Heaven's Light is Our Guide


# HeteroGNN Architectures for Multi-Class Network Attack Detection: SAGE vs Transformer Approaches

A thesis is submitted in partial fulfilment for the requirement of the degree of

**Bachelor of Science in**

**Electrical & Computer Engineering**

By

**Sadia Afrin**
**Roll No. 1910014**

To the

Department of Electrical & Computer Engineering
Rajshahi University of Engineering & Technology

June 13, 2025

# HeteroGNN Architectures for Multi-Class Network Attack Detection: SAGE vs Transformer Approaches

A thesis is submitted in partial fulfilment for the requirement of the degree of

**Bachelor of Science in**

**Electrical & Computer Engineering**

By

**Sadia Afrin**
**Roll No. 1910014**

To the

Department of Electrical & Computer Engineering
Rajshahi University of Engineering & Technology

June 13, 2025

# Acknowledgement

This thesis has been submitted to the Department of Electrical and Computer Engineering of Rajshahi University of Engineering & Technology (RUET), Rajshahi-6204, Bangladesh, for the partial fulfillment of the requirements for the degree of B.Sc. in Electrical & Computer Engineering. Thesis title regards to "**HeteroGNN Architectures for Multi-Class Network Attack Detection: SAGE vs Transformer Approaches**".

First and foremost, I offer my sincere gratitude and ineptness to my thesis supervisor, **Fariya Tabassum**, Assistant Professor, Department of Electrical & Computer Engineering who has supported me throughout my thesis with his patience and knowledge. I shall ever remain grateful to her for her valuable guidance, advice, encouragement, cordial and amiable contribution to my thesis.

I wish to thank once again **Dr. Md. Anwar Hossain** as the head of the Department of Electrical & Computer Engineering for his support and encouragement and also for providing all kind of laboratory facilities.

I am also grateful to the administration of Rajshahi University of Engineering & Technology for providing a self-sufficient Under Graduate (UG) lab.

Finally, I want to thank the most important and the closest persons of my life and my parents for giving a big support to me.

Sadia Afrin — June 13, 2025

Roll: 1910014 — RUET, Rajshahi

Heavens Light is our Guide


## Department of Electrical & Computer Engineering

## Rajshahi University of Engineering & Technology

# CERTIFICATE

This is to certify that the thesis entitled "**HeteroGNN Architectures for Multi-Class Network Attack Detection: SAGE vs Transformer Approaches**" by Sadia Afrin (Roll: 1910014), has been carried out under my supervision. To the best of my knowledge, this thesis is an original one and has not been submitted anywhere for any degree or diploma.

**Thesis Supervisor**

_____

**Fariya Tabassum**

Assistant Professor

Department of Electrical & Computer Engineering

Rajshahi University of Engineering & Technology

Heavens Light is our Guide


## Department of Electrical & Computer Engineering

## Rajshahi University of Engineering & Technology

# CERTIFICATE

This is to certify that the thesis entitled "**HeteroGNN Architectures for Multi-Class Network Attack Detection: SAGE vs Transformer Approaches**" by Sadia Afrin (Roll: 1910014), has been carried out under my supervision. To the best of my knowledge, this thesis is an original one and has not been submitted anywhere for any degree or diploma.

**External Member**

_____

**Fariya Tabassum**

Assistant Professor

Department of Electrical & Computer Engineering

Rajshahi University of Engineering & Technology

# Abstract

Highly-sophisticated cyber-attacks have become more and more popular in recent years, making it vital to have effective anomaly detection system in network security. In the world of advanced and constantly changing cyber-attacks, old-fashioned intrusion detection systems are not able to detect new and difficult size to locate attacks. This thesis compares two state-of-the-art advanced Heterogeneous Graph Neural Network (HeteroGNN) architectures GraphSAGE-based and TransformerConv-based models to perform multi-class anomaly detection on the network traffic data. By representing network entities, e.g., hosts and flows, as heterogeneous graphs, the work captures intricate relationships and structural dependencies that are often ignored in flat data representations. Two state-of-the-art graph learning architectures, GraphSAGE and Transformer-based GNNs, are implemented and evaluated on large-scale real-world dataset for multi-class intrusion detection, composed of benign, portScan, pingScan, and DoS traffic types. Model performance is compared on accuracy, macro F1-score, and confusion matrix. Results present the relative strengths of the models on network anomalies where Transformer-based models are used to capture long dependency and GraphSAGE as for efficient and scalable. This paper demonstrates the promising side of Heterogeneous-Graph Learning on the cybersecurity field to improve the robustness and accuracy of intrusion detection system, and provides a scalable solution for contemporary cybersecurity attacks.

**Keywords:**

Heterogeneous Graph Neural Networks (HeteroGNN), GraphSAGE, TransformerConv, Intrusion Detection System (IDS), Cybersecurity, Network Anomaly Detection, Graph Learning, CIDDS-001 Dataset, Multi-Class Classification, Graph Neural Networks (GNN).

# Table of Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| CIDDS | Coburg Intrusion Detection Data Set |
| DDoS | Distributed Denial of Service |
| DoS | Denial of Service |
| FDIA | False Data Injection Attack |
| GAE | Graph Autoencoder |
| GAN | Generative Adversarial Network |
| GAT | Graph Attention Network |
| GCN | Graph Convolutional Network |
| GNN | Graph Neural Network |
| HIN | Heterogeneous Information Network |
| HGNN | Heterogeneous Graph Neural Network |
| IDS | Intrusion Detection System |
| IP | Internet Protocol |
| LSTM | Long Short-Term Memory |
| ML | Machine Learning |
| MLP | Multi-Layer Perceptron |
| NIDS | Network-based Intrusion Detection System |
| NSL-KDD | Network Security Lab – Knowledge Discovery Dataset |
| ReLU | Rectified Linear Unit |
| RNN | Recurrent Neural Network |
| SMOTE | Synthetic Minority Over-sampling Technique |
| SVM | Support Vector Machine |
| TS | Time Series |
| UDP | User Datagram Protocol |
| ViT | Vision Transformer |

# Chapter One

## Introduction

Digital age has witnessed the novel dimension of cybersecurity threats and scenarios, which foster research and development of intelligent anomaly detection system. Conventional rule-based approach and statistical models are less effective in learning complex or novel attacks in high-dimensional packets. This has led to a growing interest in leveraging state-of-the-art deep learning techniques, in particular Graph Neural Networks (GNNs), for efficient anomaly detection in network infrastructures. In this paper, we explore two sophisticated graph learning models which are based on the following two advanced graph learning models: ones with SAGEConv as a layer in the architecture of Heterogeneous GNN and ones using Transformer-based GNN architecture to detect and classify diverse types of the network attacks.

Anomalies usually refer to pixels that have distinct spectral and spatial differences with their surrounding background pixels.[1] Anomaly detection is the process to identify abnormal patterns that significantly deviate from patterns that are typically observed.[2] The demand for real-time decision-making, the complexity of systems growing, and the steadily increasing amount of data in today's world have made anomaly detection vital. Usually the availability of sets of data with anomalies necessary for training and validation of detection models is small, which makes it more complex to define the abnormal behavior. Furthermore, data often include noise that are close to true anomalies, making it difficult to recognize and delete them.[3]

### 1.1 Overview:

The introduction starts by introducing the main issue addressed by the thesis, Network Attacks Detection through Heterogeneous Graph Neural Networks (HeteroGNN). The importance of early and accurate anomaly detection is emphasized in the context of cybersecurity and the potential of graph-based deep learning models to revolutionize traditional IDSs. The reason for embarking on such research is then described, centred on deficiencies in traditional and flat (non-hierarchical) machine learning methods in encapsulating relational and structural data in, for example, and network traffic. Justification for GraphSAGE and Graph Transformer The GraphSAGE and Graph Transfomer were chosen based on their capability to model heterogeneous node types and directed edges: these were the fundaments on which we based our model selection. Related work is extensively reviewed to highlight limitations in anomaly detection techniques and to place the proposed technique in the context of other work in this

area. Finally, an overview, based on the purposes and structure of each section, is given for the follow-up of the thesis.

## 1.2 Background:

### *1.2.1 Network Anomaly Detection:*

Unsupervised anomaly detection is to find a pattern to identify abnormal examples which do not comply with the normal behavior. In computer network defenses, this might be detecting malicious activity like port discovery, pinging floods, or DoS attacks. Most traditional intrusion detection systems make use of signature based detection or rule based detection, however, they cannot generalize to the novel or unseen anomalies. In order to detect cyber threats to a network, many companies operate a network-based intrusion detection system (NIDS). [4][5]

### *1.2.2 Graph-based Intrusion Detection:*

In contrast to flat tabular forms of representation, graph representations capture the structural relations among communicating entities (e.g., IP hosts, traffic flows), which are more informative. Graph Neural Networks (GNNs) are capable of utilizing such structures to learn node embeddings, which are learned by passing messages along the edges of the underlying graph and which facilitate a more meaningful representation of relational and topological properties of networked data. This approach leverages graphs' ability to create complex and dynamic relationships between nodes, allowing the detection of anomalous patterns associated with malicious activities.[6]

### *1.2.3 Heterogeneous Graph:*

A Heterogeneous Graph that consists of more than one type of nodes and/or more than one type of edges can be used to model complex systems that include various entities and relationships. Heterogeneous graphs in turn do not break the rich semantics available in homogeneous graphs as they can be preserved by differentiating between different types of entities and interactions. For instance, in the domain of cybersecurity, a heterogeneous graph could consist of host and flow nodes, connected via edges that represent communication or transfer of data. This structure yields a more informative representation where graph neural networks are more inclined to capture the hidden relationships and dependencies among various kinds of entities. With the thriving of

heterogeneous graph in real-world applications, heterogeneous graph neural networks have drawn increasing research attention recently. [7]

#### *1.2.4 Meta-Data:*

In HIN, metadata is the structural knowledge of node and edge types in the graph. That is, list of all node types (e.g., host, flow) and edge types that can be in the form of triples that indicate the source node type, the relationship type, and the target node type (e.g. ('host', 'generates', 'flow')). Metadata is important in controlling how graph neural networks deal with heterogeneous data. It enables the model to learn which operations to use on different parts of the graph and how message passing and feature aggregation should be performed for different types of entities. In fact, in systems like PyTorch Geometric, meta-information is required to set up different types of GNN layers, if it is heterogeneous GNNs such as HeteroConv, where specific functions correspond to each edge type. Without metadata, it would be challenging to utilize the expressive semantic structure of heterogeneous graphs for machine learning.

## 1.3 Motivation:

There were several direct reasons considering the motivation of this study:

- *Network data Complexity*: Network traffic is relational, multi-modal and composed by a variety of communication entities. More realistic is that a graph is itself composed of a heterogeneous structure.
- *Limitations of Classical Models:* Classical ML models, e.g., KNN, SVM, and DNNs, view input data as i.i.d. samples and neglect the inter-node dependencies which are of importance for network behavior analysis.
- *Prospect of Graph Learning:* Recent breakthroughs in GNNs have achieved remarkable achievements on various domains such as fraud detection, traffic prediction, and bioinformatics, which demonstrate the promise of them applied to anomaly detection.
- *Exploring Comparative Strengths: GraphSAGE* is renowned for its scalability and efficiency for large graphs, while the Transformer-based GNNs provide stronger expressiveness and global attention. In this work, we compare the power of the two networks for multi-class anomaly detection.

By benchmarking these two state-of-the-art GNNs on the same real-world intrusion detection dataset, in this work, we provide a better understanding of which model is more applicable for certain scenarios and how the graph learning can change the landscape of network security.

## 1.4 Graph Learning:

Anomaly Detection (e.g., in Cyber, Fraud, or AB) using Graph Learning aims to utilize the structure between data points that exists in the form of a graph for an efficient identification of the significant deviations between the data points. Anomalies in Graphs:

- Anomalies in nodes – anomaly nodes (nodes that are anomaly compare to their neighbors).
- Edge anomalies – Potentially fraudulent relationships between nodes.
- Subgraph anomalies – little subgraphs which are anomalous.
- Attributed anomalies – Nodes that do not have the features expected from a particular structural pattern.

We briefly describe in this section the most publicized methods of anomaly detection.

### *1.4.1 Heterogeneous Graph Neural Networks (HeteroGNN):*

The heterogeneous graph is composed of multiple node types and edge types, indicating the various relations in the real data. Different from traditional GNNs, HeteroGNNs, such as, HeteroConv, generalize to learn independent transforms for each node and edge type, making it well-suited for modeling network traffic, where hosts and flows have distinct semantics. Unlike traditional graph neural networks (GNNs), which primarily operate on graph data in non-Euclidean space to explore interactions between nodes, heterogeneous graph neural networks (HGNNs) are specifically designed to handle heterogeneous graphs consisting of multiple types of nodes and edges. [8]

### *1.4.2 Graph Sample and Aggregate (GraphSAGE):*

GraphSAGE (Graph Sample and Aggregate) is a powerful inductive method for the purposes of generating node embeddings on graphs. GraphSAGE is able to generalize to unseen nodes, thus being suitable for dynamic or large graphs. The main concept of GraphSAGE is to sample a fixed number of neighbours for every node and aggregate to their features using a selected pooling strategy like mean, LSTM or max pooling. Instead of training individual embeddings for each node, we learn a function that generates embeddings by sampling and aggregating features from a node's local neighborhood. [9] This assembled information is then concatenated with the node's own features (usually, with the concatenation and neural transformation) to obtain the updated node embeddings. By repeating the sample-and aggregate process over multiple

layers, GraphSAGE can learn multi-hop neighborhood information in a scalable way. It employs multiple aggregation mechanisms to adjust to different kinds of graph structures and applications including node classification, link prediction, and anomaly detection. With its inductive and sampling-based nature, GraphSAGE is efficient and flexible for practical graph based machine learning problems.

### *1.4.3 Graph Attention Networks (GAT):*

GATs[10] employ attention mechanisms to tune the impact of neighboring nodes during feature aggregation so that the model can focus on more informative nodes. This adaptiveness is advantageous for anomaly detection, since it can automatically down-weight irrelevant or noisy neighbors, and hence make the abnormal behaviors more salient.

### *1.4.4 Graph Transformers:*

Graph Transformers (from NLP) extends self-attention mechanisms to graph structured data, and unlike CNS, it can capture long-range structural context information without restricting it to local neighborhoods. In particular, it is beneficial for anomaly detection to access long-range correlations as well as global structural deviations, which are hard to detect by the GCN-type model.

### *1.4.5 Graph Neural Networks (GNNs):*

Because GNNs are built to operate with graph-structured data, they can be used to discover anomalies in systems such as chemical structures, social networks, and communication networks. GNNs are able to identify abnormal nodes, edges, or substructures that diverge from typical patterns by learning node embeddings and linkages.
The study of GNNs for TS data analysis is becoming more prominent in recent years. The majority of works focus on multivariate TS problems [11], [12] but univariate TS problems have also started gaining traction in different domains.[13]

### *1.4.6* **Generative** *Adversarial Networks (GANs):*

A discriminator network and a generator network that have been trained adversarially make up a GAN. GANs are used to create synthetic "normal" data for anomaly

identification. Generative adversarial networks (GANs) [14] as a recent branch of unsupervised learning methods can learn to model the distribution of highly complex medical imaging data. [15]The discriminator then uses the degree of deviation between the real data and the created "normal" data to identify abnormalities. They are very useful for picking up on minute irregularities. On the other hand, AnoGAN[16] is able to do unsupervised learning with normal data which has no labeled.[17]

#### *1.4.7* Graph *Autoencoders (GAE):*

In short, GAEs[18] are unsupervised models that encode node features into some latent space and then try to reconstruct the original graph (usually given by the adjacency matrix). Anomalies may be identified by comparing the reconstruction errors: the more anomalous nodes or links would have worse reconstructions because their patterns on the graph are rare or unexpected.

## 1.5 Literature Review:

Various classification approaches have been employed to forecast the anomaly detection by the researchers. This section provides a brief summary of the work reviewed from the literature analyzed in this area.

### 1.5.1 Case 1

In [19], Xue et al.'s. "Research on Time Series Anomaly Detection Based on Graph Neural Network", 2023. The research aims at addressing the drawbacks of traditional time series anomaly detection in learning the inter-variable dependencies, which are especially crucial in intricate systems like a CPS. Xue introduces two novel GNN models to improve the capability of temporal feature extraction: a novel GNN design, which replaces the prediction module with a Transformer, and a hybrid model that integrates GNN with Gated Recurrent Units. Both models find solutions to the time gradient and anomaly scoring issues due to the vanishing/exploding gradient manner and improved scoring precision due to the learning of the many-to-many dependencies. There is a lack of deep learning models in learning the concealed temporal relationships, and the GNN-Transformer hybrid is a motivator for multivariate time series anomaly detection.

### 1.5.2 Case 2

In [20], Chen et al. 's "An Improved GraphSAGE to Detect Power System Anomaly Based on Time-Neighbor Feature" (2023) solves one of the difficult task of anomaly detection on power system, and with the advent of smart grid, the power system becomes more and more susceptible to cyber attacks, such as false data injection attack (FDIA). The work proposes a time-neighbor based GraphSAGE for anomaly detection model, which is to exploit graph structure of power systems to aggregate features learned from the neighbouring nodes to better detect anomaly. This method utilizes features of the time-series to learn indirect anomalies that could be missed by hostile traffic imposing false data over time. The experiments verify the superiority of the model compared to traditional methods like Logistic Regression, MLP, GCN, especially in the aspect of Recall and AUC, which indicates that our method is effective to detect abnormal data and to increase precision of predictions. The model's capacity to pool information from the local nodes and consider the temporal aspect makes it a good candidate for the anomaly detection problems of power systems.

### 1.5.3 Case 3

In [21], Kim et al. s paper "Graph Anomaly Detection with Graph Neural Networks: A Survey" (2022) offers a survey on latest developments on graph anomaly detection with Graph Neural Networks (GNNs). The paper serves to emphasize that GNNs provide a promising solution to anomaly detection problem in complex graph structure environment including both dynamic and static graphs. The work classifies anomaly detection methods into static graph vs. dynamic graph and node vs. edge vs. subgraph vs. graph level. The survey presents several important GNN models including Graph Convolutional Networks (GCNs), GraphSAGE and Graph Attention Network (GAT), and how they are useful for detecting structural and attribute-based anomalies. It highlights the significance of addressing the challenges related with real-world graphs like class imbalance, noisy data and complexity of modeling intricate topologies. The paper also points out a few promising future research directions such as learning GNNs to be interpretable, handling the class imbalance problem, and for better capturing anomalies in dynamic and heterogeneous graphs. This review is integral to understanding GNN-based AD as well as the challenges we face when trying to improve the real-world performance for applying the proposed detection strategies.

### 1.5.4 Case 4

The study in [22], Y. Liu, "Anomaly Detection on Attributed Networks via Contrastive Self-Supervised Learning", Deep learning methods like DOMINANT, SpecAE adopt autoencoders to reconstruct the features, which can effectively detect graph representations for anomaly discovery. These techniques are important in the process of learning meaningful embeddings for graphs. An interesting comparison with CoLA (Contextualized Local Anomaly detection) is that, it proposes the use of "target node versus local subgraph" instance pairs, which is specifically leveraging the local subgraph components, providing a finer-grained mechanism for anomaly detection in graphs and considering local context. But these methods also have disadvantages. They often suffer from scalability limitations, especially when applied to larger graphs, and they often are unable to effectively take advantage of attributed graph data that may complement the detection of anomalous vertices. Moreover, although such approaches are successful for learning generic representations, they are not suitable for outlier detection, since they do not aim at finding local anomalies. In addition, high-dimension of features and nontrivial graph structures make them more challenging to efficiently achieve. Furthermore, these methods are highly hyper-parameter dependent, with subgraph size and embedding size requiring careful tuning for good performance.

### 1.5.5 Case 5

In [23], Yiling Zou, “Heterogeneous Network Node Classification Based on Graph Neural Networks “, Graph Neural Networks have proven efficient in addressing the unique challenges facing node classification in Heterogeneous Networks, HNNC. By bridging the contextualization gap via relational information between nodes across disparate data sources, HNNC-GNN significantly improves classification performance. However, this implementation approach also comes with its own set of limitations. Building on meta-paths, HNNC-GNN becomes heavily reliant on prior causative knowledge, inhibiting the detection of unexpected or novel anomalies. Furthermore, the additional components bolted onto HNNC meant for anomaly detections add to the complexity when the nodes classifier is the primary focus of the architecture. Finally, the construction of numerous s graphs and their subsequent aggregation using other TAGCN-based techniques are computationally expensive, especially with large-scale networks or considering network dynamics.

### 1.5.6 Case 6

In [24], Y. Zheng, “Generative and Contrastive Self-Supervised Learning for Graph Anomaly Detection”, Recently, self-supervised learning methods on graph anomaly detection have been gaining popularity, such as SL-GAD that employs generative and contrastive learning for scalable and effective scheme. This fusion allows improved abnormality detection from a self-supervised perspective, without fully depending on labels. For example, one problem in deep learning is how to efficiently learn the representation from complex structured data, such as graphs, especially on both graph embedding learning(especially) and graph generative learning. However, these approaches have several down-sides. They commonly do not fully utilize the designed graph's context information causing the worse approximation behavior based on some other fields. Further, these methods are hyperparameter sensitive, and one needs to adequately tune them for optimal performance. High-degree graphs face a specific challenge: information loss following subgraph sampling, resulting in incomplete or incorrect representation. Furthermore, they can be computationally intensive, especially on large graphs, where training and inference steps include sampling multiple graph views and performing iterative evaluations.

### 1.5.7 Case 7

The work in [25] "Research and application of Transformer based anomaly detection model: A literature review" (Ma et al.) focuses on the development and application of the transformer model for anomaly detection. It explains the powerful ability of the Transformer model to extract contextual features, which makes it appropriate for applications in anomaly detection, and particularly for time series-based and logging data. The performance of the Transformer model has been greatly improved with derivatives such as BERT, ViT and Informer. In particular, ViT achieves remarkable success in vision-based anomaly detection, by transforming the images to sequence data and taking advantage of the attention mechanism in Transformer. Nevertheless, these models suffer from high computational cost, particularly in high-resolution data, and limitations in scalability when the mode´ls are used with large datasets. Though transformer-based methods can well model long-range dependency, they usually need heavy computation and careful tuning, which limits their application to real-time detection systems. However, Transformer-based anomaly detection models, especially when they are fused with other techniques such as GAN and VAE, have been promising for making anomaly detection more efficient for a wide range of applications. Regarding future work is the improvement of scalability and efficiency of these models, particularly in the context of edge computing.

### 1.5.8 Case 8

The work in [26], introduces combine flow-level and packet-level data synthesis in a heterogeneous graph representation for real-time intrusion detection with HGNNs and explanations by LLMs 97% F1. There are still some limitations for the framework such as that it is heavily relies on availability and quality of labeled network traffic data which could be hard to obtain. It is also reported that the computational complexity of GNNs can also lead to elevated processing time and resource requirements, which can affect scalability. There is also false positive to contend with, thereby making the explanations generated by XAI not always granular enough for immediate actionable decisions.

## 1.6 Research Gap:

Despite the increased popularity of Graph Neural Networks (GNNs) for anomaly and intrusion detection, most related prior art is:

#### *1.6.1. Homogeneous Graph Modeling:*

GNN-based IDS systems Many IDS models using GNNs (e.g., GCN-GraphS (Bai & Kdd, 2019)) encode the network as a graph with single type of node (e.g., according to the set of hosts or flows) that doesn't take into account the rich semantic relationships between heterogeneous entities (hosts, flows, services, etc.).

#### *1.6.2. Lack of Comparison Between Heterogeneous GNN Architectures:*

Although works present isolated papers on HGNN, there is no or few comparative studies existing between architectures: GraphSAGE style on HGNN, and Transformer style on HGNN.

#### *1.6.3. Multi-Class Intrusion Detection with HGNNs:*

Most existing GNN-NIDS papers deal with binary classification (benign vs attack) or few attack types.

#### *1.6.4. Under-studied Edge Semantics & Handling of the Metadata:*

Few works can well model the edge feature (such as: byte count, protocol) and directly transform IPs or timestamps into node/edge features in graph view. The majority use flat table data.

#### *1.6.5. Simulation of a realistic network topology:*

Many existing GNN-based IDS benchmarks actually construct machine learning datasets directly with synthetic or poorly-structured data without including any semantics into the graph representation.

## 1.7 Objectives:

The main goal of this thesis is to propose, implement and evaluate different GraphSAGE-based and Graph Transformer-based HeteroGNN architectures for multi-class network intrusion detection. This research aims to:

- Represent network traffic as a heterogeneous graph, taking account of complex relationships among various elements (i.e., hosts, flows);
- Apply advanced graph learning technology to enhancing the accuracy, generalization and semantic comprehension of intrusion detection mechanism;
- Assess, and contrast, the efficiency of two promising HeteroGNN (GraphSAGE vs. Transformer) architectures when applied to a real-world large scale network traffic dataset;
- Which model will be selected for detecting and classifying more than one category of attack in the network (dos, portScan, pingScan, benign)?
- Summarize the virtues and limitations of leveraging graph-based deep learning techniques for intrusion detection: Make a step towards intelligent, adaptive and scalable cyber security systems.

## 1.8 Thesis Planning:

The overall organization and outline of the thesis predicated on clear ideas and intentions that, I hope the result will, in a straightforward and coherent study of anomaly detection from the advanced point of view in several application domain. It is critically important to have a well-formulated research plan to assure that these questions will be appropriate, thorough, and

scientifically sound. The thesis planning is intended to provide a roadmap of the research from the early survey of anomaly detection to establish basic metrics for evaluating the model performance and inferencing the results. The goal is to make each phase of the study a building block towards a more profound knowledge in anomaly detection.

### 1.8.1 Work Plan:

This thesis takes a systematically way for research into the subject of anomaly detection with GNNs by designing and measuring GraphSAGE and Graph Transformer models. The workflow starts with an overview of the state-of-the-art concerning anomaly detection methods, graph learning and GNNs, and continues with dataset inspection and preprocessing. They build a heterogeneous graph with the combination of the host and flow node types and perform feature engineering to take important features. To evaluate performance and optimize the hyperparameter, we design and train two GNN models, i.e., GraphSAGE and Graph Transformer, and fine-tune them.

The models are going to be examined and compared based on criteria such as accuracy, F1 score, and confusion matrix, and finally analyze each model in LDAD tasks in detail. Lastly, we will summarize our results in regards to the graph type, detection of anomalies, and model performance and propose their further utilization for anomaly detection in graph-based systems. The full report will include introduction, methods, results, and conclusions, edited and proofread to ensure that both language and academic standards are met.

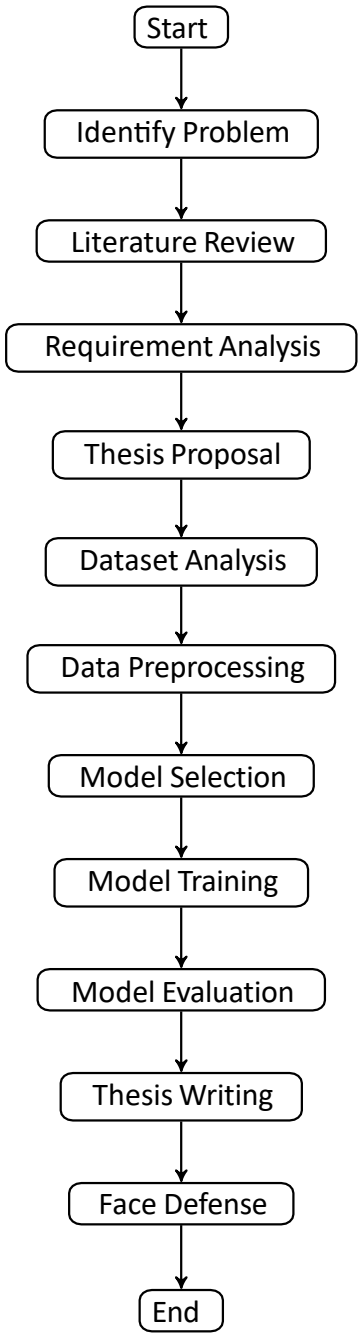


Figure 1.1: Flowchart of the Thesis Process

### 1.8.2 Timeline Diagram:

The thesis timetable has been meticulously structured to ensure systematic and efficient advancement through each phase of the research process. Table 1.1 presents the projected timeline for the thesis.

Table 1.1: Thesis Timeline

| S.No. | Task | Duration (weeks) | Date |
|---|---|---|---|
| 1 | Identify Problem | 5 | April - May 2024 |
| 2 | Literature Review | 8 | May - June 2024 |
| 3 | Requirement Analysis | 2 | July 2024 |
| 4 | Thesis Proposal | 1 | July 2024 |
| 5 | Dataset Analysis | 5 | July - August 2024 |
| 6 | Data Preprocessing | 8 | September - October 2024 |
| 7 | Pre-defense | 2 | November 2025 |
| 8 | Model Selection | 8 | November 2024 - January 2025 |
| 9 | Model Training | 6 | January - February 2025 |
| 10 | Model Evaluation | 6 | March - April 2025 |
| 11 | Thesis Writing | 2 | April 2025 |
| 12 | Defense Preparation | 1 | April 2025 |

## 1.9 Thesis Estimation:

A financial summary of the resources needed to carry out the research is given by thesis estimations. When assessing the total financial requirements, both direct costs such as (labor and data acquisition) as well as indirect costs such as (hardware and software) are included.

### 1.9.1 Overall Estimation:

The resources that are needed in order to successfully complete this thesis have been detailed in the given evaluation. Including all of the mandatory auxiliary devices, equipment, and technologies for effective data processing and model training. This not only involves hardware (e.g., computers and servers), but also software tools and platforms to create, test and optimize the AD system. The estimation also takes into account the data storage needs in order to allow for secure and safe storage of large datasets, such that they can be used and accessed throughout the research process.

The financial forecast is also complete with information on the costs of purchasing hardware and software licenses, and money spent on data collection. This includes plans for purchase of any special hardware that may be required for model training, as well as software tools that will be used in the design and implementation of the research. In addition, budget planning also includes expenses for obtaining the datasets, which are required to train and to test the proposed abnormal detection models. These costs might also involve acquisition of third-party service providers say cloud storage or data aggregators – where these will be relevant to the research.

Other costs associated with research are additionally included in the projection. Possible other quanta (these are potential extras that may be incurred apart from student fees and tuition): collaboration costs, travel for conference presentations of the theses work, any other resources for the thesis. The estimate accounts for all stages of the research process, starting at data collection/development of the model and ending at analysis and reporting.

This estimate ensures that the requisitioned resources are planned and budgeted and provides a detailed breakdown of the materials and services required and creates a financial forecast for project. It provides a guide to the logistics and financial management of the thesis, to ensure the successful and timely completion of the study, and to preempt issues of resource scarcity and budget constraints.

### 1.9.2 Equipment Estimation:

The computing resources required for data preprocessing, model training and evaluation is as follows:

Table 1.2: System Configuration

| Specification | Details |
|---|---|
| Processor | Intel Core i3 9th Gen |
| Installed RAM | 8.0 GB DDR4 |
| System Type | 64-bit operating system, x64-based processor |
| GPU | Intel Iris Xe graphics |
| Software Tools | Jupyter Notebook, Kaggle, Google Colab |

### 1.9.3 Financial Estimation:

The financial expenditures associated with conducting this thesis have been carefully considered and are outlined as follows:

- **Hardware Costs**: This research was conducted using a personal computer owned by the researcher. As no additional computing hardware or peripheral devices were purchased specifically for this project, there were no hardware-related expenses incurred during the thesis work.
- **Software Costs**: All the software applications and tools utilized throughout the thesis are open-source and freely available. These include programming environments, data analysis tools, machine learning libraries, and visualization frameworks. Consequently, there were no financial costs associated with software acquisition or licensing.
- **Data Acquisition Costs**: The dataset employed in this study was sourced from Kaggle, a reputable and widely used open data platform. The dataset is publicly available and was accessed without any associated fees or subscription requirements. Thus, no expenditure was necessary for data procurement.

In conclusion, the thesis was completed without incurring any direct financial costs, as the hardware, software, and data resources were all accessible without charge. This highlights the feasibility of conducting high-impact academic research through efficient utilization of freely available tools and resources.

## 1.10 Thesis Outline:

The remainder of this thesis is organized as follows:

- Chapter 2 presents a concise overview of the selected case study, followed by a detailed explanation of the proposed methodology.
- Chapter 3 focuses on the design and implementation aspects of the research. It includes comprehensive descriptions of the dataset and the models employed in this study.
- Chapter 4 discusses the results obtained through the experimental process. This chapter includes an in-depth analysis of the outcomes and a comparative assessment with previous studies.
- Chapter 5 explores the broader implications of the research, particularly its potential social and environmental impacts.

- Chapter 6 identifies and discusses the complex engineering problems and activities addressed throughout the course of this thesis.
- Chapter 7 concludes the report by summarizing the key contributions of the study and proposing directions for future research,

# Chapter 2

# Methodology

A structured process is crucial to get organized the way it is researched. It guarantees that all aspects are structured, transparent, and rigorous, ultimately leading to validity and reproducibility. Productivity boosters for researchers, are resource management, Time effectiveness as well as using the protocols.

## 2.1 Overview:

This chapter provides an overview of the methods that were used to detect anomaly in this research using complex graph learning methods. The goal is to construct an accurate, scalable and robust detection mechanism that is also able to respond to the heterogenous behavior that is prevalent in real time data. The approach includes main elements: a literature review of existing models, the development of a new model, model architecture, and justification for the study approach and methods requested.

The objective of this approach is to develop a machine learning model for the classification of network traffic data to different types of attack (normal, port scan, denial of service (DoS) and ping scan). The method employs a graph-based neural network architecture that is capable of processing heterogeneous information as source-destination IP addresses, network protocol flags, and other characteristics relating to traffic characterization.

## 2.2 Case Study:

In this section, discussion of two paper is done which were selected to study.

The first paper [20], “An improved GraphSAGE for power system anomaly detection with time-neighbor feature” by Chen et al. (2023), proposes an original solution for Defence against Anomalous behaviour in PowEr system (DAPE), with particular attention to the behavior of False Data Injection Attacks (FDIA). In this paper, we present a graph-based model where nodes correspond to power grid components and edges represent the physical or logical connections. The key novelty is to incorporate spatial and temporal dependencies in the learning algorithm through a variant of

GraphSAGE architecture. In particular, the model measures the dissimilarity between a node and its neighbors based on not only spatial characteristics (topological proximity) but historical tendency (behavioral changes over history), and this strengthens the model's detection ability on latent and temporal characters for anomalies. To address the issue of insufficient training labeled data in practical, the model uses unsupervised label estimation method to train effectively. The experiments are performed on the Elliptic database which is originally used for anomaly detection in cryptocurrency transactions but is here modified due to similarities in the transaction graph structure to a power system. The experimental results show that our proposed time-neighbor-based GraphSAGE model achieves notable improvements over the baselines. It attained precision of 0.618, a recall of 0.802, F1-score of 0.661, and AUC score of 0.913 showing its effectiveness for identification of anomalous activities with high recall in the presence of skewed data.

The work As there are many kinds of GNN for the unsupervised learning tasks, we focus on "Anomaly Detection Using Graph Neural Network" used by while the study proposed in [19] creates a one class master, who is trained in the variant of one class GNN used for the unsupervised learning tasks, and we fo-cus on "Anomaly Detection Using Graph Neural the more complex deep herein, we present the generated, grouped data used to the police officer class.

It shows results on ACCURACY of 94. 68%, but there are several limitations and concerns that deserve attention. A major disadvantage of the above approach is that it may overfit. The low difference between the train and test accuracy could indicate good generalization of the network, but high training accuracy is an indication that a model has learned the patterns in the train too seriously) it, which will not be able to generalise as well with real-life situations that are either more complicated or changing over time. This problem is especially relevant in the field of cybersecurity as new attack types and adversarial actions emerge commonly 2 Situation Understanding in Wireless Network Deployment that has seen some changes in the problems and constraints landscape frequently emerge. Furthermore, the selected usage of the NSL-KDD data set, which is popular when researching the subject, serves also as an issue since this dataset is rather outdated, since it does not offer any tooling to pen testers through which they would be able to test that their IDS will not detect their own attacks, and does not have any availability(annotation)beyond classification. some recent attack features and it deals with class imbalanced data, especially for rare but meaningful attack types such as User-to-Root (U2R) and Remote-to-Local (R2L). The dependence on overall accuracy for a

performance measure may be misleading if it is not supported by class-specific precision, recall, and F1-score.

While graph neural networks (GNNs) are successful in learning relational structures, their application can be computationally costly, particularly for large-scale data of graphs. from real-time traffic logs. These limitations underscore the importance of further testing, optimization and validation before applying these models. trustworthy over (operational)cyber-physical systems.

## 2.3 Proposed Methodology:

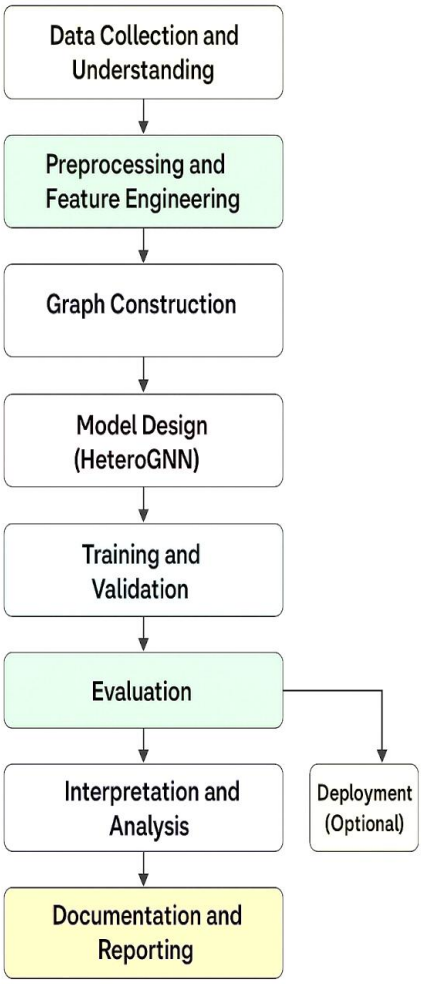


Figure 2.1: The Proposed Methodology

Here each block in the process plays a specific role. The components are described below in detail:

### *1. Data Collection and Initial Preprocessing*

**Input**: The initial dataset containing network traffic data is read using pandas.

**Step 1.1**: **Column Removal**:

- Unnecessary columnsare dropped.
- Data Flow: Data remains as tabular information in the DataFrame.

**Step 1.2**: **Attack Type Normalization**:

- The attackType column is cleaned by replacing missing values (---) with benign.
- Data Flow: The DataFrame is updated with clean attackType labels.

### *2. Feature Engineering*

**Step 2.1**: **One-Hot Encoding of Flags**:

- The Flags column is one-hot encoded into binary features
- Data Flow: The new one-hot encoded columns are added to the DataFrame.

**Step 2.2**: **IP Address Engineering**:

- **Source IP Address**:
  - The Src IP Addr is split into four octets and transformed into binary strings.
  - The octets are then concatenated to create a new feature representing the full binary IP address.
- **Destination IP Address**:
  - Similarly, the Dst IP Addr is split into octets and converted into a binary string.
- Data Flow: The DataFrame is updated with new columns representing the binary format of the source and destination IP addresses.

**Step 2.3**: **Bytes Handling**:

- Missing Bytes values are handled by converting any string representation into numeric values.
- Data Flow: The Bytes column is updated with numeric values for further analysis.

**Step 2.4**: **Categorical Encoding**:

- The Proto and attackType columns are one-hot encoded to convert categorical data into numerical form.
- Data Flow: The encoded columns are added to the DataFrame.

### *3. Data Transformation and Scaling*

**Step 3.1**: **Feature Scaling**:

- The continuous features Duration, Packets, and Bytes are scaled using the PowerTransformer to normalize their distribution.
- Data Flow: The scaled values replace the original columns in the DataFrame.

### *4. Data Splitting*

**Step 4.1**: **Training, Validation, and Test Split**:

- The dataset is split into training, validation, and test sets using stratified sampling based on attack types.
- Data Flow:
  - df_train: 80% of the data for training.
  - df_val: 10% of the data for validation.
  - df_test: 10% of the data for testing.

### *5. Graph Construction*

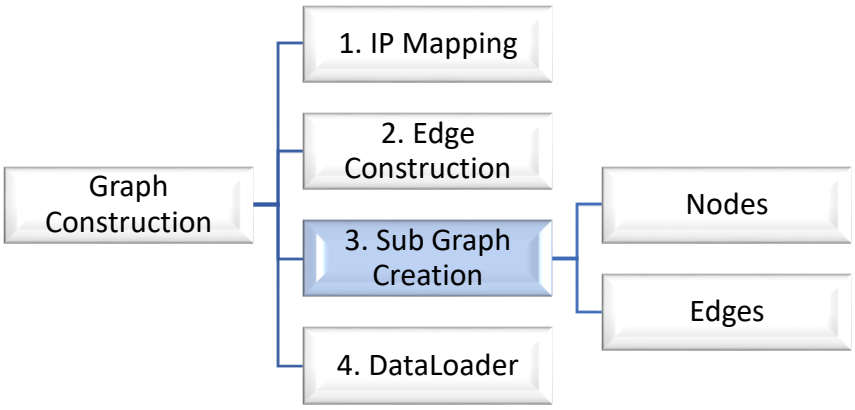


Figure 2.2: Detailed Graph Construction

**Step 5.1**: **Source and Destination IP Mappings**:

- **Source IP**: Unique source IP addresses are mapped to indices.
- **Destination IP**: Similarly, unique destination IP addresses are mapped.
- Data Flow: A dictionary (ip_map) is created to represent the mappings.

**Step 5.2**: **Edge Construction**:

- The get_connections() function creates edges between the source and destination IPs for each flow.
- Data Flow: Edges are represented as pairs of indices that define relationships between hosts (source and destination IPs) and network traffic flows.

**Step 5.3**: **Subgraph Creation**:

- The dataset is divided into smaller subgraphs, each representing a batch of network traffic data.
- Data Flow: For each subgraph, host features (source and destination IP features) and flow features (traffic-related features) are processed.
- **Nodes**:
  - **Host Nodes**: Represent IP addresses (source and destination).
  - **Flow Nodes**: Represent the network traffic features.

- **Edges**: Connect host nodes and flow nodes, representing interactions between source and destination IPs.

**Step 5.4**: **Heterogeneous Data Creation**:

- Using HeteroData from torch-geometric, the graph structure is created to hold different types of nodes (hosts and flows) and edges (connections between hosts and flows).
- Data Flow: The subgraphs are created and stored as batches in the DataLoader.

### *6. Model Architecture*

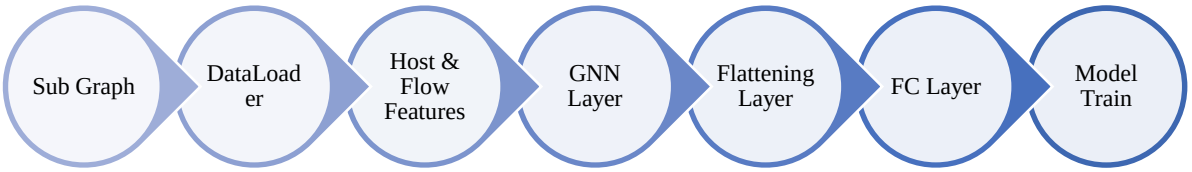


Figure 2.3: Detailed Methodology

**Step 6.1**: **Host and Flow Features**:

The host and flow features are used as inputs to the Graph Neural Network.

- **Host Features**: Consist of binary representations of source and destination IP addresses.
- **Flow Features**: Include network traffic metrics such as Duration, Packets, Bytes, and one-hot encoded flags.

**Step 6.2**: **Graph Neural Network Layer**:

- The data passes through the Graph Neural Network (GNN) layer where the relationships between host nodes and flow nodes are learned.
- Data Flow: Host and flow features interact via graph convolutions to produce feature embeddings for each node.

**Step 6.3**: **Flattening Layer**:

- The output from the GNN layer is flattened to make it suitable for the fully connected layers.
- Data Flow: Node embeddings are reshaped into a 1D format to pass to the subsequent layers.

**Step 6.4**: **Fully Connected Layer**:

- The flattened features are passed through a fully connected layer to predict the final classification.
- Data Flow: The output is the prediction for the type of attack or benign traffic.

***7. Training***

**Step 7.1**: **Loss Calculation and Backpropagation**:

- The model is trained using the Adam optimizer, and the loss is calculated using Cross-Entropy Loss (since this is a classification task).
- Data Flow: Gradients are calculated and used to update the model weights during training.

Due to the multi-class classification both models are trained with cross-entropy loss function.

The model is trained using cross-entropy loss, which for multi-class classification is:

$$\text{Loss} = -\sum_{i=1}^{N} y_i \log(p_i) \qquad 2.1$$

Where:

- $y_i$ is the true label of the $i$-th instance
- $p_i$ is the predicted probability for the $i$-th class
- $N$ is the number of instances in the batch

**Optimizer:** We use Adam optimizer with a learning rate of 0.001.

**Batch Size:** A batch size of 16 is used for the two models.

**Validation :** The models are validated on a validation set, tuning of hyperparameters for best performance.

**DataLoader**: DataLoader is implemented for the efficient batched data input in the models. The function create_dataloader() does the following things:

- It groups the subgraphs into batches.
- It preprocesses the node features (host. x, flow. x) and labels (flow. y) for training.
- It initializes the edge indices of the host and flow nodes.
- The DataLoader is for efficient loading of data during training and evaluation.
- The DataLoader performs batching using the following equation for the number of batches $B$ :

$$B = \left\lceil \frac{N}{\text{Batch Size}} \right\rceil \qquad 2.2$$

Where $N$ is the number of subgraphs.

## *8. Evaluation*

**Step 8.1**: **Model Evaluation**:

- After training, the model is evaluated on the test dataset.
- Data Flow: The test set is passed through the trained model to predict attack types.

**Step 8.2**: **Metrics Calculation**:

- The model's performance is evaluated using metrics such as F1-Score**,** Confusion Matrix**,** and Classification Report**.**
- Data Flow: Predictions and ground truth are compared to calculate the classification metrics.

## 2.4 Model Overview

For attack detection two graph-based architectures are considered: GraphSAGE (a GNN based model) and Transformer based GNN models.

### *2.4.a. GraphSAGE:*

**Input Representation:** The input features of the GraphSAGE model are the host and flow node features.

**Aggregation:** The aggregation process of the GraphSAGE to get the optimal representation by considering the information of its neighbors. Aggregation approach For aggregation, we have used mean aggregation method in this study.

**Learning:** The GraphSAGE model learns embeddings using subgraphs, with the edge indices (relationship between nodes) and their corresponding features to train the model.

The input feature matrix $X$ for the GraphSAGE model contains node features. The aggregation for GraphSAGE can be represented as:

$$h_v^{(k)} - \text{Aggregator}\left(\left\{h_u^{(k-1)}: u \in \mathcal{N}(v)\right\}\right) \quad 2.3$$

Where:

- $h_v^{(k)}$ is the embedding of node $v$ at the $k$-th layer
- $\mathcal{N}(v)$ is the set of neighbors of node $v$
- Aggregator is a function (e.g., Mean Aggregator) that computes a fixed-size representation of the neighbors' features.

### *2.4.b. Transformer:*

**Input Representation:** The host and a flow node features are used as input features, as in GraphSAGE. But the Transformer models can use self attention to pay attention only to the neighbor nodes and semantics while considering long-distance correlations.

**Model organization:** This model is a Transformer-like model with the encoder-decoder framework, of which the encoder is constructed through multi-head self-

attention over the graph node features, and the decoder is designed to complete the classification on the provided graph.

The Transformer uses self-attention to compute the output representation for each node. The self-attention mechanism computes the attention score for node $i$ as:

$$\text{Attention } (Q, K, V) - \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \qquad 2.4$$

Where:

- $Q$ is the query matrix for the node
- $K$ is the key matrix for the node
- $V$ is the value matrix for the node
- $d_k$ is the dimension of the key matrix

The output of the self-attention mechanism is then used as input to the encoder-decoder structure.

## 2.5 Justification of the Study:

The contemporary network traffic is characterised by the diversity and complexity of traffic patterns which is extremely challenging for conventional IDS. As cyber attacks advance in complexity of operations, and in particular with the adoption of multi-layered and distributed attacking strategies, the importance of more sophisticated and effective methods which are able to effectively capture relationship of irrelevant attacks to the network behaviour becomes indispensable. In this direction, the research of Heterogeneous Graph Neural Networks (HeteroGNNs) for multi-class network attack detection is a promising direction.

The conventional approaches falls short of capturing complex relationships between different network entities including hosts, flows and ports. These methods suffer from failing to exploit the rich, multi-entity relationships naturally existing within network data. By employing heterogeneous graph, where nodes correspond to entities (e.g., hosts and flows) while edge stands for the connections (e.g., host ↔ flow), this paper aims to fill this vacuum. By doing so, HeteroGNNs can better model the heterogeneous nature of network traffic, and in turn capture more realistic network settings. The capacity for HeteroGNNs

to retain cross-entity interactions leads to better feature extraction of information that is essential for recognizing sophisticated attack patterns.

This paper evaluates and compares two HeteroGNN architectures, i.e., the latest SAGE and Transformer-based models to detect and categorize multiple MNN attacks such as benign, portScan, pingScan and DoS. The multi-class characteristic of this task calls for discriminative classifiers that can distinguish between small variations of attacks, something which simple classifiers might not capture.

What we are exploring here is not just this comparison of architectures (SAGE vs Transformer models), but the novel graph creation strategies we utilize. For example, by using one-hot encoding for the ports, IP-to-ID transformation, and linking between flow and host, the important metadata are well embedded into the graph structure. Such approaches retain the intrinsic relation between network entities, and further increase the capacity of the model to apply network-level metadata to the decision making process. This is particularly important to make the accurate determination of an attack, especially in large scale networks where the metadata provides a significant input of the differentiation between normal and un-normal behaviour.

In addition, the dataset used in this study was realistic in nature and very large in scale with over 539,000 records, which adequately represents real enterprise-scale networks. That these results are not only theoretically accurate but also empirically significant is guaranteed. This work aims to provide an influential reference for the following researches and consider multi-class attack detection as an important detection problem in network security to improve security by using effective and scalable, machine learning models that is trained on a wide range of Network attack scenarios and behaviors.

Finally, this study justifies the further investigation of HeteroGNNs for multi-class network attack detection since they can capture the complicated interactions among the traffic features and offer a more sophisticated way to classify threats. Through the comparison of SAGE and Transformer models, the paper hopes to bridge an essential empty in the evaluation of these graph-based architectures for network security, to provide useful insights that may inspire more efficient, accurate, and scalable IDS solutions.

# Chapter 3

## Design and Implementation

The particular set of activities required to accomplish the project task are outlined in this chapter. In the Design chapter This includes the experiment setup for this project. After that, the mathematics of the models is explained. The dataset and model are described and the model is done in Section Implementation.

### 3.1 Design Overview

In this work, we develop a framework to identify multi-class network attacks based on heterogeneous graph neural networks. Two variants of GNN architectures SAGEConv and TransformerConv are adopted and compared. The pipeline includes a number of essential steps that direct the entire workflow, from data preprocessing to model evaluation:

***Step 1: Understanding and Gathering Data:***

The dataset contains 539,998 network traffic samples labeled as multi-class as Benign, DoS, PortScan, and PingScan. Each sample contains meta informationof the flows, the hosts and the labels for differentiation between normal and attack traffic.

***Step 2: Preprocessing and Feature Engineering***

Preprocessing stage includes handling of missing values and normalization of numerical features to a common scale. For categorical variables, one-hot encoding is applied. Features are of two types: host features and flow features, and the target labels (Benign, DoS, PortScan, PingScan) are label encoded for classification purposes.

***Step 3: Graph Construction***

We form the heterogeneous graph by viewing it as a PyTorch Geometric graph. Two types of nodes exist: host and flow, which are connected by two types of edges: host → flow and flow → host. Features vectors are associated to each type of node to create a directed, bidirectional graph. This structure represents the relationships between the hosts and the flows.

***Step 4: Model Design***

(A) SAGEConv HeteroGNN: A 4-layer model with the SAGEConv target aggregation mechanism performing edge-type-dependent feature aggregation. LeakyReLU is the choice of the activation function, while flow node embeddings are input to a linear classifier.

(B) TransformerConv-based HeteroGNN: A 4-layer HeteroGNN, where TransformerConv [velivckovic2017graph] for attention-based edge representation. Attention is used to enhance the performance for different types of edges, and the embeddings are then passed through a linear output layer.

***Step 6: Evaluation***

Various performance measures are calculated such as the accuracy, precision, recall and the macro F1-score. Class-level performance of each model is assessed through confusion matrices. We discussed the pros and cons of our proposed transformation in comparison to SAGEConv and TransformerConv.

***Step 7: Data Interpretation and Analysis***

The models are evaluated for their competency to detect various attack scenarios. In order to compare the performances of the models on different types of attacks like DoS or PortS-c an, the models will also be compared in their performance per attack class.

***Phase 8: Documentation and Reporting***

The thesis report will consist of Introduction, Related Works, Methodology, Results, Discussion, and Conclusion chapters. Illustrative key visual elements like graphs on structures, model architecture diagrams, confusion matrices and tables for comparison will be added to better convey the findings.

### 3.1.1 Experimental Setup:

We implemented our experiments in Python 3.10 in a Jupyter Notebook with a GPU implementation using Google Colab. All results were obtained under controlled conditions from reproducible random seeds to ensure fair comparison. We ran the

experiments repeating train-test splits to show the stability and generalization ability of our method.

### 3.1.2. Python Libraries:

For construction and evaluation of the models, Python libraries were employed for building and assessing the models, the following Python libraries were used. They give us very useful tools to build better data-driven application.

- **NumPy:** NumPy is the fundamental library for array and matrix operations. "It provides extensive facilities for the high-level types and operations, but also for the low-level linear algebra operations and random number generation." It is a popular tool for scientific computing because it enables simple manipulation of large datasets and multidimensional arrays. The high-level array interface also makes it easy to work as fast as possible by running operations in parallel using multiple processors.
- **Pandas:** Pandas gem for data manipulation and analysis. It offers two primary data structures — Series and DataFrame, both of which are easy to use and enable fast manipulation of structured data(like tables). Pandas allows to do things like roughly filtering, merging, grouping, and reshaping data - which is glorious for data prepping. It's frequently used early in the process of analyzing data or when preparing data for machine-learning.
- **Matplotlib, Seaborn:** Matplotlib is a flexible plotting library which can be used to generate static, animated, and interactive visualizations in Python. It underpins the vast majority of visualizations we see in Python.

  By using Seaborn it can create complex plots in a very short amount of time - Seaborn actually provides a very high-level interface to draw attractive and informative statistical graphics. Both are very popular libraries among the data science community to plot graphs, in particular during data exploration for incidence matrices, statistical analysis and confusion matrices.

- **scikit-learn:** scikit-learn is the most common library used for machine learning. It is equipped with a variety of functions for creating models, comparing their performance, and recording results including accuracy, precision, F1 score, and confusion matrix etc. It also provides algorithms for classification, regression, clustering, and dimensionality reduction, therefore easy to applied to a wide range of machine learning scenarios. It is popular for its easy-to-use API and the performance of the machine learning algorithms that it offers.

- **Torch(PyTorch):** Torch, also known as PyTorch, is a deep learning framework that offers flexibility and speed to build neural networks. It has dynamic computation graphs which is suitable in research and practical use. Use cases for PyTorch include applications in image recognition, natural language processing, and reinforced learning. It works well with other libraries like NumPy and scikit-learn and follows a deep learning workflow making it easy to use with machine learning tasks.
- **torch_geometric:** This is a library that extends PyTorch to work with graphbased data structures. It provides tools for constructing graph-related deep learning models,offers node classification and link prediction support and it is designed to facilitate fastcustomization alongside other GNNbased on computational graphs operations. torch_geometric is adopted in scenarios where data itself has a natural graph structure, like social network, molecular chemistry and recommendation systems, and supports more complex learning models.
- **NetworkX:** A stable library for establishing the structure of, and analyzing the dynamics of, complex networks. This can help to visually analyze graph structures, fabricate relationships among nodes in networks, and so on. It is best for working with networks such as social networks, transport routes, or communication networks and is well suited to network analysis.
- **tqdm:** tqdm is short for "taqaddum," which is Arabic for progress. It's a library that allows to put a progress bar in loops, which is very very useful in longer running processes such as model training. With just an innocuous line of code, tqdm gives a visual feedback on the progress of the process at hand, making it easy to monitor how things are progressing, and approximating the time it's all going to come to an end. It is especially helpful when to deal with pretty big size of data set or time-consuming calculations.
- **imblearn:** imblearn (imbalanced-learn) is a library of Python that helps combat class imbalance in machine learning. It provides us with algorithms such as the Synthetic Minority Over-sampling Technique(SMOTE) to create synthetic samples of the minority class and different kinds of methods to under-sample the majority class. This library increases the model´s generalization capacity on imbalanced datasets, given that in most real world classification problems some classes have more examples than another ones.
- **datetime:** The Python datetime module provides classes to work with date and time. It lets do things like work with dates and times, perform date arithmetic and formatting, and parse and format time strings. It provides the noir date and

datetime classes which allow simple manipulation of dates and times without even having to think about it.

- **itertools:** Python itertools module provides a set of fast, memory efficient tools known as iterators. It provides capabilities such as count() that yields an infinite sequence of numbers (counting upward given the first element) and cycle() that repeats an iterable as many times as pass to it and repeat() that simply repeats an element. chain() can be used to concatenate iterables together, and zip_longest() can be used to perform the same zip operation as zip, but filling short iterables with a provided value. It also provides utilities to generate permutations (permutations()) and combinations (combinations()) of a sequence, as well as to compute the Cartesian product of iterables using product(). These are useful to deal with large dataset and keeps iterating small amount chunk without loading huge data in memory.

## 3.2 Mathematical Formulation:

In this section, the mathematical algorithms of the models used in this thesis are discussed.

Model-1 Algorithm:

---

**Algorithm 1** Heterogeneous GraphSAGE-based Classification

---

**Input:** Heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathscr{E})$,

Node features $\{X_v \mid v \in \mathcal{V}\}$, Edge types $\{e_i\}$,

Labelled flow nodes Y

**Output:** Predicted class labels $\hat{Y}$ for flow nodes

1: Initialize node embeddings $H^0 \leftarrow X_v$

2: for l = 1 to L do ▷ L = 4 layers

3: for each edge type $e_i \in \mathscr{E}$ do

4: $H_i \leftarrow$ SAGEConv(H^{(l-1)}, $e_i$) ▷ Type-specific SAGE aggregation

5: end for

6: H^{(l)} $\leftarrow$ LeakyReLU($H_i$) ▷ Combine edge-type messages

7: end for

8: Z $\leftarrow$ H^{(L)} ▷ Final flow embeddings

9: $\hat{Y} \leftarrow \text{Softmax}(W_o Z + b_o)$ ▷ Linear classifier

10: **Return** $\hat{Y}$

---

Model-2 Algorithm:

---

**Algorithm 2** Heterogeneous Transformer-based Classification

---

**Input:** Heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathscr{E})$,

Node features $\{X_v \mid v \in \mathcal{V}\}$,

Edge types $\{e_i\}$,

Labelled flow nodes Y

**Output:** Predicted class labels $\hat{Y}$ for flow nodes

1: Initialize node embeddings $H^0 \leftarrow X_v$

2: for l = 1 to L do ▷ L = 4 layers

3: for each edge type $e_i \in \mathscr{E}$ do

4: Compute queries, keys, values:

$Q_i = W\_Q\ H^{(l-1)}$, $K_i = W\_K\ H^{(l-1)}$, $V_i = W\_V\ H^{(l-1)}$

5: Compute attention weights:

$\alpha_i = \text{softmax}(Q_i K_i^T / \sqrt{d})$

6: $H_i \leftarrow \alpha_i V_i$ ▷ Attention-weighted sum

7: end for

8: $H^{(l)} \leftarrow \text{LeakyReLU}(H_i)$ ▷ Update node embeddings

9: end for

10: $Z \leftarrow H^{(L)}$ ▷ Final flow embeddings

11: $\hat{Y} \leftarrow \text{Softmax}(W_o Z + b_o)$ ▷ Linear classifier

10: **Return** $\hat{Y}$

---

## 3.3 Implementation:

In this chapter will be the talk about the dataset used in the thesis and the selected models.

### *3.3.1 Dataset Selection:*

The dataset used in this study is a network traffic dataset that was manually designed for the purpose of multi-class anomaly detection in cybersecurity related problems.

The CIDDS-001 data set was captured over a period of four weeks and contains nearly 32 millions flows. Thereof, about 31 millions flows were captured within the OpenStack environment. About 0.7 million flows were captured at the external server. 11 The CIDDS-001 data set includes 92 attacks. 70 attacks were executed within the OpenStack environment and 22 attacks targeted the external server.[27]

The selection of this dataset is based on:

- It includes sample network traffic logs in the form of both attack and benign traffic traces.
- It is natural from a graph theoretic modeling perspective, because it captures the interactions between hosts (devices) and flows (connections).
- One of the reasons is that it is capable of multi-class classification in which we are able to identify different categories of attacks.
- It is a metadata dataset that can be used in the creation of a heterogeneous graph where nodes and edges have different types and meanings.

This rendered the dataset a good candidate to be used for benchmarking advanced graph neural networks (e.g., SAGEConv and TransformerConv) in a heterogeneous GNN framework.

### *3.3.2 Dataset Description:*

CIDDS-001 is a labeled flow-based data set designed for testing the capabilities of anomaly-based network intrusion detection systems. OpenStack was used to simulate a small business environment where CIDDS-001 is generated. This environment is composed of numerous clients and common servers such as E-Mail servers, or Web servers. The client side is written with Python scripts which imitate typical user actions. The CIDDS-001 dataset consists of unidirectional Netflows. Table 3.1 provides a

summary of the characteristics of the CIDDS-001 data set. The attributes 1 to 10 are default NetFlow attributes whereas the attributes 11 to 14 are added by us during the labelling process.[27]

Table 3.1: Attribute information of the Dataset

| S.I. | Attribute Name | Attribute Description | Attribute Type |
|---|---|---|---|
| 1 | Src IP | Source IP Address | Categorical |
| 2 | Src Port | Source Port | Numeric |
| 3 | Dest IP | Destination IP Address | Categorical |
| 4 | Dest Port | Destination Port | Numeric |
| 5 | Proto | Transport Protocol (e.g., ICMP, TCP, or UDP) | Categorical |
| 6 | Date first seen | Start time flow first seen | Timestamp / DateTime |
| 7 | Duration | Duration of the flow | Numeric (Time) |
| 8 | Bytes | Number of transmitted bytes | Numeric |
| 9 | Packets | Number of transmitted packets | Numeric |
| 10 | Flags | Concatenation of all TCP Flags | Categorical |
| 11 | Class | Class label (normal, attacker, victim, suspicious or unknown) | Categorical |
| 12 | AttackType | Type of Attack (portScan, dos, bruteForce, —) | Categorical |
| 13 | AttackID | Unique attack id. All flows which belong to the same attack carry the same attack id. | Categorical / ID |
| 14 | AttackDescription | Additional info about the set attack parameters (e.g., number of attempted password guesses for SSH-Brute-Force attacks) | Text / String |

**Comprehensive Feature Descriptions**

1. **Src IP (Source IP Address):**
   This feature represents the IP address of the source host that initiated the connection or data flow. It is useful for identifying the origin of network traffic, which is critical in network forensics and intrusion detection.
2. **Src Port (Source Port):**
   The source port number used by the sending host in the communication. This is usually a random high-numbered port used for establishing outbound connections, and it helps distinguish different sessions initiated from the same IP.

3. **Dest IP (Destination IP Address):**
   The IP address of the target or receiving host in the network communication. This attribute helps in identifying which system was targeted or communicated with.
4. **Dest Port (Destination Port):**
   The port on the destination host that is being accessed or attacked. Common values include port 80 (HTTP), port 443 (HTTPS), port 22 (SSH), etc., and it plays a significant role in categorizing the type of service involved.
5. **Proto (Protocol):**
   Indicates the transport-layer protocol used for the communication. Examples include TCP, UDP, and ICMP. This field is essential in classifying the behavior of the network flow and identifying protocol-specific attacks.
6. **Date first seen:**
   This denotes the timestamp of when the data flow or connection was first observed. It is useful for constructing timelines of activity and detecting abnormal patterns over time.
7. **Duration:**
   The total length of time, typically in seconds or milliseconds, that the flow lasted. It is useful for distinguishing between short-lived scanning activities and long-duration sessions that may indicate data exfiltration or prolonged intrusion.
8. **Bytes:**
   Represents the total number of bytes transferred during the session or flow. High or unusually low byte counts may indicate abnormal behavior or inefficient communication attempts.
9. **Packets:**
   This is the total number of data packets transmitted in the flow. It helps in understanding the flow density and can be used in conjunction with the Bytes feature to assess the nature of the traffic.
10. **Flags:**
    Contains a combination of TCP flags (e.g., SYN, ACK, FIN, RST) observed during the session. These flags signal control messages in TCP and can be analyzed to detect scanning, connection attempts, or session termination.
11. **Class:**
    A label indicating the classification of the flow or connection. Classes can include "normal" (benign traffic), "attacker", "victim", "suspicious", or "unknown", and are critical for supervised learning models in cybersecurity.
12. **AttackType:**
    This field describes the specific category of attack associated with the connection, if

any. Examples include "portScan", "DoS" etc. It enables more granular analysis of attack vectors and patterns.

13. **AttackID:**
    A unique identifier assigned to each attack. Flows sharing the same AttackID are considered part of the same attack instance. This facilitates grouping and tracking of complex or distributed attacks.
14. **AttackDescription:**
    Provides detailed contextual information about the attack, such as the number of password attempts in a brute-force attack. This narrative can aid in forensic analysis and in generating descriptive analytics.

### *3.3.3 Exploratory Data Analysis*

In this step, the characteristics of the dataset will be examined and visualized to gain insights and identify patterns.

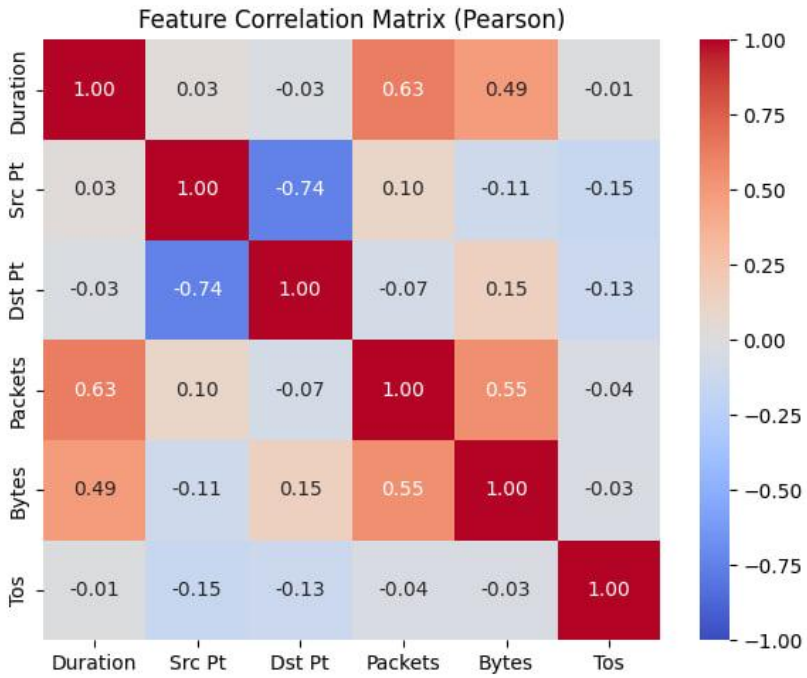


Figure 3.1: Correlation Matrix Heatmap of Dataset

The correlation analysis shows that no feature pairs are strongly correlated ($|r| > 0.8$), suggesting that the redundancy among features is not high, which is also beneficial for

modeling. While the deleted features should be included in the list of features with lesser importance, some moderate positive relationships are observed between Packets, Duration, Bytes and the rest of the features | implying that those features are meaningful flow behavior indicative features. A high negative correlation between Source Port and Destination Port (-0.74) could suggest that there is some scanning or some sort of service pattern scan going on that is highly useful for identifying anomalies. In general feature selection is easy and we can use models that are sensitive for multicollinearity.

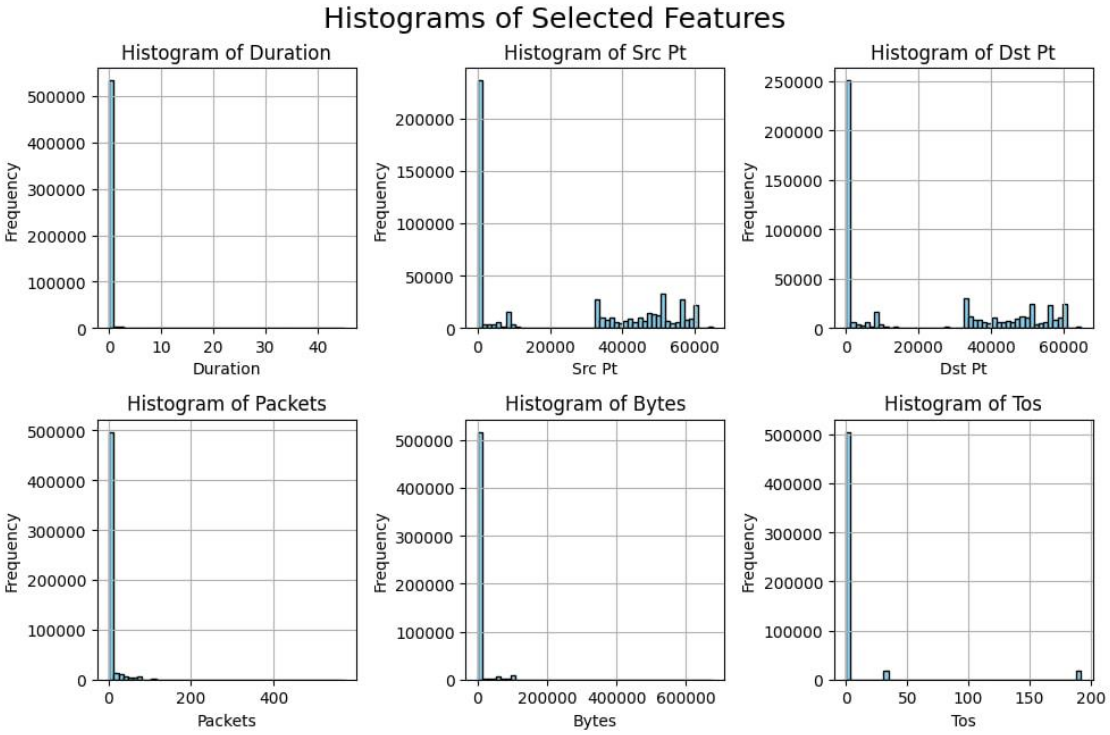


Figure 3.2: Histograms of Features

The quality of the data per characteristic:

**Duration:** The histogram for the "Duration" attribute reveals that it is heavily skewed towards 0, with only a small number of entries for high duration values. This may mean an intake dataset littered with short connections or bouts of low activity, for example a network trace, but they could want to con rm that the duration distribution is represented well for certain attack detection uses.

**Src Pt and Dst Pt (Source Port and Destination Port):** Both histograms of these features have high occurrences in lower values whereas are more dump for the higher values. These characteristics appear to exhibit some variability, and can be used to distinguish between different network traffic. But for such skewed distributions, we need more processings, normalizations.

**Packets:** Like “Duration” this feature is highly skewed, where majority of the packets have very low count. This might indicate that the data set contains much low-packet type of traffic that is important, or needs to be treated such as log transform to highlight the patterns for detecting the attacks.

**Bytes:** The "Bytes" field has a very skewed distribution too, showing a few datagrams that have a large number of bytes. It might be interesting trying to understand if such type of outliers prevent the model to learn efficiently.

**Tos (Type of Service):** This attribute seems to have a short scope as the majority of values are present in a small range. It may require some more processing to see if it also was a discriminative feature in model as attack evidence.

**Processing required:**

**Feature engineering:** Use between and add newly generated features from the exsisting ones, where for example, packet-byte ratio, or duration-packet ratio between, etc to help model better capture the real pattern.

**Outliers:** In case of outliers (e.g., large packet sizes or byte counts), it may need to treat them by filtering, transformation, or by using robust machine learning algorithms that can handle violations to the underlying assumptions of the model.results.

### *3.3.4 Data Preprocessing*

The datasets of this work are preprocessed to be ready for the model input. It is easy to presume the dataset as one having (purely user’s) network traffic (say) and that the features comprise of IP addresses, protocol flags, time length, packet counts along with bytes of transaction. The preprocessing of the dataset is as follows:

#### *3.3.4.a Flag One-Hot Encoding:*

The ‘Flags’ column is mode with one-hot encoding which are protocol flags (e.g., ACK, PSH, SYN). A custom function (one_hot_flags()) is used to one hot encode

individual protocol flags for each row in the ‘Flags’ feature. The flagsare then split into individualcolumns for eachflag(ACK, PSH, RST, SYN, FIN).

For a given flag $f_i$ (where $i$ can represent the flag type like ACK, PSH, SYN, etc.), the one-hot encoding transforms it into a binary vector:

$$f_i = \begin{cases} 1, & \text{if flag is present} \\ 0, & \text{if flag is absent} \end{cases} \quad 3.1$$

### *3.3.4.b IP Address Processing:*

The source and destination IP address values (Src IP Addr, Dst IP Addr) are subject to binary encoding. We split IP addresses into octets and each octet of address is converted to binary and then flattened to 16 binary features for each IP address. This allows our GNN to leverage binary encoding for graph learning.

Each IP address is split into four octets, each represented as a binary number. For an IP address $IP = a.b.c.d$, where $a, b, c, d$ are the four octets:

$$IP = [\text{bin}(a), \text{bin}(b), \text{bin}(c), \text{bin}(d)]$$

This representation expands into 16 binary features ( 4 octets × 4 bits).

### *3.3.4.c Treating Missing and Inconsistensc Values:*

The ' Bytes ' field for packet size may not be a numeric value. These are processed by numericalising them through a cleaning step. Any which are not a number will be assigned a special value (like a large constant) before converting the column into a number. Assume $x$ is the non-numeric value in the 'Bytes' column. Replace $x$ with a constant value $C$

### *3.3.4.d Scaling of Continuous Features:*

For these columns 'Duration', 'Packets', 'Bytes', normalized by PowerTransformer. This helps to normalize these features to have a mean of 0 and a standard deviation of

1. This rescaling is helpful for many machine learning algorithms to have better convergence during training.

For continuous features $X - [x_1, x_2, \dots, x_n]$, the PowerTransformer normalization ensures that the features have a mean of 0 and a standard deviation of 1 :

$$X_{\text{mormalixed}} = \frac{X-\mu}{\sigma} \quad 3.2$$

Where:

- $\mu$ is the mean of the feature
- $\sigma$ is the standard deviation of the feature

### *3.3.5. Graph Construction*

The network traffic is structured in graph, in which hosts (source and destination IPs) and flows (network communication sessions) are nodes of the graph and edges connecting hosts represent communication between them.

#### *3.3.5.a Subgraph Creation:*

A subgraph consists of nodes representing hosts and flows, and edges representing the connection between these nodes. For a given subgraph, nodes $v_1, v_2, \dots, v_n$ represent hosts and flows. The adjacency matrix $A$ for the subgraph, with $A_{ij} - 1$ if there is an edge between nodes $v_i$ and $v_j$, is:

$$A = \begin{pmatrix} 0 & 1 & 0 & \dots \\ 1 & 0 & 1 & \dots \\ 0 & 1 & 0 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \quad 3.3$$

The dataset is then fragmented into subgraphs to make model training more efficient. Each subgraph has a fixed number of rows (e.g., 1024) that linearises into a structure representing a graph of network traffic. Each subgraph consists of:

**Host Node :** It consists the source and the destination IP along with their respective binary features.

**Flow Nodes:** Those described by the features involved in the network flow (Duration, Packets, Bytes, protocol flags, etc.).

### *3.3.5.b Connection Mapping:*

Host nodes (source and destination IPs) are connected and edges are added where connections occur. get_connections() function creates two set of edges as host_to_flow and flow_to_host. These edges connect host nodes to flow nodes and vice versa, and constitute a bipartite graph.

Edges are defined by the connection mapping between host nodes and flow nodes. Let host $_i$ and flow $_j$ be the nodes, and let the connection matrix $M$ represent the edges:

$$M = [\text{host}_i \leftrightarrow \text{flow}_j] \quad 3.4$$

### *3.3.6 Data Splitting:*

The dataset is split into training, validation, and test sets using stratified sampling to ensure class distribution is preserved:

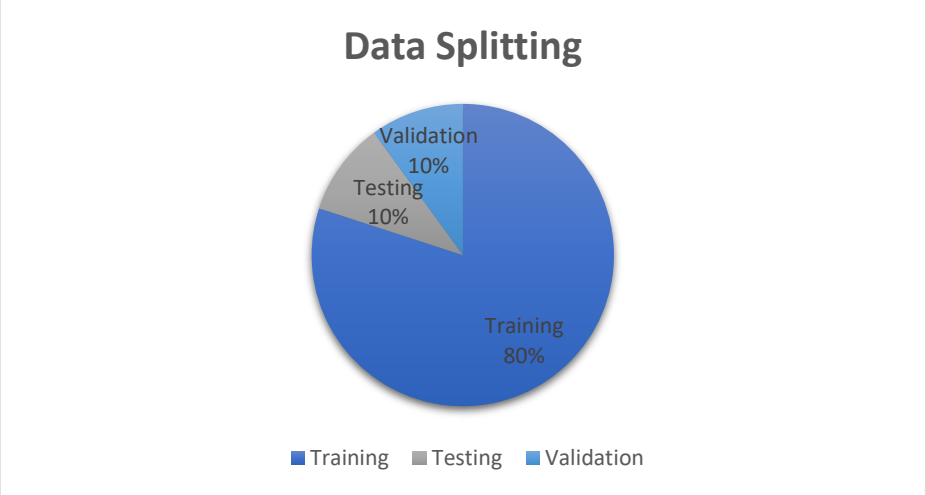


Figure 3.3. Dataset Splitting

Stratification is performed to guarantee that the ratio is preserved between the varieties of attacks classes (e.g., benign, portScan, dos, pingScan) in comparison with every set.

### *3.3.7 Class Description:*

4 classes are found in the dataset. These are the attack type the network system has faced. They are:

**i. Benign:**

In cyberspace, benign means anything; behavior, activity, or action that has no impact on the damage of a system or network. If something is benign it does not describe a violation. Innocuous activities could be ordinary processes such as users logging in, automatic software upgrade, or benign network traffic. As an example, benign traffic are user's web browsing or software when operates as its construction. Because benign activities are acceptable, it is of interest to differentiate these from malicious behaviors when considering cybersecurity systems, in order to prevent false positives and trigger the alarms only when real threats are present.

**ii. Port Scan:**

A port scan is another method performed to identify which ports are open on a computer or a network. The open ports are the ports that can use to communicate with the system and it serves as a potential attack method. Port Scanning Techniques The Analysis Report will likely mention these various types of port scanning techniques: Different kinds of techniques are possible to perform a port scan some of them are listed below:

- TCP Connect Scan: Completes a three-part handshake to the target port. If it get a connection, the port is open.
- SYN Scan: Send SYN packets (one of the packages of the TCP handshake) and check for the responses. It's faster because it isn't doing all the handshaking.
- UDP Scan: A variation on the TCP port scan, this one is used for UDP (User Datagram Protocol) networks which are connectionless and more difficult to detect.

Port scanning is not necessarily a malicious activity, but in cyber-attacks it is commonly used to find open ports that are open to exploit for unauthorized access. Port scans are frequently identified and reported as possible attacks.

**iii. Denial of Service (DoS):**

A Denial of Service (DoS) attack is an attempt to make a computer or network resource unavailable to its intended users, typically by temporarily or indefinitely disrupting a host, which could be a computer, a server or even an entire network. DoS attacks can occur in multiple forms:

- Flooding attacks: Such attacks flood large amounts of data or requests to the target system and it is incapable to process that volume of traffic. This might take the form of an HTTP flood or UDP flood.
- Resource Depletion: Attacker could flood the system or application with the vulnerabilities which are used to consume all system resources like CPU and memory hence slow down or crash the system.

While simple DoS attacks come from one source, Distributed Denial of Service (DDoS) come from from multiple sources, which complicates countermeasures. A DoS attack can range from a website or server to network infrastructure. DoS attacks can be mitigated by applying defensive mechanisms, such as traffic filtering, rate limiting or dedicated DDoS protection service.

**iv. Ping Scan:**

A ping sweep is a simple method for determining which IP addresses on a network are live. It's what's called ICMP (Internet Control Message Protocol) scanning. The simplest form of a ping scan is pinging each IP in a particular range to see if it is up. An ICMP Echo Reply from a device shows that the device is reachable and up and running. Key points about a ping scan:

- Network Discovery: Mostly ping scans are used to find active hosts in a network. It’s quick and easy way to check, which devices are being online & active.
- ICMP-based 'ping scan': May be blocked/filtered by firewalls unknowing about the -PN option, so only to be used by privileged users.

Although a ping scan is commonly employed for legitimate reasons, such as network administration or troubleshooting, attackers leverage it for the purpose of reconnaissance, to identify alive targets, so as to subsequently attack (i.e., port scan or DOS).

### *3.3.8 Model Selection:*

In order to accurately detect and categorize many different anomalies in network traffic by means of a graph-based paradigm, two state-of-the-art Heterogeneous Graph Neural Network (HeteroGNN) models are considered:

- HeteroConv using SAGEConv layers
- HeteroConv with TransformerConv layer

These models are selected specifically for their capability to deal with heterogeneous graphs with myriad node and edge types, and for their efficiency in inductive learning on real-world large-scale graph data. Their architecture can facilitate fine-grained anomaly detection by supporting dynamic information flow between entities that are structurally and semantically diverse such as hosts and flows.

### *3.3.9 Model Description:*

1. **HeteroConv + SAGEConv:**

   HeteroConv: A module to aggregate different GNN layers for each edge type in a heterogeneous graph (built with\\ PyTorch Geometric).

   SAGEConv (GraphSAGE): Does neighborhood aggregation with sampling, supporting inductive learning and scale up.

   - 4-layer model
   - Each HeteroConv layer also includes one SAGEConv layer for each different relation type.
   - Mean strategy for aggregation over neighbors

   **Use Case Strength:**

   - Works well on structured graph data
   - Efficient and scalable
   - Good performance on moderately heterogeneous graphs

2. **HeteroConv + TransformerConv:**

TransformerConv: Utilizes the attention mechanisms of Transformer model for graphs.

HeteroConv is used again to handle various types of edges.

- 4-layer model
- TransformerConv layers use attention to model global and local interdependencies.
- Positional encoding and edge features enhance the context interpretation process

**Use Case Strength:**

- Applicable in the graph with high heterogeneity and complex semantic
- Captures subtle dependencies that vanilla convolutions can miss

## 3.4 Model Comparison:

These models are not only part of graph learning, they represent two state-of-the-art directions in it:

- SAGEConv: efficient, scalable inductive learning
- TransformerGNN: expressive, global attention modeling.

Table 3.2: Key Model Comparison

| Criteria | SAGEConv Model | TransformerConv Model |
|---|---|---|
| Aggregation Mechanism | Mean of sampled neighbors | Attention over neighbors |
| Interpretability | Moderate | High (via attention weights) |
| Message Passing Strategy | Aggregation from neighbors + learned transformation (SAGEConv) | Attention mechanism over neighbors (TransformerConv) |
| Neighbor Sampling | Fixed-size neighborhood sampling | Attention-based full neighborhood (or sampled, if large) |
| Readout / Pooling | Mean aggregation | Attention-weighted aggregation |
| Training Time | Faster (fewer parameters, lightweight) | Slower (due to attention and more parameters) |
| Model Size | Smaller | Larger |
| Interpretability | Higher (aggregation-based) | Lower (black-box attention scores) |

## 3.5 Model Architecture:

Heterogeneous Graph Input:

- Node Types: host, flow
- Edge Types: host → flow, flow → host
- Node Features: Extracted from network metadata (e.g., IP, protocol, byte count)

Each graph $G = (V, E, T_V, T_E)$ is defined as:

- $V$ : set of nodes

- $E$ : set of edges
- $T_V: V \rightarrow$ { host, flow } : node type mapping
- $T_E: E \rightarrow$ {( host,$R$, flow ), ( flow,$R$, host )} : edge type mapping

Node feature matrix:

$$X_v \in \mathbb{R}^{N_v \times F_v} \quad 3.5$$

This describes the feature matrix of nodes of type $v$ in a heterogeneous graph.

- $X_v$ : Feature matrix for all nodes of type $v$
- $N_v$ : Total number of nodes of that type (e.g., number of host nodes)
- $F_v$ : Number of features for each node (feature dimension)
- $\mathbb{R}^{N_v \times F_v}$ : Real-valued matrix of shape $N_v$ rows and $F_v$ columns

- **Model 1: HeteroConv + SAGEConv:**

  Each HeteroConw layer aggregates over different relation types $r$ using a distinct sage conv .
  SAGEConv Mathematical Formula
  Far a node $i$, let $\mathcal{N}(i)$ be the set of its neighbors under relation $r$.

  1. Neighborhood Aggregation:

  $$h_{\mathcal{N}_r(i)}^{\cdot} = \frac{1}{|\mathcal{N}_r(i)|} \sum_{j \in \mathcal{N}_r(i)} h_j \quad 3.6$$

  2. Update Rule:

  $$h_i' = \sigma\left(W^{(r)} - \left[h_i \| h_{\mathcal{N}_r(i)}\right]\right) \quad 3.7$$

  Where:

  - $h_i$ : input feature of node $i$
  - $W^{(r)}$ : learnable weight for relation $r$
  - $\|$ : concatenation
  - $\sigma$ : non-linear activation (e.g., ReLU)
  - HeteroConv Aggregation:

Let $\{R\}$ be the set of all relations.

$$h_i' = \text{AGG}(\{\text{SAGEConv}_r(h_i, \mathcal{N}_r(i)) \mid r \in R\}) \quad 3.8$$

AGG: aggregation across relations (e.g., sum, mean, or attention-based).

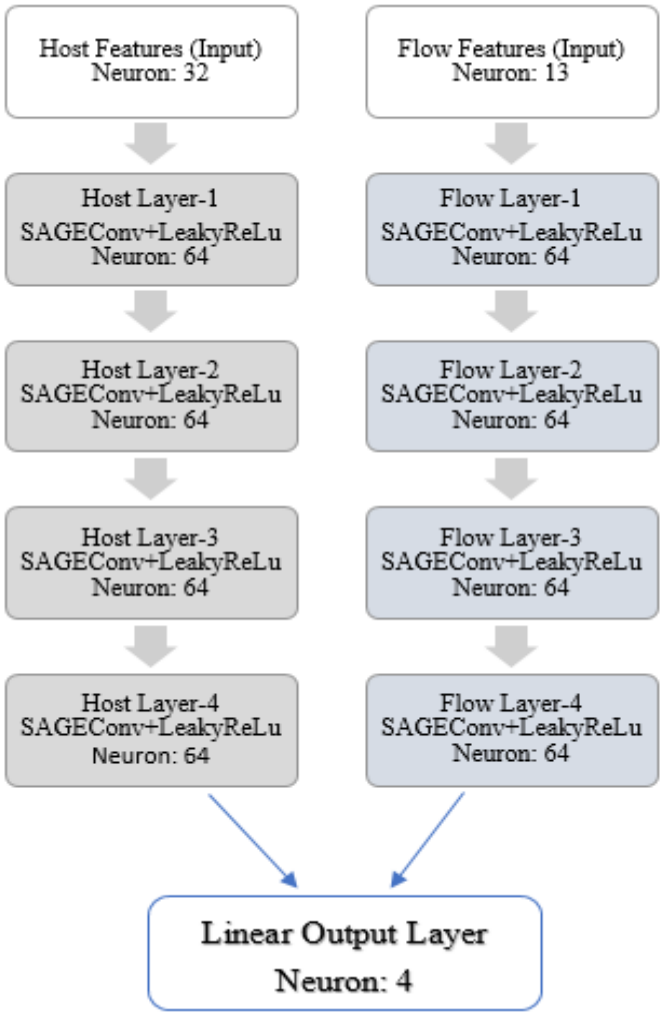


Figure 3.4: Model 1 Architecture

- **Model 2: HeteroConv + TransformerConv:**

  Instead of mean aggregation, TransformerConv uses attention over the neighbors.

  TransformerConv Mathematical Formula

  Let $h_i \in \mathbb{R}^F$ be the feature of node $i$, and $\mathcal{N}(i)$ its neighbors.

1. Query, Key, Value projection:

$$q_i = W^Q h_i, k_j = W^K h_j, v_j = W^V h_j \quad 3.9$$

$W^Q, W^K, W^V \in \mathbb{R}^F$ are learnable weight matrices.

2. Attention Score:

$$\alpha_{ij} = \frac{\exp\left((q_i^\top k_j)/\sqrt{d}\right)}{\sum_{k \in \mathcal{N}(i)} \exp\left((q_i^\top k_k)/\sqrt{d}\right)} \quad 3.10$$

Where:

- αij: attention coefficient showing how much node i attends to node j
- d: dimension of key vectors (used for normalization)

3. Aggregation:

$$h_i' = \sum_{j \in \mathcal{N}(i)} \alpha_{ij} v_j \quad 3.11$$

Dimension of query/key vectors, softmax is applied to ensure normalized attention weights. This allows the model to learn which neighbors are more important in the current context.

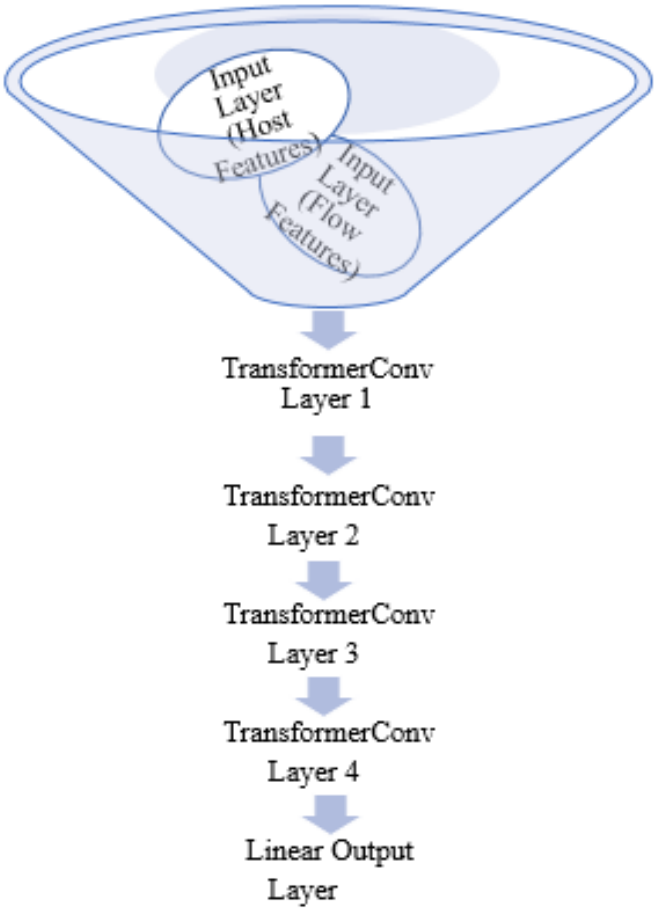


Figure 3.5: Model 2 Architecture

- **Output Layer (Both Models):**
  Fully Connected Classifier

$$y_i = \text{softmax}\left(W_o h_i^{'} + b\right) \quad 3.12$$

Where:

- $h_i^{'}$ : coutput feature after last GNN layer
- Wa: classification weight matrix
- $y$ :predicted class probabilities for node $i$

- **Activation function:**

  i. LeakyReLU:

  Mathematically:

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha x & \text{if } x < 0 \end{cases} \quad 3.13$$

  where $\alpha$ (negative slope) is a small constant like 0.01 .

  ii. Softmax:

  Ensures output values are in range $[0,1]$ and sum to 1

  Mathematically:

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \quad 3.14$$

  where $C$ is the number of classes.

  Table 3.3: Layer wise activation function

| Layer Type | Activation Used | Purpose |
|---|---|---|
| TransformerConv | LeakyReLU | Non-linearity & gradient flow |
| SAGEConv | LeakyReLU | Non-linearity & avoids dead neurons |
| Final Classification Layer | Softmax | Probabilistic multi-class output |

- **Loss Function:**

  Far multi-class classification:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{e=1}^{C} y_{i,c}^{\text{lnwi}} \log\left(y_{i,c}^{\text{Fecl}}\right) \quad 3.15$$

  - Crass-Entropy Lass
  - $N$ : number of labeled nodes
  - C: number of classes

# Chapter 4

## Results and Discussion

In this chapter, the study results are presented and discussed in detail. Table, figure, and graph presenting the key findings are followed by a comprehensive interpretation. Compare and contrast the results with those in prior studies in the literature to emphasize overlapping or divergent insights. This chapter presents the interpretation of the findings with respect to the research objectives, implications based on the findings and evidence in the data to support or refute the hypotheses. Such comparison with existing studies do provide for a stronger grasp of the subject matter, and highlights the value added of the current study.

### 4.1 Overview:

This chapter provides a summary of the findings and how they address the research aims. It starts with a summary of findings in tables, figures and graphs and then interprets the data carefully. The results are compared to the relevant literature and both similarities and differences are emphasized in order to interpret the results. This chapter attempts to illustrate how findings can be of relevance to the field more generally and how they may be interpreted or applied in practical terms or used as a base for theoretical discussion.

### 4.2 Performance Metrics:

The models have been analysed with the the following metrics:

- **Classification Report:**

  Classification Report is used to get the results of the performance for a classification model. It can also offer a great deal of other metrics to see how well is the model doing in each of the available classes.

  The statement usually is: Precision, Recall, F1-Score, Support etc.

  **Support**: The number of true instances for each class in the dataset.

  **Accuracy**: The overall percentage of correct predictions for the entire dataset.

**Macro Average**: The unweighted mean of precision, recall, and F1 score across all classes.

**Weighted Average**: The mean of precision, recall, and F1 score across all classes, weighted by the number of true instances in each class. This helps to account for the imbalance in class distribution.

- **Validation Loss Curve:**

A validation loss curve is a plot of how the loss (or error) on the validation set changes as the model is trained over many epochs (iterations). It is a measure to determine how well a model is able to generalize on new, untrained data it also shows whether a model is overfitting, underfitting or working fine.

**Key Concepts:**

*Loss Function:* Measures the error difference between the model's predictions and the target. Typical loss functions are Mean Squared Error (MSE) for regression and Cross-Entropy for classification.

*Epochs:* Number of times to train on full training set.

*Validation Set:* Part of the dataset not used during training, but to measure how well the model is doing during training.

**Learning from the Validation Loss Curve:**

*Validation Loss Lower:* Model is getting better at generalizing. The model is improving its fit to the validation data.

*Inflection Point of Validation Loss:* When the curve flattens, then the model could be in its best situation, so extra epochs of training might not make the model any better.

*Growing Validation Loss:* The training data while the model are doing good is not generalizing on the validation data. This could be a sign that the model is too rich or not trained enough for too many epochs.

*Underfitting (early)*: In case of high training and validation loss that aren't improving with further training, the model may be too simple or the training process may not be set right.

*Overfitting:* When training loss decreasing while validation loss is increasing, then the model is memorizing the training data, not learning to generalize.

- **Confusion Matrix:**

The confusion matrix is used to understand how good the model is at each of the different attack classes.The confusion matrix $C$ for a multi-class classification problem has elements $C_{ij}$ representing the number of times class $i$ is predicted as class $j$. For a $k$-class problem:

$$C = \begin{pmatrix} C_{11} & C_{12} & \dots & C_{1k} \\ C_{21} & C_{22} & \dots & C_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ C_{k1} & C_{k2} & \dots & C_{kk} \end{pmatrix} \quad 4.1$$

Confusion metrics, commonly used in machine learning and classification models, is a group of metrics, abstraction of the confusion matrix (an overlay of values representing the performance of a classification model). The confusion matrix contrasts predicted vs actual categories (ground truth), categorizing them as true positives, false positives, true negatives, and false negatives.

Table 4.1: Confusion Matrix for multi class classification

| | | Predicted Values | | | |
|---|---|---|---|---|---|
| | | a | b | c | d |
| Actual Values | a | TN | FP | TN | TN |
| | b | FN | TP | FN | FN |
| | c | TN | FP | TN | TN |
| | d | TN | FP | TN | TN |

Where a,b,c,d represents the classes. For this thesis (**benign, portScan, dos, pingScan**) 4 classes are taken.

Elements of the Confusion Matrix:

**i. True Positive (TP):**

*Definition:* True Positive is the number of correctly identified positive observations.

*Example:* Suppose a medical test that tests for the existence of a disease. A person knows he or she has the disease and the test shows that the person has the disease is a True Positive. For instance, if a test predicts “disease present” and the patient does actually have the disease, this will be a True Positive.

**ii. False Positive (FP):**

*Definition:* A False Positive is when the negative prediction was a model mistake.

*Example:* In the same medical test, if a non-infected person tests positive for the disease (i.e., the disease is predicted to be "present," but the person is actually OK), then it is a False Positive. Here, the model has mislabeled a healthy individual as sick.

**iii. True Negative (TN):**

*Definition:* True Negative is the number of negative instances that are correctly predicted as negative by the model.

*Example:* Carrying further with the medical test example, if the test correctly identifies a healthy person as healthy (i.e., the test returns “disease absent” and the person is actually disease free), it is a True Negative. The model is having it correct that it is not a disease.

**iv. False Negative (FN):**

*Definition:* A False Negative occurs when the model predicts a negative instance when it is actually positive.

*Example:* If the person actually is diseased and the test erroneously says "disease absent", then it is a False Negative. This is dangerous because the individual will not be treated even if they are ill.

From these four quantities, many important statistics can be computed:

**Accuracy:** The ratio between the correctly classified examples and the total number of predictions.

Accuracy is the proportion of correctly classified instances:

$$\text{Accuracy } - \frac{\text{Number of Correct Predictions}}{\text{Total Predictions}} \quad 4.2$$

**Precision, Recall, and F1-Score:** These numbers represent the sensitivity (ability of the model to detect the class of attacker) of the model with low false positives ( f1 measures the model's accuracy) and false negatives.

- Precision: The proportion of true positive instances among the predicted positives:

$$\text{Precision } = \frac{TP}{TP+FP} \quad 4.3$$

- Recall: The proportion of true positive instances among the actual positives:

$$\text{Recall } = \frac{TP}{TP+FN} \quad 4.4$$

- F1-Score: The harmonic mean of precision and recall:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad 4.5$$

Where:

- $TP =$ True Positive
- $FP =$ False Positive
- $FN =$ False Negative

## 4.3 Model Performance:

The performance of models employed in the thesis will be presented in this section. The proposed approach aims at the development of more precise models by the use of advanced ensemble methods in order to detect (and avoid) anomalies at the early stage.

- **Model 1: HeteroConv + SAGEConv:**

### i. Classification Report:

Table 4.2 Classification report of SAGEConv model

| | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| benign | 0.9271 | 0.9235 | 0.9253 | 16845 |
| portScan | 0.9752 | 0.9661 | 0.9706 | 12807 |
| dos | 1.0000 | 1.0000 | 1.0000 | 10918 |
| pingScan | 0.9285 | 0.9412 | 0.9348 | 13430 |
| accuracy | | | 0.9535 | 54000 |
| macro avg | 0.9577 | 0.9577 | 0.9577 | 54000 |
| weighted avg | 0.9536 | 0.9535 | 0.9535 | 54000 |

### ii. Validation loss curve:

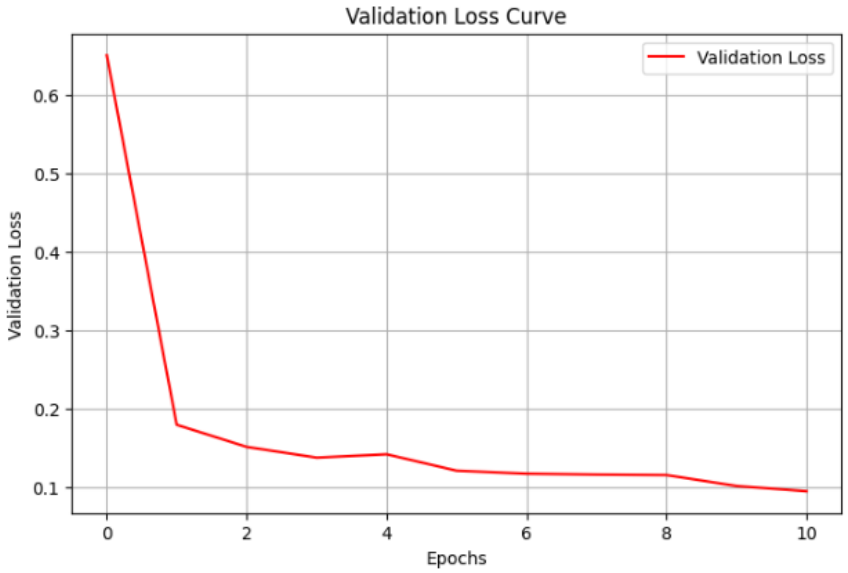


Figure 4.1: Model 1 Validation Loss Curve

### iii. Confusion Matrix:

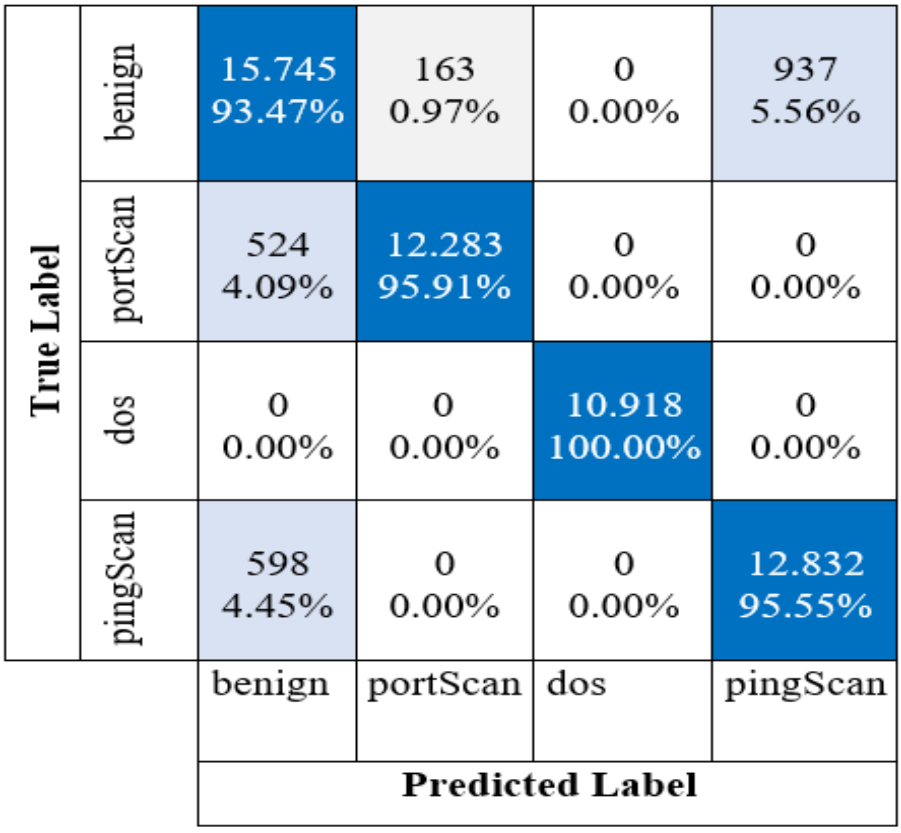


Figure 4.2: Model 1 Confusion Matrix

- **Model 2: HeteroConv + TransformerConv:**

### i. Classification Report:

Table 4.3 Classification report of TransformerConv model

| | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| benign | 0.9247 | 0.9121 | 0.9184 | 16845 |
| portScan | 0.9580 | 0.9736 | 0.9658 | 12807 |
| dos | 1.0000 | 1.0000 | 1.0000 | 10918 |
| pingScan | 0.9306 | 0.9320 | 0.9313 | 13430 |
| accuracy | | | 0.9494 | 54000 |
| macro avg | 0.9533 | 0.9544 | 0.9539 | 54000 |
| weighted avg | 0.9493 | 0.9494 | 0.9493 | 54000 |

### ii. Validation loss curve:

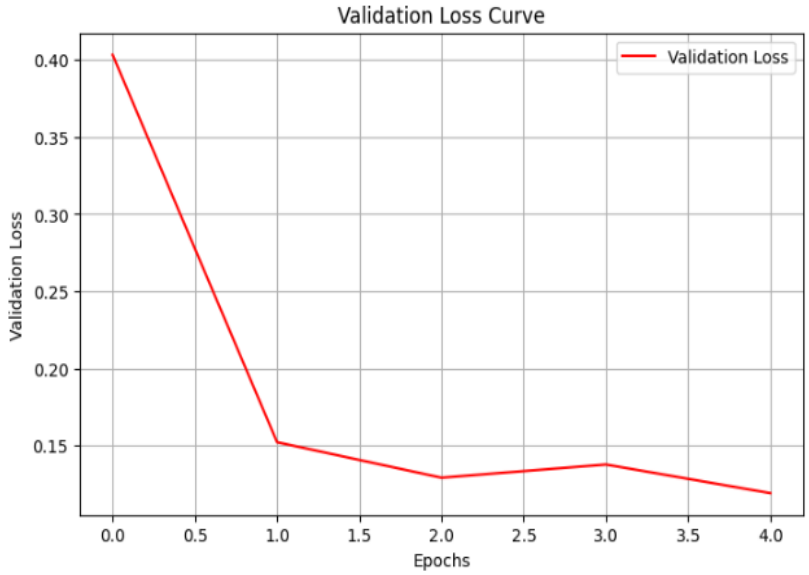


Figure 4.3: Model 2 Validation Loss Curve

### iii. Confusion Matrix:

| True Label | | Predicted Label: benign | Predicted Label: portScan | Predicted Label: dos | Predicted Label: pingScan |
|---|---|---|---|---|---|
| | benign | 15.365<br>91.21% | 546<br>3.24% | 0<br>0.00% | 934<br>5.54% |
| | portScan | 338<br>2.64% | 12.469<br>97.36% | 0<br>0.00% | 0<br>0.00% |
| | dos | 0<br>0.00% | 0<br>0.00% | 10.918<br>100.00% | 0<br>0.00% |
| | pingScan | 913<br>6.80% | 0<br>0.00% | 0<br>0.00% | 12.517<br>93.20% |

Figure 4.4: Model 2 Confusion Matrix

In this case, each models are capable of pinpointing anomalous node behavior via feature aggregation of neighbored nodes, spotting malicious interactions using directed edge semantics, and capturing structured attack patterns by multi-hop message passing. It is this capability to represent complex structure and interactions that renders GNNs so effective for attack detection, potentially revealing information that might slip through rule-based defenses.

## 4.4 Model Comparison:

In this section, the performance of the models used in the research will be compared.

1. **Classification Report Comparison**

Table 4.4 Classification report Comparison

| Metric | SAGEConv Model | TransformerGNN Model | Observation |
|---|---|---|---|
| Accuracy | 0.9535 | 0.9494 | Slightly better accuracy for SAGEConv |
| Macro F1 | 0.9577 | 0.9539 | SAGEConv better in macro average |
| Weighted F1 | 0.9535 | 0.9493 | SAGEConv better for class imbalance |

Class-wise F1-score:

Table 4.5 Class wise Comparison

| Class | SAGEConv F1 | TransformerGNN F1 | Winner |
|---|---|---|---|
| Benign | 0.9253 | 0.9184 | SAGEConv |
| PortScan | 0.9706 | 0.9658 | SAGEConv |
| DoS | 1.0000 | 1.0000 | Tie |
| PingScan | 0.9348 | 0.9313 | SAGEConv |

2. **Confusion Matrix Comparison:**

**Correct Predictions** (Diagonal elements):

Table 4.6 Confusion Matrix Comparison

| Class | SAGEConv | TransformerGNN | Observation |
|---|---|---|---|
| Benign | 15,745 | 15,365 | SAGEConv better |
| PortScan | 12,283 | 12,469 | TransformerGNN better |
| DoS | 10,918 | 10,918 | Same |
| PingScan | 12,832 | 12,517 | SAGEConv better |

3. **Accuracy Comparison:**

The following is the comparison of F1 score, class-wise for each of the two models, SAGEConv and TransformerGNN. The F1 score per class, SAGEConv does slightly better for most of the class except for "DoS" in which they both gain perfect score.

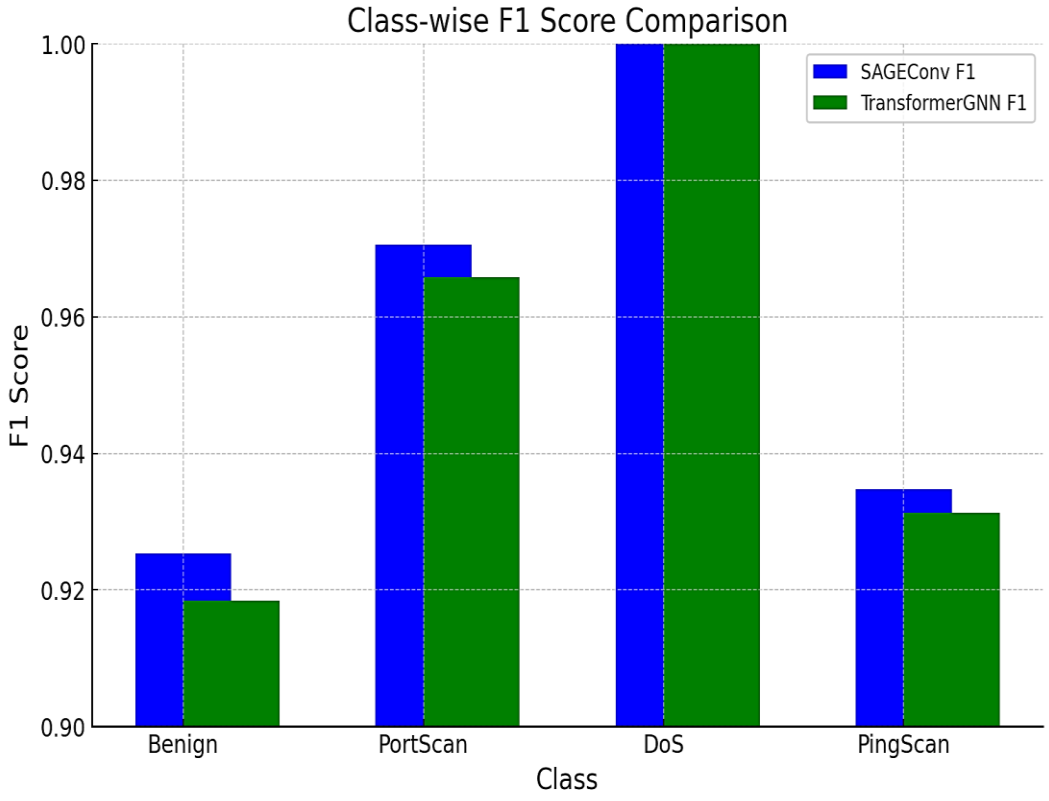


Figure 4.5: Class-wise Model Accuracy Comparison

# Chapter 5

## Social and Environmental Influence

Social and Environmental Impact is concerned with how the technology or system interacts with and impacts on society (human well-being, culture, financial stability and law) and the environment (in terms of resource usage, energy consumption and sustainability). As an example, a cyber security anomaly detection system may have positive societal impacts on digital safety and privacy, and an environmental impact due to the enegy used in processing the data.

## 5.1 Social Impacts:

Social impact is concerned with the ways in which a technology influences a person, a community, or an organization. Some of these include:

### *5.1.1 Financial and Health Influences*

The developed anomaly detection framework has potential both in the terms of enhancing corporate security and in the aspect of responding to outside influence from a societal point of view. This all encompasses priority areas such as health, finance and essential infrastructure. In such areas, the early detection and prevention of malicious activities including Denial of Service (DoS) attacks, port scans and ping scans, for example, may reduce the cost of attacks to the organization concerned. Further, all know that security incidents often are an expensive proposition with the World Bank putting losses at the hands of cybercrime at over the millions a year (directly and indirectly). This number doesn't even include the impact that these attacks have on brand reputation or customer trust, or the legal liabilities that companies can face due to these attacks.

Nowhere is this more the case than in health care, which holds some of their most intimate and sensitive data. Health care providers manage a trove of sensitive data such as medical records, health insurance information and other sensitive information which if breached can have serious consequences. There should always be a heightened sense of cybersecurity in a situation like this. "The relationship of trust between patients and healthcare providers informs how patient data should be treated: securely and confidentially at all times." Patient data must be protected Patient privacy and security

are of utmost importance for good reason – compromised patient data can lead to invasion of privacy, expensive lawsuits and, more importantly, loss of public trust. The willingness to trust that information with providers can have severely negative effects on the entire trust the nation has in its health care system if they believe their own personal health care information is no longer being kept safe.

Thus, an anomaly detection system described in this paper is significant for public health security to guarantee the reliability of healthcare IT systems. By This way the system is preventing and can stop attacks which would otherwise result in the hospital record to be locked, to disable a medical equipment or damaging patient health data/ Report. It's not just a matter of keeping systems online, but a question of maintaining the accuracy and availability of medical information. The quality of patient data is crucial when it is a matter of life and death, like in case of a medical emergency.

The benefit such a system could offer is immeasurable: it not only improves the cyber resilience of healthcare institutions but also indirectly benefits the public at large, by preserving the ability of healthcare systems to function — and to function securely and reliably. Ultimately, the development and use of these technologies will play a key role in ensuring a secure digital infrastructure in sectors where privacy, continuity, reliability and trust are most at stake.

### *5.1.2 Safety, Legal, and Cultural Issues:*

The motivation behind this work is to be able to support safe internet practices by recognizing bad patterns of traffic and to enable compliance with data privacy laws like the General Data Protection Regulation (GDPR). To promote responsible internet usage by allowing organizations to identify malicious traffic patterns and meet important data privacy regulations, including GDPR. GDPR has set a precedent for personal data protection, demanding the highest standard of privacy for individuals from organizations. Designed to satisfy those strict legal requirements and still let organizations have true visibility into their systems and monitor that and protect against threats, even for their hosted data while preserving user privacy with the EDR system. In this way, the system provides for a possibility to continue meeting security requirements without violating data privacy regulations, thereby helping to maintain confidence of consumers in digital infrastructures. Now, with data breaches and misuse of personal information rampant, that balance is more important than ever.

This study is a contribution to the cultural aspect of AI used in documentary films and cybersecurity responsibly. AI changes and is applied in across different areas, among the in network security the ethical use of this technologies has becoming a priority. These works, however, emphasizes the ethical obligation of rolling out AI systems that are not enriched using individual's sensitive information or breach the privacy of users. As AI is applied in a wider range of industries (especially cybersecurity), it is important that the cultural norms that inform privacy and data protection change to reflect the increasing prevalence of these technologies.

The study suggests that if AI does become more integrated in cybersecurity, there should be transparent practices and ethical guidelines ensuring its use in areas such as network security. Here, the ethical use of AI is not only a technical issue, but also a social and cultural issue. It's about earning people's trust so that AI systems are made to serve privacy, fairness and transparency It's about winning peoples' trust, so that AI systems are designed specifically to put privacy, fairness and transparency first. Ultimately, this paper aims to define a framework that not only facilitates addressing security issues, but also preserves the principles of privacy and integrity, so that AI cybersecurity, can work both for the interest of individuals and corporates, while not violating any ethical standard.

### *5.1.3 Ethical considerations:*

Ethical issues in cyber security and anomaly detection systems are crucial to ensure that these technologies are used in a manner which upholds basic human rights, fairness, and hence trust from the public. Privacy and ethical transparency, bias controls and balancing national security requirements with civil rights are all important elements of an ethical cybersecurity approach. By being proactive in addressing these challenges, a cyber security systems which not only protect our digital habitats, but that also serve society more broadly can be developed.

Ethical concerns in cybersecurity and anomaly detection systems are critical to see that these technologies are built and applied in a manner that protects basic human rights, and supports fairness, thereby earning the trust of the public. Cybersecurity is penetrating deeper into every aspect of our online lives, and there needs to be an ethical backbone underlying the development of this technology. Issues include safeguarding privacy, transparency about how these systems work, avoiding bias in data and decisions, and balancing national security with civil liberties.

Respecting privacy is one of the ethical aspects of cybersecurity. The huge amounts of personal data flow, gathered, processed and analysed by these systems must be treated with the maximum respect and do not have allowed to misuse or unauthorized access. Ethical transparency is about being open about the manor in which data is used and giving people who use a system/control over their data. With no such guarantees, people can feel exposed or even cheated, and trust in the technology can collapse.

Control of biases is equally crucial. Anomaly detection algorithms based on AI and machine learning algorithms can reinforce or amplify inherited biases if not properly managed. For instance, if these systems are biasedly trained they might over-identify certain groups or people as risks, which will be a discriminatory outcome. There needs to be mechanisms to protect against these kinds of risks to ensure fairness and prevent us from entrenching existing social inequalities.

Moreover, the interplay between national security and civil rights is very fine. So, while strong cybersecurity is essential to counter online threats and protect key infrastructure, it is also crucial that uphold people's right not to be spied on. For an ethical cybersecurity process such systems should at least function within something like a framework of freedom but also within the real (genuine) security interests of the state.

By meeting these ethical challenges, cyber security systems which not only protect all our digital environments, but also add to society in general can be developed. This is security that protect against the bad while tending the good, shielding our online home, habitat and environment and are also regard for trust and fairness, and basic respect for human dignity. Ultimately it should strive for technologies that allow people to do more, protect the freedoms that people should hold, and generally improve the well being of society and protect our extremely interconnected/networked world.

### 5.2 Environmental Impact:

Since learning complex graphs based models can be time consuming, the final aim of the system is to guarantee that executing them is light weight at run time. After being trained, such models should be optimized so that they respond quickly with a low amount of computing resources (so-called inference). This is useful to be able to can easily spot, react, and get information on anomalies in content, without adding too much overhead on the infrastructure, so it is fit to be applied in near-realtime industrial applications. But it is important to bear in mind that while increased productivity is a welcome change, it does come at an environmental

price; an increase in the amount of computational power available could result in an increase of energy consumed if it is not carefully watched.

The proposed solution also takes a preventive standpoint, by promoting a low consumption of resources (bandwidth and energy), and therefore being an efficient countermeasure against the environment pollution caused by perpetual or large degree cyber-attacks, such as DoS. These assaults usually work by flooding the target with an extremely high volume of traffic which overloads both computational and network resources, resulting in a dissipation of resources and bandwidth that can be put to more constructive uses. Through the discovery and neutralization of these threats, before such threats are fully realized, the system serves to protect the rare resources, so that, energy, intensive infrastructure elements are not unduly stressed.

Also, the system's intelligent recognition capability and adaptive countermeasures to such attacks should be the key factors in reducing overall environmental impact. Early detection of attacks such as DoS also means that the infrastructure does not require huge resources to handle or recover from such events. Such proactive prevention, that affects the overall load on the network infrastructure, implys the decrease of the energy consumption in a mode of continuous operation, operation during and after a massive cyber attack. Therefore, the inkblot detection scheme will benefit not only the safety and reliability of the connected world, but also as a method for addressing the environmental cost of security operations.

Further, by diminishing the frequency and magnitude of security-related actions to combat wide-spread attacks, this system lessens the need for energy-intensive back-up systems and duplicate operations. That means consume less power, charge less money, and promote a greener, more sustainable cybersecurity. When living in a world that’s increasingly interwoven and interconnected, need products that are just a stitch in time for the future of cybersecurity. It serves to help find and use resources when nothing may be absolutely required to get the job done and get up to speed once everything has finally been said and 9 unusually requested inalienable intervention.

### 5.3 Sustainability Issues:

The sustainability of network infrastructure has become more important than ever as enterprises expand their digital presence and depend heavier on cloud services. The proposed anomaly detection system enhances the long-term port reliability and prevention of system downtime. This will help keep digital services functioning without interruption, which are now central to modern life and the global economy.

With businesses and organizations scaling up their digital footprint, adopting new technologies, and becoming more dependent on cloud services, the sustainability of network infrastructure is more important than ever. In an ever-changing digital world, having a strong and reliable network infrastructure is key to keeping vital digital services running round the clock essential to life today and the modern global economy. The introduced anomaly detection method serves to improve the overall long-term port reliabilities of network equipment by identifying the underlie nefarious behavior beforehand and reducing the rate of network downtime. Through their proactive identification and resolution, the system provides for uninterrupted, efficient service delivery that would otherwise disrupted business and end-user (customer) operations.

In addition, the employment of advanced methods, such a propriate binarization and energy-based decoding mechanisms in combination with using on-line model followed traffic logs, results in the application of a highly favorable system. Such approaches enable to apply an intelligent instance based approach to anomaly detection and do not require massive data gathering or redundant data copy. This effective data management does not just boost the system performance but also adhere to the principles of Green AI, a new field aimed at building energy-efficient and sustainable AI models and solutions.

The focus on reduced data redundancy and avoiding unnecessary computations is also crucial in decreasing the environmental footprints of AI and machine learning (ML) computations. This method contributes to sustainability by minimizing the power draw of heavy data gathering and carbon footprint footprint of AI/ML. Moreover, the established trend is likely to encourage the development of the greenest, most cost-effective solutions also for protecting valuable digital assets in a complex cybersecurity world.

As the digital battlefield grows and develops, it is all the more important to take into account the legacy of technological solutions on efficiency and green footprint. The study contributes, therefore, to a comprehensive map for the exploitation of AI and its subsets in designing for sustainable secure network solutions for longterm. Intelligently aiming at minimizing the unnecessary consumption of resources, while still ensuring the system's security and reliability, this novel design proposal contributes for a more sustainable, resilient, and ecological network infrastructure which is able to efficiently scale in the endless digital world.

In the end, social and environmental costs of any technology—especially something as essential to modern life as cybersecurity—must be weighed. By recognizing the societal benefits as well as the environmental costs, to build solutions that not only protect users and their data, but actively benefit broader goals for a more sustainable and environmentally responsible world.

# Chapter 6

## Complex Engineering Problems and Activities

Complex Engineering Problems Complex engineering problems are multidimensional, requiring both the application of multiple technical skills and working in unchartered environments, at scale and in the presence of uncertainty. These are problems in which the process of using the engineering method directly leads to solutions; a town might have frozen water pipes, for example, or no safe drinking water. Activities are projects in which the applied (systematic) engineering method is used to design, develop, test, and refine solutions to open-ended problems that are current or historical. Both are critical in the development of strong, functional systems, particularly in fields like cyber security, where systems will need to manage massive amounts of data, change with the threat environment and survive over time.

## 6.1 Addressing Complex Engineering Problems

Finding anomalies in such large-scale networks is an extremely challenging task, as the amount of network traffic is huge and also very diverse. Network traffic trends and patterns generally have a low signal-to-noise ratio, and rarely do evolving attacks show themselves even to traditional systems. Such attacks can be very subtle, sophisticated, and obscured among normal data traffic, and are very difficult to detect. To mitigate these problems by proposing Heterogeneous Graph Neural Networks (HeteroGNN) that are able to model complicated relations from different entities that include hosts, flows, and their interactions in the network.

They are able to learn and predict rare behaviors which may indicate a breach of a network, by capturing structural as well as semantic complexity. Traditional methods, which usually rely on the simpler, homogeneous presentation of the data, may not catch up with the complexity of the modern network traffic (multi-dimensional based). In contrast, HeteroGNNs can handle dynamic relationships between different types of entities in a network and therefore facilitate more accurate and powerful anomaly detection.

## 6.2 Addressing Complex Engineering Activities

They had to combine ideas coming from different fields, such as machine learning, cryptography, graph theory and programming. The development of the anomaly detection system was a large process of complex engineering:

- **Heterogeneous Graph Construction:** The first key step is to construct graphs to describe multiple types of network entities (e.g., hosts, flows) and their relationships. This includes information collection from numerous network logs, the data of which needs to be processed and formatted with caution.
- **Design of Graph Neural Network Architecture:** The subsequent step was the architecture design for the GNN. This work used state-of-the-art architectures such as GraphSAGE and TransformerConv, allowing the model to capture both local as well as global patterns in the graph. Such architectures are especially suitable to model big and imbalanced datasets, as they capture most relevant information across the feature space, at the same time keeping the computational burden low.
- **Imbalance Data:** Multi-class classification under imbalance During the last step, To performed a multi-class classification task based on the different types of anomaly present in our dataset and analyzed the performance of our model on imbalance data. This entailed devising methods to address class imbalance, such as oversampling methods, modifying the loss function, or incorporating more complex evaluation metrics like precision-recall-F1.
- **Absence of Labeled Anomalies:** Labeling anomalies can be challenging, particularly in big datasets. The lack or scarcity of labeled anomalies in many situations makes supervised learning challenging.
- **False Positives and False Negatives:** It is crucial, though challenging, to balance decreasing false positives—normal data that is mistakenly identified as anomalies—and false negatives—anomalies that are mistakenly categorized as normal data in anomaly detection.
- **Scalability:** In real-time anomaly detection, when quick decision-making and low latency are crucial, neural networks may find it difficult to grow to big datasets.

This work showcases some advanced engineering techniques, which transcend recommendations in several disciplines, thus showing the interdisciplinary aspect of current cyber security solutions.

# Chapter 7

## Conclusion and Future Plan

In the final section, the major findings and practical implications of the research are discussed. This is a good conclusion and demonstrates the importance of the study. It also allows do to draw conclusions and suggest further work.

### 7.1 Conclusion:

This thesis compared two of the latest Heterogeneous Graph Neural Network (HeteroGNN) architectures, GraphSAGE and TransformerGNN, applied on multi-class anomaly detection in network traffic. Both models performed well, and TransformerGNN was slightly better than GraphSAGE in accuracy. Utilization of heterogeneous graph was demonstrated to be very promising in learning heterogeneous nature of network traffic behavior, and even subtle attack patterns that traditional models are hardly to cover, can be detected.

These results highlight the necessity of using sophisticated graph-based methods to model and protect contemporary network ecosystems. It is a significant step toward creating systems that are able to respond to new types of threats as they emerge in the cyber landscape, which are becoming increasingly sophisticated.

### 7.2 Challenges:

The study, however, had its limitations:

Fixed Anomaly Types: Fixed types of anomalies that the system was capable of detecting were benign traffic, ping scanning, port scanning and DoS attacks. The capabilities of the system may also be further extended in future work to securely cope with other types of attacks such as more advanced or unknown attacks.

Static database: Models were trained on a static database and real-time adaptability was not considered. In reality the network traffic is dynamic and the attack methods are always developed. The models presented here could be extended in future work by incorporating real-

time data streams, which could model how the models evolve in order to capture and respond to new and unseen threats.

Graph Construction in Large-Scale Networks: Though the graph construction approach that followed in this study is efficient, real-time construction of graph in large-scale network can create the bottleneck. Scalability of our approach for large-scale scenarios is a direction for further refinements.

## 7.3 Future Plan:

Prospective research directions Many exciting prospects lie ahead in the direction of next generation and future work:

- Integration of Temporal and Dynamic Graph Structures To better deal with the evolving attacks, in the future, it could consider to integrate the temporal and dynamic graph structures. This would enable the model to recognize and react to evolving attacks, also known as advanced persistent threats (APT).
- Unsupervised / Self-supervised Learning: Another direction worth exploring in future would be to extend and apply other unsupervised or self-supervised learning approaches. Such a method is important in practice where new types of attacks appear and therefore it need to be able to detect unseen anomalies.
- Real-time Network Security Pipeline: As an ambitious next step, it is a goal to take the system into a real-time network security pipeline. This would involve leveraging streaming data and incremental learning methods to regularly update the model in order to spot threats as they unfold, ensuring such a system can respond in real-time to threats.

# References:


[1] "Hyperspectral Anomaly Detection Using the Spectral–Spatial Graph | IEEE Journals & Magazine | IEEE Xplore." Accessed: Sep. 5, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/9930793

[2] H. Kim, B. S. Lee, W. Y. Shin, and S. Lim, "Graph Anomaly Detection With Graph Neural Networks: Current Status and Challenges," *IEEE Access*, vol. 10, pp. 111820–111829, 2022, doi: 10.1109/ACCESS.2022.3211306.

[3] J. E. De Albuquerque Filho, L. C. P. Brandao, B. J. T. Fernandes, and A. M. A. Maciel, "A Review of Neural Networks for Anomaly Detection," *IEEE Access*, vol. 10, pp. 112342–112367, 2022, doi: 10.1109/ACCESS.2022.3216007.

[4] L. N. Tidjon, M. Frappier, and A. Mammar, "Intrusion detection systems: A cross-domain overview," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3639–3681, 2019, doi: 10.1109/comst.2019.2922584.

[5] S. B. Park, H. J. Jo, and D. H. Lee, "G-IDCS: Graph-based intrusion detection and classification system for CAN protocol," *IEEE Access*, vol. 11, pp. 39213-39227, 2023, doi: 10.1109/ACCESS.2023.3268519.

[6] W. Villegas-Ch, J. Govea, A. Maldonado Navarro, and P. Palacios Játiva, "Intrusion detection in IoT networks using dynamic graph modeling and graph-based neural networks," *IEEE Access*, vol. 13, pp. 65356-65375, 2025, doi: 10.1109/ACCESS.2025.3559325.

[7] X. Li, H. Wei, and Y. Ding, "PHGNN: Pre-training heterogeneous graph neural networks," *IEEE Access*, vol. 12, pp. 135411-135418, 2024, doi: 10.1109/ACCESS.2024.3409429.

[8] C. Qian, "Heterogeneous graph neural network model based on edge feature generation and meta-graph similarity for feature extraction," *IEEE Access*, vol. 13, pp. 49672-49682, 2025, doi: 10.1109/ACCESS.2025.3552125.

[9] W.L. Hamilton, R. Ying, and J. Leskovec, "*Inductive representation learning on large graphs*," Stanford Univ., Stanford, CA, USA, 2017.

[10] P. Velikovi, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "*Graph attention networks*," Québec Artif. Intell. Inst., Mila, Montreal, QC, Canada, 2017.

[11] D. Cao *et al.*, “Spectral Temporal Graph Neural Network for Multivariate Time-series Forecasting,” *Adv Neural Inf Process Syst*, vol. 2020-December, Mar. 2021, Accessed: Sep. 20, 2024. [Online]. Available: https://arxiv.org/abs/2103.07719v1

[12] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, “Connecting the Dots: Multivariate Time Series Forecasting with Graph Neural Networks,” *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol. 20, pp. 753–763, May 2020, doi: 10.1145/3394486.3403118.

[13] B. Bertalanic, J. Hribar, and C. Fortuna, “Visibility Graph-Based Wireless Anomaly Detection for Digital Twin Edge Networks,” *IEEE Open Journal of the Communications Society*, vol. 5, pp. 3050–3065, 2024, doi: 10.1109/OJCOMS.2024.3393853.

[14] I. Goodfellow *et al.*, “Generative adversarial networks,” *Commun ACM*, vol. 63, no. 11, pp. 139–144, Oct. 2020, doi: 10.1145/3422622.

[15] M. Esmaeili *et al.*, “Generative Adversarial Networks for Anomaly Detection in Biomedical Imaging: A Study on Seven Medical Image Datasets,” *IEEE Access*, vol. 11, pp. 17906–17921, 2023, doi: 10.1109/ACCESS.2023.3244741.

[16] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, “Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10265 LNCS, pp. 146–147, Mar. 2017, doi: 10.1007/978-3-319-59050-9_12.

[17] D. Kim, J. Cha, S. Oh, and J. Jeong, “AnoGAN-Based Anomaly Filtering for Intelligent Edge Device in Smart Factory,” *Proceedings of the 2021 15th International Conference on Ubiquitous Information Management and Communication, IMCOM 2021*, Jan. 2021, doi: 10.1109/IMCOM51814.2021.9377409.

[18] T. N. Kipf and M. Welling, “Variational graph auto-encoders,” 2016, *arXiv:1611.07308*.

[19] Y. Xue, "Research on Time Series Anomaly Detection Based on Graph Neural Network," *2023 IEEE International Conference on Electrical, Automation and Computer Engineering (ICEACE)*, Changchun, China, 2023, pp. 1670-1674, doi: 10.1109/ICEACE60673.2023.10442027.

[20] C. Chen, Q. Li, L. Chen, Y. Liang, and H. Huang, "An improved GraphSAGE to detect power system anomaly based on time-neighbor feature," *Energy Reports*, vol. 9, Suppl. 1, pp. 930-937, 2023, ISSN 2352-4847, doi: 10.1016/j.egyr.2022.11.116.

[21] H. Kim, B. S. Lee, W. -Y. Shin, and S. Lim, "Graph Anomaly Detection With Graph Neural Networks: Current Status and Challenges," *IEEE Access*, vol. 10, pp. 111820-111829, 2022, doi: 10.1109/ACCESS.2022.3211306.

[22] Y. Liu, Z. Li, S. Pan, C. Gong, C. Zhou, and G. Karypis, "Anomaly Detection on Attributed Networks via Contrastive Self-Supervised Learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 6, pp. 2378–2392, Feb. 2021, doi: 10.1109/TNNLS.2021.3068344.

[23] Yiling Zou, Jian Shu "Heterogeneous Network Node Classification Based on Graph Neural Networks "2023 3rd International Conference on Intelligent Communications and Computing (ICC) | 979-8-3503-0832-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICC59986.2023.10421668

[24] Zheng, Y., Jin, M., Liu, Y., Chi, L., Phan, K. T., & Chen, Y.-P. P. (2023). Generative and contrastive self-supervised learning for graph anomaly detection. *IEEE Transactions on Knowledge and Data Engineering, 35*(12), 10067089. https://doi.org/10.1109/TKDE.2023.3271771

[25] M. Ma, L. Han, and C. Zhou, "Research and application of Transformer based anomaly detection model: A literature review," *arXiv*, Feb. 2024. [Online].

[26] Y. A. Farrukh, S. Wali, I. Khan, and N. D. Bastian, "XG-NID: Dual-modality network intrusion detection using a heterogeneous graph neural network and large language model," *Expert Systems with Applications*, vol. 287, p. 128089, 2025.

[27] D. Hoogland, "CIDDS-001: Coburg Intrusion Detection Data Set," *Kaggle*, May 21, 2019.[Online].Available:https://www.kaggle.com/datasets/dhoogla/cidds001/data [Accessed: Dec. 28, 2024].