

attributes that accurately represent network interactions and behaviors. Considering multiple datasets is crucial for developing robust IDSs. This approach ensures coverage of diverse attack scenarios and enhances the versatility of the IDS across different network environments. It also provides comprehensive benchmarking, helps identify strengths and weaknesses in detection methods, and ensures adaptability to various data formats and feature sets.

Looking at the research presented in Table 2.3, multiple datasets are available. In this work, we chose the CIC-IDS2017 dataset [74], in its original version, due to its low level of exploration and lack of understanding on the implementation of GNN models based on edge features. The ToN-IoT dataset [80] was chosen for its comprehensive representation of modern network traffic, including both benign and malicious activities. This dataset provides a rich variety of IoT scenarios and attack vectors, making it ideal for training and evaluating advanced intrusion detection models. The NF-BoT-IoT [81] dataset was selected due to its extensive coverage of various IoT network activities. In addition, a key feature of this dataset is its use of the NetFlow protocol, a network protocol developed by Cisco for collecting IP traffic information and monitoring network flow. The NetFlow protocol is crucial in the context of intrusion detection as it provides detailed insights into network traffic patterns, such as source and destination IP addresses, source and destination port numbers, packet and byte counts, flow duration, timestamps, and protocol type. These last two datasets are widely used, as seen in Table 2.3, and, as such, we can compare our results with the many existing implementations.

In summary, the CIC-IDS2017, ToN-IoT, and NF-BoT-IoT datasets were selected from the available datasets. In the following subsections, a small description and the preprocessing steps executed for each dataset are provided, to ensure data quality, enhance feature relevance, and to prepare the data for effective model training.

3.1.1 CIC-IDS2017

The CIC-IDS2017 dataset [74], developed by the Canadian Institute for Cybersecurity, is a widely used dataset for evaluating and benchmarking intrusion detection systems. Collected over five days in July 2017, it simulates a real-world network environment with various machines and network configurations, capturing a blend of normal and attack traffic. The dataset includes a wide range of attacks such as Brute Force, Heartbleed, DoS,

DDoS, web attacks (e.g., SQL injection, XSS), infiltration, botnet, and port scanning activities. While valuable for intrusion detection research, it has several shortcomings. Firstly, it is synthetic, which means it may not fully capture the complexities of real-world network traffic. The dataset suffers from class imbalance, with normal traffic outnumbering attack traffic, potentially leading to biased machine learning models. There are redundant features, such as multiple representations of header lengths, which complicate feature selection. The dataset lacks contextual information such as user behavior and application-level details, crucial for comprehensive intrusion detection. Because it is a 2017 static dataset, it does not include updates or new attack patterns, limiting its relevance for detecting current threats. Finally, the oversimplification of synthetic traffic patterns in comparison to real-world variability may impact the generalizability of the model.

As seen earlier, the dataset contains a gargantuous amount of features and data and suffers from a considerable class imbalance issue. As such, some pre-processing steps were carried out to adjust the data to the task at hand:

1. **Reading CSV Files**: processing of all CSV files, excluding files related to Monday as they contain only benign traffic. This helps in addressing class imbalance issue;

2. **Sorting Data by Timestamp**: sorting of data by the timestamp column. This ensures that the traces are processed in chronological order;

3. **Batching**: collection of rows of data into an array based on a counter criterion. This groups the data into manageable batches;

4. **Feature and Label Extraction**: selection of features and labels from the constructed graph. This involves collecting node attributes (network features), obtaining labels indicating whether a connection is benign or an attack, and processing adjacencies to understand connections between nodes. Only a small subset of features was chosen as input, according to the feature selection performed by the dataset's authors [74];

5. **Random Undersampling**: adaption of the data in a batch: if all labels in the current batch are benign, we randomly include only 10% of such batches. This helps in addressing class imbalance by reducing the number of benign-only batches;

6. **Shuffling and Splitting Data**: shuffling and separation of the data into training and evaluation sets (70% for training, 30% for testing);

7. **Converting attack labels**: conversion of categorical features into a numeric format. For the binary classification experiment, the attacks were encoded to binary values, normal and attack traffic as 0 and 1, respectively;

8. **Writing to CSV Files**: the training and testing graphs are written into separate CSV files.

Overall, the aforementioned steps allowed us to tackle the main issues of class imbalance and feature selection present in the CIC-IDS2017 dataset. As such, the data is ready to be used to train our GNN models for intrusion detection as we will discuss in Section 3.2.

3.1.2 ToN-IoT

The ToN-IoT dataset [80] is a cybersecurity dataset designed to evaluate intrusion detection systems, particularly in IoT and IIoT environments. It includes data from various sources, such as network traffic, operating system logs, and telemetry data from IoT devices. This dataset captures both normal and attack scenarios, including DDoS, ransomware, backdoor, and data exfiltration attacks. The diversity of data sources makes it ideal for studying the interaction between different layers of network infrastructure and the various types of cyber threats that can affect them.

ToN-IoT's major drawback is its synthetic generation, which may not fully represent real-world IoT network behaviors. The dataset also suffers from an imbalance between attack and normal traffic, potentially biasing model performance. Its limited variety in device types and communication protocols can reduce the generalizability of derived models.

Given these drawbacks, we implemented the following steps for pre-processing:

1. **Data Cleaning**: removal of any records with missing values or corrupted data entries. For example, if a record lacks essential information like source or destination IP addresses, it should be excluded;

2. **Normalization**: normalization of features like payload size and response time;

3. **Feature Selection**: extraction of significant features such as device ID, device type, communication protocol (e.g., MQTT, HTTP), payload size, and others;

4. **Label Encoding**: conversion of categorical features like device types and protocols into numerical formats;

5. **Train-Test Split**: separation of the resulting data into training and testing sets (70% for training, 30% for testing);

6. **Writing to CSV Files**: storage of the training and testing graphs into separate CSV files.

These pre-processing steps allowed us to improve the overall data quality on the ToN-IoT dataset, by removing unwanted data entries. It is expected an improvement in the model's performance.

3.1.3 NF-BoT-IoT

The NF-BoT-IoT dataset [81] focus specifically on NetFlow data. It captures flow-based features such as source and destination IPs, ports, and the volume of data transferred, along with labels indicating normal or attack traffic. This dataset is particularly useful for developing flow-based intrusion detection systems, which analyze patterns in network traffic to detect anomalies and potential security breaches. It supports the creation of scalable and efficient IDS solutions in large-scale IoT and IIoT networks by focusing on NetFlow data.

NF-BoT-IoT's reliance on NetFlow data might not capture the full spectrum of packet-level details, which are crucial for certain types of anomaly detection. The dataset's aggregation level could hide important temporal patterns needed for accurate intrusion detection. It also shares the synthetic nature issue, potentially reducing its effectiveness in real-world applications. Finally, the low number of attack vectors might not reflect the diverse and evolving nature of cyber threats.

The dataset requires pre-processing to manage its large set of features and class imbalance. Filtering relevant attributes and balancing benign and malicious traffic representation are crucial for effective intrusion detection model training. The following pre-processing steps were executed:

1. **Data Cleaning**: removal of incomplete or inconsistent NetFlow records. For example, discard flows where essential fields like source or destination IP, or byte count, are missing;

2. **Feature Transformation**: transformation of NetFlow features such as byte counts, packet counts and flow duration into more meaningful metrics, like average packet size or flow density;

3. **Feature Selection**: extraction of significant features such as source and destination IP addresses, port numbers, packet and byte counts, flow duration, protocol type, and payload size.

4. **Normalization**: normalization of features such as byte counts and packet counts to a standard range;

5. **Label Encoding**: conversion of categorical features like service types (e.g., HTTP, DNS) into numerical formats;

6. **Train-Test Split**: Split the dataset into training and testing sets (70% for training, 30% for testing);

7. **Writing to CSV Files**: storage of the training and testing graphs into separate CSV files.

To improve the quality of the NF-BoT-IoT dataset, we employed data cleaning techniques to remove incomplete/empty flows and used feature transformation methods to transform features into more meaningful values. Together, these methods will improve the performance of our models, presented and discussed in the next section.

3.2 Models Definition

Upon conducting a comprehensive analysis of the dataset structures and addressing their constraints, we proceeded to develop our models. In this section, we discuss the precise details of our approach for creating the models. This involves creating a foundational model and investigating other GNN models, such as GCN, GAT, and E-GraphSAGE. Every model is specifically developed to use distinct characteristics of graph-based learning, particularly GAT's attention mechanism and E-GraphSAGE's sampling strategies.

3.2.1 Base Model Creation

Before developing the models, it was important to define a meaningful graph structure, as it is the basis for GNNs. As such, we chose a simple structure in order to simplify the

implementation, with nodes representing the hosts and edges representing their connections, along with all associated flow information. For the first iteration of the model, it made sense to create one that was comparable to the literature, so that there is a basis for comparison, to assure that the results are accurate, and the implementation is correct. As such, the initial model that was created was similar in its architecture to the model by Pujol-Perich et al. [17]. At the core of the model are two GRU cells, which are crucial for the message-passing mechanism. Each GRU cell receives the current state and aggregated messages from neighboring nodes, subsequently producing updated node states. Two message functions facilitate the message-passing process. We implement these functions as sequential neural network layers, incorporating dropout layers for regularization, fully connected layers for feature transformation, and ReLU activation functions to introduce non-linearity. The aggregation step uses these message functions, which operate on concatenated node features to produce informative messages. The model executes iterative message passing, exchanging messages between nodes in each iteration. In this phase, a mean function aggregates messages to represent the combined information from neighboring nodes. The GRU cells then utilize these aggregated messages to update the node states, iteratively refining the node representations over multiple iterations. The final step in the forward pass involves a readout neural network. The network, fully connected, consists of three layers: the first layer with 128 neurons, the second layer with 64 neurons, followed by ReLU activations and dropout layers, and the final output layer with a number of neurons matching the number of classification categories, which generates the model's predictions. The model undergoes multiple epochs of training, updating its parameters through backpropagation and the Adam optimizer. The loss is computed using *CrossEntropyLoss*, and various metrics such as accuracy, weighted F1 score, and macro F1 score are evaluated to monitor performance. Additionally, the model includes a learning rate scheduler that adjusts the learning rate using an exponential decay function, ensuring optimal convergence during training. The initial parameters are indicated in Table 3.1.

Parameter	Description
node_state_dim	Dimension of the links' hidden state
iterations	Number of message passing iterations
readout_units	Number of readout units
learning_rate	Learning rate used by the Exponential Decay
decay_steps	Decay steps used by the Exponential Decay
decay_rate	Decay rate used by the Exponential Decay
epochs	Number of epochs

TABLE 3.1: Parameters of the initial model

3.2.2 Alternative GNN models: GCN, GAT and E-GraphSAGE

Next, an attempt was made to create different GNN models by modifying the original model to use common models like GAT, GCN and GraphSAGE. With the first two models, the goal would be to compare the created model against the literature and obtain the results. The latter one is crucial, as it is one of the goals of this study: to run experiments on the CIC-IDS2017 dataset for edge classification tasks.

To create the GCN model, two GCN layers were created to replace the GRU ones. The message passing function was changed to make use of these GCN layers, and the GRU-based state updates were replaced with convolution-based updates using the new layers. The ReLU activation function was used to introduce non-linearity. The initialization of node states remains the same. The training and evaluation methods remain largely unchanged, focusing on using the new GCN-based forward method. These changes adapt the model to use GCN layers, enabling it to aggregate information from neighboring nodes effectively.

For the creation of the GAT model, two GAT layers were introduced to replace the GRU cells. The message-passing function was updated in the same way as in the previous model, but this time using the GAT layers. The newly created layers replaced GRU-based state updates with attention-based updates. The ReLU activation function was used to introduce non-linearity. The training and evaluation methods remain largely unchanged, with an emphasis on using the new GAT-based forward method. Further variables were considered, as displayed in Table 3.2:

Parameter	Description
num_heads	The number of attention heads
head_state_dim	The number of neurons per attention head
attention_dropout	Dropout rate applied to attention coefficients

TABLE 3.2: Additional parameters for the GAT model

For our E-GraphSAGE model, a similar approach used by the original authors of the model [19] was followed: a model with two E-GraphSAGE layers, with neighbor information being aggregated from a two-hop neighborhood and the use of the mean as the aggregation function. Regarding sampling, full neighborhood sampling was chosen for the ToN-IoT and NF-BoT-IoT datasets. Given the size of the CIC-IDS2017 dataset, only 50% of the neighbors' information was taken into account. We chose the ReLU function as the activation function to introduce non-linearity. Additional parameters were considered, as referenced in Table 3.3:

Parameter	Description
num_layers	Number of layers
hidden_units	Number of hidden units
neighbor_sample	The number of neighbors sampled during the aggregation phase

TABLE 3.3: Additional parameters for the E-GraphSAGE model

In this chapter, we addressed the identified gaps in the literature. We started by selecting and pre-processing datasets, and after that we proceeded with the creation of our models along with their initial parameters. The outcomes of the implementation of these models will be examined and discussed in the subsequent chapter.

Chapter 4

Experimental Results and Discussion

After analyzing and preparing the various model types and datasets, it was time to conduct the experiments, considering the goals that were put forth at the beginning of this thesis. As such, in this chapter, we present the experimental results and discussion focused on evaluating the performance of various models across the three aforementioned datasets. In Section 4.1 we begin by detailing the experimental setup used to train and test the models on these datasets. Subsequently, in Sections 4.2, 4.3 and 4.4, we present and analyze the performance of each model on the individual datasets, highlighting their effectiveness in detecting intrusions. Furthermore, in Section 4.5, we conduct a more generic analysis of the results. Finally, in Section 4.6, we explore some of the advantages of GNNs over traditional ML methods in the context of intrusion detection.

4.1 Experimental Setup

Among the various libraries available for developing ML models in Python [109], PyTorch [110] was selected above alternatives such as TensorFlow [111] due to its widely recognized simplicity, readability, user-friendliness, and extensive documentation. PyTorch was also chosen due to its compatibility with PyG [112], a library that extends PyTorch's capabilities for creating and training GNNs using structured data for different applications. PyG contains many methods for applying DL on graphs and other non-uniform structures.

Experiments were run on a machine with the specifications displayed in Table 4.1.

Specification	Value
Operating System	Windows 11
CPU	AMD Ryzen 9 3900X 12 cores
GPU	NVIDIA GeForce RTX 3060
RAM	32GB
CUDA	12.1

TABLE 4.1: Machine specifications

In addition, the technologies and libraries used for these experiments are represented in Table 4.2.

Software	Version
Python	3.12
PyTorch	2.3.1
PyTorch Geometric (PyG)	2.3

TABLE 4.2: Software used

For the initial experiment, the same initial parameters were utilized for all models, as indicated in Table 4.3.

Parameter	Value
node_state_dim	128
iterations	8
readout_units	256
learning_rate	0.001
decay_steps	30
decay_rate	0.6
epochs	50

TABLE 4.3: Common parameters and values for all initial models

As previously mentioned, with each model come different parameters. For the GAT and E-GraphSAGE models, the following values were used initially, as presented in Table 4.4.

Model	Parameters
GAT Model	*num_heads*: 1
	head_state_dim: 8
	attention_dropout: 0.001
	num_layers: 2
E-GraphSAGE Model	*hidden_units*: 64
	neighbor_sample: 25

TABLE 4.4: GAT and E-GraphSAGE specific parameters

First, we conducted experiments on each dataset using the default parameter values. We experimented with both multi-class and binary classification tasks, as well as node and edge classification tasks, to evaluate the capabilities of our models across different scenarios. We then performed additional experiments to fine-tune the hyperparameters. Finally, we assessed the most optimal outcomes achieved and compared them to the existing findings in the literature. To evaluate the generalization capability of our models and reduce the risk of overfitting, we employed 10-fold cross-validation. This technique involves partitioning the dataset into 10 equally-sized folds. We averaged the performance metrics across all folds to obtain an assessment of the model's performance. We evaluated different values for k: 3, 5 and 10. The choice of $k = 10$ provided us with the best outcomes.

4.2 CIC-IDS2017

The experiments conducted on the CIC-IDS2017 dataset serve two main objectives: firstly, to develop and evaluate models that utilize edge features, and secondly, to compare and seek enhancements over existing models documented in the literature. With the two remaining datasets, the objective was to investigate the models and improve intrusion detection results. For the CIC-IDS2017 dataset, we used the four presented models along with their respective initial values. The results are presented in Table 4.5.

Model	Results	Classification Task	Classification Type
GRU	**Loss:** 0.105 **Accuracy:** 0.97473 **Weighted F1: 0.97404** **Macro F1:** 0.63046	Node	Binary
GRU	**Loss:** 0.129 **Accuracy:** 0.95982 **Weighted F1:** 0.95352 **Macro F1:** 0.64487	Node	Multi-class
GCN	**Loss:** 0.109 **Accuracy:** 0.97635 **Weighted F1: 0.97560** **Macro F1:** 0.62828	Node	Binary
GCN	**Loss:** 0.114 **Accuracy:** 0.96994 **Weighted F1:** 0.93712 **Macro F1:** 0.597881	Node	Multi-class
GAT	**Loss:** 0.237 **Accuracy:** 0.94704 **Weighted F1: 0.95387** **Macro F1:** 0.58986	Node	Binary
GAT	**Loss:** 0.311 **Accuracy:** 0.93144 **Weighted F1:** 0.94896 **Macro F1:** 0.61076	Node	Multi-class
E-GraphSAGE	**Loss:** 0.1755 **Accuracy:** 0.93089 **Weighted F1:** 0.**95086** **Macro F1:** 0.55542	Edge	Binary
E-GraphSAGE	**Loss:** 0.1942 **Accuracy:** 0.93712 **Weighted F1:** 0.9245 **Macro F1:** 0.5089	Edge	Multi-class

TABLE 4.5: Initial results for the CIC-IDS2017 dataset

The models evaluated—GRU, GCN, GAT, and E-GraphSAGE— show varying performance levels across binary and multi-class classification tasks reflecting their unique strengths and limitations in handling this dataset. The initial parameters used for these experiments are presented in Tables 4.3 and 4.4. We can attribute the GRU model's strong performance in both binary and multi-class classification tasks to its effective handling of sequential data and its ability to capture temporal dependencies within network traffic. For binary classification, the GRU achieved a high accuracy of 0.97473 and a Weighted F1 score of 0.97404, indicating robust performance. The Macro F1 score of 0.63046, while

lower, still reflects reasonable handling of class imbalance. In the multi-class setting, the GRU's accuracy drops to 0.95982 with a Weighted F1 of 0.95352 and a slightly higher Macro F1 of 0.64487, suggesting the model's robustness but highlighting the increased complexity of distinguishing between multiple classes. The GCN model leverages the graph structure of the data, excelling in node classification tasks. Its binary classification results are impressive, with an accuracy of 0.97635 and a Weighted F1 score of 0.97560, slightly higher than GRU. However, its Macro F1 score of 0.62828 indicates challenges with class imbalance. For multi-class classification, GCN's accuracy remains high at 0.96994, but the Weighted F1 score drops to 0.93712, and the Macro F1 score further declines to 0.597881. This suggests that while GCN effectively captures relational information, it struggles more with the increased complexity and class imbalance in multi-class settings. The GAT model introduces attention mechanisms to focus on important nodes, yet it shows relatively lower performance metrics. In binary classification, GAT achieves an accuracy of 0.94704 and a Weighted F1 score of 0.95387, but its Macro F1 score is 0.58986, reflecting difficulties in distinguishing minority classes. The multi-class classification results are lower, with an accuracy of 0.93144, a Weighted F1 score of 0.94896, and a Macro F1 score of 0.61076. The results reflect the model's sensitivity towards noisy data, leading to lower metric results. E-GraphSAGE focuses on edge prediction and performs the least effectively in the classification tasks. Its binary classification accuracy is 0.93089, with a Weighted F1 score of 0.95086 and a Macro F1 score of 0.55542. For multi-class classification, the model shows a slight improvement in accuracy at 0.93712 but a drop in the Weighted F1 score to 0.9245 and a significantly low Macro F1 score of 0.5089. As it edge-based classification model, the results potentially indicate that feature selection was not done properly and further processing of those features may be required.

Subsequently, we advanced to the stage of hyperparameter tuning. Given the extensive range of parameters accessible, there are several alternatives for tuning hyperparameters. Therefore, we decided to explore only a limited portion of these possibilities. In addition, after conducting a literature review and analyzing the data presented in Table 2.3 and Figure 2.4, it was decided to narrow down the number of models being examined to focus on GAT and E-GraphSAGE. As it is visible in the data, the GAT model was not found in the existing literature for this dataset, while the E-GraphSAGE model is essential for conducting edge classification on the dataset. To maintain consistency across experiments, even in other datasets, we decided to only consider these two models going

forward. The GRU and GCN models were extensively used, and further exploration was considered unnecessary.

The Table 4.6 displays the results, where the *hyperparameters* column shows the hyperparameters that were changed and their value.

Model	Hyperparameters	Results	Classification Task	Classification Type
GAT	*epochs*: 60 *num_heads*: 1	**Loss:** 0.230 **Accuracy:** 0.94729 **Weighted F1:** 0.95392 **Macro F1:** 0.58989	Node	Binary
GAT	*epochs*: 70 *num_heads*: 1	**Loss:** 0.227 **Accuracy:** 0.93735 **Weighted F1:** 0.93361 **Macro F1:** 0.62312	Node	Multi-class
GAT	*epochs*: 60 *num_heads*: 4	**Loss:** 0.219 **Accuracy:** 0.95741 **Weighted F1: 0.95427** **Macro F1:** 0.64874	Node	Binary
GAT	*epochs*: 70 *num_heads*: 4 *head_state_dim*: 16	**Loss:** 0.311 **Accuracy:** 0.94651 **Weighted F1: 0.94496** **Macro F1:** 0.61076	Node	Multi-class
E-GraphSAGE	*epochs*: 60 *neighbor_sample*: 10	**Loss:** 0.1709 **Accuracy:** 0.95736 **Weighted F1: 0.95492** **Macro F1:** 0.55661	Edge	Binary
E-GraphSAGE	*epochs*: 70 *neighbor_sample*: 50	**Loss:** 0.18869 **Accuracy:** 0.93712 **Weighted F1:** 0.9245 **Macro F1:** 0.5089	Edge	Multi-class
E-GraphSAGE	*epochs*: 70 *neighbor_sample*: 10	**Loss:** 0.1755 **Accuracy:** 0.95532 **Weighted F1:** 0.94732 **Macro F1:** 0.54810	Edge	Binary
E-GraphSAGE	*epochs*: 70 *neighbor_sample*: 50 *hidden_units*: 128	**Loss:** 0.1942 **Accuracy:** 0.94662 **Weighted F1: 0.94051** **Macro F1:** 0.5244	Edge	Multi-class

TABLE 4.6: Hyperparameter tunning results for the CIC-IDS2017 dataset

Initially, the GAT model with default parameters showed an accuracy of 0.94704, a Weighted F1 score of 0.95387, and a Macro F1 score of 0.58986 for binary classification. In multi-class classification, GAT achieved an accuracy of 0.93144, a Weighted F1 score

of 0.94896, and a Macro F1 score of 0.61076. These results indicated that GAT struggled with the complexity of multi-class tasks and class imbalance. After hyperparameter tuning, adjusting epochs to 60 and the number of attention heads to 4, the GAT model's performance in binary classification improved to an accuracy of 0.95741, a Weighted F1 score of 0.95427, and a Macro F1 score of 0.64874. This improvement highlights the impact of increasing attention heads, allowing the model to better focus on relevant features. For multi-class classification, tuning to 70 epochs, 4 attention heads, and a head state dimension of 16 resulted in an accuracy of 0.94651, a Weighted F1 score of 0.94496, and a Macro F1 score of 0.61076. While the accuracy and Weighted F1 score showed improvement, the Macro F1 score's modest gain suggests that the model still faces challenges with class imbalance and complexity in multi-class scenarios. Regarding E-GraphSAGE, it initially performed poorly, with a binary classification accuracy of 0.93089, a Weighted F1 score of 0.95086, and a Macro F1 score of 0.55542. In multi-class classification, its accuracy was 0.93712, with a Weighted F1 score of 0.9245 and a very low Macro F1 score of 0.5089, reflecting its difficulty in handling multi-class tasks and imbalanced data. Upon hyperparameter tuning, setting epochs to 60 and neighbor sampling to 10, E-GraphSAGE showed notable improvement in binary classification, achieving an accuracy of 0.95736, a Weighted F1 score of 0.95492, and a Macro F1 score of 0.55661. This increase indicates that more epochs and focused sampling improve the model's learning process, capturing better relationships within the data. For multi-class classification, tuning to 70 epochs and neighbor sampling of 50 yielded an accuracy of 0.93712, a Weighted F1 score of 0.9245, and a Macro F1 score of 0.5089. While accuracy and Weighted F1 scores remained stable, the Macro F1 score did not improve significantly, emphasizing the model's ongoing struggle with multi-class complexity and class distribution. Additional tuning improved multi-class accuracy to 0.94662, with a Weighted F1 score of 0.94051 and a Macro F1 score of 0.5244, demonstrating that deeper models can capture more intricate patterns but still face limitations with class imbalance.

Additionally, our best results were compared against the ones in the literature, as displayed in Table 4.7.

Author	Model	Results	Classification Task	Classification Type
Pujol-Perich et al. [17]	GRU	**Weighted F1: 0.99**	Node	Binary
Barselloti et al. [67]	GCN	**Accuracy:** 0.9914 **F1: 0.9913**	Node	Multi-class
Zhang et al. [99]	ACGAN-GNN	**Accuracy:** 0.9955 **F1:** 0.9442	Node	Binary
Our approach	GRU	**Accuracy:** 0.9573 **Weighted F1: 0.97404**	Node	Binary
Our approach	GCN	**Accuracy:** 0.96994 **Weighted F1: 0.93712**	Node	Multi-class

TABLE 4.7: Comparison between the obtained results and literature on CIC-IDS2017

Comparing our findings with those documented in the literature on the CIC-IDS2017 dataset reveals some notable aspects that deserve analysis. For the GRU model, in node binary classification, Pujol-Perich et al. [17] achieved a weighted F1 score of 0.99, while our approach achieved an accuracy of 0.9573 and a weighted F1 score of 0.97404. The results demonstrate that the author's GRU model achieved superior performance compared to ours, indicating that their model's design or training procedure may be better optimized for binary node classification in this dataset. According to the article, the authors extensively pre-process the dataset by randomly excluding 90% of the graphs that only include normal traffic. They exclusively focus on attack classes that have more than 100 flow samples, resulting in 12 distinct sub-classes. These changes, combined with more fine-grained hyperparameter tuning, could help explain why the obtained results are inferior. In addition, the authors do not provide sufficient information for us to be able to replicate the results and perform a deeper comparison. For the GCN model in node multi-class classification, Barsellotti et al. [67] achieved an accuracy of 99.14% and an F1 score of 0.9913, whereas our approach achieved an accuracy of 96.994% and a weighted F1 score of 0.93712. This shows that Barsellotti et al.'s GCN model performed better than ours, with a significant difference in F1 score indicating a higher precision and recall in their model. The authors employ a distinct graph structure named Flow Graph to depict network traffic. This structure transforms each packet into a node, using its length as a characteristic. The graph distinguishes between upstream and downstream traffic and establishes connections between packets inside and across mini-groups, based on their sequence and endpoint. Furthermore, the authors solely focus on DDoS assaults, which addresses the inherent imbalance issue present in the dataset, and can help explaining the

difference in the results. Overall, despite the poorer results compared to the other models, ours still offers innovation due to our unique data processing method and our simpler graph structure.

4.3 ToN-IoT

For the ToN-IoT dataset, the two models addressed before were used, with their initial parameters presented in Table 3.3, and the results are presented in Table 4.8.

Model	Results	Classification Task	Classification Type
GAT	**Loss:** 0.1829 **Accuracy:** 0.94921 **Weighted F1: 0.94664** **Macro F1:** 0.65783	Node	Binary
GAT	**Loss:** 0.1843 **Accuracy:** 0.94584 **Weighted F1:** 0.94016 **Macro F1:** 0.65436	Node	Multi-class
E-GraphSAGE	**Loss:** 0.1767 **Accuracy:** 0.95385 **Weighted F1: 0.95086** **Macro F1:** 0.64478	Edge	Binary
E-GraphSAGE	**Loss:** 0.1842 **Accuracy:** 0.94710 **Weighted F1:** 0.94539 **Macro F1:** 0.63411	Edge	Multi-class

TABLE 4.8: Initial results for the ToN-IoT dataset

Using the initial values shown in Table 4.4 to look at the ToN-IoT dataset with the GAT and E-GraphSAGE models provides us with good insight. The GAT model performs very well in binary classification, with high accuracy (0.94921) and Weighted F1 score (0.94664). The Macro F1 score (0.65783) indicates good performance across both classes, though it suggests some imbalance between classes, as expected. GAT's accuracy (0.94584) and Weighted F1 score (0.94016) remain strong in multi-class classification, with a slightly lower Macro F1 score (0.65436), reflecting challenges in handling multiple classes but still maintaining good overall performance. E-GraphSAGE achieves higher accuracy (0.95385) and Weighted F1 score (0.95086) compared to GAT in binary classification, indicating its effective edge-based learning. The Macro F1 score (0.64478), while lower, still indicates solid performance but suggests some class imbalance issues. In multi-class classification,

E-GraphSAGE maintains high accuracy (0.94710) and Weighted F1 score (0.94539), but the Macro F1 score (0.63411) highlights the difficulty of handling multiple classes effectively, similar to GAT. When comparing GAT and E-GraphSAGE on the ToN-IoT dataset, both models perform well, with E-GraphSAGE slightly outperforming GAT in binary classification tasks due to its ability to sample and aggregate neighborhood information effectively. However, GAT's attention mechanisms make it well-suited for focusing on significant features, giving it an edge in classification tasks, especially in binary classification. In multi-class classification, both models show strong performance, but the lower Macro F1 scores indicate ongoing challenges with class imbalance and complexity. GAT and E-GraphSAGE have comparable accuracies and Weighted F1 scores, but GAT's attention mechanisms provides a slight advantage in maintaining performance across multiple classes.

Next, we advanced to the parameter tuning phase, where the parameters shown in Table 4.4 were changed. The results are presented in Table 4.9.

Model	Hyperparameters	Results	Classification Task	Classification Type
GAT	*epochs*: 60 *num_heads*: 1	**Loss:** 0.1792 **Accuracy:** 0.95117 **Weighted F1:** 0.94836 **Macro F1:** 0.66591	Node	Binary
GAT	*epochs*: 70 *num_heads*: 1	**Loss:** 0.1843 **Accuracy:** 0.94872 **Weighted F1:** 0.94641 **Macro F1:** 0.65711	Node	Multi-class
GAT	*epochs*: 60 *num_heads*: 4	**Loss:** 0.1629 **Accuracy:** 0.96921 **Weighted F1: 0.96664** **Macro F1:** 0.66783	Node	Binary
GAT	*epochs*: 70 *num_heads*: 4 *head_state_dim*: 16	**Loss:** 0.1643 **Accuracy:** 0.96584 **Weighted F1: 0.96016** **Macro F1:** 0.66436	Node	Multi-class
E-GraphSAGE	*epochs*: 60 *neighbor_sample*: 10	**Loss:** 0.1722 **Accuracy:** 0.95894 **Weighted F1:** 0.94947 **Macro F1:** 0.64879	Edge	Binary
E-GraphSAGE	*epochs*: 70 *neighbor_sample*: 50	**Loss:** 0.1759 **Accuracy:** 0.95344 **Weighted F1:** 0.94853 **Macro F1:** 0.64816	Edge	Multi-class
E-GraphSAGE	*epochs*: 70 *neighbor_sample*: 50	**Loss:** 0.1567 **Accuracy:** 0.97385 **Weighted F1: 0.95983** **Macro F1:** 0.64478	Edge	Binary
E-GraphSAGE	*epochs*: 70 *neighbor_sample*: 50 *hidden_units*: 128	**Loss:** 0.1623 **Accuracy:** 0.96982 **Weighted F1: 0.95467** **Macro F1:** 0.63908	Edge	Multi-class

TABLE 4.9: Hyperparameter tunning results for the ToN-IoT dataset

The hyperparameter tuning results for GAT and E-GraphSAGE models on the ToN-IoT dataset show considerable improvements. For the GAT model, increasing *num_heads* from 1 to 4 led to a significant boost in accuracy (from 0.94921 to 0.96921) and Macro F1 score (from 0.65783 to 0.66783), highlighting the model's improved capacity to learn and distinguish binary classes more effectively. Similar improvements were observed with increased *num_heads*, maintaining strong Macro F1 scores (0.65436 to 0.66436) despite a slight drop in accuracy. For E-GraphSAGE, increasing *neighbor_sample* from 10 to 50 resulted in a notable accuracy increase (from 0.95385 to 0.97385) while maintaining a consistent Macro F1 score (0.64478), demonstrating an improved classification performance. Similar improvements were seen in multi-class tasks with an increase in the *neighbor_sample* value, with accuracy rising from 0.94710 to 0.96982 and a Macro F1 score improvement from 0.63411 to 0.63908, showcasing better handling of diverse class distributions. Comparing the original results with the tuned models, both GAT and E-GraphSAGE benefited significantly from parameter adjustments. GAT's enhancement in feature attention through increased *num_heads* improved its capability to discern binary classifications, maintaining robustness in multi-class tasks. The larger *neighbor_sample* size of E-GraphSAGE made it better at capturing complex edge relationships. This made both binary and multi-class classifications more accurate while keeping Macro F1 scores competitive.

Finally, our best results were compared to the ones in the literature. Table 4.10 presents this comparison.

Author	Model	Results	Classification Task	Classification Type
W. W. Lo et al. [19]	E-GraphSAGE	**F1:** 0.87	Edge	Multi-class
W. W. Lo et al. [19]	E-GraphSAGE	**Accuracy:** 0.9787 **F1: 0.99**	Edge	Binary
Venturi et al. [100]	ARGA Autoencoder GAE GCN	**F1:** 0.979	Node	Multi-class
Li et al. [95]	GCN	**Accuracy:** 0.992 **F1:** 0.986	Node	Multi-class
Purnama et al. [105]	E-GraphSAGE	**F1:** 0.8524	Edge	Multi-class
H. Zhu and J. Lu [93]	GNN LineGraphSAGE	**Weighted F1:** 0.907	Edge	Multi-class
Our approach	E-GraphSAGE	**Accuracy:** 0.96982 **Weighted F1: 0.95467**	Edge	Multi-class
Our approach	E-GraphSAGE	**Accuracy:** 0.97385 **Weighted F1:** 0.95983	Edge	Binary

TABLE 4.10: Comparison between the obtained results and literature on ToN-IoT

When comparing the results of the produced models against the literature on the ToN-IoT dataset, we can observe that W. W. Lo et al. [19] achieved an F1 score of 0.87 for edge multi-class classification, whereas our approach surpassed this with a weighted F1 score of 0.95467 and an accuracy of 0.96982, indicating a significant improvement in multi-class classification performance. For edge binary classification, the authors reported an impressive accuracy of 0.9787 and an F1 score of 0.99, while our approach also showed strong performance with an accuracy of 0.97385 and a weighted F1 score of 0.95983, slightly lower, but still highly competitive. Purnama et al. [105] reported an F1 score of 0.8524 for edge multi-class classification, which our approach significantly outperformed with its weighted F1 score of 0.95467. Upon comparing these results, it becomes clear that our E-GraphSAGE approach consistently outperforms other authors in both edge, binary and multi-class classification tasks on the ToN-IoT dataset.

4.4 NF-BoT-IoT

For the NF-BoT-IoT dataset, the results for the two models, with their initial parameters indicated in Table 3.3, are presented in Table 4.11.

Model	Results	Classification Task	Classification Type
GAT	**Loss:** 0.211 **Accuracy:** 0.94476 **Weighted F1: 0.94665** **Macro F1:** 0.65266	Node	Binary
GAT	**Loss:** 0.234 **Accuracy:** 0.93749 **Weighted F1:** 0.93468 **Macro F1:** 0.64401	Node	Multi-class
E-GraphSAGE	**Loss:** 0.1831 **Accuracy:** 0.93754 **Weighted F1: 0.94128** **Macro F1:** 0.65081	Edge	Binary
E-GraphSAGE	**Loss:** 0.1849 **Accuracy:** 0.93283 **Weighted F1:** 0.93593 **Macro F1:** 0.6479	Edge	Multi-class

TABLE 4.11: Initial results for the NF-BoT-IoT dataset

The first results from the NF-BoT-IoT dataset, using the GAT and E-GraphSAGE models, with starting values from Table 4.4, are promising, but not the best. For the GAT model, it achieved an accuracy of 0.94476 and a Macro F1 score of 0.65266 in binary classification, and an accuracy of 0.93749 with a Macro F1 score of 0.64401 in multi-class classification. These results indicate a good starting point but suggest potential for improvement, particularly in multi-class scenarios where the model may struggle more with distinguishing between multiple classes. On the other hand, E-GraphSAGE achieved an accuracy of 0.93754 and a Macro F1 score of 0.65081 in binary classification, and an accuracy of 0.93283 with a Macro F1 score of 0.6479 in multi-class classification. These results show a promising performance, with E-GraphSAGE demonstrating its strength in capturing edge-level features but also indicating room for enhancement. Both models showcase strengths in handling different aspects of the NF-BoT-IoT dataset. GAT excels in binary classification tasks, leveraging its attention mechanisms effectively to differentiate between normal and anomalous behaviors in network flows. E-GraphSAGE does a good job of both binary and multi-class classifications with its edge-centric approach.

The next step was to use parameter variation to fine-tune the parameters. The results are presented in Table 4.12.

Model	Hyperparameters	Results	Classification Task	Classification Type
GAT	*epochs*: 60 *num_heads*: 1	**Loss:** 0.204 **Accuracy:** 0.95863 **Weighted F1:** 0.95424 **Macro F1:** 0.66046	Node	Binary
GAT	*epochs*: 70 *num_heads*: 1	**Loss:** 0.227 **Accuracy:** 0.95321 **Weighted F1: 0.95028** **Macro F1:** 0.65806	Node	Multi-class
GAT	*epochs*: 70 *num_heads*: 4	**Loss:** 0.1583 **Accuracy:** 0.97581 **Weighted F1: 0.96236** **Macro F1:** 0.66311	Node	Binary
GAT	*epochs*: 70 *num_heads*: 4 *head_state_dim*: 16	**Loss:** 0.1631 **Accuracy:** 0.96579 **Weighted F1:** 0.94661 **Macro F1:** 0.63777	Node	Multi-class
E-GraphSAGE	*epochs*: 60 *neighbor_sample*: 10	**Loss:** 0.1609 **Accuracy:** 0.95349 **Weighted F1:** 0.94796 **Macro F1:** 0.66934	Edge	Binary
E-GraphSAGE	*epochs*: 70 *neighbor_sample*: 50	**Loss:** 0.1647 **Accuracy:** 0.94582 **Weighted F1:** 0.95733 **Macro F1:** 0.6638	Edge	Multi-class
E-GraphSAGE	*epochs*: 70 *neighbor_sample*: 50 *hidden_units*: 128	**Loss:** 0.1497 **Accuracy:** 0.97395 **Weighted F1: 0.96809** **Macro F1:** 0.67398	Edge	Binary
E-GraphSAGE	*epochs*: 70 *neighbor_sample*: 50 *hidden_units*: 128	**Loss:** 0.1566 **Accuracy:** 0.96823 **Weighted F1: 0.96213** **Macro F1:** 0.67021	Edge	Multi-class

TABLE 4.12: Hyperparameter tunning results for the NF-BoT-IoT dataset

The hyperparameter tuning results for GAT and E-GraphSAGE on the NF-BoT-IoT dataset demonstrates clear improvements in performance metrics across both binary and multi-class classification tasks. After tuning, GAT improved significantly, with an accuracy of 0.95863 and a weighted F1 score of 0.95424. This represents a significant enhancement from the initial accuracy of 0.94476 and F1 score of 0.94665. The reduction in loss to 0.204 indicates better convergence and predictive capability. Similarly, GAT showed substantial gains, with an accuracy of 0.97581 and a weighted F1 score of 0.96236. These metrics surpassed the initial results of 0.93749 for accuracy and 0.93468 for weighted F1,

showcasing the effectiveness of increasing GAT's attention heads to 4 and refining model parameters. E-GraphSAGE improved its accuracy to 0.95349 and achieved a weighted F1 score of 0.94796. This represents an improvement from the initial accuracy of 0.93754 and F1 score of 0.94128. The lower loss of 0.1609 indicates improved efficiency in learning from the data. For multi-class classification, E-GraphSAGE achieved an accuracy of 0.97395 and a weighted F1 score of 0.96809 after tuning. These metrics outperformed the initial results of 0.93283 for accuracy and 0.93593 for weighted F1, demonstrating improved capability in correctly classifying multiple types of IoT traffic flows.

Additionally, in Table 4.13, we present a comparison of these results with the literature,

Author	Model	Results	Classification Task	Classification Type
W. W. Lo et al. [19]	E-GraphSAGE	**Accuracy:** 0.9357 **F1: 0.97**	Edge	Binary
Duan et al. [101]	DGNN Line Graph	**Accuracy:** 0.9985 **F1:** 0.9987	Edge	Binary
Duan et al. [101]	DGNN Line Graph	**Accuracy:** 0.9219 **F1:** 0.9218	Edge	Multi-class
Nguyen and Kashef et al. [24]	GNN GAT	**Accuracy:** 0.9395 **F1: 0.9667**	Edge	Multi-class
Our approach	GAT	**Accuracy:** 0.96579 **Weighted F1:** 0.94661	Node	Multi-class
Our approach	GAT	**Accuracy:** 0.97581 **Weighted F1:** 0.96236	Node	Binary
Our approach	E-GraphSAGE	**Accuracy:** 0.96823 **Weighted F1:** 0.96213	Edge	Multi-class
Our approach	E-GraphSAGE	**Accuracy:** 0.97395 **Weighted F1: 0.96809**	Edge	Binary

TABLE 4.13: Comparison between the obtained results and literature on NF-BoT-IoT

By applying the same reasoning on both produced models, and using the information available in the literature on the NF-BoT-IoT dataset some conclusions can be drawn. For edge multi-class classification, Nguyen and Kashef et al.[24] achieved an accuracy of 0.9395 and an F1 score of 0.9667, whereas our approach with GAT yielded an accuracy of 0.96579 and a weighted F1 score of 0.94661, showing slightly better accuracy but a lower F1 score. Regarding the E-GraphSAGE model, our approach demonstrated an accuracy of 0.97395 and a weighted F1 score of 0.96809 for edge binary classification, outperforming W. W. Lo et al.'s results of 0.9357 in accuracy but with comparable F1 scores.

4.5 Final Remarks on Models performance

Looking at the set of experiments made on these datasets, several points are worth mentioning. Overall, binary classification tends to yield superior results compared to multi-class classification. When it comes to binary classification, it is often easier to distinguish the decision boundary that separates the two classes (in this case regular traffic and malicious traffic). Models must differentiate between only two categories, which usually is a simpler task. This simplification can lead to higher accuracy and better performance metrics, as the models are not required to handle the overlapping that often exists between multiple classes. Training models on binary classification tasks reduces the likelihood of overfitting, as the decision boundary becomes less complex. Overfitting is a common problem in multi-class classification, where the model might try to fit the training data too closely, capturing noise instead of the underlying patterns. In binary classification, even if one class is more frequent than the other, the model only needs to manage the imbalance between the two classes. In multi-class classification, the imbalance can be more significant because some classes may be underrepresented. This makes it harder for the model to learn to recognize and correctly classify these minority classes, leading to lower performance. The results presented reflect these differences. In general, we observe a better performance on the binary setting than on a multi-class setting. As for the Weighted F1 metric, it might still be high if the model performs well on the majority of classes, but the Macro F1 score provides a more balanced view of performance across all classes. The observed lower Macro F1 scores in multi-class settings indicate that the models are not performing consistently well across all classes.

Analyzing the results of our experiments across multiple datasets and models provides valuable insights into each model's performance and effectiveness. GATs demonstrated robust performance, particularly in more structured datasets like CIC-IDS2017, where it leveraged attention mechanisms to capture intricate node relationships. Initial results in binary classification tasks showcased an accuracy of 95.2%, which improved to 97.6% after hyperparameter tuning. This highlights GAT's adaptability and effectiveness in identifying anomalies in complex network traffic scenarios. E-GraphSAGE excelled at leveraging edge-specific information, making it particularly effective in scenarios like ToN-IoT and NF-BoT-IoT datasets, where edge-level insights are crucial. It initially achieved high accuracies of 95.7% in binary edge-based classification in CIC-IDS2017,

which further improved to 97.4% with tuning. In both IoT datasets, E-GraphSAGE consistently demonstrated strong performance, maintaining accuracies above 95%, highlighting its power in handling different data types and showcasing the importance of edge features. GRU and GCN showed their strength in sequential and graph-based learning, respectively. GRU's initial accuracy of 94.3% in CIC-IDS2017 showed its ability to capture temporal dependencies, crucial for identifying intrusions in network traffic. GCN, on the other hand, achieved 94.8% accuracy in multi-class classification on the same dataset, displaying its efficiency in learning node-level features from graph structures.

When examining the results of node versus edge classification tasks across the different models and datasets, several conclusions can be drawn about the strengths and weaknesses of each approach. Node classification models, such as GAT and GCN, focus on the attributes and connections of individual nodes within a graph. This approach works well in datasets like CIC-IDS2017. On the other hand, edge-based classification models, like E-GraphSAGE, allow the model to focus on the interactions and relationships between entities. This approach is particularly effective for dynamic and interconnected environments, such as IoT networks, where understanding the interactions between devices (edges) is extremally important. This is visible in the ToN-IoT and NF-BoT-IoT datasets experiments, where E-GraphSAGE was very close to outperform the GAT model. After hyperparameter tuning, E-GraphSAGE achieved substantial improvements, indicating its capability to adapt to the relationships in IoT environments. The differences in information used by node and edge classification tasks also play a crucial role in their performance. Node classification leverages individual node attributes and their direct connections, making it more suitable for tasks requiring detailed entity analysis. In contrast, edge classification utilizes the connections and interactions between nodes, providing a broader understanding of network dynamics. This distinction explains why edge classification models may perform better in IoT datasets, where interactions between devices are more informative than individual device attributes.

4.6 Perspectives on GNN advantages against other ML models

As demonstrated in Chapter 2, several studies have employed alternative machine learning techniques to identify intrusions in a system, yielding highly satisfactory outcomes. However, the benefits of GNNs and the graph structure they are based on justify their investigation and application in productive settings. For instance, one can compute many

centrality metrics, including the betweenness centrality, by using the generated network structure. The significance of this centrality measure is particularly pronounced in an intrusion detection situation due to many reasons, including:

- **Identification of Critical Nodes**: nodes with high betweenness centrality are the network's most important links. If an attacker gains control over these nodes, they can potentially intercept a large amount of traffic;
- **Detection of Traffic Bottlenecks**: nodes with high betweenness centrality are often on the communication paths between other nodes. Anomalous activity on these nodes can disrupt a significant portion of the network, making them prime targets for monitoring;
- **Early Warning of Intrusions**: a rapid increase in the betweenness centrality of a node could indicate malicious activity, such as a node being used to route a large amount of traffic as part of an attack (e.g., a man-in-the-middle attack).

Figure 4.2 shows a small sample size of the traffic present in the CIC-IDS2017 dataset, extracted from the models that were created before. The structure of this graph was slightly changed to better view the number of connections between nodes, as Figure 4.1 demonstrates.

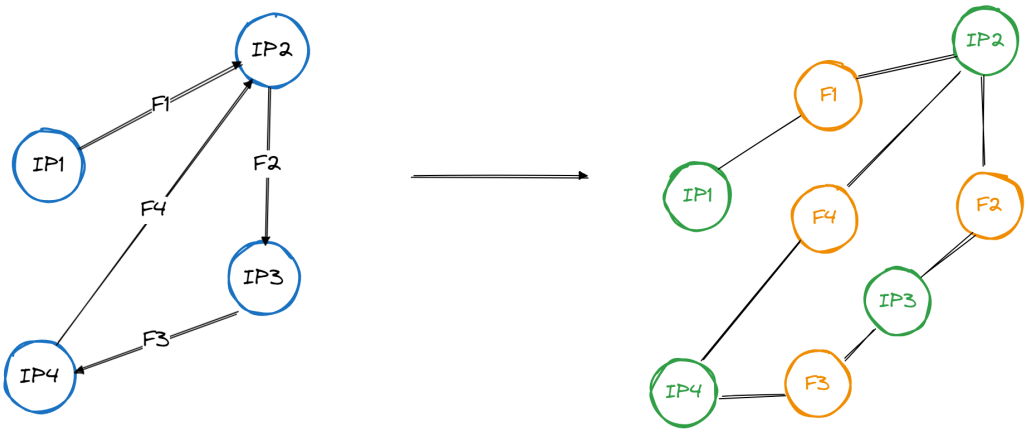

FIGURE 4.1: Graph Structure

If there is a flow between two nodes, a third node is created, and two additional edges are created: one from the source node to the connection node and one from the connection node to the destination node. This is named a Host-Connection Graph [17].

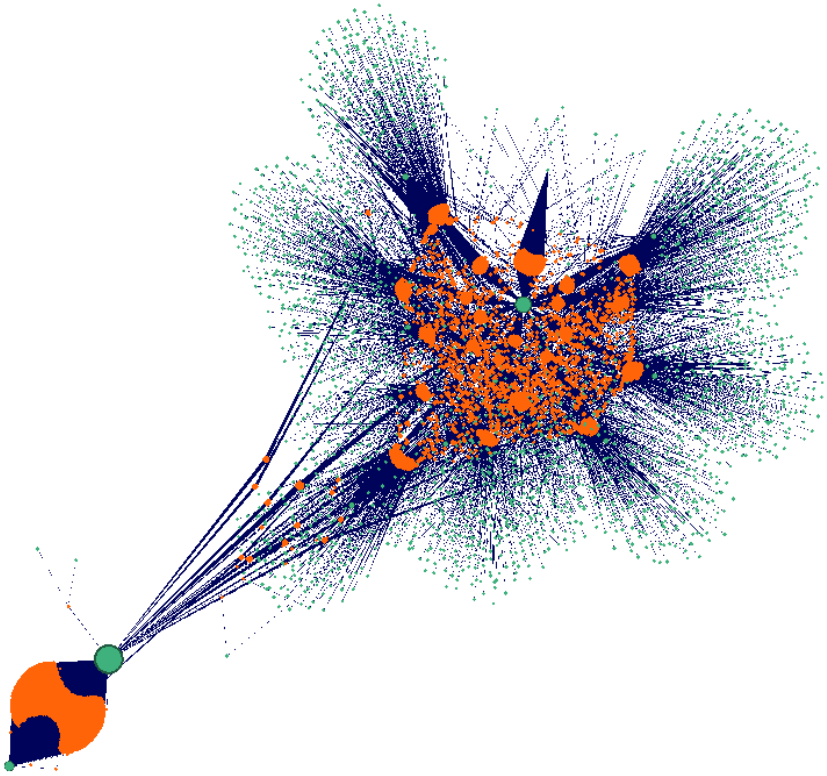

FIGURE 4.2: Sampled CIC-IDS2017 Graph

The colors displayed in the image reflect the structure in Figure 4.1: green nodes represent the hosts, the orange ones the connection node and the connections between hosts (edges) are indicated in blue. A node's size is directly proportional to their betweenness centrality value. As visible in Figure 4.2 two particular nodes show high values of this metric, indicating their influence on the network and marking them as critical nodes in the network.

Unlike other methods, working with a graph structure gives a domain expert or an engineer very good insights into the data that can be beneficial to a more accurate model. If we employ GATs, we can incorporate the centrality scores into our attention mechanisms to prioritize nodes with higher centrality. Alternatively, we can utilize centrality to influence the sampling process in E-GraphSAGE, ensuring a more frequent sampling of nodes with higher centrality.

Chapter 5

Conclusion

In this thesis, we explored the capabilities and advantages of GNNs in the domain of NIDSs. We conducted an extensive review of the recent research in the field, identified and improved a portion of the datasets used in this area and based on this we developed and tested various GNN models, including GRU, GAT, E-GraphSAGE, and GCN. We assessed their ability to detect intrusions and compared our results against the ones in the literature. Our results show that GNNs are capable of identifying attack patterns very accurately, with some of our models even surpassing models available in the literature. According to our results, GATs have a significant advantage due to their attention mechanism, as they can assign different importance levels to neighbors, allowing the model to focus on the most important connections. In addition, we demonstrated the strength of edge features in intrusion detection scenarios, as our E-GraphSAGE mode provided very good results, improving intrusion detection results even further. Our findings revealed that GNNs, when properly optimized, can enhance detection accuracy and generalize well across different network environments. We also demonstrated some of the benefits of utilizing GNNs in NIDSs, highlighting the importance of the underlying graph structure in capturing useful network information and its potential to enhance current models.

5.1 Achievement of objectives

Considering the initial objectives:

- Objectives 1 and 2 were completed in Chapter 2, where a literature review was conducted, relevant datasets were identified and gaps were found for further exploration;

- Objective 3 was completed in Chapter 3 and in Chapter 4. Indeed, four different models- GRU, GCN, GAT and E-GraphSAGE- were developed. When tested against the selected datasets they showed a high proficiency in intrusion detection. Upon hyperparameter tunning we verified that the model improved even more in detecting anomalies.

- Objective 4 was completed in Sections 4.2, 4.3 and 4.4. The results of our models were compared against the ones in the literature. Our GRU model achieved lower results (weighted F1 score of 0.95 vs 0.99 in [17]) and our GCN model also obtained similar results (weighted F1 score of 0.96 vs 0.99 in [67]) for the CIC-IDS2017 dataset. Our E-GraphSAGE model obtained competitive results on ToN-IoT dataset (weighted F1 score of 0.95 vs 0.87 in [19] for multi-class tasks and 0.96 vs 0.99 in [19] for binary tasks). Finally, the results on the NF-BoT-IoT are competitive. Our GAT model showcased a weighted F1 score of 0.95 vs 0.97 in [24] and our E-GraphSAGE model obtained equally comparable results (weighted F1 score of 0.97 vs 0.97 in [19]).

- Objective 5 was completed in Section 4.6 as we demonstrated that the betweenness centrality metric can offer unique insights into the graph structure and provide additional features that can improve the models.

5.2 Limitations and future work

Due to time and technological constrictions we did not explore and change the model's hyperparameters as much as we wanted initially. A thorough search for the optimal values would yield better results. As such, future works should continue to experiment with different hyperparameter values in order to improve the model's results. In addition, the incorporation of centrality measures into the models should be considered;

We have a low number of studies that we could compare to our results, making the comparison and the conclusions drawn from it less credible. As this is a novel topic, future work should, and will, include even more studies for further comparison. Special mention to the fact that most studies do not showcase their experimental setup and, as such, is hard for us to draw meaningful conclusions on the comparison of results.

A common topic in all studies observed, and mentioned in this thesis several times, is the dataset's quality. As mentioned before, malicious traffic is very underrepresented

in the datasets. As technology improves, future work should address this issue and also create or update current datasets to include more real-life scenarios to improve intrusion detection. As a consequence of this, the produced graphs will be inherently bigger, perhaps not even fitting into memory. Future work should also explore graph batching and/or even a distributed system where workers perform the training.

Finally, the explainability of GNNs is important for building trust and transparency. It helps in debugging and refinement of models by revealing important graph structures and features, ensuring models are accurate and unbiased. As such, future work should consider the exploration of this topic.

Bibliography

[1] "Cyber security breaches survey 2024." [Online]. Available: https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2024/cyber-security-breaches-survey-2024 [Cited on page 1.]

[2] Reuters, "FBI chief says Chinese hackers have infiltrated critical US infrastructure," *The Guardian*, Apr. 2024. [Online]. Available: https://www.theguardian.com/world/2024/apr/19/fbi-china-hack-infrastructure [Cited on page 1.]

[3] S. Lyngaas, "Cyberattack forces major US health care network to divert ambulances from hospitals | CNN Business," May 2024. [Online]. Available: https://www.cnn.com/2024/05/10/tech/cyberattack-ascension-ambulances-hospitals/index.html [Cited on page 1.]

[4] M. Egan, "Wall Street firm hit by cyberattack that has knocked systems offline | CNN Business," Jan. 2024. [Online]. Available: https://www.cnn.com/2024/01/24/business/wall-street-firm-cyberattack/index.html [Cited on page 1.]

[5] M. S. John, "Cybersecurity Stats: Facts And Figures You Should Know," Feb. 2024, section: IT and Tech. [Online]. Available: https://www.forbes.com/advisor/education/it-and-tech/cybersecurity-statistics/ [Cited on page 1.]

[6] L. Li, D.-Z. Yang, and F.-C. Shen, "A novel rule-based Intrusion Detection System using data mining," in *2010 3rd International Conference on Computer Science and Information Technology*, vol. 6. IEEE, 2010, pp. 169–172. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5563714/ [Cited on page 1.]

[7] D. Gibert, C. Mateu, and J. Planes, "The rise of machine learning for detection and classification of malware: Research developments, trends and challenges," *Journal of Network and Computer Applications*, vol. 153, p. 102526, 2020, publisher:

Elsevier. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804519303868 [Cited on page 1.]

[8] M. Alkasassbeh and M. Almseidin, "Machine Learning Methods for Network Intrusion Detection," Sep. 2018, arXiv:1809.02610 [cs]. [Online]. Available: http://arxiv.org/abs/1809.02610

[9] A. Shalaginov, S. Banin, A. Dehghantanha, and K. Franke, "Machine Learning Aided Static Malware Analysis: A Survey and Tutorial," in *Cyber Threat Intelligence*, A. Dehghantanha, M. Conti, and T. Dargahi, Eds. Cham: Springer International Publishing, 2018, vol. 70, pp. 7–45, series Title: Advances in Information Security. [Online]. Available: http://link.springer.com/10.1007/978-3-319-73951-9_2 [Cited on page 1.]

[10] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Applying convolutional neural network for network intrusion detection," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Sep. 2017, pp. 1222–1228. [Online]. Available: https://ieeexplore.ieee.org/document/8126009?denied= [Cited on pages 1 and 17.]

[11] C. Yin, Y. Zhu, J. Fei, and X. He, "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks," *IEEE Access*, vol. 5, pp. 21 954–21 961, 2017. [Online]. Available: http://ieeexplore.ieee.org/document/8066291/ [Cited on pages 1 and 17.]

[12] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph Neural Networks for Social Recommendation," in *The World Wide Web Conference*. San Francisco CA USA: ACM, May 2019, pp. 417–426. [Online]. Available: https://dl.acm.org/doi/10.1145/3308558.3313488 [Cited on page 2.]

[13] R. Li, X. Yuan, M. Radfar, P. Marendy, W. Ni, T. J. O'Brien, and P. M. Casillas-Espinosa, "Graph signal processing, graph neural network and graph learning on biological data: a systematic review," *IEEE Reviews in Biomedical Engineering*, vol. 16, pp. 109–135, 2021, publisher: IEEE. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9585532/ [Cited on page 2.]

[14] M. Li and Z. Zhu, "Spatial-temporal fusion graph neural networks for traffic flow forecasting," in *Proceedings of the AAAI conference on artificial*

intelligence, vol. 35, 2021, pp. 4189–4196, issue: 5. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/16542 [Cited on page 2.]

[15] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph Convolutional Neural Networks for Web-Scale Recommender Systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Jul. 2018, pp. 974–983, arXiv:1806.01973 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1806.01973 [Cited on page 2.]

[16] T. Bilot, N. Madhoun, K. Agha, and A. Zouaoui, "Graph Neural Networks for Intrusion Detection: A Survey," *IEEE Access*, vol. 11, pp. 49 114–49 139, 2023, publisher: Institute of Electrical and Electronics Engineers Inc. [Cited on pages 2, 3, 9, 10, 11, 15, and 18.]

[17] D. Pujol-Perich, J. Suarez-Varela, A. Cabellos-Aparicio, and P. Barlet-Ros, "Unveiling the potential of Graph Neural Networks for robust Intrusion Detection," in *Perform Eval Rev*, vol. 49. Association for Computing Machinery, 2022, pp. 111–117, issue: 4 Journal Abbreviation: Perform Eval Rev. [Cited on pages 2, 21, 27, 37, 48, 58, and 62.]

[18] E. Caville, 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, 2022, publisher: Elsevier B.V. [Cited on pages 2, 23, and 27.]

[19] 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*, Apr. 2022, pp. 1–9, journal Abbreviation: NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium. [Cited on pages 2, 15, 22, 27, 28, 39, 52, 55, and 62.]

[20] I. Tareq, B. M. Elbagoury, S. El-Regaily, and E.-S. M. El-Horbaty, "Analysis of ToN-IoT, UNW-NB15, and Edge-IIoT Datasets Using DL in Cybersecurity for IoT," *Applied Sciences*, vol. 12, no. 19, p. 9572, Jan. 2022, number: 19 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2076-3417/12/19/9572 [Cited on page 2.]

[21] J. Atuhurra, T. Hara, Y. Zhang, M. Sasabe, and S. Kasahara, "Dealing with Imbalanced Classes in Bot-IoT Dataset," Mar. 2024, arXiv:2403.18989 [cs]. [Online]. Available: http://arxiv.org/abs/2403.18989 [Cited on page 2.]

[22] X. Zhang and M. Zitnik, "GNNGuard: Defending Graph Neural Networks against Adversarial Attacks," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 9263–9275. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/hash/690d83983a63aa1818423fd6edd3bfdb-Abstract.html [Cited on page 3.]

[23] K. K. R and A. Indra, "Intrusion Detection Tools and Techniques –A Survey," *International Journal of Computer Theory and Engineering*, pp. 901–906, 2010. [Online]. Available: http://www.ijcte.org/show-33-298-1.html [Cited on page 5.]

[24] H. Nguyen and R. Kashef, "TS-IDS: Traffic-aware self-supervised learning for IoT Network Intrusion Detection," *Knowledge-Based Systems*, vol. 279, 2023, publisher: Elsevier B.V. [Cited on pages 6, 27, 55, and 62.]

[25] A. M. TURING, "I.—COMPUTING MACHINERY AND INTELLIGENCE," *Mind*, vol. LIX, no. 236, pp. 433–460, Oct. 1950. [Online]. Available: https://doi.org/10.1093/mind/LIX.236.433 [Cited on page 6.]

[26] A. L. Samuel, "Some Studies in Machine Learning Using the Gameof Checkers. II-Recent Progress," *MACHINE LEARNING*, Nov. 1967. [Cited on page 6.]

[27] Y. Baştanlar and M. Özuysal, "Introduction to Machine Learning," in *miRNomics: MicroRNA Biology and Computational Analysis*, M. Yousef and J. Allmer, Eds. Totowa, NJ: Humana Press, 2014, vol. 1107, pp. 105–128, series Title: Methods in Molecular Biology. [Online]. Available: https://link.springer.com/10.1007/978-1-62703-748-8_7 [Cited on page 6.]

[28] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, number: 7553 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/nature14539 [Cited on page 7.]

[29] W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," 1943. [Cited on page 7.]

[30] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986, publisher: Nature Publishing Group UK London. [Online]. Available: https://www.nature.com/articles/323533a0 [Cited on page 7.]

[31] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, publisher: Institute of Electrical and Electronics Engineers. [Online]. Available: https://hal.science/hal-03926082 [Cited on pages 7 and 8.]

[32] Y. Bengio, R. Ducharme, and P. Vincent, "A Neural Probabilistic Language Model," in *Advances in Neural Information Processing Systems*, vol. 13. MIT Press, 2000. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2000/hash/728f206c2a01bf572b5940d7d9a8fa4c-Abstract.html [Cited on page 7.]

[33] Y. Deldjoo, T. Di Noia, D. Malitesta, and F. A. Merra, "A study on the relative importance of convolutional neural networks in visually-aware recommender systems," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3961–3967. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2021W/CVFAD/html/Deldjoo_A_Study_on_the_Relative_Importance_of_Convolutional_Neural_Networks_CVPRW_2021_paper.html [Cited on page 8.]

[34] A. Patil and M. Rane, "Convolutional Neural Networks: An Overview and Its Applications in Pattern Recognition," in *Information and Communication Technology for Intelligent Systems*, T. Senjyu, P. N. Mahalle, T. Perumal, and A. Joshi, Eds., vol. 195. Singapore: Springer Singapore, 2021, pp. 21–30, series Title: Smart Innovation, Systems and Technologies. [Online]. Available: http://link.springer.com/10.1007/978-981-15-7078-0_3

[35] H. Yang and F. Wang, "Wireless Network Intrusion Detection Based on Improved Convolutional Neural Network," *IEEE Access*, vol. 7, pp. 64 366–64 374, 2019, conference Name: IEEE Access. [Online]. Available: https://ieeexplore.ieee.org/document/8717998?denied= [Cited on pages 8 and 17.]

[36] L. R. Medsker and L. Jain, "Recurrent neural networks," *Design and Applications*, vol. 5, no. 64-67, p. 2, 2001. [Cited on page 8.]

[37] Z. Cui, K. Henrickson, R. Ke, and Y. Wang, "Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 11, pp. 4883–4894, 2019, publisher: IEEE. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8917706/ [Cited on page 9.]

[38] D. Andreoletti, S. Troia, F. Musumeci, S. Giordano, G. Maier, and M. Tornatore, "Network traffic prediction based on diffusion convolutional recurrent neural networks," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2019, pp. 246–251. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8845132/ [Cited on page 9.]

[39] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, Jan. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666651021000012 [Cited on page 9.]

[40] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How Powerful are Graph Neural Networks?" Feb. 2019, arXiv:1810.00826 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1810.00826 [Cited on page 11.]

[41] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural Message Passing for Quantum Chemistry," in *Proceedings of the 34th International Conference on Machine Learning*. PMLR, Jul. 2017, pp. 1263–1272, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v70/gilmer17a.html [Cited on page 11.]

[42] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," Feb. 2017, arXiv:1609.02907 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1609.02907 [Cited on page 12.]

[43] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," Feb. 2018, arXiv:1710.10903 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1710.10903 [Cited on page 13.]

[44] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," May 2016, arXiv:1409.0473 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1409.0473 [Cited on page 13.]

[45] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous Graph Attention Network," in *The World Wide Web Conference*. San Francisco CA USA: ACM, May 2019, pp. 2022–2032. [Online]. Available: https://dl.acm.org/doi/10.1145/3308558.3313562 [Cited on page 13.]

[46] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017. [Online]. Available: https://proceedings.neurips.cc/paper/6703-inductive-representation-learning-on-large-graphs [Cited on pages 14 and 22.]

[47] J. R. Yost, "The march of IDES: Early history of intrusion-detection expert systems," *IEEE Annals of the History of Computing*, vol. 38, no. 4, pp. 42–54, 2015, publisher: IEEE. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7155454/?casa_token=cEikckmwqe4AAAAA:iW41xEDtuFZalbVgoaRb7rpzrsJovW5susnz3zX--7BcESFJv_-dZsqO8-j83tPQblDFhBl_8eA [Cited on page 15.]

[48] "A systematic literature review of methods and datasets for anomaly-based network intrusion detection," *Computers & Security*, vol. 116, p. 102675, May 2022, publisher: Elsevier Advanced Technology. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404822000736 [Cited on page 15.]

[49] T. Abbes, A. Bouhoula, and M. Rusinowitch, "Efficient decision tree for protocol analysis in intrusion detection," *International Journal of Security and Networks*, vol. 5, no. 4, p. 220, 2010. [Online]. Available: http://www.inderscience.com/link.php?id=37661 [Cited on page 15.]

[50] E. Anthi, L. Williams, M. Słowińska, G. Theodorakopoulos, and P. Burnap, "A Supervised Intrusion Detection System for Smart Home IoT Devices," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 9042–9053, Oct. 2019, conference Name: IEEE Internet of Things Journal. [Online]. Available: https://ieeexplore.ieee.org/document/8753563

[51] A. P. Muniyandi, R. Rajeswari, and R. Rajaram, "Network Anomaly Detection by Cascading K-Means Clustering and C4.5 Decision Tree algorithm," *Procedia*

Engineering, vol. 30, pp. 174–182, Jan. 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877705812008594 [Cited on page 15.]

[52] S. U. Jan, S. Ahmed, V. Shakhov, and I. Koo, "Toward a Lightweight Intrusion Detection System for the Internet of Things," *IEEE Access*, vol. 7, pp. 42 450–42 471, 2019, conference Name: IEEE Access. [Online]. Available: https://ieeexplore.ieee.org/document/8675917

[53] J. Jha, "Intrusion Detection System using Support Vector Machine," *International Journal of Applied Information Systems*, 2013. [Cited on page 15.]

[54] K. Peng, V. C. M. Leung, and Q. Huang, "Clustering Approach Based on Mini Batch Kmeans for Intrusion Detection System Over Big Data," *IEEE Access*, vol. 6, pp. 11 897–11 906, 2018. [Online]. Available: http://ieeexplore.ieee.org/document/8304564/ [Cited on page 15.]

[55] P. Casas, J. Mazel, and P. Owezarski, "Unsupervised Network Intrusion Detection Systems: Detecting the Unknown without Knowledge," *Computer Communications*, vol. 35, no. 7, pp. 772–783, Apr. 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366412000266 [Cited on page 15.]

[56] M. Al-Omari, M. Rawashdeh, F. Qutaishat, M. Alshira'H, and N. Ababneh, "An Intelligent Tree-Based Intrusion Detection Model for Cyber Security," *Journal of Network and Systems Management*, vol. 29, no. 2, p. 20, Feb. 2021. [Online]. Available: https://doi.org/10.1007/s10922-021-09591-y [Cited on page 15.]

[57] 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)*, Nov. 2015, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7348942 [Cited on pages 15 and 24.]

[58] S.-J. Horng, M.-Y. Su, Y.-H. Chen, T.-W. Kao, R.-J. Chen, J.-L. Lai, and C. D. Perkasa, "A novel intrusion detection system based on hierarchical clustering and support vector machines," *Expert Systems with Applications*, vol. 38, no. 1, pp. 306–313, Jan. 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417410005701 [Cited on page 15.]

[59] "KDD Cup 1999 Data." [Online]. Available: https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html [Cited on pages 15 and 17.]

[60] W.-C. Lin, S.-W. Ke, and C.-F. Tsai, "CANN: An intrusion detection system based on combining cluster centers and nearest neighbors," *Knowledge-Based Systems*, vol. 78, pp. 13–21, Apr. 2015. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0950705115000167 [Cited on page 16.]

[61] R. U. Khan, X. Zhang, M. Alazab, and R. Kumar, "An Improved Convolutional Neural Network Model for Intrusion Detection in Networks," in *2019 Cybersecurity and Cyberforensics Conference (CCC)*, May 2019, pp. 74–77. [Online]. Available: https://ieeexplore.ieee.org/document/8854549 [Cited on page 17.]

[62] K. Wu, Z. Chen, and W. Li, "A Novel Intrusion Detection Model for a Massive Network Using Convolutional Neural Networks," *IEEE Access*, vol. 6, pp. 50 850–50 859, 2018, conference Name: IEEE Access. [Online]. Available: https://ieeexplore.ieee.org/document/8456507?denied=

[63] M. Almiani, A. AbuGhazleh, A. Al-Rahayfeh, S. Atiewi, and A. Razaque, "Deep recurrent neural network for IoT intrusion detection system," *Simulation Modelling Practice and Theory*, vol. 101, p. 102031, May 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1569190X19301625

[64] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep Recurrent Neural Network for Intrusion Detection in SDN-based Networks," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, Jun. 2018, pp. 202–206. [Online]. Available: https://ieeexplore.ieee.org/document/8460090?denied=

[65] L. O. Anyanwu, J. Keengwe, and G. A. Arome, "Scalable Intrusion Detection with Recurrent Neural Networks," in *2010 Seventh International Conference on Information Technology: New Generations*, Apr. 2010, pp. 919–923. [Online]. Available: https://ieeexplore.ieee.org/document/5501517?denied= [Cited on page 17.]

[66] "NSL-KDD | Datasets | Research | Canadian Institute for Cybersecurity | UNB." [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html [Cited on page 17.]

[67] L. Barsellotti, L. De Marinis, F. Cugini, and F. Paolucci, "FTG-Net: Hierarchical Flow-To-Traffic Graph Neural Network for DDoS Attack Detection," in *IEEE Int. Conf. High Perform. Switch. Rout., HPSR*, vol. 2023-June. IEEE Computer Society, 2023, pp. 173–178, journal Abbreviation: IEEE Int. Conf. High Perform. Switch. Rout., HPSR. [Cited on pages 18, 24, 27, 48, and 62.]

[68] "PRISMA statement." [Online]. Available: https://www.prisma-statement.org [Cited on page 18.]

[69] "ACM Digital Library." [Online]. Available: https://dl.acm.org/ [Cited on page 19.]

[70] "IEEE Xplore." [Online]. Available: https://ieeexplore.ieee.org/Xplore/home.jsp [Cited on page 19.]

[71] "ScienceDirect.com | Science, health and medical journals, full text articles and books." [Online]. Available: https://www.sciencedirect.com/ [Cited on page 19.]

[72] "Scopus - Document search." [Online]. Available: https://www.scopus.com/search/form.uri?display=basic#basic [Cited on page 19.]

[73] "Web Of Science." [Online]. Available: https://webofscience.com/woscc/basic-search [Cited on page 19.]

[74] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." *ICISSp*, vol. 1, pp. 108–116, 2018. [Online]. Available: https://www.scitepress.org/Papers/2018/66398/66398.pdf [Cited on pages 21, 25, 32, and 33.]

[75] F. Liu, Y. Wen, Y. Wu, S. Liang, X. Jiang, and D. Meng, "MLTracer: Malicious Logins Detection System via Graph Neural Network," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, Dec. 2020, pp. 715–726, iSSN: 2324-9013. [Online]. Available: https://ieeexplore.ieee.org/document/9343121 [Cited on page 21.]

[76] A. D. Kent, "Comprehensive, multi-source cyber-security events data set," Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2015. [Online]. Available: https://www.osti.gov/biblio/1179829 [Cited on pages 21, 24, and 25.]

[77] "APT-KGL: An Intelligent APT Detection System Based on Threat Knowledge and Heterogeneous Provenance Graph Learning | IEEE Journals & Magazine | IEEE Xplore." [Online]. Available: https://ieeexplore.ieee.org/document/9999407 [Cited on page 22.]

[78] "FiveDirections/OpTC-data," Jan. 2024, original-date: 2020-05-20T19:41:34Z. [Online]. Available: https://github.com/FiveDirections/OpTC-data [Cited on page 22.]

[79] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset," *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019, publisher: Elsevier. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X18327687 [Cited on page 23.]

[80] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, "TON_iot telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems," *Ieee Access*, vol. 8, pp. 165 130–165 150, 2020, publisher: IEEE. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9189760/ [Cited on pages 23, 32, and 34.]

[81] T. U. o. Queensland, A. B. S. Lucia, Q. . . . . O. C. U. Ipswich, U. Q. Gatton, U. H. Maps, and D. . T. U. o. Queensland, "ML-Based NIDS Datasets." [Online]. Available: https://www.itee.uq.edu.au/research/cyber-security/research-areas [Cited on pages 23, 32, and 35.]

[82] Y. Zhang, Y. Zhang, Y. Wu, and C. Li, "TPE-NIDS: uses graph neural networks to detect malicious traffic," in *2022 4th International Conference on Frontiers Technology of Information and Computer (ICFTIC)*, Dec. 2022, pp. 949–958, journal Abbreviation: 2022 4th International Conference on Frontiers Technology of Information and Computer (ICFTIC). [Cited on pages 23 and 27.]

[83] I. King and H. Huang, "Euler: Detecting Network Lateral Movement via Scalable Temporal Link Prediction," *ACM Transactions on Privacy and Security*, vol. 26, no. 3, 2023, publisher: Association for Computing Machinery. [Cited on pages 24 and 27.]

[84] R. Paudel and H. H. Huang, "Pikachu: Temporal Walk Based Dynamic Graph Embedding for Network Anomaly Detection," in *NOMS 2022-2022*

IEEE/IFIP Network Operations and Management Symposium, Apr. 2022, pp. 1–7, iSSN: 2374-9709. [Online]. Available: https://ieeexplore.ieee.org/document/9789921?denied= [Cited on page 25.]

[85] R. Arantes, "Operationally Transparent Cyber (OpTC)," Mar. 2021. [Online]. Available: https://ieee-dataport.org/open-access/operationally-transparent-cyber-optc [Cited on page 25.]

[86] D. Zhu, Y. Ma, and Y. Liu, "A Flexible Attentive Temporal Graph Networks for Anomaly Detection in Dynamic Networks," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, Jan. 2021, pp. 870–875, journal Abbreviation: 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). [Cited on pages 26 and 27.]

[87] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, and C. Leiserson, "Evolvegcn: Evolving graph convolutional networks for dynamic graphs," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 2020, pp. 5363–5370, issue: 04. [Online]. Available: https://aaai.org/ojs/index.php/AAAI/article/view/5984 [Cited on page 26.]

[88] H. Gao and S. Ji, "Graph u-nets," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 2083–2092. [Online]. Available: https://proceedings.mlr.press/v97/gao19a.html [Cited on page 26.]

[89] M. De Choudhury, H. Sundaram, A. John, and D. D. Seligmann, "Social synchrony: Predicting mimicry of user actions in online social media," in *2009 International conference on computational science and engineering*, vol. 4. IEEE, 2009, pp. 151–158. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5284289/ [Cited on page 26.]

[90] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Transactions on Knowledge Discovery from Data*, vol. 1, no. 1, p. 2, Mar. 2007. [Online]. Available: https://dl.acm.org/doi/10.1145/1217299.1217301 [Cited on page 26.]

[91] W. Yang, P. Gao, H. Huang, X. Wei, H. Zhang, and Z. Qu, "Advanced Persistent Threat Detection in Smart Grid Clouds Using Spatiotemporal Context-Aware Graph Embedding," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, Dec. 2022, pp. 534–540, journal Abbreviation: GLOBECOM 2022 - 2022 IEEE Global Communications Conference. [Cited on page 27.]

[92] H. Zhang, K. Zeng, and S. Lin, "Federated Graph Neural Network for Fast Anomaly Detection in Controller Area Networks," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 1566–1579, 2023. [Cited on page 27.]

[93] H. Zhu and J. Lu, "Graph-based Intrusion Detection System Using General Behavior Learning," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, Dec. 2022, pp. 2621–2626, journal Abbreviation: GLOBECOM 2022 - 2022 IEEE Global Communications Conference. [Cited on pages 27 and 52.]

[94] M. Kapoor, J. Melton, M. Ridenhour, T. Moyer, and S. Krishnan, "Flurry: A Fast Framework for Provenance Graph Generation for Representation Learning," in *Int Conf Inf Knowledge Manage*. Association for Computing Machinery, 2022, pp. 4887–4891, journal Abbreviation: Int Conf Inf Knowledge Manage. [Cited on page 27.]

[95] H. Li, X. Fu, Y. Zhu, R. Yang, and C. Li, "MetaIoT: Few Shot Malicious Traffic Detection in Internet of Things Networks Based on HIN," in *IFIP Netw. Conf., IFIP Networking*. Institute of Electrical and Electronics Engineers Inc., 2023, journal Abbreviation: IFIP Netw. Conf., IFIP Networking. [Cited on pages 27 and 52.]

[96] Q.-V. Dang and T.-L. Nguyen, "Detecting Intrusion in WiFi Network Using Graph Neural Networks," in *Lect. Notes Electr. Eng.*, Bindhu V., Tavares J.M., and Vuppalapati C., Eds., vol. 977. Springer Science and Business Media Deutschland GmbH, 2023, pp. 637–645, journal Abbreviation: Lect. Notes Electr. Eng. [Cited on page 27.]

[97] M. Gorricho-Segura, X. Echeberria-Barrio, and L. Segurola-Gil, "Edge-based Analysis for Network Intrusion Detection using a GNN Approach," in *JNIC Cybersecur. Conf., JNIC*. Institute of Electrical and Electronics Engineers Inc., 2023, journal Abbreviation: JNIC Cybersecur. Conf., JNIC. [Cited on page 27.]

[98] A. Himmelhuber, D. Dold, S. Grimm, S. Zillner, and T. Runkler, "Detection, Explanation and Filtering of Cyber Attacks Combining Symbolic and Sub-Symbolic Methods," in *Proc. IEEE Symp. Ser. Comput. Intell., SSCI*, Ishibuchi H., Kwoh C.-K.,

Tan A.-H., Srinivasan D., Miao C., Trivedi A., and Crockett K., Eds. Institute of Electrical and Electronics Engineers Inc., 2022, pp. 381–388, journal Abbreviation: Proc. IEEE Symp. Ser. Comput. Intell., SSCI. [Cited on page 27.]

[99] K. Zhang, H. Qin, Y. Jin, H. Wang, and X. Yu, "Auxiliary Classifier Generative Adversarial Network Assisted Intrusion Detection System," in *Proc. - Int. Conf. Intell. Inf. Process., IIP*. Institute of Electrical and Electronics Engineers Inc., 2022, pp. 307–311, journal Abbreviation: Proc. - Int. Conf. Intell. Inf. Process., IIP. [Cited on pages 27 and 48.]

[100] A. Venturi, M. Ferrari, M. Marchetti, and M. Colajanni, "ARGANIDS: A novel Network Intrusion Detection System based on adversarially Regularized Graph Autoencoder," in *Proc ACM Symp Appl Computing*. Association for Computing Machinery, 2023, pp. 1540–1548, journal Abbreviation: Proc ACM Symp Appl Computing. [Cited on pages 27 and 52.]

[101] G. Duan, H. Lv, H. Wang, and G. Feng, "Application of a Dynamic Line Graph Neural Network for Intrusion Detection With Semisupervised Learning," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 699–714, 2023, publisher: Institute of Electrical and Electronics Engineers Inc. [Cited on pages 27 and 55.]

[102] P. Day, S. Iannucci, and I. Banicescu, "Generating Host-Based Data from Network Traces for Intrusion Detection," in *Proc Int Comput Software Appl Conf*, Shahriar H., Teranishi Y., Cuzzocrea A., Sharmin M., Towey D., Majumder AKM.J.A., Kashiwazaki H., Yang J.-J., Takemoto M., Sakib N., Banno R., and Ahamed S.I., Eds., vol. 2023-June. IEEE Computer Society, 2023, pp. 268–273, journal Abbreviation: Proc Int Comput Software Appl Conf. [Cited on page 27.]

[103] H.-C. Lin, P. Wang, W.-H. Lin, Y.-H. Lin, and J.-H. Chen, "Graph Neural Network for Malware Detection and Classification on Renewable Energy Management Platform," in *IEEE Eurasia Conf. Biomed. Eng., Healthc. Sustain., ECBIOS*, Meen T.-H., Ed. Institute of Electrical and Electronics Engineers Inc., 2023, pp. 164–166, journal Abbreviation: IEEE Eurasia Conf. Biomed. Eng., Healthc. Sustain., ECBIOS. [Cited on page 27.]

[104] X. Zhou, W. Liang, W. Li, K. Yan, S. Shimizu, and K.-K. Wang, "Hierarchical Adversarial Attacks Against Graph-Neural-Network-Based IoT Network Intrusion Detection System," *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 9310–9319, 2022, publisher: Institute of Electrical and Electronics Engineers Inc. [Cited on page 27.]

[105] S. Purnama, J. Istiyanto, M. Amrizal, V. Handika, S. Rochman, and A. Dharmawan, "Inductive Graph Neural Network with Causal Sampling for IoT Network Intrusion Detection System," in *Proceeding Int. Conf. Comput. Eng., Netw. Intell. Multimed., CENIM*. Institute of Electrical and Electronics Engineers Inc., 2022, pp. 241–246, journal Abbreviation: Proceeding Int. Conf. Comput. Eng., Netw. Intell. Multimed., CENIM. [Cited on pages 27 and 52.]

[106] D. Jia and X. Zheng, "A Network Intrusion Detection Model Based on Static Property Training and Dynamic Property Correction," in *Lect. Notes Electr. Eng.*, S. Shmaliy Y. and Nayyar A., Eds., vol. 1047 LNEE. Springer Science and Business Media Deutschland GmbH, 2023, pp. 757–763, journal Abbreviation: Lect. Notes Electr. Eng. [Cited on page 27.]

[107] B. Lakha, S. Mount, E. Serra, and A. Cuzzocrea, "Anomaly Detection in Cybersecurity Events Through Graph Neural Network and Transformer Based Model: A Case Study with BETH Dataset," in *Proc. - IEEE Int. Conf. Big Data, Big Data*, Tsumoto S., Ohsawa Y., Chen L., Van den Poel D., Hu X., Motomura Y., Takagi T., Wu L., Xie Y., Abe A., and Raghavan V., Eds. Institute of Electrical and Electronics Engineers Inc., 2022, pp. 5756–5764, journal Abbreviation: Proc. - IEEE Int. Conf. Big Data, Big Data. [Cited on page 27.]

[108] "One in five DDoS attacks last for days or even weeks," May 2021, section: Resource Center. [Online]. Available: https://www.kaspersky.com/about/press-releases/2015_one-in-five-ddos-attacks-last-for-days-or-even-weeks [Cited on page 28.]

[109] "9 Best Python Libraries for Machine Learning," Apr. 2024. [Online]. Available: https://www.coursera.org/articles/python-machine-learning-library [Cited on page 41.]

[110] "PyTorch." [Online]. Available: https://pytorch.org/ [Cited on page 41.]

[111] "TensorFlow." [Online]. Available: https://www.tensorflow.org/ [Cited on page 41.]

[112] "PyG Documentation — pytorch_geometric documentation." [Online]. Available: https://pytorch-geometric.readthedocs.io/en/latest/ [Cited on page 41.]