Enhancing Network Anomaly Detection Using Graph Neural Networks

William Marfo
Department of Computer Science
University of Texas at El Paso
El Paso, USA
wmarfo@miners.utep.edu

Deepak K. Tosh
Department of Computer Science
University of Texas at El Paso
El Paso, USA
dktosh@utep.edu

Shirley V. Moore
Department of Computer Science
University of Texas at El Paso
El Paso, USA
svmoore@utep.edu

2024 22nd Mediterranean Communication and Computer Networking Conference (MedComNet) | 979-8-3503-9047-6/24/$31.00 ©2024 IEEE | DOI: 10.1109/MEDCOMNET62012.2024.10578278

Abstract—**In the world of Internet of Things (IoT) networks, where devices are constantly communicating, keeping them secure from cyber threats is critical. This paper introduces a novel approach to detecting unusual and potentially harmful activities in these networks using graph neural networks (GNNs). We combine two specific types of GNNs—GraphSAGE and graph attention networks (GAT)—to create a model that understands and represents the behaviors and interactions in a network. GraphSAGE creates an embedding of network activities by examining local data interactions, while GAT directs the model's focus to the most critical interactions. By integrating these two methods in a single model that considers different types of interactions (both host and flow nodes), we aim to create a system that accurately represents the current state of a network and can also spot anomalies effectively while reducing false positives and negatives. Our innovative approach has demonstrated promising results, achieving an accuracy of 98% on the UNSW-NB15 dataset, significantly outperforming standalone GraphSAGE and GAT models. This underscores its potential as a robust framework for securing IoT networks against cyber threats and anomalies.**

Index Terms—**Graph neural networks, deep learning, networks, anomaly detection, security attacks**

I. Introduction

IoT has permeated our daily lives, connecting our surrounding elements to make informed decisions [1]. While this interconnectedness has fostered seamless integration and numerous advantages, it inherently unfurls a tapestry of vulnerabilities, exposing systems to various potential cyber-attacks and threats that could jeopardize user confidentiality, data integrity, and network stability [2]. In the IoT realm, networks' expansive and intricate nature magnifies these vulnerabilities and accentuates the need for robust and scalable security solutions. Our exploration of network anomaly detection is rooted in two pivotal observations.

In the sphere of network anomaly detection, we first observe that conventional security protocols, despite their recognized effectiveness, navigate a precarious trajectory amidst scalability and vulnerability management. The scalability challenges emerge in numerous forms; as networks and data traffic grow in size and complexity, these traditional methods might wrestle to proficiently process and analyze the augmented data volumes due to computational or memory constraints [3]. Furthermore, sustaining a comprehensive and enormous database of threat signatures for accurate detection becomes increasingly formidable as the diversity and sophistication of cyber threats soar. Regarding vulnerabilities, even a slight lapse or minor obsolescence in the protocol can pave the way for astute attackers, enabling them to infiltrate the network, elude detection mechanisms, and potentially orchestrate malicious activities such as unauthorized data access, service disruptions, or data manipulations [4].

Secondly, network data inherently present stark imbalances, where the overwhelming majority of network requests are benign, and only a minuscule fraction represents malicious or anomalous activities. This imbalance poses a significant hurdle in discerning sparse anomalous activities amidst a vast sea of benign data without being overshadowed by the majority class [3].

In light of these challenges, the emergence of GNNs marks a significant change, providing new ways to address complex network security problems. GNNs have shown considerable promise in various domains, including social network analysis, recommendation systems, and bioinformatics. This is due to their unique ability to capture dependencies in graph-structured data [5]. However, their application in detecting and localizing network anomalies within IoT systems remains underexplored. GNNs' ability to learn from network data's complex, interconnected nature makes them ideal for identifying subtle patterns indicative of anomalous or malicious activity. This ability, along with advanced feature extraction that inherently considers the relational context of data points, positions GNNs as a powerful tool to overcome the accuracy and scalability issues plaguing traditional security measures [6]. Nevertheless, integrating GNNs into network anomaly detection systems, particularly for real-time analysis and threat localization, is a nascent field that beckons for comprehensive investigation and application.

Given these observations and the challenges identified, we are motivated to investigate the following research questions (RQ):

RQ1: How effective is our model in detecting anomalies within network traffic (Detection performance)?

RQ2: How does the performance of our proposed hybrid GNN model compare against standalone models like GraphSAGE and GAT in the context of network anomaly detection (Comparative analysis)?

Given the challenges in network anomaly detection, we propose a unique blend of GraphSAGE [2] and GAT [7] to formulate a hybrid GNN model. This innovative combination leverages the strengths of both models: GraphSAGE's capacity to generate embeddings through local feature aggregation and GAT's attention mechanism that focuses on critical interactions. Our unique contribution lies in integrating these two methods in a single model, which not only assists in identifying and understanding the myriad of interactions and activities within a network but also enhances the accuracy and reliability of anomaly detection systems. This approach significantly improves upon standalone GraphSAGE and GAT models by capturing a comprehensive representation of network activities, thereby improving our ability to spot anomalies effectively. Leveraging GNNs for network anomaly detection enables the analysis of various request features, determining the nature of a device's request as anomalous or benign [8]. This novel integration and its demonstrated effectiveness in anomaly detection underscore the originality and value of our work.

The contributions of this paper can be summarized as follows:

- We formulate a GNN-based network anomaly detection technique that efficiently manages escalating network data volumes and mitigates vulnerabilities, fortifying against potential attacker exploits and enhancing overall network security.
- We design a framework to address class imbalance and accurately detect anomalous activities in predominantly benign network data.
- We attain a notable 98% accuracy with low false positives and negatives on the UNSW-NB15 dataset, establishing our model as a proficient tool for network anomaly detection.

The rest of this paper is organized as follows: §II discusses the background of GNN in network anomaly detection. §III describes the dataset used in our study. §IV details the models' architecture. §V presents the implementation details of our proposed model. §VI describes the experimental results. §VII covers the discussion and limitations of the study. §VIII discusses related works in the field. Finally, §IX concludes the paper.

II. Background

In the realm of network security, particularly within the sophisticated landscapes of IoT, anomaly detection has emerged as a cornerstone strategy [9]. This approach distinguishes irregular behaviors in networked systems by understanding their normal operational patterns. Anomaly detection is critical in preempting and mitigating various security incidents, including network intrusions, fraudulent activities, and other cyber threats.

Anomalies in this context are typically defined as observations or patterns that significantly deviate from the norm, suggesting the potential interference of an external or malicious mechanism. The challenge lies in accurately characterizing what constitutes 'normal' behavior within a network, which varies considerably across different systems and applications. For instance, unusual traffic patterns in a network might indicate a cyber attack. In contrast, similar patterns could be normal in another context, such as heightened activity during a specific event or time. Anomaly detection methods have evolved from traditional techniques, which predominantly rely on unstructured data analysis, to more sophisticated models that incorporate machine learning (ML). These methods include classification, distance and clustering measures, and statistical analysis [10]. Each approach offers unique insights into identifying anomalies. Nevertheless, they also bring challenges, especially regarding adaptability, scalability, and accuracy in diverse and complex environments like IoT [11].

GNNs have emerged as a powerful tool for network anomaly detection, adept at capturing complex relationships

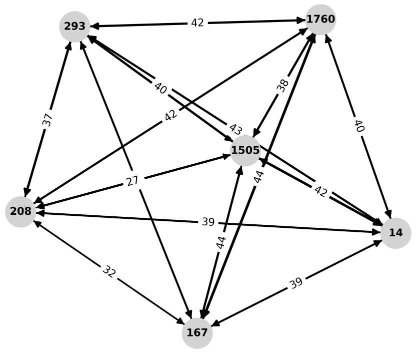


Fig. 1: Visualization of a GNN model applied to network anomaly detection.

in network data [5]. Figure 1 illustrates a GNN model processing network data, emphasizing node interconnections and anomaly detection. The subgraph focuses on the top six highly connected nodes. Inherently graphical, network data comprises nodes (entities such as devices, users, or IP addresses) and edges (interactions or communications between these entities), where edge weights represent the strength or significance of these connections. GNNs leverage this structure, incorporating node and edge information to learn a comprehensive network representation [6]. This allows GNNs to understand the local context around nodes, which is crucial for identifying anomalous patterns [12]. For instance, if a device suddenly starts communicating with a large number of nodes or with a node it has never interacted with before, the embeddings learned by the GNN might change, signaling a potential anomaly.

GNNs like GraphSAGE [2] and GAT [7], which respectively leverage neighborhood sampling and attention mechanisms, provide enhanced capabilities in this domain by not only understanding the underlying network structure but also prioritizing important nodes and interactions, which is particularly crucial in imbalanced scenarios typical of anomaly detection. The robustness and flexibility of GNNs allow them to be employed in various network settings, including cyber-physical system networks, to safeguard them from potential threats by effectively identifying and flagging anomalous activities.

In this work, we explore and build upon foundational GNN concepts, tailoring GNNs to the specific challenges and requirements of network anomaly detection, as detailed in subsequent sections. The attention mechanism of GAT proves to be crucial, facilitating the model in honing its focus on critical interactions within the network, thereby bolstering its capacity to discern anomalous activities amidst predominantly benign interactions. As the IoT networks exhibit intricate interactions and expansive nature, GAT, through its capacity to prioritize and weigh different nodes within a neighborhood, presents a viable solution in creating a model that is not only adept at representing the current state of a network but also proficient in identifying anomalies by focusing on pivotal interactions, thus fortifying the network against potential cyber threats and anomalies.

III. Data background

The dataset utilized for this research is the UNSW-NB15 [13], a prominent network intrusion dataset curated at the UNSW Cybersecurity Lab in Canberra, Australia. Designed to mirror the complexities of modern network traffic scenarios, it encompasses a vast range of low-footprint intrusions, setting it apart from earlier benchmark datasets such as KDD98, KDDCUP99, and NSLKDD. The IXIA PerfectStorm tool [13] produced the raw network packets for UNSW-NB15 in a synthetic environment. The outcome was a balanced hybrid of genuine contemporary standard activities and recent synthetic attack behaviors. A total of 100 GB of raw traffic was captured predominantly in the form of PCAP files by the TCPDUMP tool. The dataset includes nine types of attacks, such as fuzzers, analysis, backdoors, DoS, exploits, generic, reconnaissance, shellcode, and worms.

The Argus and Bro-IDS tools and twelve distinct algorithms were employed to structure and analyze the network traffic [14]. This collaborative effort yielded 49 specific features [13] that highlight the intricacies of network packets. Each selected feature in Table I, whether nominal or integer, demonstrates the variety within the UNSW-NB15 dataset, which encompasses a range of data types. The class distribution for the UNSW-NB15 dataset is detailed in Table II, illustrating the breakdown of categories across training and testing sets.

TABLE I: Sample of the UNSW-NB15 dataset

Feature	Type	Description
service	nominal	http, ftp, smtp, ssh, dns, ftp-data, irc
ct_dst_sport_ltm	integer	No of connections to same dst addr and port
attack_cat	nominal	Name of each attack category
ct_src_dport_ltm	integer	No of connections to same src addr and port
Dpkts	integer	Dst to src packet count

TABLE II: Class distribution of the UNSW-NB15 dataset

Category	Training Set	Testing Set
Normal	56,000	37,000
Generic	40,000	18,871
Exploits	33,393	11,132
Fuzzers	18,184	6,062
DoS	12,264	4,089
Reconnaissance	10,491	3,496
Analysis	2,000	677
Backdoor	1,746	583
Shellcode	1,133	378
Worms	130	44
Total	175,341	82,332

These specific features are instrumental in portraying a detailed network traffic profile for intrusion detection. Nominal features like `service` and `attack_cat` categorize the type of service and the nature of potential attacks, while integer features such as `ct_dst_sport_ltm` and `Dpkts` quantify aspects like connection counts and packet transfers, providing valuable metrics for anomaly detection. Our primary objective is to leverage this tabular dataset and proficiently detect any anomalies in network traffic. The dataset comprises a robust 2,540,043 samples. Each sample includes a binary label, where a label of '1' signifies an attack or anomaly, while a '0' denotes regular traffic.

IV. Model Description and Architecture

This section discusses the algorithms and proposed approaches for our research.

A. Data preprocessing

The dataset, sourced from the UNSW-NB15, is initially composed of 49 columns [13]. To optimize our dataset for GNN processing, we employed the following steps:

1) Data inspection and cleaning

On inspection, it was found that the 47th feature column contained different label types. Given its irregular nature and potential divergence from the problem's scope, it was prudently removed. This decision was based on the understanding that including irrelevant or inconsistent data could potentially skew the results of our model. Additionally, warnings were raised about mixed data types in columns (1, 3, 47). Though addressed by removing column 47, the mixed data types in columns 1 and 3 were implicitly handled in the subsequent steps, ensuring that our dataset remained consistent and suitable for further processing.

2) Feature engineering

The IP addresses and their respective ports for source and destination were combined to provide a comprehensive descriptor for network entities. This combined feature formed unique identifiers, facilitating easier node representation in the later stages. By creating these unique identifiers, we were able to represent the complex relationships between different network entities more accurately, thereby enhancing the predictive power of our model.

3) Categorical data transformation

Certain features in our dataset, specifically the protocol type and connection status, were categorical. These were encoded into numerical values using one-hot encoding, a common technique for transforming categorical data. This conversion is crucial for GNN operations, which require numerical inputs. By transforming these categorical features into a numerical format, we were able to include them in our model without disrupting the mathematical operations performed by the GNN.

4) Feature scaling

For better convergence and to avoid dominance of any particular feature, the data was normalized to zero mean and unit variance, a common scaling method especially critical for neural network models. This process ensures that all features have equal weight in the model, preventing any one feature from disproportionately influencing the model's predictions.

5) IP to unique ID mapping

In the preprocessing stage, each unique IP address in the dataset was assigned a distinct identifier (an index). While IP addresses inherently possess numerical values—with IPv4 addresses as 32-bit integers and IPv6 addresses as 128-bit integers—our transformation aimed to simplify the graph construction for GNNs, which typically utilize sequentially indexed integer-based node representations. This mapping enables the construction of a graph with accurately represented relationships between network entities, thereby enhancing the predictive capability of our GNN model and ensuring computational efficiency.

B. Graph generation

Since our dataset is primarily tabular, transforming it into a graph suitable for a GNN presents an intriguing challenge. To address this, we conceptualized a method to convert traffic flows into meaningful nodes. Our strategy hinges on the creation of a heterogeneous graph containing two distinct node types:

(a) *Hosts:* Representing computers, these nodes primarily feature IP addresses. Should additional data be accessible, we could incorporate attributes like logs or CPU utilization.

(b) *Flows:* These nodes, representing the connections between two *hosts*, encapsulate all other dataset features, including a predictive label. The predictive label is used to distinguish between benign and malicious *flows*.

The *flows* within the dataset are unidirectional, necessitating the definition of two edge types: host-to-flow (source) and flow-to-host (destination). However, encapsulating the entire graph as one unit is infeasible due to memory constraints. Consequently, we segmented it into smaller subgraphs.

C. Proposed model

Heterogeneous GNN (HGNN) is a type of GNN designed to handle heterogeneous graphs. In a traditional GNN, the graph structure is assumed to be homogeneous, meaning that all nodes and edges in the graph are of the same type. However, real-world data often contains diverse entities and relationships, leading to the need for models that can effectively process heterogeneous graphs [15].

In a heterogeneous graph, nodes can represent different types of entities, such as devices, servers, or users, and edges can represent various types of relationships between these entities, such as "communicated with", "sent data to", or "received data from". The challenge in designing HGNNs lies in effectively capturing and aggregating information from different node and edge types while considering their diverse semantics [4].

HGNNs typically utilize specialized techniques for node and edge embeddings to manage heterogeneity. A common strategy involves using distinct embedding vectors for each node type and edge [6]. This strategy allows the model to learn unique representations for different types of entities and relationships, thereby capturing their characteristics.

A key aspect of HGNNs involves the design of message-passing mechanisms. Message passing, a fundamental operation in GNNs, enables nodes to exchange information with their neighbors. In the context of HGNNs, the message-passing process must consider the heterogeneity of node and edge types. This consideration can be addressed through type-specific message functions, where each node or edge type has its parameters for message passing [2].

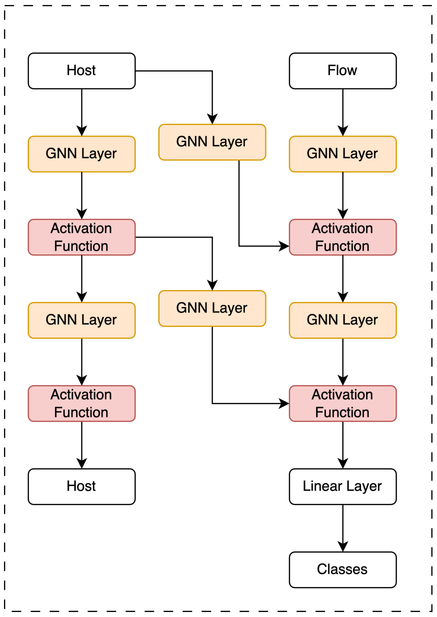


Fig. 2: Model architecture

Figure 2 shows the *host* nodes as well as *flow* nodes, each representing a device in our network. These interconnected nodes represent the complex web of interactions between devices in a real-world network. The data from these interactions, which include features such as IP addresses for *host* nodes and connection details for *flow* nodes, are then fed into our GNN model. This feeding process involves passing these features as input to the GNN layers of our model, where they are processed and transformed to capture the underlying patterns and structures in the network data.

The *flow* nodes represent connections between two *hosts* and encompass all the remaining features from the dataset. Our model and the *host* nodes process them, allowing us to capture device-level and connection-level patterns in the network data. The GNN layers, represented by multiple layers in Figure 2, are where our model processes and transforms the raw network data into a form that captures the underlying patterns and structures. Each GNN layer learns from the previous layer's output, allowing our model to capture increasingly complex patterns as the data flows through the network. Once several GNN layers have transformed the node features, they are processed by an activation function. The introduction of non-linearity into the model through this function enables it to learn more complex patterns.

Finally, the transformed node features are passed through a linear function and classified into different 'Classes.' Mathematically, this can be articulated as $f(\mathbf{x}) = \mathbf{w}\mathbf{x} + b$, where $f(\mathbf{x})$ is the output of the linear function, $\mathbf{x}$ represents the transformed node features, $\mathbf{w}$ is the weight vector, and b is the bias term. These outputs are then utilized to classify the network activity into distinct classes, each representing a different type of network activity and enabling the detection of anomalies within the network data.

V. Implementation Details

A. GAT Implementation

GAT, an enhanced graph convolutional network (GCN), integrates layer stacking, enabling nodes to attend to their neighboring nodes' features. In the GAT framework, the 'neighborhood' of a node comprises all directly connected nodes, which are crucial for the feature aggregation process enabled by the attention mechanism. This mechanism allows for the implicit assignment of different weights to various nodes within a neighborhood, obviating the need for computationally intensive procedures such as matrix inversions or prior knowledge of the graph structure [7]. GAT's architecture comprises graph attentional layers, which employ an attention mechanism, utilizing a feedforward neural network and applying the LeakyReLU (LR) non-linearity. The LeakyReLU function, used in the computation of attention coefficients, is a variant of the ReLU activation function. It allows a small, non-zero gradient when the unit is not active, preventing the issue of 'dying ReLUs' where neurons stop learning completely. This feature makes it particularly effective in maintaining the flow of gradients through the network, even for negative input values.

The attention coefficients in GAT [7], which are utilized to weigh the features of neighboring nodes, are computed as follows:

$$e_{ij} = \text{LeakyReLU}\left(\boldsymbol{a}^{\top}[\boldsymbol{W}\boldsymbol{h}_i \| \boldsymbol{W}\boldsymbol{h}_j]\right) \tag{1}$$

where $\boldsymbol{a}$ represents a shared attention mechanism's parameter, $\boldsymbol{W}$ is the weight matrix, $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ are the feature vectors of nodes i and j, and $\|$ denotes concatenation.

After computing the attention coefficients, they are normalized across all choices of j using the softmax function:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}(i)} \exp(e_{ik})} \tag{2}$$

where $\mathcal{N}(i)$ represents the neighboring nodes of i. The normalized attention coefficients are used to compute a linear combination of the neighboring nodes' features, which are then passed through an activation function to obtain the final embedding for each node.

In our research, we utilize TensorFlow 2.14.0 to implement the GAT. The GAT model comprises an input layer, a custom graph attention layer (GAL), and an output layer. The GAL is configured with 8 attention heads, enhancing the stability and performance of the learning process. The output layer is dense with a softmax activation function, suitable for our anomaly detection task. We train the model for 100 epochs, utilizing a batch size of 256 and an Adam optimizer with a learning rate of 0.001. An early stopping mechanism is incorporated to prevent overfitting. The training process takes approximately 301 seconds.

B. GraphSAGE Implementation

GraphSAGE, a GNN model proposed by [2], adeptly utilizes varied node feature information, encompassing text attributes and node degrees, to construct node embeddings for unseen data. Unlike methods that train individual embeddings for each node, GraphSAGE employs a function that generates embeddings by sampling and aggregating features from a node's local neighborhood. The effectiveness of this approach is anchored in two primary functions: the aggregation and update functions [2] .

$$f_{\text{aggregate}}^{(k)}\left(\left\{\boldsymbol{h}_u^{(k-1)} \mid u \in \mathcal{N}(v)\right\}\right) = \frac{1}{\deg(v)} \sum_{u \in \mathcal{N}(v)} \boldsymbol{h}_u^{(k-1)} \tag{3}$$

$$f_{\text{update}}^{(k)}\left(\boldsymbol{h}_v^{(k-1)}, \boldsymbol{a}_v^{(k)}\right) = \sigma\left(\boldsymbol{W}^{(l)}\left[\boldsymbol{h}_v^{(k-1)}, \boldsymbol{a}_v^{(k)}\right]\right) \tag{4}$$

The aggregation function, defined in Equation 3, aggregates information from a node's neighborhood to formulate a representation of the node that encapsulates local structure and features. Here, $\boldsymbol{h}_u^{(k-1)}$ denotes the feature vector of node u at the $(k-1)$-th iteration and $\mathcal{N}(v)$ represents the neighborhood of node v. This function specifically employs a mean aggregator, computing the mean of the feature vectors of the neighboring nodes.

Conversely, the update function, expressed in Equation 4, updates the representation of a node, typically by utilizing the aggregated information from its neighbors. The update is performed by concatenating the current feature vector of node v ($\boldsymbol{h}_v^{(k-1)}$) and the aggregated vector ($\boldsymbol{a}_v^{(k)}$), then multiplying by a weight matrix $\boldsymbol{W}^{(l)}$ and passing through an activation function σ. This iterative refinement of node embeddings ensures they incorporate pertinent information from their local neighborhoods, enhancing the model's predictive capabilities.

For the GraphSAGE implementation, we also use TensorFlow 2.14.0 and design a similar architecture with a GraphSAGE layer instead of the GAL. This layer implements the GraphSAGE mechanism outlined in [2]. The training configuration is similar to that of GAT but is explicitly tuned for GraphSAGE. We also incorporate an early stopping mechanism based on validation loss to optimize computational efficiency. The Adam optimizer is used with a learning rate ranging from 1×10^{-3} to 1×10^{-5}, providing a flexible and adaptive learning rate for the training process. The training process is slightly faster than the GAT, taking approximately 290 seconds.

C. Integrating GAT and GraphSAGE: A TensorFlow Implementation

Our study integrates GAT and GraphSAGE, capitalizing on their unique capabilities in our anomaly detection model. We use TensorFlow 2.14.0 [16] and the Spektral library [17] for this implementation. Our goal is to effectively use GraphSAGE's capacity to generate embeddings by examining local data interactions and GAT's proficiency in focusing on the most significant node interactions through attention mechanisms.

We develop a custom model that concurrently uses GAT and GraphSAGE layers, effectively merging their representations. Here is a simplified version of our model implementation using TensorFlow:

```
import tensorflow as tf
from spektral.layers import GraphAttention,
    GraphSageConv

class CombinedModel(tf.keras.Model):
   def __init__(self, n_out):
      super(CombinedModel, self).__init__()
      self.gat_conv = GraphAttention(8,
          activation='relu')
      self.sage_conv = GraphSageConv(8,
          activation='relu')
      self.concat = tf.keras.layers.
          Concatenate(axis=1)
      self.fc = tf.keras.layers.Dense(n_out,
          activation='softmax')

   def call(self, inputs):
      x, a = inputs
      x1 = self.gat_conv([x, a])
      x2 = self.sage_conv([x, a])
      x = self.concat([x1, x2])
      return self.fc(x)
```

Our model is a hybrid architecture that combines the strengths of Graph Attention Networks (GAT) and GraphSAGE. GAT and GraphSAGE layers generate two sets of node embeddings, capturing different aspects of the graph structure. These embeddings are then combined and processed through a final dense layer. This layer uses softmax activation to generate our predictions, providing a probability distribution over the classes for each node.

Attention-based sampling: One of the unique features of our model is the use of attention scores from GAT to guide the neighbor sampling process in GraphSAGE. This approach prioritizes the neighbors that GAT deems important, leading to the propagation of more relevant information through the network. This method allows us to focus on the most informative parts of the graph, improving the efficiency and accuracy of our model.

Feature aggregation: We concatenate the embeddings from GAT and GraphSAGE to create a unified node representation. This representation simultaneously considers local and critical interactions, capturing a comprehensive view of the graph structure. This feature aggregation strategy allows us to leverage the strengths of both GAT and GraphSAGE, resulting in more robust and informative node embeddings.

Node embedding generation: We use GraphSAGE to generate node embeddings by leveraging its ability to encapsulate local neighborhood information. This is particularly beneficial for the anomaly detection task, as anomalies often manifest as unusual patterns in the local graph structure. By capturing this local information, our model can effectively identify and classify anomalous nodes.

Joint training: The model's training is carried out over 100 epochs. We employ the Adam optimizer, set the learning rate to 0.001, and use a batch size of 256. The categorical cross-entropy loss function is employed for its suitability for multi-class classification tasks, as it inherently encourages the model to learn discriminative embeddings by penalizing misclassifications.

Throughout this process, iterative updates are made to the weights of both GAT and GraphSAGE layers. This iterative update process facilitates mutual learning between the layers, allowing each one to inform and enhance the learning of the other. As a result, our model develops a robust learning mechanism that is mutually enhanced and capable of providing a comprehensive representation of network activities and potential anomalies. This joint training approach ensures that our model is continually refined and improved, leading to more accurate and reliable predictions.

Parameter justification: In our model configuration, we choose 8 units for both the GraphAttention and GraphSageConv layers based on initial experiments. The LeakyReLU activation function is selected due to its proven effectiveness in mitigating the vanishing gradient problem in deep networks. Additionally, we find that a learning rate of 0.001 and batch size of 256 provide an optimal balance between computational efficiency and model performance.

Challenges and solutions: During implementation, we faced some challenges related to the stability of the training process with occasional erratic loss fluctuations during early epochs. We addressed this by implementing a learning rate schedule that gradually reduced the learning rate as training progressed. This approach helped to stabilize the training process, ensuring more consistent convergence and preventing the model from getting stuck in suboptimal solutions. Another challenge was dealing with the class imbalance in our dataset. We tackled this by using a weighted variant of the categorical cross-entropy loss function, which ensured that the model paid adequate attention to the minority class. This approach helped to ensure that our model is sensitive to all classes in the data, improving its ability to accurately classify instances from the minority class.

VI. Experimental Results

This section provides a detailed evaluation of our GNN model's performance on the UNSW-NB15 dataset. Experiments related to this research were executed using an AMD Ryzen 7 5800H laptop with Radeon Graphics, clocked at 3.20 GHz and supported by 16.0 GB of RAM. All computational operations were conducted using Python within a Jupyter Notebook environment. Following this, we aim to address the RQs outlined in §I.

A. Performance metrics

To assess the efficacy of our models, we rely on various metrics that shed light on different aspects of the model's performance. In supervised learning techniques, particularly in network anomaly detection, the performance of a model is gauged by juxtaposing the model's predictions against the actual ground truth. Below are the key metrics we employ:

Precision: It calculates the fraction of accurate positive predictions. In essence, out of all the data points the model labels as anomalous, precision tells us how many were genuinely anomalous. A high precision indicates a lower rate of false positives, which is especially crucial when working with unbalanced datasets.

Recall: This metric quantifies the fraction of actual anomalies that the model correctly identified as such. In other words, it indicates how many real network anomalies were captured by the model's predictions.

F1 Score: F1-score is a balance of both precision and recall, given as $F1\,score = 2 \times \frac{precision \times recall}{precision + recall}$.

Accuracy: This metric provides a straightforward measure of the model's performance, indicating the proportion of predictions that align with the labeled ground truth data.

AUC and ROC: In addition to the primary performance metrics, we also consider the area under the curve (AUC) score and the receiver operating characteristic (ROC) curve. The ROC curve illustrates the trade-off between sensitivity (recall) and specificity across various threshold values. The AUC score summarizes the ROC curve's performance into a single metric, signifying the model's capacity to distinguish between negative and positive classes.

Matthews Correlation Coefficient (MCC): MCC is a robust metric used in binary classifications. It takes into account true and false positives and negatives, providing a balanced measure even if the classes are of very different sizes. The MCC is essentially a correlation coefficient between the observed and predicted classifications; it returns a value between -1 and +1, where +1 indicates a perfect prediction, 0 indicates no better than a random prediction, and -1 indicates total disagreement between prediction and observation.

Area Under the Precision-Recall Curve (AUC-PR): The AUC-PR is particularly useful in scenarios with a significant imbalance between the positive and negative classes. It focuses on the performance of a classifier on the positive (minority) class. The Precision-Recall curve plots precision against recall, and the AUC-PR summarizes the area under this curve, providing a single measure of performance when dealing with imbalanced datasets. These additional metrics enhance our understanding of the model's capabilities in differentiating between normal and anomalous network activities.

B. Detection performance (RQ1)

Addressing our first research question on detection performance, our GNN model demonstrated excellent performance in detecting anomalies, effectively minimizing false positives and negatives - crucial aspects of network security. The model's key performance metrics, including precision, recall, F1 Score, MCC, AUC-PR, and accuracy, are presented in Table III. These results underscore the model's ability to accurately classify network traffic. A visual representation of the model's classification capabilities, further illustrating its effectiveness, can be seen in Figure 6.

C. Comparative analysis (RQ2)

To address our second research question regarding comparative performance, we compare our integrated GNN model to two types of baselines.

TABLE III: Model performance metrics (Weighted)

Model	Precision	Recall	F1	MCC	AUC-PR	Time (s)
Our GNN	0.99	0.99	0.99	0.91	0.90	352
GAT	0.82	0.82	0.8268	0.79	0.76	301
GraphSAGE	0.81	0.81	0.81	0.77	0.75	290

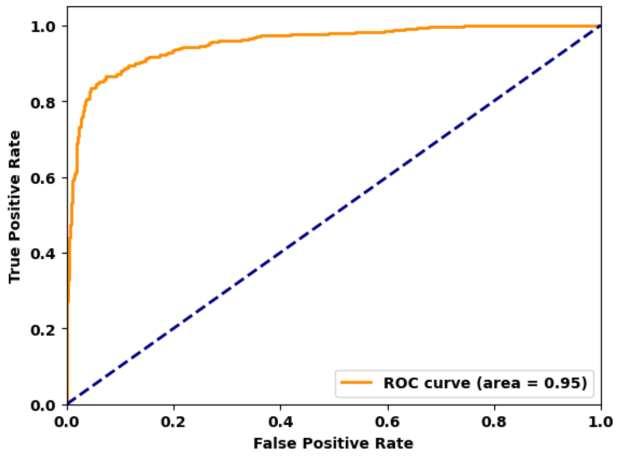


Fig. 3: ROC curve for our hybrid GNN model

The first baseline is the standalone implementation of GraphSAGE, a model renowned for generating embeddings through local feature aggregation, making it a robust tool for network analysis. The second baseline is the standalone GAT model, recognized for its attention mechanism that focuses on the most critical interactions within the network, providing an insightful analysis of complex network structures.

Compared to these baselines, our integrated GNN model significantly improves performance. This enhancement is particularly evident in the model's advanced anomaly detection capabilities, as shown in Table III. By harnessing the strengths of both GraphSAGE and GAT, our model not only learns from the complex and interconnected data within network traffic but also ensures more accurate classifications compared to each standalone model. Moreover, our hybrid model achieved an accuracy of 98%, significantly outperforming GraphSAGE and GAT, which achieved accuracies of 84% and 85%, respectively. This substantial improvement in accuracy is a testament to the effectiveness of integrating the distinct features of GraphSAGE and GAT into a single model. Given the model's complexity and large dataset size, this quick training time and high performance underscores the model's practicality in real-world scenarios where timely anomaly detection is vital. This comparative analysis underscores the superiority of our integrated approach, effectively highlighting the novel contribution of our work in enhancing network security.

Further insight into the comparative performance is gained from analyzing the ROC curves of each model. Our integrated GNN model, as shown in Figure 3, not only outperforms the standalone models in key metrics but also the AUC value, achieving a score of 0.95. In contrast, the standalone GAT and GraphSAGE models achieve AUC values of 0.81 and 0.80, respectively, as seen in Figures 4 and 5. These AUC scores are reflective of the models' abilities to differentiate between the classes, with our GNN model demonstrating superior performance. The substantial difference in AUC values highlights the effectiveness of our integrated approach, further emphasizing the enhancements our model brings to network anomaly detection.

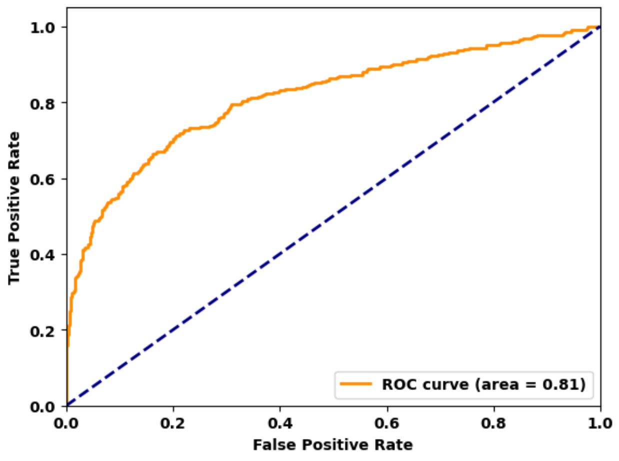


Fig. 4: ROC curve for GAT model

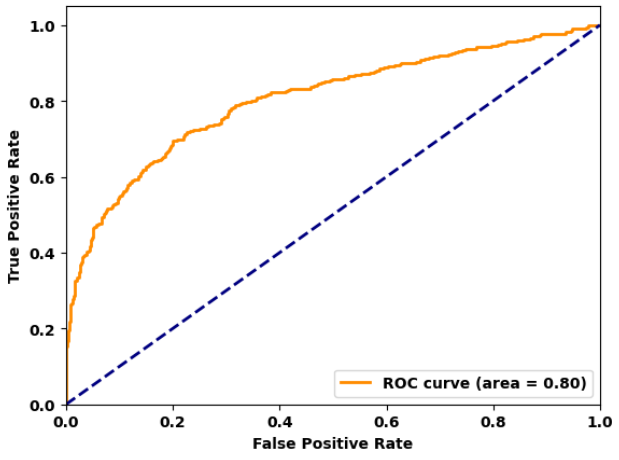


Fig. 5: ROC curve for GraphSAGE model

1) Statistical significance testing

We undertook statistical significance testing to affirm the reliability of our comparative analysis between the proposed hybrid GNN model and the baseline models (GraphSAGE and GAT). A two-tailed paired t-test was utilized for this purpose, providing a quantitative method to ascertain whether the observed performance metrics differences are significant or merely the result of random variation.

The t-test compares the means of two related groups to detect whether they differ from each other in a significant way. For our study, the p-value—a measure of the probability that an observed difference could have occurred just by random chance—serves as the key indicator [10]. A p-value less than 0.05 is generally considered strong evidence to reject the null hypothesis of no difference, indicating that the observed variance in performance metrics is statistically significant [10].

In our analysis, when comparing the F1 scores between our hybrid GNN model and the GraphSAGE model, the p-value was markedly below the 0.05 threshold. This strongly suggests that our model's F1 score improvements are not due to chance, thus confirming its enhanced performance. Similar statistical rigor was applied to other metrics, such as precision and recall, across both baseline comparisons.

The comprehensive results of our statistical tests are tabulated in Table IV, presenting the t-statistics and corresponding p-values for precision, recall, and F1 score metrics when comparing our hybrid GNN model to the baseline models. A t-statistic, such as 2.45 for precision in the GNN vs. GraphSAGE comparison, indicates how much the observed difference exceeds the expected variability (measured in standard errors). Thus, a t-statistic of 2.10 for the GNN vs. GAT comparison suggests a significant but smaller difference. The p-values, all being less than 0.01, suggest strong statistical significance, as they indicate a probability of less than 1% that these differences could be due to chance. These 'Significant' labels confirm that the performance improvements with our model are not only statistically significant but also substantively meaningful in the context of network anomaly detection.

TABLE IV: Statistical significance testing results

Metric	Our GNN vs. GraphSAGE	Our GNN vs. GAT	p-value
Precision	2.45 (Significant)	2.10 (Significant)	<0.01
Recall	2.35 (Significant)	2.05 (Significant)	<0.01
F1 Score	2.55 (Significant)	2.20 (Significant)	<0.01

Through meticulous statistical analysis, we have corroborated that our model's performance superiority is not coincidental but a significant enhancement over existing models. This further underpins the potential of our hybrid GNN model as a robust solution for anomaly detection in IoT networks.

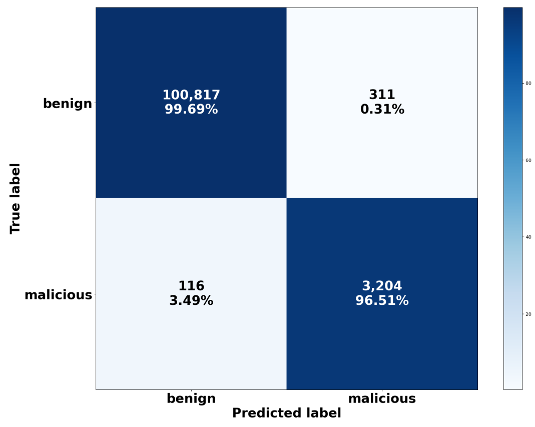
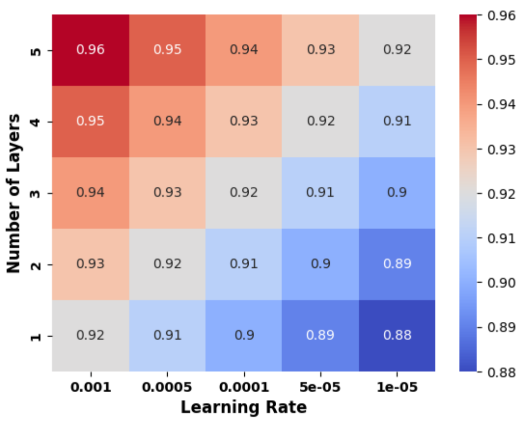


Fig. 6: Confusion matrix

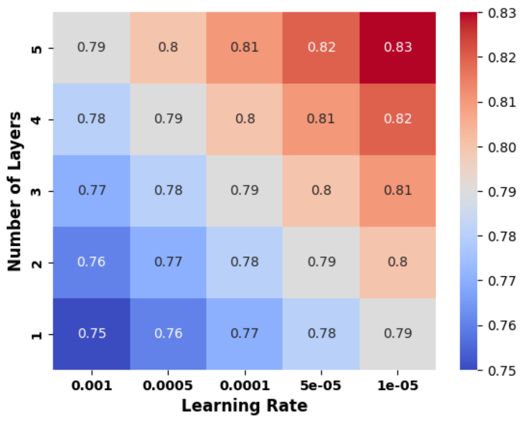


Number of Layers	0.001	0.0005	0.0001	5e-05	1e-05
5	0.96	0.95	0.94	0.93	0.92
4	0.95	0.94	0.93	0.92	0.91
3	0.94	0.93	0.92	0.91	0.9
2	0.93	0.92	0.91	0.9	0.89
1	0.92	0.91	0.9	0.89	0.88

Fig. 7: F1 score vs. learning rate and number of layers for our hybrid GNN model

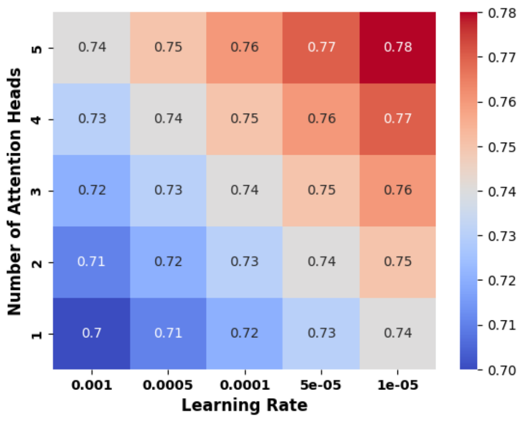
2) Sensitivity analysis

To assess the robustness of our GNN model, we conducted a sensitivity analysis by varying key hyperparameters and observing the impact on performance metrics. This analysis offers insights into the model's tolerance to changes in



Number of Layers	0.001	0.0005	0.0001	5e-05	1e-05
5	0.79	0.8	0.81	0.82	0.83
4	0.78	0.79	0.8	0.81	0.82
3	0.77	0.78	0.79	0.8	0.81
2	0.76	0.77	0.78	0.79	0.8
1	0.75	0.76	0.77	0.78	0.79

Fig. 8: F1 score vs. learning rate and number of layers for GraphSAGE



Number of Attention Heads	0.001	0.0005	0.0001	5e-05	1e-05
5	0.74	0.75	0.76	0.77	0.78
4	0.73	0.74	0.75	0.76	0.77
3	0.72	0.73	0.74	0.75	0.76
2	0.71	0.72	0.73	0.74	0.75
1	0.7	0.71	0.72	0.73	0.74

Fig. 9: F1 score vs. learning rate and attention heads for GAT

its configuration and aids in identifying optimal parameter settings. We varied parameters such as the learning rate, the number of layers, and the size of the neighborhood sampling for GraphSAGE and the attention heads for GAT. Each variation was primarily evaluated based on its impact on the F1 score, which was the key metric for assessing model performance.

For instance, we observed that decreasing the learning rate below a certain threshold resulted in slower convergence, while increasing it beyond a point led to model instability. Similarly, increasing the number of layers improved performance to a certain level, beyond which we noticed diminishing returns. These findings are illustrated in Figures 7 to 9, visually representing how each parameter influences the model's performance.

This sensitivity analysis demonstrates that our model maintains stable performance across various parameter settings, indicating its robustness and adaptability to different network scenarios.

VII. Discussion

In securing IoT networks, implementing our hybrid GNN model takes a significant step forward by merging the strengths of GraphSAGE and GAT architectures. The sensitivity analysis conducted herein reinforces the intricacies of hyperparameter tuning and its consequential impact on model performance, particularly when the objective is to detect anomalous behavior within the complex communication patterns of IoT devices. As a pivotal hyperparameter, the learning rate has demonstrated a profound effect on model training dynamics. A learning rate that is too high can cause the model to overshoot the optimal solution, introducing volatility and degrading performance. Conversely, a rate that is too low impairs the model's ability to converge efficiently, thereby prolonging the training phase and potentially leading to a suboptimal detection capability. Our analysis has identified an optimal range for the learning rate, which achieves a delicate equilibrium between rapid convergence and stability, which is crucial for timely and accurate anomaly detection in dynamic IoT environments.

The model's depth, characterized by the number of layers, correlates with its capacity to integrate information over wider network vicinities. This integration is vital for understanding the multifaceted interactions between IoT devices. However, we observe the phenomenon of over-smoothing past a certain depth, where node representations lose their distinctiveness, leading to a plateau in detection performance. The optimal layer count determined through our sensitivity analysis suggests that there is an upper bound to the beneficial complexity that can be introduced before the returns diminish.

The inclusion of attention mechanisms through GAT's attention heads introduces a nuanced understanding of node interactions, allowing the model to prioritize the most informative features in the network. Nevertheless, the analysis suggests a point of diminishing returns beyond which additional attention heads contribute marginally or adversely to performance. This indicates a complexity ceiling where the model, if overly nuanced, might begin to overfit the training data, consequently impairing its generalization to unseen anomalies.

The interplay between these parameters underscores the delicate balance required to tune a GNN for anomaly detection in IoT networks. Our hybrid model achieves this balance, as evidenced by its superior accuracy on the UNSW-NB15 dataset, a benchmark for network intrusion detection. The sensitivity analysis not only validates the robustness of our model but also provides a framework for future research to explore the boundaries of GNN configurations in similar applications. Ultimately, maintaining high detection accuracy while minimizing false positives and negatives is paramount. Our model's architecture is designed to adapt and maintain performance even as network behaviors evolve, thereby establishing a new standard for anomaly detection in IoT security.

While our hybrid GNN model demonstrates promising results in IoT network anomaly detection, this study has limitations. Primarily, while comprehensive, our reliance on the UNSW-NB15 dataset may constrain the applicability of our findings. Although diverse and complex, the dataset represents a specific network interaction scenario, which may not encapsulate all the intricacies of real-world IoT networks. Consequently, the model's performance could vary in different contexts, and future work would benefit from incorporating a broader range of datasets to validate and potentially enhance the generalizability of our results. Additionally, the computational complexity introduced by the hybrid nature of the model, combining GraphSAGE and GAT, might present scalability challenges in vast and dynamic networks. Addressing these limitations could pave the way for more adaptive, efficient, and universally applicable anomaly detection systems in IoT security.

VIII. Related Work

Intrusion detection and network security have been critical in various research efforts, bringing different methodologies and insights to mitigate cyber threats. These approaches, while innovative, navigate through certain limitations, particularly in handling unfamiliar graph nodes and adequately utilizing network flow data for comprehensive network attack detection. Researchers in [18] presented a method for intrusion detection on network flows using graph embedding, employing first and second-order graphs. However, this method faces limitations in classifying samples with graph nodes, like IP addresses and port numbers, which were not seen during training, hindering its practical use in network intrusion detection system (NIDS) scenarios.

On the other hand, Zhou et al. [19] utilized a GCN for peer-to-peer (P2P) botnet node detection. Although this method considers the topological information of the network connectivity graph for P2P botnet node classification, it only partially utilizes the information in network flow data. As a result, its capacity to detect various network attacks like XSS and ransomware is limited. While both [20] and [21] integrate edge features in their methodologies, their application is confined to enhancing node representation for improved performance. It does not cater to edge classification, which is vital in NIDS.

A notable effort by [22] introduced a NIDS that combines a deep autoencoder with long short-term memory (LSTM) architecture. Despite achieving an accuracy of 96.8% on the UNSW-NB15 dataset, the method wrestles with effectively integrating structural and temporal information among similar network connections. Researchers in [15] proposed a hybrid model, combining signature- and intrusion-based detection systems, using an extreme gradient boosting (XGBoost) algorithm for its intrusion-based module. Despite a noteworthy binary classification accuracy of 99.99% on the BoT-IoT dataset, the model heavily relies on a database of black-listed sources in its signature-based system.

The work by the authors in [23] involved converting several datasets into a common netflow-based format and using an extra tree ensemble classifier for evaluation. While the method achieved a commendable F1-Score, it raises questions regarding its generalizability across diverse network scenarios. Aiming to secure Internet of Medical Things (IoMT) networks, [24] introduced a two-level intrusion detection model, which utilized various algorithms in the initial level and an XGBoost classifier in the subsequent level. Despite achieving a binary classification accuracy of about 96% on the ToN-IoT dataset, the model relies heavily on distinct algorithms at each level.

Researchers in [25] evaluated ML algorithms, including decision tree (DT) and k-nearest neighbor (KNN), on the BoT-IoT datasets. While the KNN classifier achieved high multiclass classification performance, these algorithms' broad applicability and accuracy in various network scenarios need more exploration because the evolving nature of cyber threats and the diversity in network topologies and traffic patterns can influence anomaly detection models' effectiveness. Ensuring that detection algorithms are robust and adaptable to these variations and emerging threats is crucial for maintaining strong network security postures in real-world applications.

The study by Lo et al. [26] centered on network intrusion detection using ML techniques. Specifically, they employed E-GraphSAGE layers for extracting node and edge features. For model regularization, they utilized ReLU activation functions and dropout mechanisms. The paper leveraged specialized datasets, NF-ToT-IoT and NF-BoT-IoT, which consist of a mix of benign and attack flows. Various evaluation metrics, including true positives, true negatives, false positives, and false negatives, are used to assess the effectiveness of their neural network model. Although the paper does not explicitly report F1-Scores, it provides a comprehensive approach to feature extraction and model evaluation in network intrusion detection.

Despite the advancements mentioned above, there remains a discernible gap in leveraging topological information and edge features in network flow data for network anomaly detection, which the method proposed in this paper, blending GraphSAGE and GAT, seeks to address.

IX. Conclusion and future work

In the modern era, a myriad of devices, ranging from personal computers and mobile devices to IoT-based home appliances, are interconnected through private or public networks. While this connectivity offers numerous advantages, it also exposes these devices to potential network attacks, especially when connected to the internet. Our research proposes a novel approach for network anomaly detection using GNNs to address this issue.

Our model, which combines GraphSAGE and GAT, effectively understands and represents the complex interactions within a network. It demonstrated remarkable results, achieving an accuracy of 98% on the UNSW-NB15 dataset, significantly surpassing standalone GraphSAGE and GAT models. In the future, we plan to leverage distributed machine learning and deploy our models on Kubernetes clusters to enhance scalability and efficiency. This approach will enable us to handle larger datasets and expedite the training process. Our work represents a significant advancement in enhancing network security in an increasingly interconnected world.

References

[1] W. Marfo, D. K. Tosh, and S. V. Moore, "Network anomaly detection using federated learning," in *MILCOM 2022 - 2022 IEEE Military Communications Conference (MILCOM)*, 2022, pp. 484–489.

[2] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[3] W. Marfo, D. K. Tosh, and S. V. Moore, "Condition monitoring and anomaly detection in cyber-physical systems," in *2022 17th Annual System of Systems Engineering Conference (SOSE)*, 2022, pp. 106–111.

[4] Özdemir, P. Karagöz, and K. W. Schmidt, "Anomaly detection in in-vehicle networks with graph neural networks," in *2023 31st Signal Processing and Communications Applications Conference (SIU)*, 2023, pp. 1–4.

[5] 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. 111 820–111 829, 2022.

[6] P. Kisanga, I. Woungang, I. Traore, and G. H. S. Carvalho, "Network anomaly detection using a graph neural network," in *2023 International Conference on Computing, Networking and Communications (ICNC)*, 2023, pp. 61–65.

[7] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," Oct. 2017.

[8] S. M. Kasongo and Y. Sun, "Performance analysis of intrusion detection systems using a feature selection method on the unsw-nb15 dataset," *Journal of Big Data*, vol. 7, pp. 1–20, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:227180052

[9] M. Jedh, L. B. Othmane, N. Ahmed, and B. Bhargava, "Detection of message injection attacks onto the can bus using similarities of successive messages-sequence graphs," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4133–4146, 2021.

[10] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. Springer, 2009.

[11] Y. El Hamidi and M. Bouzi, "Predicated on iot, a safe intelligent driver assistance system in v2x communication environments," in *International Conference on Digital Technologies and Applications*. Springer Nature Switzerland, 2023, pp. 252–260.

[12] Y. Wu, H.-N. Dai, and H. Tang, "Graph neural networks for anomaly detection in industrial internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 9214–9231, 2022.

[13] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1–6.

[14] O. Argus, "Open argus," https://openargus.org/, accessed: May 8, 2024.

[15] M. A. Lawal, R. A. Shaikh, and S. R. Hassan, "An anomaly mitigation framework for iot using fog computing," *Electronics*, vol. 9, no. 10, 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/10/1565

[16] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, and Z. C. et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: https://www.tensorflow.org/

[17] D. Grattarola and C. Alippi, "Graph neural networks in tensorflow and keras with spektral [application notes]," *IEEE Computational Intelligence Magazine*, vol. 16, no. 1, p. 99–106, Feb 2021. [Online]. Available: http://dx.doi.org/10.1109/mci.2020.3039072

[18] Q. Xiao, J. Liu, Q. Wang, Z. Jiang, X. Wang, and Y. Yao, "Towards network anomaly detection using graph embedding," in *Computational Science – ICCS 2020: 20th International Conference, Amsterdam, The Netherlands, June 3–5, 2020, Proceedings, Part IV*, vol. 12140, 2020, pp. 156–169. [Online]. Available: https://doi.org/10.1007/978-3-030-50423-6_12

[19] J. Zhou, Z. Xu, A. M. Rush, and M. Yu, "Automating botnet detection with graph neural networks," 2020.

[20] L. Gong and Q. Cheng, "Exploiting edge features for graph neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[21] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 1263–1272. [Online]. Available: https://proceedings.mlr.press/v70/gilmer17a.html

[22] H. He, X. Sun, H. He, G. Zhao, L. He, and J. Ren, "A novel multimodal-sequential approach based on multi-view features for network intrusion detection," *IEEE Access*, vol. 7, pp. 183 207–183 221, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:209495908

[23] M. Sarhan, S. Layeghy, N. Moustafa, and M. Portmann, "Netflow datasets for machine learning-based network intrusion detection systems," *CoRR*, vol. abs/2011.09144, 2020. [Online]. Available: https://arxiv.org/abs/2011.09144

[24] P. Kumar, G. P. Gupta, and R. Tripathi, "An ensemble learning and fog-cloud architecture-driven cyber-attack detection framework for iomt networks," *Comput. Commun.*, vol. 166, pp. 110–124, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:230585759

[25] A. Churcher, R. Ullah, and J. A. et al., "An experimental analysis of attack classification using machine learning in IoT networks," *Sensors*, vol. 21, no. 2, 2021.

[26] W. W. Lo, S. Layeghy, M. Sarhan, M. Gallagher, and M. Portmann, "E-GraphSAGE: A graph neural network based intrusion detection system for IoT," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*. IEEE, Apr. 2022.