SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Anomaly Detection in Team Communication Platforms

Fabian Höltke

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Anomaly Detection in Team Communication Platforms

Erkennung von Anomalien in Team-Kommunikationsplattformen

Author:	Fabian Höltke
Supervisor:	Prof. Dr. Claudia Eckert
Advisor:	Ching-Yu Kao and Wei Herng Choong
Submission Date:	17.07.2023

I confirm that this master's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 17.07.2023 Fabian Höltke

Abstract

Team communication platforms have emerged as vital tools in our professional and personal lives. They facilitate collaboration, streamline workflow, and serve as indispensable links that connect us to our colleagues, classmates, and peers. However, as these platforms evolve and gain widespread acceptance, they simultaneously expose new vulnerabilities for malicious activities, resulting in significant data security challenges. Detecting attackers on these platforms is particularly daunting, given the high level of assumed trust among users.
Graph Neural Networks (GNNs), a type of machine learning algorithm specifically designed for graph-based data, are emerging as a promising solution to tackle evolving security challenges on graph-based data networks. Over recent years, GNNs have proven to be superior in the field of anomaly detection on graph networks, outperforming traditional machine learning or heuristic-based approaches.
In this thesis, we introduce ECONAD, a specialized GNN model developed to detect anomalies in team communication platforms. ECONAD distinguishes itself by incorporating human knowledge about known data breaches through innovative augmentation strategies and processing team communication platforms through various attack vector-specific perspectives. In addition, we present a novel dataset, detailing the activities of 250 users on a team communication platform over a period of three years. This dataset serves as the foundation for testing and evaluating the effectiveness of our anomaly detection model. Through our experiments, we show that our model surpasses state-of-the-art GNN anomaly detection models when applied to this unique dataset, outperforming the baseline by up to 35% in recall and 14% in the overall f1 score. Moreover, our approach unveils graph-based anomalies that existing threat detection methods are unable to identify.

Contents

1 Introduction

As digital technology advances at an unprecedented pace, team communication platforms emerged as vital tools in our professional and personal lives. They facilitate collaboration, streamline workflow, and serve as indispensable links that connect us to our colleagues, classmates, and peers. As of now, over 83% of professionals worldwide depend on these technologies for team communication [1]. Simultaneously, this market is projected to be worth 23.8 billion USD by 2030, underscoring its growing prominence in our digital society [2].
However, as these platforms evolve and gain widespread acceptance, they simultaneously open up new vulnerabilities for malicious activities, posing significant challenges to data security [3, 4]. Many users often overlook the potential security risks embedded within these communication platforms, falsely presuming that threats are unlikely to originate from familiar co-workers or acquaintances. This complacency allows an attacker to exploit a single compromised account and leverage it to deceive other users, thus gaining unauthorized access to more sensitive information [5, 6, 7, 8].

Graph Neural Networks (GNNs) [9] are rising as a promising solution to tackle evolving security challenges on graph-based data networks [10]. In recent years, GNNs have proven to be superior in the field of anomaly detection on graph networks, outperforming traditional machine learning or heuristic-based approaches [11, 12]. As team communication platforms can be represented as heterogeneous graphs, we propose to apply a GNN to find threats and malicious actors within a team communication platform.
However, this heterogeneity poses a challenge for existing GNNs to process, as existing models for malicious node classifications are primarily designed for homogeneous graph structures [10]. Also, incorporating team communication-specific human knowledge about previous attacks into a model is not trivial. Evaluated breaches often use several different attack vectors, which are team communication platform-specific and have not been investigated by other GNN-based research.

In this thesis, we present ECONAD, a GNN model specifically designed to detect anomalies in team communication platforms. ECONAD is based on a state-of-the-art anomaly detection model for social networks, which is further enhanced by introducing multiple views to handle the graph's heterogeneous structure. Additionally, novel custom data augmentation strategies are introduced to identify multiple types of attacks that were recorded within team communication platforms.
To evaluate ECONAD, we create a custom team communication dataset, containing the activity of 250 users over three years. On this dataset, each enhancement of ECONAD is extensively evaluated, with the top-performing strategies combined to create our final

model.

The following chapter introduces the background of this thesis, including previously occured breaches in team communication platforms, and defined attack vectors in this domain. Related models and approaches for anomaly detection on team communication platforms are further described in Chapter 3. Chapter 4 highlights the need for a new dataset and describes the process of creating a custom dataset for this thesis. The dataset is further processed with our novel model in Chapter 5. Here, we outline the improvements made to the base model. In Chapter 6, we evaluate our approach, to understand what impact the different enhancements had on the model and how they compare to the baseline. Our findings are discussed in Chapter 7. In Chapter 8, we conclude our work and outline future work.

2 Background

More and more companies and organizations are using team collaboration platforms as their primary tool for internal communication and organization [13][14]. A centralized team communication platform offers much more comfort and ease to use for communication than traditional email communication.
However, while years of phishing attacks have created users suspicious of unusual emails, few suspect a message from a coworker within their own organization's platform. Therefore, compromising a single account within a platform can be easily leveraged by an attacker to deceive other users and gain additional access.
Many organizations maintain multiple channels to encourage participation and facilitate knowledge sharing. Unfortunately, the question of who has access to these channels is frequently overlooked, leading to the unintentional sharing of confidential information, including sensitive messages such as passwords or API keys. Additionally, few individuals consider the long-term storage of their messages and the potential access by compromised accounts. Once the information is within the platform, it can be indefinitely accessible, creating future security risks [3].
Not only does lateral phishing pose a threat within team communication platforms, but the extensive configuration options for workspaces also present risks. For example, users can install third-party integrations, which can potentially act maliciously and have been exploited in the past to extract sensitive data from the platform [15, 16]. Furthermore, all users can modify their profile settings without any countermeasures in place to prevent potential impersonation attacks [4].

2.1 Recent Attacks on Team Collaboration Platforms

No exact number exists regarding the frequency of breaches on team communication platforms. Usually, only the most impactful breaches are published and recorded by the media or post-mortems.

At the beginning of 2023, two prominent companies, the ride-booking company Uber [17] and the video game developer Activision Blizzard [18], faced breaches on their internal communication platform, Slack [19] [5, 6]. These breaches occurred within a short span of five months, and it is believed that the same attacker was responsible for both incidents. In both cases, the attacker used spear-phishing SMS attacks to bypass Slack's 2FA authorization. Once an employee fell victim to the attack, the attacker gained control over their Slack account and

proceeded to exploit all publicly available channels. The attacker's actions resulted in the unauthorized access of sensitive information about unreleased games for Activision Blizzard and the discovery of passwords for other services in the case of Uber. Furthermore, the attacker managed to breach the administration panels of several internal services associated with Uber. As a consequence, Uber experienced a 5.2% drop in share prices, resulting in an estimated loss of 4.53 billion USD [20].
Similarly, in 2021 and 2022, two video game developers, Electronic Arts [21] and Rockstar [22], also encountered breaches on their internal communication tool[7, 8]. In both instances, the attacker purchased stolen session cookies from the dark web and utilized them to log into the victims' Slack accounts. Subsequently, the attacker internally contacted IT support to request new authentication tokens. In the case of Electronic Arts, they pretended to have lost their phone.

2.2 Attack Vectors

The reports of breaches described in the previous Section 2.1 differ as follows: In the first two scenarios, involving Uber and Activision Blizzard, the attacker downloaded and processed messages to find sensitive information. In the second two scenarios, the attackers conducted active spear phishing attacks within the team collaboration platform, extending their attack on other services outside the team communication platform.

During the investigation of these recent attacks, we identified the following attack vectors (AV) that the attackers used to leverage their attack after the initial access to the team collaboration platform:

AV1 **Lateral Neighborhood Phising**: The attacker attempts to gain access to sensitive information by sending phishing messages to known members of the victim's immediate communication circle, which could include the victim's team or individuals with whom they had previous message interactions.

AV2 **Lateral Phishing to "Powerusers"**: "Powerusers" are defined users within a platform that manage a security-relevant resource. An example of this is an IT administration that holds credentials for specific services. By targeting power users with elevated administrative privileges, the attacker can attempt to build up persistence in the platform (e.g. requesting new credentials) or leverage the attack to new services. Typically, these attacks require the attacker to contact users with whom the victim had no prior interaction.

AV3 **Impersonation**: The attacker impersonates a user of the team communication platform by copying the victim's avatar and profile information. They attempt to gain access to sensitive information by pretending to be a well-known co-worker within the organization.

AV4 **Channel Sniffing**: The attacker attempts to uncover sensitive information by infiltrating multiple public channels. This information may include plain-text passwords or confidential documents that have been previously shared, as message history remains preserved within team communication platforms and does not disappear over time.

In this thesis, our objective is to identify the four attack vectors within a team communication platform before the attacker can exploit them and inflict further damage.

3 Related Work

The following sections review and compare existing threat detection systems and methods for social networks. With this, state-of-the-art anomaly detection algorithms for graph networks are presented.Various system designs are compared, including the underlying algorithm on which our own model is based.
Next, this chapter introduces currently accessible datasets from team communication platforms that are freely available for research. Each dataset is evaluated based on its relevance to the research problem and data quality.
Finally, we examine existing commercial solutions for anomaly detection in team collaboration platforms and evaluate their capabilities to detect the attack vectors described in Section 2.2.

3.1 Anomaly Detection on Social Networks

Current strategies for anomaly detection on (social) networks can be categorized into two leading methods: Non-Deep Learning (Non-DL) methods and Deep Learning (DL) methods. Non-DL methods typically rely on various types of heuristic anomaly measurements to detect anomalies, while DL methods often resort to feature learning or Graph Neural Networks (GNNs) for the detection of anomalies. In recent years, DL-based approaches have shown superior performance over traditional Non-DL methods [12, 11]. The following subsections provide a brief overview of each category.

3.1.1 Heuristic Based Anomaly Detection

Heuristic-based anomaly detection significantly affected graph analysis, paving the way for novel techniques to identify and interpret anomalous nodes or edges. Innovative methods such as the measure of "normality" [23], the user-oriented FocusCO algorithm [24] (used for user preference attribute extraction), outlier classification models [25], or methods for selecting congruent subspaces in multivariate attributed graphs [26] all emerged from this field. Taking into account both the structural and attribute-based characteristics of graphs, these heuristic-based approaches demonstrated notable advancements over traditional techniques that prioritize structural features exclusively.

However, the rise of deep learning-based techniques introduced a significant shift in anomaly detection, redefining the methods and mechanisms used to identify anomalies. The ability of deep learning-based approaches to discern complex patterns and dependencies within data outperformed the performance of heuristic-based methods. For example, Li et al. [12] introduced a learning framework that leveraged residual analysis for anomaly detection in attributed networks. This approach successfully identified anomalies that significantly deviated from expected behavior, outperforming traditional heuristic-based methods. Furthermore, Müller et al. [11] introduced the Graph Outlier Ranking (GOutRank) method that offered a unified approach to anomaly detection in graph and attribute data spaces. The authors demonstrated that deep learning-based techniques outperformed heuristic-based methods.

3.1.2 Non-Graph DL Based Anomaly Detection on Social Networks

Before shifting the focus to graph anomaly detection, we took a broader look at the topic and investigated how other researchers detected anomalies in the domain of social networks using non-graph-based approaches.

Several studies chose to focus their entire attention on the attributes of social network users. For example, Kawase et al. [27] delved into the issue of account takeover, a form of online identity theft prevalent in online vehicle marketplaces. The authors presented a dual-faceted approach to prevent and detect unauthorized account activities. To prevent account takeovers, they performed a behavioral analysis of fraudsters' operations and implemented a mutual two-factor authentication method, resulting in a significant reduction of 43% in account takeovers. For detecting fraudulent activities, a concept drift-sensitive machine learning training approach was introduced, which improved detection rates by 18% over baseline methods. Consequently, the automated detection resulted in a safer marketplace by reducing the exposure of fraudulent listings by 69%.

In another study, Ho et al. [28] focused entirely on lateral email phishing attacks. They conducted a large-scale characterization of these attacks using a dataset of 113 million employee-sent emails from 92 enterprise organizations. To detect phishing emails, they developed a classifier that could identify hundreds of real-world lateral phishing emails. This study offered valuable insights into the nature and scale of enterprise phishing attacks.

The research conducted by Shen et al. [29] and He et al. [30] focused on analyzing user features within dating apps to detect fake accounts. Shen et al. [29] developed a trust-aware detection framework to detect malicious users in dating social networks. They proposed a user trust model and a novel data-balancing method within their framework to enhance the recall rate of malicious user detection. This approach significantly outperformed other baseline algorithms.
He et al. [30], on the other hand, developed a novel system named DatingSec to counteract the hidden signals in the textual information of user interactions. This system combined Long Short-Term Memory neural networks (LSTMs) and an attentive module to capture users' temporal-spatial behaviors and user-generated textual content. When evaluated on a real-world dataset from Momo [31], a widely used dating app, DatingSec claimed it outperformed

state-of-the-art methods.

Much like He et al.[30], who employed LSTMs to predict anomalous users within dating networks, numerous other studies relied solely on Natural Language Processing (NLP) approaches to process user messages for anomaly detection. For example, Seyler et al. [32] developed a novel general framework for the semantic analysis of text messages to detect compromised accounts on social networks. Their approach, based on the difference in language usage between normal users and adversaries, proposed new semantic features for measuring semantic incoherence in a message stream. When tested using a Twitter dataset [33], their approach proved effective in detecting compromised accounts, with the KL-divergence-based language model feature performing the best.
Similarly, Ilias et al. [34] addressed the issue of detecting automated accounts or bots on Twitter [35], which spread harmful content. They proposed two methods, based primarily on NLP, for the early detection of these bots. The first method utilized feature extraction and machine learning algorithms to identify accounts that post automated messages. The second method introduced a deep learning architecture, unique in its use of an attention mechanism for bot identification. Both methods, when evaluated using real Twitter datasets, demonstrated advantages over existing techniques to identify malicious users on social networks.

Although these papers produced good results for their specific use cases, they did not incorporate any graph topology for their classification.

3.1.3 Graph Based DL Anomaly Detection on Social Networks

The rapid advancements and wide adoption of graph neural networks (GNNs) in recent years generated considerable attention and traction in the field of anomaly detection on attributed graph networks [36]. In this regard, the papers of Ma et al. [10] and Kim et al. [37] presented an overview of the current status quo in the field of GNN anomaly detection and reviewed recent advances in detecting graph anomalies using GNN models. Both studies highlighted the significant role of deep learning in handling high-dimensional network data, proposed new taxonomies for different state-of-the-art methods, and outlined potential future research directions.

General purpose anomaly detection models such as ANOMALOUS [38] and DOMINANT [39] paved the way for more specialized models. ANOMALOUS [38] addressed the issues of noisy and irrelevant node attributes in anomaly detection by proposing a new framework that jointly conducted attribute selection and anomaly detection based on CUR [40] matrix decomposition and residual analysis. On the other hand, DOMINANT [39] tackled the problem of anomaly detection on attributed networks by proposing a novel deep learning model that integrated topological graph structure and node attributes for node embedding learning, using graph convolutional networks [41] and deep autoencoders [42].

These advances led to the development of specialized models for social networks [43, 44, 45].

Chaudhary et al. [43] proposed a Graph Neural Network for anomaly detection in email and Twitter networks by studying the graph structure and understanding the functioning of anomalous nodes through the use of deep neural networks. Similarly, Dou et al. [44] proposed a novel framework, which exploited user preference for fake news detection by jointly modeling content and graph, addressing the issue of disinformation and fake news. Xu et al. [45] addressed the problem of modeling and integrating human knowledge of different types of anomalies for attributed network anomaly detection. Their approach modeled prior human knowledge through novel data augmentation strategies and integrated them in a Siamese graph neural network encoder through a well-designed contrastive loss. Their proposed model *Contrastive Anomaly Detection* (CONAD) is further evaluated in Section 3.3.

The technique of using different graph transformations based on node features and relations to improve the performance of anomaly detection models on (heterogeneous) graphs was extensively discussed in several papers [10, 46, 47]. Zhang et al. [46] used this technique to identify malicious users in underground forums, introducing an intelligent system, iDetective, that employed a meta-path-based approach for user representation and a method named Player2Vec for key player identification [46]. Meanwhile, Peng et al. [47] proposed ALARM, an extendible framework that combined multiple GNN models for feature-specific predictions. ALARM took into account user preferences and heterogeneous attribute characteristics through multiple graph encoders and a well-designed aggregator supporting self-learning and user-guided learning.

The papers discussed in this section demonstrated the potential of graph-based methods for anomaly detection on social networks, not only in understanding complex patterns but also in capturing subtle, nuanced deviations that might otherwise go unnoticed. In the next section, we take a closer look at three different system designs used by the previously introduced GNN anomaly detection models for social networks [44, 39, 47].

3.2 System Designs

This section investigates common techniques and approaches of the previously mentioned papers used to detect anomalies in social media datasets [44, 39, 47]. We identified the following three system designs and took a closer look at them.

- The first approach, as described in the paper "User Preference-aware Fake News Detection" by Dou et al. [44], was employed to detect fake news within a Twitter [35] network. For each message transmitted, a unique graph was constructed based on its propagation throughout the Twitter community. Utilizing a dual-encoder design structure, the final classification of each graph depended on both the message's interaction graph structure and its embeddings. The complete system design is illustrated in Figure 3.1. Although this approach appeared reasonable for a vast open social network such as Twitter, which includes millions of users, it may be less applicable to a team communication platform characterized by a more limited user base.

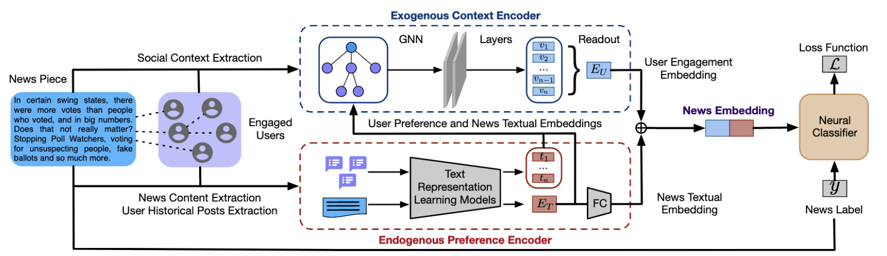

Exogenous Context Encoder
GNN
Layers
Readout
E_U
Social Context Extraction
News Piece
In certain swing states, there were more votes than people who voted, and in big numbers. Does that not really matter? Stopping Poll Watchers, voting for unsuspecting people, fake ballots and so much more.
Engaged Users
User Engagement Embedding
User Preference and News Textual Embeddings
News Embedding
Loss Function
$\mathcal{L}$
Neural Classifier
Text Representation Learning Models
E_T
FC
News Content Extraction
User Historical Posts Extraction
News Textual Embedding
$\hat{y}$
News Label
Endogenous Preference Encoder

Figure 3.1: Event-focused graph and model design, as demonstrated by Dou et al. [44]. For each written message, a user-interaction graph and text embeddings were created and processed separately.

- The next investigated design approach was based on processing a single attributed full graph that represented an entire social network and its interactions. Unlike the previously mentioned system design, where the graph was divided into multiple subgraphs based on events such as sent messages, this approach processed the entire graph at once. This allowed for greater flexibility in sampling and detecting various types of anomaly events. According to the number of research papers focusing on anomaly detection tasks, this graph processing approach was the most popular [10].
  To process graph structures derived from social networks, the paper "Deep Anomaly Detection on Attributed Networks" [39] introduced an autoencoder [42] approach for anomaly detection based on reconstruction error (see Figure 3.2). This approach was further refined for the application on social networks in the paper "Contrastive Attributed Network Anomaly Detection with Data Augmentation" [45], which is evaluated in more detail in Section 3.3.

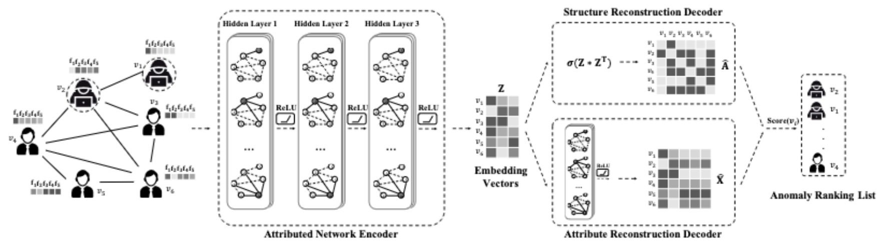


Figure 3.2: DOMINANT [39] employed an autoencoder design in which the entire graph is processed at once. The classification was based on the reconstruction error of a siamese decoder, which was specifically trained to reconstruct the original graph.

- The final investigated system design assumes that distinct graph views can represent various graph features. Each graph view is subsequently processed individually by a GNN. The final classification is determined by combining the outputs of all GNNs. An exemplary architecture, introduced in the paper titled "A deep multi-view framework for anomaly detection on attributed networks" by Peng et al. [47], is illustrated in Figure 3.3.

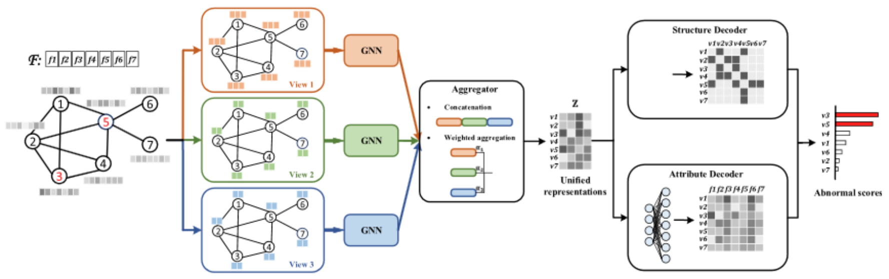


Figure 3.3: Design of a multi view graph system, as described in the work by Peng et al. [47]. This approach involved splitting the graph into distinct views according to its features. Each graph view was subsequently processed by an independent Graph Neural Network (GNN). The final classification was determined by combining the individual GNN classifications.

Compared to graphs derived from traditional social networks, such as Twitter [35] or Facebook [48], team communication platforms are more limited in memory size and graph complexity. There is no need to create a new subgraph for single events, instead, we can process the full graph at once. Therefore, this thesis focuses on the second system design. This approach offers the most flexibility for sampling and detecting different kinds of anomaly events and we believe that it is the most suitable for detecting the observed attack vectors described in Section 2.2

.

3.3 Contrastive Anomaly Detection (CONAD)

As mentioned in Sections 3.1.3 and 3.2, multiple GNN-based models were fine-tuned for anomaly detection on social networks. CONAD, the model introduced in the paper *Contrastive Attributed Network Anomaly* [45], was chosen for further evaluation, as it leveraged human knowledge into its model design to detect different anomaly types. Initially based on DOMINANT [39], this model could process a single social network graph at once. This approach seemed reasonable for our use case, as we can leverage the knowledge of the observed attack vectors described in Section 2.2.

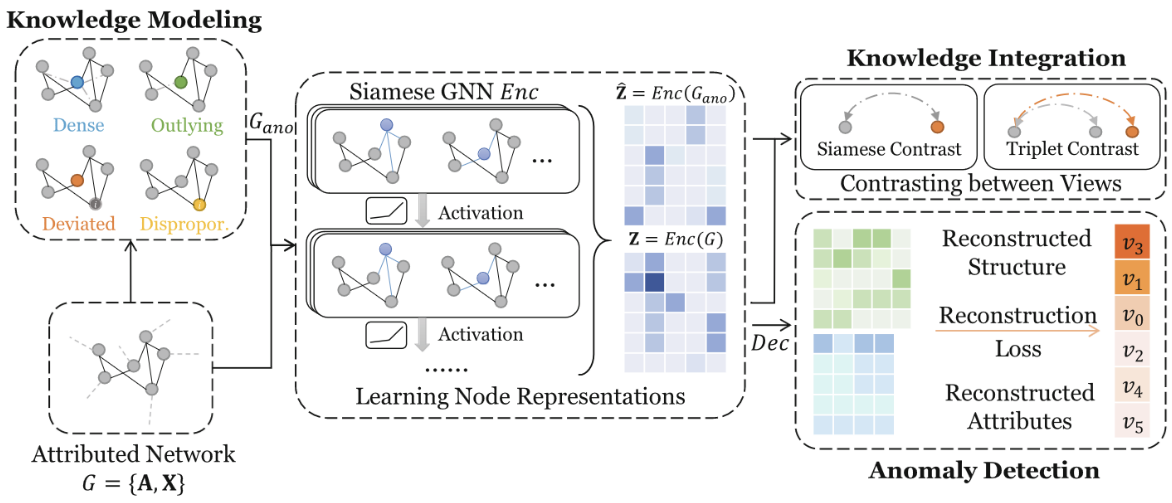


Figure 3.4: CONAD [45] system design. In contrast to other GNN-based anomaly detection approaches for social networks, CONAD incorporates human knowledge about anomalies into its model design.

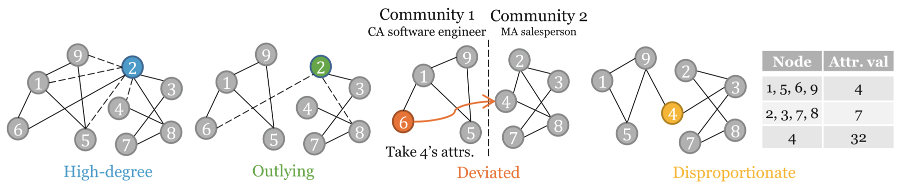
3.3.1 Design

The CONAD model design, as shown in Figure 3.4, consists of three main components.
The first *Knowledge Modeling Module* component introduces augmentation strategies for each defined anomaly type ξ to the input attributed graph $\mathcal{G}$ and generates the augmented attributed graph $\mathcal{G}_{ano}$ accordingly.
Then, the *Knowledge Integration Module* is used to feed $\mathcal{G}$ and $\mathcal{G}_{ano}$ into the graph encoder, a Siamese GNN, to learn the graph node representations. Using a Siamese encoder, both graphs are encoded into the same latent space, making it possible to contrast between the node representations of $\mathcal{G}$ and $\mathcal{G}_{ano}$. After the encoding, a unique contrastive loss is used to guide the encoder to represent normal nodes on the input-attributed network and contrastive samples on the augmented attributed network differently. Consequently, this captures anomaly patterns of the augmented nodes.
Finally, the *Anomaly Detection Module* is used to reconstruct the graph structure and node attributes from the learned node representations. The reconstruction errors are then leveraged as suspicion scores, quantifying how likely a node is to be abnormal, to detect anomalies in $\mathcal{G}$.

3.3.2 Knowledge Modeling Module

By default, CONAD has four predefined augmented anomaly types, from both the structural site and the attribute side.

- **high-degree (structural)** augmentation for detecting nodes with a high degree centrality.
- **outlying (structural)** augmentation for detecting outlying nodes.



Node	Attr. val
1, 5, 6, 9	4
2, 3, 7, 8	7
4	32

Figure 3.5: Four different kinds of anomalies are modeled through a data augmentation strategy. These augmentations can be based on the graph structure or node attributes.

- **deviated (attribute)** augmentation for detecting nodes with deviated attributes from their neighbors.
- **disproportionate (attribute)** for detecting nodes with disproportionate attributes, e.g., unreasonable low or high attribute values.

After applying the augmentation methods, the augmented attributed graph $\mathcal{G}_{ano}$ is obtained. Within $\mathcal{G}_{ano}$, a label vector y denotes if a node corresponds to one of the introduced anomaly augmentation types.
In Chapter 5, we will introduce our own novel augmentation strategies specifically designed for team communication platforms.

3.3.3 Knowledge Integration Module

The modeled human knowledge within the augmentation types is now integrated into the detecting model through learning node representations and contrasting between different views.

Learning Node Representations. In order to encode both $\mathcal{G}$ and $\mathcal{G}_{ano}$, CONAD employs a Siamese GNN architecture as an encoder.
Here, the aggregation mechanism ${h_i}^{(l+1)} = AGG(\{{h_i}^{(l)}\} \cup \{h_j^{(l)} : j \in N_i\})$ is used to learn the node representations, where ${h_i}^{(l)}$ denotes the representation of node i in the l-th layer, and ${h_i}^{(0)}$ is the input attribute of node i. N_i is the set of all neighbors of node i, $AGG(\cdot)$ is the aggregation function that can be implemented by mean pooling, max pooling, or many others [49]. On default, CONAD specifies the information aggregation based on the self-attention mechanism in Graph Attention Networks (GAT) [50].
Multiple GAT layers are stacked to form the encoder Enc for node representation learning.

Contrasting Between Views. To fully harness the power of human knowledge in $\mathcal{G}_{ano}$, CONAD contrasts between the $\mathcal{G}_{ano}$ and the normal view $\mathcal{G}$. The anomalous patterns on the attributed network are expected to be well characterized through this contrastive process. Two different contrast strategies are used to contrast between the two views, Siamese contrast

and Triplet contrast.
Siamese contrast: Suppose *Enc* encoded $\mathcal{G}$ and $\mathcal{G}_{ano}$ through stacked GAT layers into the representations $\mathbf{Z}$ and $\hat{\mathbf{Z}}$, then siamese contrast is performed between $\mathbf{z}_i$ and $\hat{\mathbf{z}}_i$, the representations of node i in $\mathcal{G}$ and $\mathcal{G}_{ano}$, respectively. The contrastive loss is defined as:

$$\mathcal{L}^{sc} = \frac{1}{n}\sum_{i=1}^{n}\left(I_{y_i=0}\cdot d(z_i,\hat{z}_i) + I_{y_i=1}\cdot \max\{0, m - d(z_i,\hat{z}_i)\}\right)$$

where I is the indicator function, y_i is the label of node i, $d(\cdot)$ is the Euclidean distance, and m is the margin. When applying Siamese contrastive loss ($y_i = 1$), node i is considered abnormal in $\mathcal{G}_{ano}$, the distance between the nodes representation in $\mathcal{G}$ and $\mathcal{G}_{ano}$, $d(z_i,\hat{z}_i)$, will be maximized with margin no smaller than m. When $y_i = 0$, the node i is considered normal in $\mathcal{G}_{ano}$, the distance between the nodes representation in $\mathcal{G}$ and $\mathcal{G}_{ano}$, $d(z_i,\hat{z}_i)$, will be minimized.
Triplet contrast: To further enhance the contrastive learning, CONAD also employs triplet contrast that works on the triplet of nodes $\{z_i, z_j, \hat{z}_j\}$, where z_i z_j are normal nodes in $\mathcal{G}$, $\hat{z}_j$ is an abnormal node in $\mathcal{G}_{ano}$. The triplet contrastive loss is defined as:

$$\mathcal{L}^{tc} = \sum_{\substack{\forall A_{ij}=1,\\ y_i=0, y_j=1}} \max\left\{0, m - \left(d(z_i,\hat{z}_j) - d(z_i,z_j)\right)\right\}$$

By minimizing the contrastive triplet loss, the model will increase the gap between two distances with a margin no smaller than m.

3.3.4 Anomaly Detection Module

Besides learning from human knowledge through augmentation strategies in $\mathcal{G}_{ano}$, CONAD also learns from the original attributed network $\mathcal{G}$ to detect anomalies in it. The aim is to reconstruct the graph structure and node attributes from the learned node representation view $\mathbf{Z}$. Since anomalies cannot be well reconstructed, the reconstruction errors are leveraged as suspicion scores to detect anomalies in $\mathcal{G}$. The model uses a decoder function *Dec*, that consists of a GAT layer to reconstruct the adjacency and attribute matrix from $\mathbf{Z}$. The reconstruction error is then defined as:

$$\hat{A} = \sigma(Z\cdot Z^{\top}), \quad \hat{X} = \mathrm{GATLayer}(A, Z)$$

$$\mathcal{L}^{\mathrm{recon}} = \lambda\left\|A - \hat{A}\right\|_F + (1-\lambda)\cdot\left\|X - \hat{X}\right\|_F$$

Here, $\sigma(\cdot)$ is a nonlinear activation function, e.g ReLU [51], $(\cdot)^{\top}$ and $\|\cdot\|_F$ are the transpose and Frobenius norm (L2 norm for a matrix), respectively. As often, λ is a hyperparameter to balance the reconstruction loss between the adjacency and attribute matrix. In Chapter 6, we experiment with different values for λ to determine which reconstruction error is more important for our introduced augmentation strategies.

3.4 Datasets

In this section, we take a closer look at potential datasets that can be used to evaluate our model. We distinguish between two kinds of datasets: First, datasets containing messages from team collaboration platforms. Second, datasets that contain messages from other platforms, such as social networks, that were used to train and evaluate models for anomaly detection.

3.4.1 Team Communication Datasets

Recent research on team communication platforms focused on message disentanglement. For this task, Chatterjee et al. [52] monitored various programming-related Slack workspaces over a period of two years. The dataset contained *38,955* conversations from *12,171* unique users and was split into four workspaces: clojurians (7918 conversations), elmlang (22172 conversations) and pythondev (8887 conversations). After the initial publication, it was also extended with messages from more programming-related workspaces, such as racket (about 1900 conversations).

In addition to the monitored Slack workspaces by Chatterjee et al., the paper *GitterCom* [53] and the website FreeCodeCamp.org [54] published datasets of up to 10,000 messages collected from several Gitter [55] communities. As in the previous dataset, all workspaces were associated with software engineers working on open-source software or specific programming languages. The workspaces were publicly available without any restrictions for communication or joining a community.

3.4.2 Non-Team Communication Datasets

Most GNN anomaly detection algorithms for message communication are not trained on team communication datasets. Instead, they used data from social networks. Here, the most common datasets originated from Twitter [35], Reddit [56], and Facebook [48] [57, 58, 59]. However, none of these datasets included internal communication from a larger company or organization.
The first Twitter and Facebook datasets [57] were initially introduced to discover social circles in ego networks. Here, the authors developed a model that combined network structure and user profile information to predict these circles.
In the second Twitter dataset, called *FakeNewsNet* [58], the authors presented a fake news data repository, which contained two comprehensive datasets with various features in news content, social context, and spatiotemporal information.
The Reddit dataset [59] was introduced to train a model for node classification. Here, the authors classified the category of unseen nodes in evolving information graphs based on citations and Reddit post data.

All the mentioned social network datasets included message data from public conversations.

The most popular dataset that only contained private messages was Enron [60], published in 2004. This dataset contained 0.5M emails of 150 employees of the Enron Cooperation [61], a former US-based energy company. In recent years, this dataset was often used for the task of email classification [62].

The presented datasets from team collaboration platforms and datasets that contained messages from other social networks, are not very similar regarding graph topology and message content in contrast to a private organization's team communication platform. In Chapter 4, we will conduct a more comprehensive evaluation of these identified differences.

3.5 Commercial Solutions

During our research, we found the following commercial solutions that promised different levels of threat detection for team collaboration platforms:

- Avanan [63] was a cloud security platform that offered a solution for popular team communication platforms. It promised to detect account takeovers and *insider threats* by using audit logs. Furthermore, it claimed to detect malicious files and sensitive information.
- Zerofox[64] was another cloud security platform that offered an integration for team communication platforms. Compared to Avanan, its focus centered more on message data, detecting abusive language, malicious links, and credential theft.
- Nightfall[65] was a cloud-native data loss prevention platform. It was entirely focused on data leakage prevention [66].

Among the most popular team communication platforms, only Slack offerd an endpoint for monitoring suspicious user activity when a customer purchased the most expensive enterprise plan [67]. This feature, known as the *Audit Logs API*, could detect ten different types of anomalous events:

First, it could keep track of Autonomous System Numbers (ASNs) to spot *bad* ones and send alters. Then, it also monitored file activity, generating alerts for abnormal downloads or file-sharing behaviors that may imply data misuse. IP address history of user tokens could be observed, raising an alert when a potentially suspicious IP, such as a cloud ASN, was identified. Alerts could also be triggered on an unusual volume of search queries, hinting at suspicious behavior. Furthermore, inconsistencies in session cookies or client fingerprints also raised alarms. The use of TOR exit nodes, often associated with anonymous and possibly malicious activities, could trigger alerts. Unexpected anomalous activities from administrative accounts, able to cause severe damage, were also detected. Finally, changes in the user token's user agent, such as a version downgrade, were noted as potential anomalies and could be investigated through audit logs.

To our best knowledge, all the commercial solutions mentioned solely relied on attributes and did not incorporate the graph topology of the team communication platform. Consequently, existing solutions could not detect graph structure-based anomalies, like **AV4** *Channel Sniffing*. However, since all the mentioned solutions were closed source, it is impossible to verify this assumption.

We believe a more holistic approach, incorporating the platform’s network structure, can detect further attacks and anomalies.

4 Dataset

This chapter begins with an evaluation of a freely available Slack [19] dataset. Subsequently, a novel dataset derived from the team communication platform used within the "Entrepreneurial Masterclass" [68] organization at the Technical University of Munich (TUM) is introduced. In contrast to existing datasets, the Masterclass dataset offers a unique opportunity for researching unprocessed message data and can be extensively examined for suspicious and anomalous activities.

The raw data from the "Entrepreneurial Masterclass" communication platform must undergo a series of transformations to be used for anomaly-detecting graph neural networks. Therefore, the following sections describe the necessary steps and techniques to convert the raw data into a suitable format. In addition, it is outlined how the attack vectors described in Chapter 2 are synthesized within the dataset. During the transformation process of the raw communication data, key features have to be extracted and converted into a standardized graph representation. The advantages of the chosen graph representation are elaborated upon in the final section of this chapter.

4.1 Choosing the Best Dataset

In the following, we compare the in Chapter 3 introduced dataset *Software related Slack Chats with Disentangled Conversations* [52] with a novel dataset, derived from the "Entrepreneurial Masterclass" team communication platform. We investigate both datasets for their features and suitability for our research objectives.

4.1.1 Existing Datasets

As investigated in Chapter 3, only a limited number of team communication platforms were monitored and recorded for research purposes. Chatterjee et al. [52] published the dataset *Software related Slack Chats with Disentangled Conversations*, in which various programming-related Slack workspaces were monitored for a period of two years. The dataset comprises 38955 conversations from 12171 unique users. It is divided into four workspaces: clojurians (7918 conversations), elmlang (22172 conversations), pythondev (8887 conversations), and racket (about 1900 conversations). The dataset includes highly active message conversations, with some days recording thousands of sent messages (see Figure 4.1). However, it is

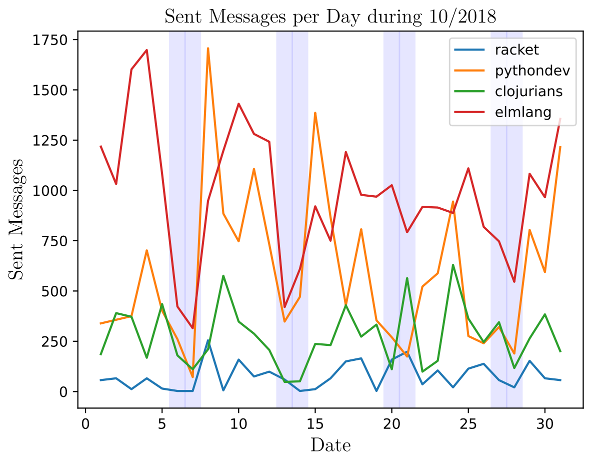


Figure 4.1: Message activity in of the Slack workspaces recorded by the paper *Software-related Slack Chats with Disentangled Conversations* [52]. Especially during weekends, indicated with blue background, less activity can be seen.

important to note that the dataset solely consists of messages from specifically selected public channels, excluding all communication outside these channels.

We further evaluate the suitability of this dataset for our research objectives by examining the communication patterns within the elmlang workspace. For this, we utiliz Gephi [69], a graph visualization tool, along with its implemented Fruchtermann-Reingold algorithm [70]. In order to apply the Fruchtermann-Reingold algorithm, we group the dataset messages into 30-minute intervals. Within each interval and for every message sent, we establish a weighted edge connecting the sender to all other active users during that same interval. Here, the weighted edge denotes an interaction between two users, with the weight increasing as the number of exchanged messages between them grows. By employing this technique across all intervals, we construct a graph that represents the communication patterns among all users in the workspace. Through the use of the Fruchtermann-Reingold algorithm, the graph undergoes a transformation: users with a greater number of weighted edges are drawn closer to the center of the graph, while users with only a few interactions can be found along the outer edge.
As can be seen in Figure 4.2, the computed interaction graph of the elmlang workspace contains many single edges toward its outer edge. This suggests that many users were active in the workspace for only a limited period of time, engaging in only a few message exchanges before leaving. Such behavior is more characteristic of workspaces accessible to the general public, rather than reflective of communication patterns within closed organizational team communication platforms.

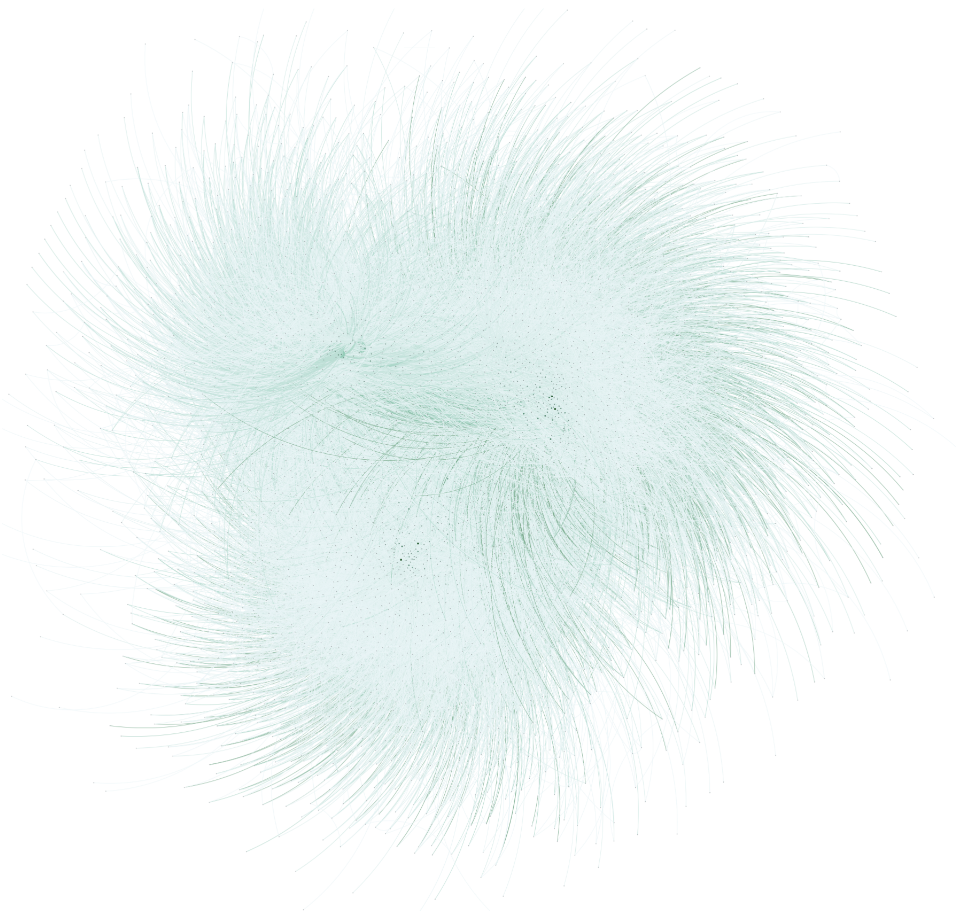


Figure 4.2: Communication cluster of the elmlang Slack workspace, recorded in the paper *Software-related Slack Chats with Disentangled Conversations* [52]. Many single edges toward the outer edge of the graph can be seen, indicating that a lot of users were only active for a limited number of time. This behavior does not represent the communication patterns of a closed organization's team communication platform.

4.1.2 Masterclass Dataset

The TUM Entrepreneurial Masterclass [68] is a program managed by the Technical University of Munich's UnternehmerTUM initiative [71]. It offers students passionate about entrepreneurial fields the opportunity to write their master's thesis about entrepreneurship-related research. The participants gain extensive integration into the entrepreneurial ecosystem at TUM and UnternehmerTUM. To effectively structure this program, an internal team communication platform (Slack) is used for organization and coordination within the masterclass.

Members of the Masterclass are divided into different task forces. Each task force is responsible for a unique area, such as marketing, technology, PR, or event organization. Additionally, task forces manage resources relevant to their function. For instance, the PR task force controls the Masterclass social media accounts, while the technology task force oversees the website and server organization.

Certain resources, such as social media account login credentials or WordPress website credentials, hold a higher security relevance. Therefore, Masterclass members are strongly advised to handle such sensitive information carefully and exercise discretion.

With the authorization of the Masterclass, this research monitored the organization's internal communication for three years, gathering data to construct a dataset that can be utilized for training machine learning models to identify potential internal threats or anomalies. Although no active threats were identified during the monitored period, sensitive information, that members were unaware of, was detected in the communication history. Overall, the dataset offers valuable insights into the communication dynamics of a thriving organization.

4.1.3 Datasets in Numbers

The unprocessed Masterclass dataset contains the activities of 234 users, each of whom is characterized by specific features such as their *timezone*, *team affiliation*, or *app usage*. As we were able to gain access to data provided by a Slack Pro [72] plan, the scope of user-specific features was expanded with more activity logs, including the *total messages* each user has posted, as well as access logs detailing the *device used*, *IP address*, and *time of access*.
The total count of conversations posted in public channels adds up to *3975*, collected from across a total of *39* different channels. The user with the highest activity level sent a total of *7620* messages throughout the recorded period, while the most active user within the last 30 days of recording contributed a total of *313* messages. On average, *78* messages were sent per week. The daily distribution of messages demonstrates a predominant pattern of messages sent on weekdays, as shown in Figure 4.3.

In Table 4.1, we compare the dataset "Software related Slack Chats with Disentangled Conversations" by Chatterjee et al. [52] examined in the previous Section 4.1.1 to the Masterclass dataset. While Chatterjee et al. monitored more conversations across 4 different workspaces, they only published a few selected channels. Additionally, no specific User features are avail-

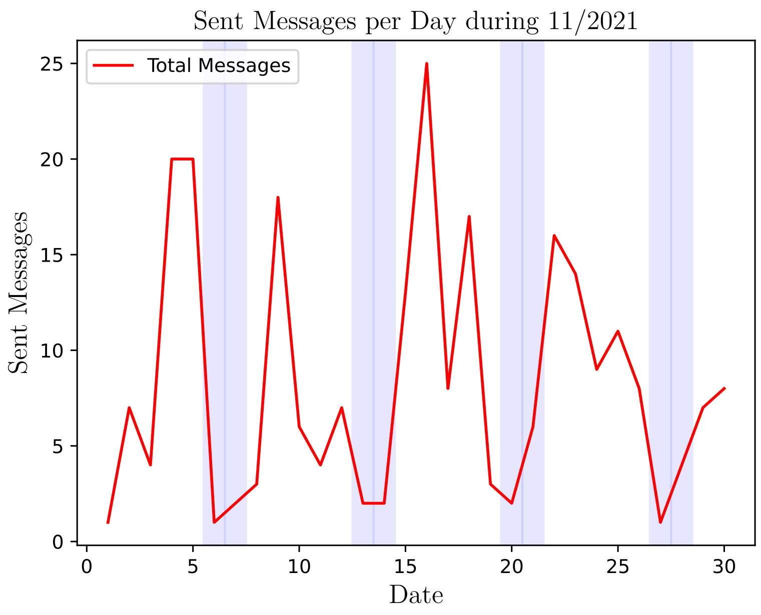


Figure 4.3: Total number of sent messages per day within the Masterclass dataset during November 2021. Most activity occurred on weekdays, with a peak on Wednesday. During weekends, indicated with blue background, less activity was observed.

able, whereas we can utilize 17 different features for each user from the Masterclass dataset. It is impossible to create a fully connected graph from Chatterjee et al., representing the entire workspace, as we cannot access non-published channels. Therefore, it is not possible to detect anomalies that occur across multiple channels. This is not a problem for the Masterclass dataset, as we can access all public channels and user data.

The limited availability of user-specific features, the absence of recorded non-published channels, and the observed user behavior of never returning users makes Chatterjee et al.'s dataset unsuitable for representing a closed organization's team communication platform. Our objective to detect the attack vectors described in Section 2.2 involves altering user features and manipulating multiple channels. Consequently, we decide to solely use the Masterclass dataset, as it resembles a closed organization's team communication platform, provides more detailed resources, and includes user features and information about all public channels.

Feature	**Masterclass [68]**	**Chatterjee et al. [52]**
Number of Workspaces	1	4
Number of Users	234	elmlang: 6454 clojurians: 2422 pythondev: 3295 racket: Unknown
Number of Teams	4	Unknown
Plan	Slack Pro [72]	Unknown
Number of Different Channels	39	elmlang: 2 clojurians: 1 pythondev: 1 racket: 1
Total Public Channel Conversations	3975	elmlang: 22172 clojurians: 7918 pythondev: 8887 racket: about 1900
Most Sent Messages by a User	7620	18498
Average Conversations per Week	78	374
Total Conversations	3975	38955
Accessible User Features	name timezone team title status text email is_admin is_bot is_app_user updated email_confirmed deleted country joined channels location IP address	None
Accessible Message Features	timestamp text is_reply replying users	timestamp text

Table 4.1: Comparison between the Masterclass dataset and the "Software-related Slack Chats with Disentangled Conversations" dataset, published by Chatterjee et al. [52]. While Chatterjee et al. published more conversations and included different workspaces, they only monitored a handful of selected channels. Additionally, no user-specific features are provided, which can be leveraged for anomaly detection.

4.2 Data Collection

The following step-by-step guide is taken from the official Slack documentation [73] and is used to export the raw data from the Masterclass Slack workspace:

```
Workspace Owners and Admins can export data from a workspace using the steps below:

    1. From your desktop, click your workspace name in the top left.
    2. Select Settings & administration from the menu, then click Workspace settings.
    3. Click Import/Export Data in the top right.
    4. Select the Export tab.
    5. Below Export date range, open the drop-down menu to select an option.
    6. Click Start Export. We'll send you an email once your export file is ready.
    7. Open the email and click Visit your workspace's export page.
    8. Click Ready for download to access the zip file.
```

The received zip file contains the raw workspace's message history in JSON format and file links from all public channels. Depending on the paid subscription to Slack, the data quality varies. For this thesis, all data from the Masterclass Slack channel is accessed with the *Slack Pro* [72] subscription. Additional data, such as user-specific activity logs, are accessed using the Slack API [74] and are manually added to the unzipped Slack data directory. For this task, an API crawler was written that can be found within this thesis' code implementation.

4.3 Data Pre-Processing

As mentioned in the previous section (Section 4.2), the message history of the received workspace is stored in JSON format. These files are organized by channel and placed in their respective directories. Within each channel directory, the message history is further divided into multiple files, each containing messages from a specific day. Each recorded message is assigned a unique *ID*, a *timestamp*, a *sender*, and a *text* field. Figure 9.1 illustrates an example of a message log. It's worth noting that responses, threads, and reactions are also stored within the message object, but they are not analyzed in detail in this thesis.
In addition to the message-related log entries, the exported files from Slack also include user-specific data, which can be found in the "users.json" file. This file provides information such as the users' *names*, *emails*, unique *user IDs*, *teams*, *devices*, and *privileges*. The user ID is used throughout the workspace to identify individuals within the message history. A final list of channels is stored in the file *channels.json*. This file contains the *name*, *id*, and the *user ids of all members* for each channel in the exported workspace.

For further research, it is necessary to normalize and preprocess the message history. This involves removing HTML links, emojis, and user-specific attributes. Since the workspace contains non-English messages, a few of them must be translated into English. To accomplish this, we leverage OpenAI's Complete API [75].

Compared to other translation services, we encounter no rate limits in API calls. Additionally, we implement a cache to avoid unnecessary API calls and reduce costs.

The preprocessed messages are stored in a pandas [76] dataframe. Table 4.2 displays the final message dataframe and its columns.
A similar dataframe is created for users in the workspace. The dataframe is created from the raw JSON but also incorporates new features computed from the sent messages. The final user dataframe columns can be seen in Table 4.3.
Lastly, a third dataframe was created to keep track of all channels within a workspace. The columns of the dataframe are shown in Table 4.4.

Message Feature	**Description**
id	The id of the message
ts	The timestamp of the message
sender	The user id of the message sender
receivers	The user id of the message receivers
message	The original message text
replying_to	An optional id of the message the current message is replying to
preprocessed	The pre-processed message text
embedding	An embedding representation of the preprocessed message text
label	Label indicating if the message is malicious or not

Table 4.2: Columns of the Message Dataframe. **Bold** indicated features are computed and cannot be found in the original JSON files.

User Feature	**Description**
user_id	The user id
name	The user name
tz	The user's timezone
team	The user's assigned team
title	The user's role in the organization
status_text	The user's status (e.g. "In a meeting", "On vacation")
email	The user email
is_admin	Whether the user is an admin
is_bot	Whether the user is a bot
is_app_user	Whether the user has the Slack app installed
updated	The timestamp of the last update
is_email_confirmed	Whether the user's email is confirmed
deleted	Whether the user is deleted
country	The user's country
channels	The channels the user is a member of
email_domain	The user's email domain
sent_message_count	The total number of messages the user sent
received_message_count	The total number of messages the user received
early_message_count	The number of messages the user sent between 6 am and 12 pm
morning_message_count	The number of messages the user sent between 12 pm and 6 pm
afternoon_message_count	The number of messages the user sent between 6 pm and 12 am
night_message_count	The number of messages the user sent between 12 am and 6 am

Table 4.3: Columns of the User Dataframe. **Bold** indicated features are computed and cannot be found in the original JSON files.

Channel Feature	**Description**
name	The channel name
description	The channel description
created	The timestamp of the channel creation
total_membership	The total number of members
messages_posted	The total number of messages posted
members_who_posted	The number of members who posted at least one message
members_who_viewed	The number of members who viewed at least one message
label	Label indicating if the channel is malicious or not

Table 4.4: Columns of the Channel Dataframe. All features are taken from JSON files.

4.4 Synthesized Attacks

As described in Section 2.2, four attack vectors are identified, that were within breaches of team communication platforms. As these attacks did not happen on the internal Masterclass communication platforms, we synthesize them programmatically by modifying the previously generated dataframes for users, messages, and channels.

1. To simulate *lateral neighborhood phishing* attacks **AV1** in team communication platforms, a group of active users, randomly selected, sends phishing messages to their common channels. This is accomplished by augmenting the message dataframes with new messages.
2. To synthesize *power user targeting* **AV2**, a batch of randomly selected users send messages to previously unaffiliated channels. This is also achieved by extending the message dataframe with new messages.
3. To simulate *impersonation attacks* **AV3**, several randomly chosen users have their name, country, and status replaced with values from different users. This is achieved by modifying the user dataframe.
4. The last attack vector *channel sniffing* **AV4** is simulated by adding selected users to multiple channels they are not members of and have not interacted with before. This is done by extending the channel dataframe with new users.

Additionally, we create a list of exemplary phishing messages in the same style and format as regular Slack messages that are used for the mentioned synthesized phishing attacks. These messages are not directly associated with the Masterclass resources. Instead, they are written generally so that they can also be used for other team communication platforms. We use OpenAI's ChatGPT [77] to generate new messages based on previously human-written messages to achieve many unique messages. In total, 260 unique messages are created, of which about 50 are hand-crafted and 210 AI-generated.

4.5 Message Embeddings

To generate features from messages exchanged among users on a team communication platform, we convert all processed messages into a fixed-size embedding representation [78]. Various methods exist for generating text embeddings for this task. As outlined by Dou et al. [44], many research papers use pre-trained word2vec [78] or Bert [79] models. However, with the emergence of large language models [80], it is also possible to obtain cutting-edge embeddings through API calls to external services [81].
For our initial approach, we employ Bert-as-a-service [82] to generate a fixed vector of 512 features as our message embeddings. This method is used effectively by previous work [44] to embed Twitter messages and identify fake news. As described in the paper, Bert-as-a-service leverages a pre-trained BERT model to encode semantic similarities among different sent

messages. However, upon closer examination, we discover that there are no significant differences in the cosine similarity (Eq. 4.1) between maliciously labeled messages and regular messages.

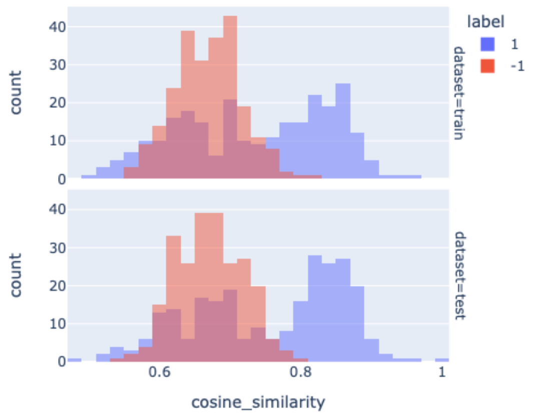
The cosine similarity is a commonly used measure for determining the similarity between feature vectors. Given two feature vectors, **A** and **B**, the cosine similarity is computed as follows:

$$\text{cosine similarity} = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\| \|\vec{B}\|} \tag{4.1}$$

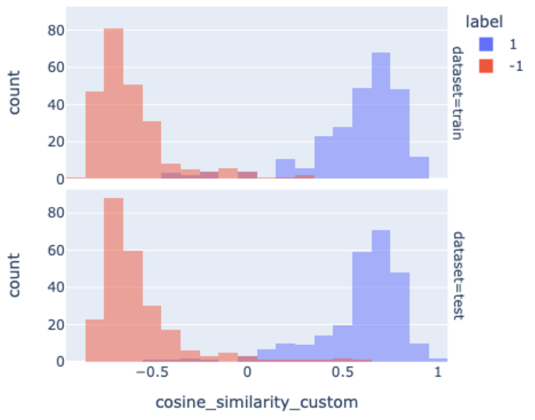
Therefore, we replace Bert-as-a-service with the OpenAI embeddings API endpoint [81]. In this case, we utilize the *babbage-similarity* engine to generate our embeddings. As demonstrated by [83], the endpoint exhibits a strong performance compared to other embedding providers in a similarity search problem. The OpenAI endpoint transforms each message into a fixed vector consisting of 2048 features. To improve the cosine similarity between flagged anomalous messages and reduce their similarity to regular messages, we further customize the embeddings for our dataset.

OpenAI provides a notebook [84] that demonstrates a method to tailor embeddings to specific classification tasks, using training data consisting of manually labeled text pairs according to their similarity. In our context, these text pairs are malicious-regular, malicious-malicious, or regular-regular messages. As a result, an optimized matrix is generated using a process of iterative optimization through gradient descent.
Initially, a random matrix is generated with dimensions matching the size of the embeddings. Then, in each training epoch, a simple Neural Network uses this matrix to project the embeddings of two text sets, in our case regular messages and malicious messages. The projected embeddings are compared using cosine similarity, which serves as the predicted similarity score. To quantify the disparity between the predicted score and the actual human-assigned similarity score, a Mean Squared Error (MSE) loss function is employed. The resulting loss is utilized to calculate the gradient, representing the rate of change of the loss in the matrix. The matrix is then updated in the opposite direction of this gradient, effectively reducing the loss. This process of prediction, loss calculation, gradient computation, and matrix update is repeated for 30 epochs or until the matrix achieves an acceptable level of accuracy. Consequently, the resulting matrix can be multiplied with the original embeddings from our messages, thereby reducing error rates when classifying message pairs as similar or dissimilar.

Figure 4.4 shows the cosine similarity between the raw and optimized embedding vectors for messages from our dataset. As a result of the optimization process, two distinct clusters emerge: one represents identical types of message pairs (malicious - malicious / regular - regular), labeled as 1, and the other corresponds to different types of message pairs (malicious-regular), labeled as -1.



(a) Cosine similarity between message pairs without applied optimization matrix. The similarity scores for the two message pairs overlap, causing the clusters to be indistinguishable.



(b) Cosine similarity of message embeddings after applied optimization matrix. Two distinct clusters can be seen after the optimization process.

Figure 4.4: Cluster 1 comprises pairs of identical message types, either *anomalous-anomalous* or *regular-regular*. In contrast, Cluster -1 encapsulates pairs of different message types, *anomalous-regular*. Optimizing embeddings has a clear effect in distinguishing the two types of message pairs.

4.6 Data Representation

As stated in Chapter 3, graph neural networks were previously utilized for anomaly detection in social networks, surpassing heuristic-based approaches or feature-based DL approaches that do not incorporate graph structure. In order to employ a Graph Neural Network on our dataset, it is necessary to transform the data into a graph representation. This section will outline different graph representations and libraries that can be utilized to represent our data.

4.6.1 Graph Dataset Libraries

Multiple libraries exist to represent and process graph data programmatically in Python [85], our preferred programming language. To this date, the most popular ones are:

- NetworkX [86], initially developed in 2002, is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.
- Deep Graph Library (DGL) [87], which is a framework-independent library and can be used with Pytorch [88], MXNet [89], and Tensorflow [90].

- Pytorch Geometric [91], which is a geometric deep learning extension library for Pytorch [88].

Although NetworkX is a widely used library for graph processing in Python, it is not intended for utilization in the realm of Graph Machine Learning. PyTorch Geometric and DGL, on the other hand, are both optimized for graph machine learning tasks. However, PyTorch Geometric is more actively developed in the field of graph anomaly detection. Hence, we opted to employ a PyTorch Geometric dataset for our graph representation.

4.6.2 Knowledge Graphs

Three types of knowledge graph implementations exist within PyTorch Geometric, which can represent data in different graph structures [92]. The most commonly used representation is the **Homogeneous Attributed Graph**. In this graph, all nodes and edges are of the same type. There is only one shared feature space, *graph.x*, for all nodes, and a single *graph.edge_index* matrix to represent all edges between nodes.
Then, there is the **Heterogeneous Attributed Graph**. Here, nodes and edges can be of different types. The node feature space is divided, and each node type has its own matrix representation, *graph[node_type].x*. Edges between different types of nodes are represented as *graph[nodetype1_relation_nodetype2].edge_index*, also using separate matrices.
Finally, PyTorch Geometric also supports **Temporal Dynamic Graphs** [93]. This type of graph is used to represent data that change over time.

4.6.3 Final Graph Design

During our initial experimentation stage, we attempted to represent the Slack dataset as a homogeneous attributed graph. In this representation, users are represented as nodes, and the edges denoted message interactions between those users. The user features are stored directly within the nodes, while the message features are associated with the attributed edges.

However, we soon discovered that this design approach is not suitable for the task of anomaly detection. As mentioned by Kim et al. [37] and Ma et al. [10], much current research and frameworks for GNN-based anomaly detection do not take into account edge features. The most recent paper to our knowledge that incorporates edge weights for anomaly prediction on graph networks is Shah et al. [94], which was published seven years ago and does not consider recent advances in the field of GNNs.

For this reason, we adopt a Heterogeneous Attributed Graph structure to represent our Masterclass dataset. In this revised approach, users, messages, and channels are all represented as nodes, while interactions such as *user-sends-message*, *message-sentTo-channel*, and *message-sentTo-user* are represented as non-attributed edges.
Compared to temporal or dynamic graph designs, heterogeneous graphs also offer better support for anomaly detection, as noted in the comprehensive study by Ma et al. [10]. Cur-

rently, there is still a significant amount of research to be conducted in the field of anomaly detection on dynamic and temporal graphs.

To initialize the graph using PyTorch Geometric's "HeterogeneousGraph" interface, the following steps are taken:

```
data = HeteroData()
data['user'].x = padding_user_x(user_x)
data['user'].y = user_y
data['message'].x = message_x
data['message'].y = message_y
data['channel'].x = padding_channel_x(channel_x)
data['channel'].y = channel_y
data['user', 'sends', 'message'].edge_index = user_sends_message
data['message', 'sentTo', 'channel'].edge_index = message_sentTo_channel
```

Note that a custom padding is applied to the user and channel feature vectors. This ensures that all feature vectors of each node type have the same dimensionality.

The final node structure of the heterogeneous graph can be observed in Figure 4.5. Each color represents a distinct node type. The edges between the nodes are non-attributed. Figure 4.6 illustrates how the graph representation displays all messages to the selected channel *emc_amlumni* during March 2023.

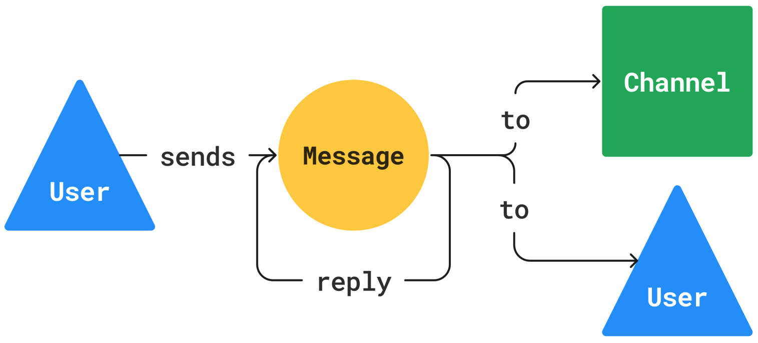


Figure 4.5: Node structure and relations of the heterogeneous knowledge graph. Different colors and shapes represent node types. Edges between the nodes are non-attributed and do not carry any features or weights.

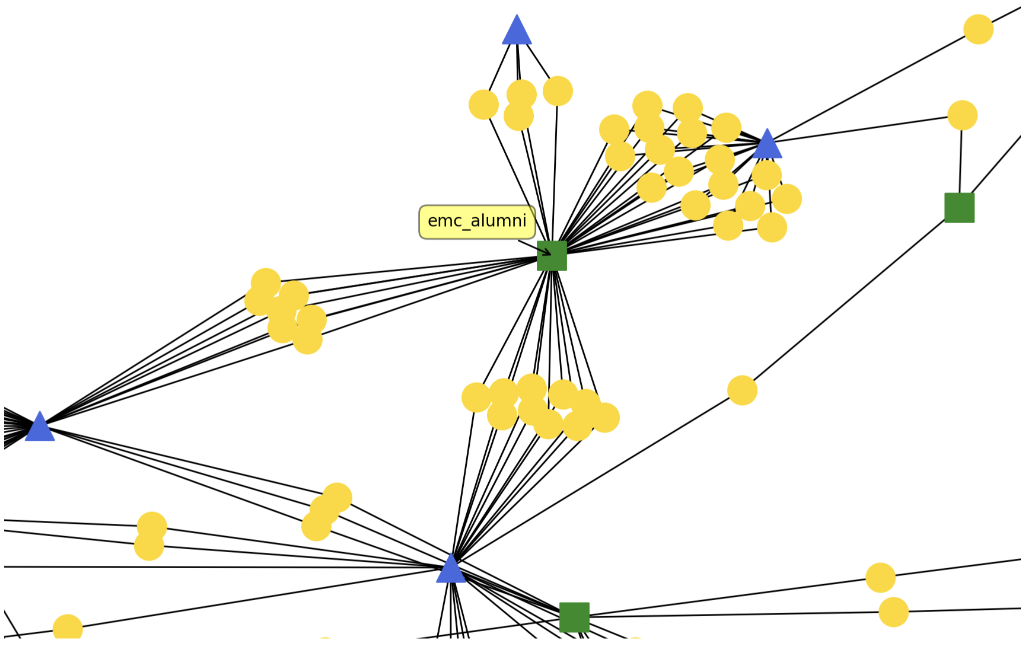


Figure 4.6: Snapshot taken of the final Masterclass knowledge graph, showing all conversations between users in the channel *emc_amlumni* within March 2023. Node types are represented by different colors and shapes.

4.7 Final Node Features

Table 4.5 showcases the final node feature list of our dataset. It is important to note that not all features from the previously generated dataframes are included in the graph representation. This includes user-specific data, such as emails, and other features irrelevant for predicting anomalies.

4.8 Dataset Creation Flow

In Figure 4.7, we present the final workflow for creating the Masterclass dataset. This workflow showcases the process of loading data from the original Slack export files and transforming it into a graph representation. Additionally, it outlines all intermediate processes, including embedding creation and attack injection.

Node Type	**Feature**
User Node	timezone
	country
	team
	title
	is_admin
	is_bot
	is_app_user
	updated
	is_email_confirmed
	number of joined channels
	number messages written in 0:00-6:00
	number messages written in 6:00-12:00
	number messages written in 12:00-18:00
	number messages written in 18:00-24:00
Message Node	text embedding
Channel Node	total members
	messages posted
	number members that posted
	number members who viewed

Table 4.5: Final node features used in the graph representation of the Masterclass dataset. Private user data, such as emails and irrelevant features for anomaly detection are excluded from the graph.

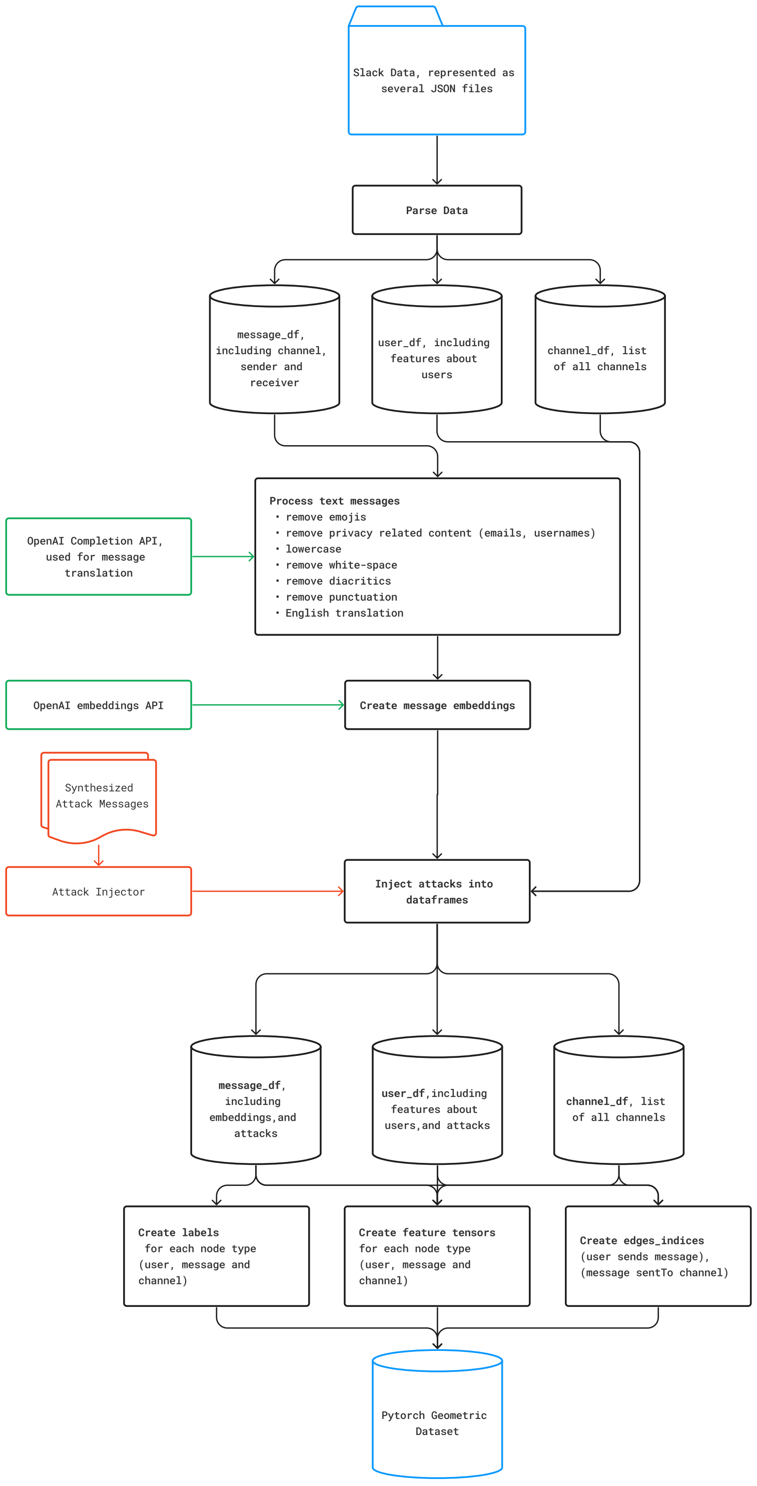

Slack Data, represented as several JSON files
Parse Data
message_df, including channel, sender and receiver
user_df, including features about users
channel_df, list of all channels
OpenAI Completion API, used for message translation
Process text messages
• remove emojis
• remove privacy related content (emails, usernames)
• lowercase
• remove white-space
• remove diacritics
• remove punctuation
• English translation
OpenAI embeddings API
Create message embeddings
Synthesized Attack Messages
Attack Injector
Inject attacks into dataframes
message_df, including embeddings, and attacks
user_df, including features about users, and attacks
channel_df, list of all channels
Create labels for each node type (user, message and channel)
Create feature tensors for each node type (user, message and channel)
Create edges_indices (user sends message), (message sentTo channel)
Pytorch Geometric Dataset

Figure 4.7: Creation flow of the Masterclass dataset. The raw data is loaded from the exported JSON files and transformed into a graph representation.

5 ECONAD

CONAD [45], which is summarized in Chapter 3.3, is selected as the baseline model for our anomaly detection task. In particular, the self-supervised learning approach of CONAD, which incorporates human knowledge to enhance the graph, proves to be a promising method for detecting anomalies in team communication platforms.
In this chapter, we present our extended CONAD model, **ECONAD**, which is specifically designed to detect anomalies in team communication platforms. First, we expand the base CONAD model by incorporating augmentation strategies tailored to team communication platforms (Section 5.1). Subsequently, we introduce a multi-view graph approach (Section 5.2). We then describe the sampling strategies used to divide the graph into smaller batches in Section 5.3, as well as the validation process in Section 5.4. Finally, we outline the prediction and system design in Section 5.5.

5.1 Custom Augmentation Strategies

Not all predefined CONAD augmentation strategies, as described in Section 3.3.2, are directly applicable to team communication platforms. For example, CONAD's *high degree* strategy is based on the assumption that a node with a high number of edges is more likely to be anomalous. However, in the context of a team communication platform, this assumption is invalid, as it would classify every channel with many active users as anomalous. Nevertheless, Xu et al. [45] demonstrated that incorporating prior human knowledge about attack patterns has a significant and positive impact on anomaly detection in social networks. Therefore, we introduce our own team communication platform-specific augmentation strategies, which are custom-tailored to the attack vectors identified in Section 2.2. Similarly to the original augmentation strategies implemented within the base CONAD model, the new augmentation strategies can be categorized into structural and attribute augmentation strategies.

A1 **Neighborhood Phishing (Attribute)** With our first custom augmentation strategy, we aim to detect lateral phishing messages sent to a compromised user's direct common interaction channels. We replace a randomly selected message node embedding vector with a malicious message embedding from the training set used to fine-tune the message embeddings in Section 4.5. By doing so, we hope the model will learn to differentiate between normal and malicious messages based on the message's embedding vector. Since the original graph structure remains unchanged, the model can only incorporate knowledge through attributes.

A2 Poweruser Phishing (Attribute, Structural) Attackers have previously gained access to more security-critical resources by contacting power users within a team communication platform, such as IT administration [7]. Since the model cannot determine which user is responsible for what service in an organization, we attempt to detect this attack by introducing new malicious message events between a user and channels not in the user's direct neighborhood. Similarly to **A1**, we use malicious embeddings for the new message node. With the introduction of this augmentation, the model can incorporate knowledge through both attributes and structure.

A3 User Impersonation (Attribute) By impersonating a user, attackers can increase their credibility and raise less suspicion when requesting access to critical infrastructure. To detect this type of attack, we replace the features of randomly chosen user nodes with the features of another random user in the same workspace. Like augmentation strategy **A1**, the model can only incorporate knowledge through attributes.

A4 Channel Sniffing (Structural) Team communication platforms often contain sensitive information in public channels. If a user joins a large number of channels that are not related to their team, it may indicate that the user is looking for sensitive information unrelated to their actual work. To detect this behavior, we introduce new edges between users and several randomly chosen channels that the user was not previously affiliated. In this augmentation strategy, the model has to learn from the graph structure, as all node attributes remain unchanged.

5.2 Multi-View Graph

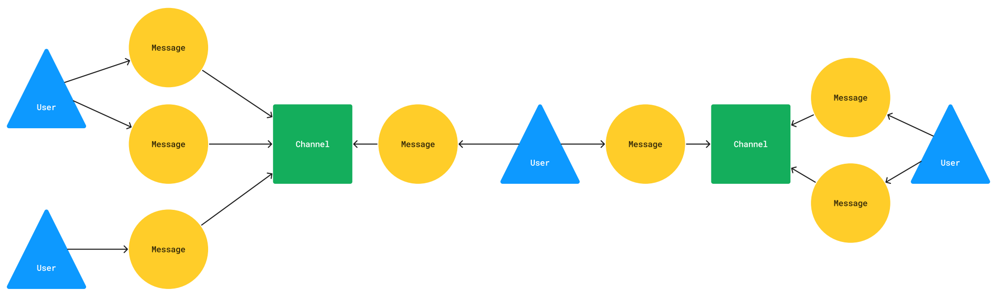
CONAD is designed to process homogeneous attributed graphs in which every node is of the same type and is assigned a feature vector with the same dimensionality. However, as described in Section 4.6.3, the graph obtained from our Masterclass dataset is heterogeneous, comprising three different node types: user nodes, message nodes, and channel nodes. To process this heterogeneous graph, we split it into multiple views based on the type of anomaly we aim to detect. Unlike the multi-view approach by Peng et al. [47] introduced in Section 3.2, we chose not to employ a specific GNN for each anomaly classification task, but to use CONAD with the extended augmentation methods introduced in the previous section. Similarly to Zhang et al. [46], we divide the graph into different views using predefined meta-paths denoted by $\mathcal{P}$.

A meta-path $\mathcal{P}$ is a path defined on the network schema $\mathcal{T}_{\mathcal{G}} = (\mathcal{A}, \mathcal{R})$, and is denoted in the form of $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} ... \xrightarrow{R_L} A_{L+1}$, which defines a composite relation $R = R_1 \cdot R_2 \cdot ... \cdot R_L$ between node types A_1 and A_{L+1}, where $\cdot$ denotes relation composition operator, and L is the length of $\mathcal{P}$ [46]. Given our heterogeneous network with different node types *user*, *message*, and *channel* and their relations, we define the following meta-paths to split the graph into two new views:

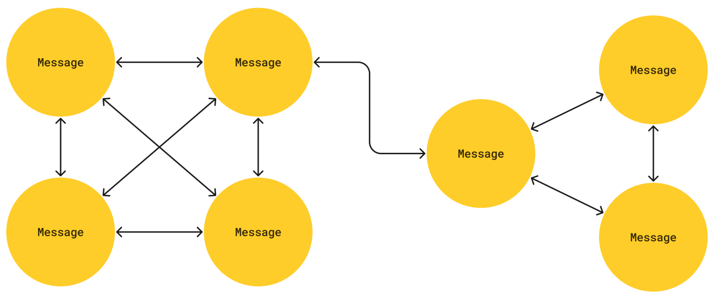
$\mathcal{P}_1$ **Message-User-Channel-User-Message** ($\mathcal{P}_1$): By removing all users and channels from the graph, we can detect anomalies based on the message's content and its relation to other messages. Since this graph view is homogeneous, we can apply CONAD's original augmentation strategies.

$\mathcal{P}_2$ **User-Message-Channel**: For augmentation **A3** and **A4**, we only use the channel and user nodes. By excluding messages, we can detect anomalies based on the user's behavior and its relation to other users and channels.

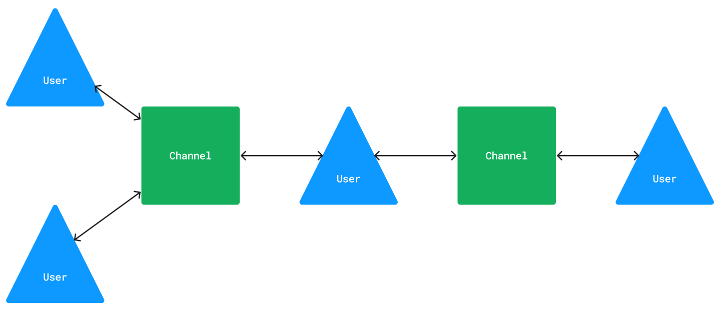
An example of how the graph is transformed by applying the meta-paths $\mathcal{P}_1$ and $\mathcal{P}_2$ can be seen in Figure 5.1.



Exemplary heterogeneous graph structure before the application of a meta-path $\mathcal{P}$. Node types are indicated by different colors and shapes.



$\mathcal{P}_1$ Message-User-Channel-User-Message view. After all users and channels are removed, messages of each channel connect to each other and form a complete cluster. Clusters are connected to other clusters when users are active in both channels. This view is applied for message augmentation strategies **A1** and **A2**.



$\mathcal{P}_2$ User-Message-Channel view. By removing all messages, only direct user-channel interactions are perceived. This view is employed for the custom augmentation methods **A3** and **A4**.

Figure 5.1: Graph transformation with meta-paths $\mathcal{P}_1$ and $\mathcal{P}_2$. After applying the meta-paths, the graph is split into two views, that can be used for different augmentation strategies.

5.3 Sampling Strategies

Our graph data set, derived from the Masterclass team communication platform, comprises 202 user nodes, 39 channel nodes, and 4132 message nodes collected over a 3-year period, as previously described in Section 4.1.3. To predict anomalies, we partition the graph into smaller batches, simulating various time periods such as days, weeks, or months that can be used to train the model.
Several approaches exist for dividing a graph into smaller subsets [95]. For our team communication platform graph, we employ multihop neighborhood sampling [59]. This method involves selecting a random source node v_i, and for each hop (representing the number of edges required to reach a neighboring node v_j), adding all nodes within that hop to the sample. Figure 5.2 illustrates a source node v_i and its 3-hop neighbors, which would be included in a batch. In our sampling strategy, we set the number of hops to 4 to ensure the inclusion of all three node types: messages, users, and channels.
In practice, multiple source nodes are chosen to create a single sample, ensuring that the source nodes represent different types. The number of source nodes is determined by the *batch size* parameter. In Chapter 6, we evaluate how the batch size parameter influences our final prediction.

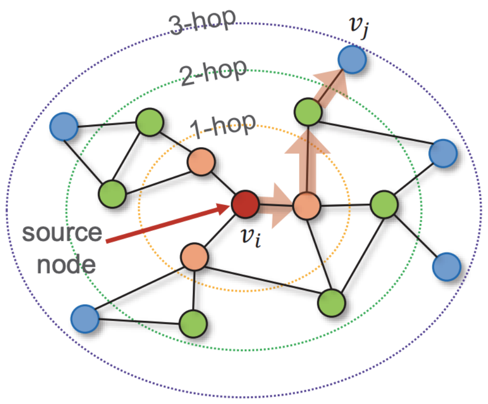


Figure 5.2: Multihop neighborhood sampling. The red node v_i is taken as source node. Nodes of different colors represent the sampled nodes at k-hops. Here v_j is one of the 3-hop neighbors of v_i. The orange arrows mark the shortest path ranging from v_i to v_j. Graphic and description taken from Xu et al. [96]

5.4 Validation

In the original implementation of the CONAD fit function, the model is trained for a fixed number of epochs without considering the performance on a separate validation set. To improve the training process and avoid overfitting, we add validation and an early stopping logic to the original implementation.

1. **Data loader for the validation set:**
   The model's training function is extended to process not only a training graph G but also an optional validation graph *val_G*.
   If a validation graph is provided, separate batches are created by using neighborhood sampling. The average loss from all batches is used to evaluate the model's performance after each training epoch.

2. **Validation loss computation:**
   After each training epoch, the model is evaluated on the validation graph using the *val_loader*. The loss on the validation graph is computed using the same loss function as used for training, and averaged across all batches of the validation graph.

3. **Early stopping based on validation performance:**
   To prevent overfitting and improve the training process, an early stopping mechanism is implemented, based on the model performance on the validation graph. Using the *patience* parameter, the number of consecutive epochs with no improvement in validation loss can be set before training is stopped by the early stopping mechanism. By default, we set *patience* to 50 epochs.
   During the training, the best-obtained validation loss and the number of epochs with no improvement are tracked. If the number of consecutive epochs with no improvement in the validation loss exceeds the value of *patience*, training is stopped early.

4. **Saving the best model state:**
   As training progresses, the model state with the best validation loss is saved. After the training is completed, either by reaching the maximum number of epochs or triggered by early stopping, the state of the model with the best performance on the validation graph is returned.

By incorporating these changes into the base CONAD training algorithm, the modified fit function now provides a more robust training procedure. The early stopping mechanism allows the model training to be more computationally efficient, as it will terminate the training early if it does not observe any improvement in the validation loss for a specified number of epochs. Furthermore, this approach also mitigates the risk of overfitting by closely monitoring the validation performance throughout the training process.

5.5 Prediction

To predict anomalies in the original graph, we initially divide the graph into multiple views, as described in Section 5.2. Then, each view undergoes processing by a distinct trained ECONAD model, which uses custom augmentation strategies tailored specifically to that view. To generate the final prediction for the original graph, we project the predictions of each view onto the original graph, as illustrated in Figure 5.3. If a node is labeled as malicious in the $\mathcal{P}_1$ message view, it will also be labeled in the original graph.

To visually represent the identified anomalous nodes, we color the nodes in the provided graph according to their predicted label. Anomalously labeled nodes are colored yellow, while normal nodes are colored purple. Channel nodes are colored green, as the dataset does not include any malicious channels. Hovering over nodes reveals additional detailed information. Node types can be distinguished by shape: user nodes are displayed as triangles, message nodes as circles, and channel nodes as squares.
Figure 5.4 showcases exemplary predictions for different types of anomalies on the original graph. The final design of the ECONAD model can be observed in Figure 5.5.

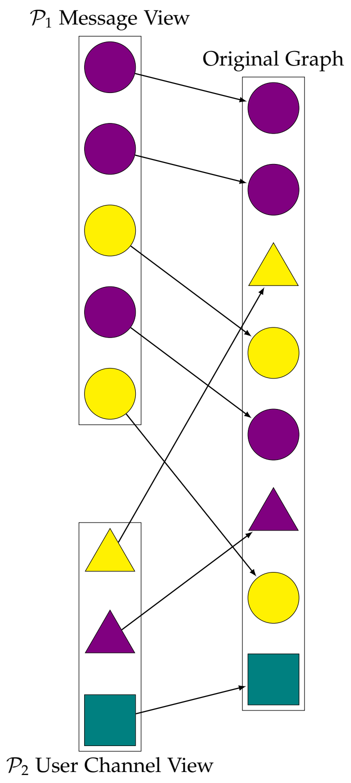
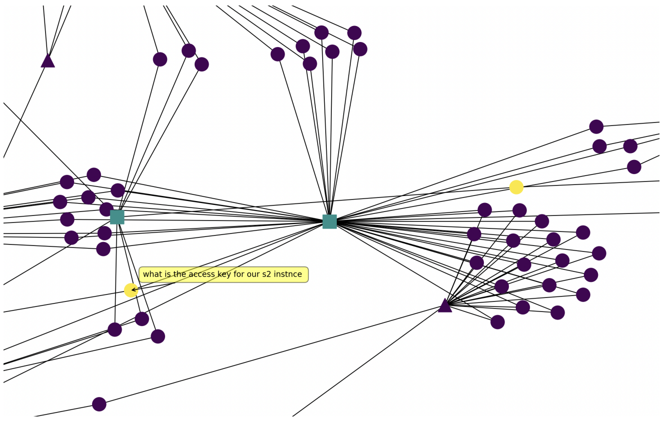
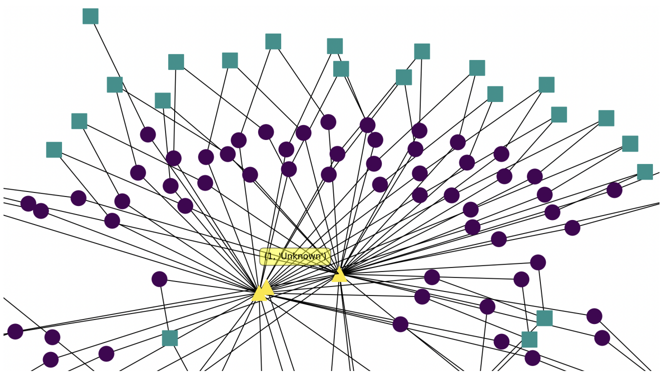

$\mathcal{P}_1$ Message View
Original Graph
$\mathcal{P}_2$ User Channel View

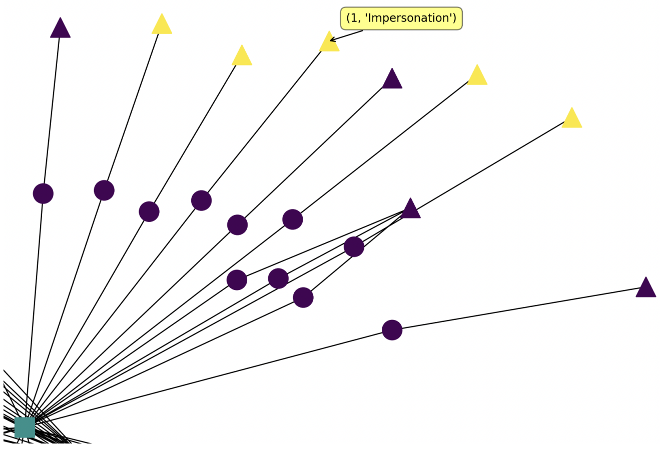
Figure 5.3: Node predictions for each view are projected onto the original graph to obtain the final prediction. If a message node is predicted as anomalous in the $\mathcal{P}_1$ message view, it will be colored yellow in the final prediction on the original graph.



Found *Lateral Phishing* attacks within the graph. In this case, anomalous message nodes are highlighted in yellow.



Found *Channel Sniffing* attacks. Anomalous user nodes are highlighted in yellow. Multiple edges connecting to various channels can be seen in this part of the graph.



Found *User Impersonation* attacks. All yellow highlighted users have interacted with the same channel and share the same attributes.

Figure 5.4: Multiple types of attacks found within the Masterclass dataset. Anomalous nodes are highlighted in yellow, normal nodes in purple. As all channels are non-malicious in the dataset, they are colored in green.

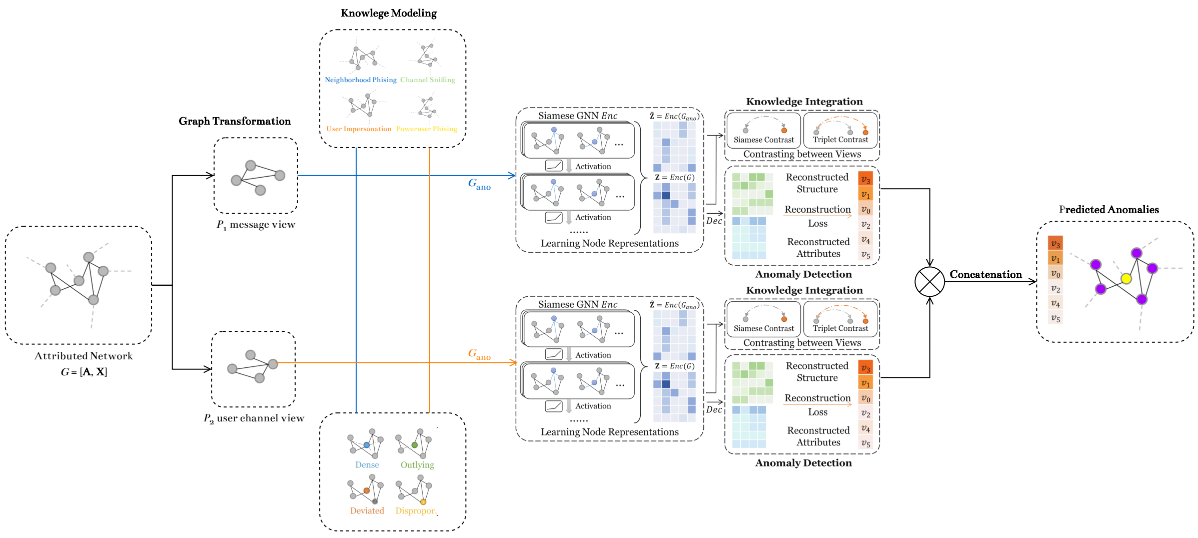


Figure 5.5: Final ECONAD model design. The graph is split into two views, based on the meta-paths $\mathcal{P}_1$ and $\mathcal{P}_2$. Each view is processed by a CONAD model, that combines team communication tailored data augmentation strategies with base CONAD augmentation strategies. The final prediction is created by projecting the predictions of each node in a single view to the original graph.

6 Experiments

This chapter describes the experiments we conducted to evaluate our proposed anomaly detection model, ECONAD. To evaluate the advantage of GNNs compared to traditional feature-based deep learning approaches, we first test the impact of graph structure and node attributes on the in Section 5.1 introduced custom augmentation strategies. To determine the optimal batch size, which represents different periods of messages sent within a workspace, we analyze the impact of various batch sizes on each augmentation strategy. As described in Section 6.7, the workspace graph is divided into multiple views to accommodate the heterogeneous structure of the graph for our introduced augmentation strategies. Hence, we evaluate the impact of each transformation. Finally, we compare our model with other graph anomaly detection models implemented in the popular graph outlier detection library pyGOD [97]. In particular, we closely examine other autoencoder-based GNN anomaly detection models, as they share similar system designs and capabilities.

6.1 Setup

We partition the graph into three subsets for the experimental setup of the base CONAD and our ECONAD model: training, validation, and test set. The training set consists of 23 months of data, while the validation set and the test set span 2 months and 1 month, respectively. These subsets are arranged in a timewise order, with the training set containing the oldest messages and the test set containing the most recent messages. To break down the training and validation graph into smaller batches, we employ neighborhood sampling with 4 hops, as described in Section 5.3. An outlier score margin of 0.2 is established, above which each node is classified as anomalous. For our augmentation strategies, we choose a rate of 0.5, which, as in the base CONAD model, determines the number of manipulated nodes for each applied augmentation strategy. The hidden dimension within the models is set to match the size of our node feature vectors, which is 2051. Furthermore, we choose a learning rate (lr) of 1e-3, as it is the default value for the base CONAD model.

6.2 Metrics

For the next sections, we employ the following metrics [98], to evaluate the performance of our model:

- **Accuracy:** The first metric represents the fraction of correct predictions made by our model. In binary classification, accuracy can be calculated as $(TP + TN)/(TP + TN + FP + FN)$, where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.
- **Precision:** Precision measures the fraction of relevant instances among the retrieved instances. In other words, it quantifies how many True Positives our model predicted out of all the positive predictions made. The precision is calculated as $TP/(TP + FP)$.
- **Recall:** Recall represents the fraction of relevant instances that were correctly retrieved out of the total number of instances. It measures how well our model captures the anomalies by labeling them as positive. The recall is calculated as $TP/(TP + FN)$. Especially for the task of anomaly detection, we are interested in a high recall value, indicating that most anomalies were found during classification.
- **Average Precision:** Average Precision (AP) is a performance metric used to evaluate the quality of a classification model. It calculates the average precision score for each class (anomalous, non-anomalous) individually and then takes the average across all classes, giving equal weight to each class. This metric is useful when there is class imbalance in the data, as it provides a balanced assessment of the model's performance across all classes. A high AP score indicates that the model performs well in correctly identifying and classifying positive instances with minimal false positives.
- **F1 Score:** The F1 score is the harmonic mean of precision and recall. While the regular mean treats all values equally, the harmonic mean assigns more weight to low values. Consequently, a high F1 score can only be achieved if both the recall and the precision are high. The F1 score is calculated as $2 * (\text{Recall} * \text{Precision})/(\text{Recall} + \text{Precision})$.
- **AUC (Area Under The Curve):** AUC represents the probability that a randomly selected node (anomalous or non-anomalous) will get a higher anomaly score than a randomly selected non-anomalous node. A higher AUC score indicates better model performance, with a perfect score of 1.

These metrics provide distinct perspectives on the model's performance, and it is crucial to take all of them into account, since the model may excel in one metric while performing poorly in another.

6.3 $\mathcal{P}_1$ Message View

In order to determine the optimal hyperparameters for our ECONAD model on the $\mathcal{P}_1$ message view, we first test the impact of structure and attribute on the classification. Subsequently, we carefully examine an appropriate batch size. Finally, we identify the most effective augmentation strategies for this view and assess their impact on the model's classification performance.

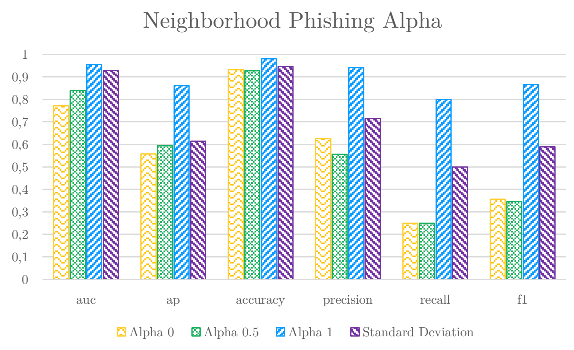
6.3.1 Impact of Structure and Attribute

In Section 3.3, we discussed how the base CONAD model and ECONAD learn from both the graph structure and the node attributes. To assess the impact of these two factors, we can adjust the weighting parameter α in the loss function, setting it to a value between 0 and 1. The loss function is defined as follows:

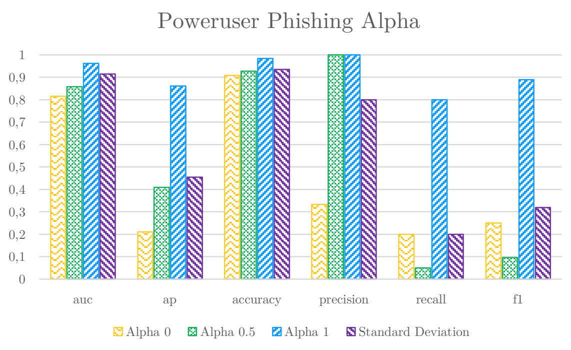
$$\mathcal{L}^{\text{recon}} = \alpha \cdot \text{attribute\_errors} + (1-\alpha) \cdot \text{structure\_errors}$$

To investigate the influence of node attributes and graph structure on our ECONAD model in the context of the $\mathcal{P}_1$ view and our custom augmentation strategies, we conduct experiments for each message augmentation strategy (**A1** and **A2**) using different α values. Additionally, we test the base CONAD model's approach of determining a meaningful α value based on standard deviation. Each experiment utilizes a batch size of 30 nodes, and no other augmentation strategies are enabled apart from **A1** and **A2**.

Figure 6.1 shows that a higher α value leads to a better classification performance for both augmentation strategies. In contrast to neighborhood phishing, the poweruser augmentation also modifies the structure of the graph by inserting new edges to unrelated message clusters. Therefore, we hypothesized that a lower α value would have a greater impact on the overall performance of this strategy. In fact, as shown in Figure 6.1b, the precision metric remains



(a) Impact of graph structure (low α) and node attributes (high α) on the neighborhood phishing augmentation strategy **A1**. The overall best prediction score is achieved using a high α value.



(b) Impact of structure (low α) and node attributes (high α) on the poweruser phishing augmentation strategy **A2**. As in the case of strategy **A1**, the best overall performance is achieved with a high α value. Especially the recall metric is most affected by the α value.

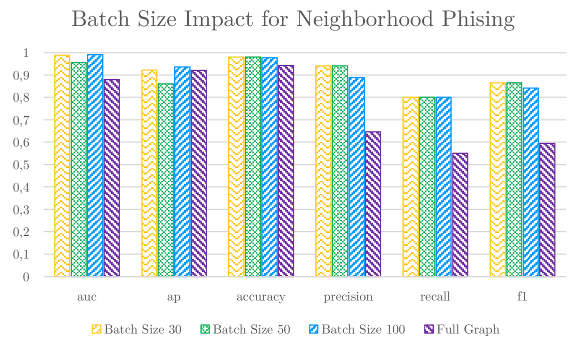
Figure 6.1: Classification metrics of single applied message augmentation strategies with a different focus on graph structure or node attributes. All experiments are conducted with a batch size of 30 nodes and no other augmentation strategies enabled. The standard deviation, used by CONAD as the default strategy to determine the α value, only produces moderate metrics compared to the as best determined α value of 1.

remarkably high with an α value of 0.5. However, this is not the case for the recall metric, where an α value of 0.5 underperforms compared to a higher α value of 1.

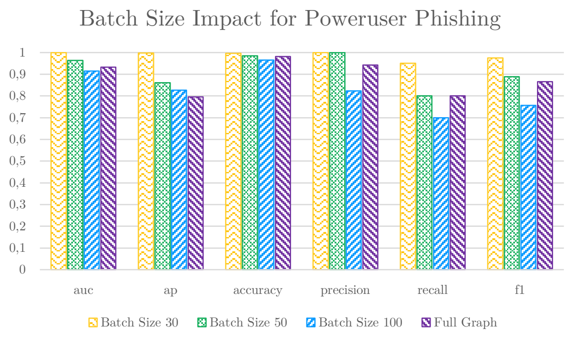
6.3.2 Impact of Batch Size

As outlined in Section 5.3, our ECONAD model employs a sampling strategy to divide the graph into smaller batches, each representing a specific time period of messages sent within the team communication platform. Here, a smaller batch size represents communication within a shorter time frame, a larger batch size represents communication within a longer time frame. Considering that our novel masterclass dataset has an average of 75 messages per week (see Chapter 4), we initially select a batch size of 50 nodes as a suitable starting point, resulting in approximately 250 messages per batch after applying neighborhood sampling. To assess the impact of batch size on the classification performance, we perform experiments in which we apply each augmentation strategy (**A1** and **A2**) with batch sizes of 30, 50, and 100 nodes, as well as the full training graph. In each experiment, we maintain an α value of 1, which was determined to be the optimal value for both augmentation strategies in the previous experiments.

As shown in Figure 6.2 the batch size has a significant impact on the classification performance. For both augmentation strategies, a smaller batch size leads to better classification performance. This is especially the case for the recall metric, which improves 25% for strategy **A1** and 15% for strategy **A2**.
As stated in Section 6.1, we utilize the last month of communication within the dataset as a



(a) Impact of the batch size on metrics for the neighborhood phishing strategy **A1**. Compared to full graph training, a lower batch size boosts overall performance on precision, recall and F1 score up by 25%.



(b) Impact of the batch size on metrics for the poweruser phishing strategy **A2**. Overall best classification could be achieved using a low batch size of 30 nodes. Especially for the recall metric, a low batch size improves the score up to 15%.

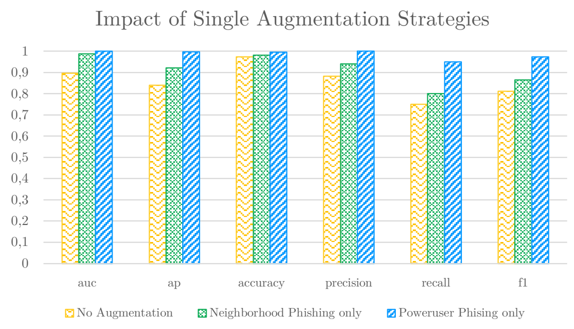
Figure 6.2: Impact of batch size on classification performance for single applied message augmentation strategies. All experiments are conducted with an α value of 1. No other augmentation strategies are enabled.

testset for our model. In this testset, approximately 50 message nodes are included. It appears that the closer the selected batch size is to the number of nodes in the testset, the greater the overall improvement in all metric performance becomes.

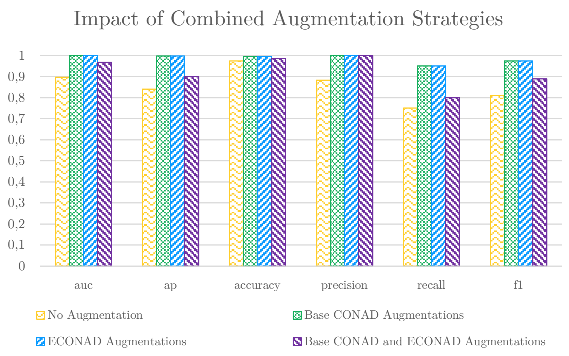
6.3.3 Comparing Augmentation Strategies

To test the impact of augmentation method **A1** for Neighborhood Phishing and **A2** for Poweruser Phishing, we first evaluate the classification performance of each novel augmentation strategy against a CONAD model without any applied augmentation in Figure 6.3a. This step ensures that the strategies are effective and do not compromise the overall performance. Then, we test the combined application of both strategies **A1** and **A2**, as employed in our final ECONAD model, against the base CONAD model and the combination of the base CONAD model with ECONAD augmentations, as illustrated in Figure 6.3b. All experiments are carried out using the optimized α value of 1, which was determined beforehand, and a batch size of 30 nodes.

Figure 6.3a demonstrates the positive impact of both our newly introduced augmentation strategies on the overall classification performance. Especially in terms of recall, **A2** outperforms the baseline by 15%. Therefore, we determine that both augmentation strategies are



(a) Metrics for augmentation strategies Neigborhood Phising **A1** and Poweruser Phishing **A2** compared to classification without any augmentation. Particularly in terms of the recall metric, **A2** outperforms the baseline by 20%.



(b) Performance of combined augmentation strategies **A1** and **A2** (ECONAD), compared to three other augmentation approaches: no augmentation, base CONAD augmentation, and combined CONAD and ECONAD augmentations. It is observed that the best overall performance is attained by either applying ECONAD or base CONAD. However, incorporating too many combined augmentation strategies appears to decrease the overall classification performance.

Figure 6.3: Impact of augmentation strategies on lateral phishing detection. All experiments were conducted with the previously determined best batch size of 30 nodes and an α value of 1.

effective and do not compromise the overall performance.
In Figure 6.3b, it is evident that the optimal overall performance is achieved by either applying the base CONAD or our ECONAD model. We cannot discern a significant difference between the augmentation strategies of the base CONAD and our ECONAD model. It appears that the base CONAD model already uses a sufficient amount of augmentation strategies to detect lateral phishing attacks. However, combining too many augmentation strategies, such as in the case of CONAD and ECONAD, diminishes the overall classification performance.

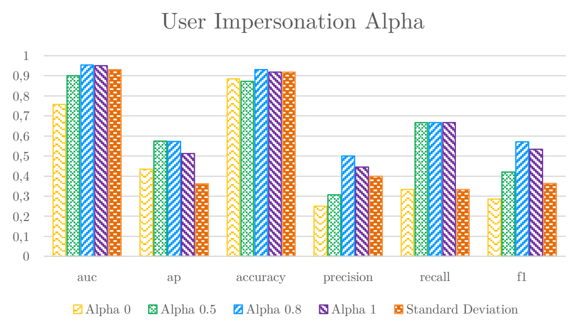
6.4 $\mathcal{P}_2$ User Channel View

In order to determine the optimal hyperparameters for our ECONAD model on the $\mathcal{P}_2$ user channel view, we first test the impact of structure and attribute on the classification. Subsequently, we carefully examine an appropriate batch size. Finally, we identify the most effective augmentation strategies for this view and assess their impact on the model's classification performance.

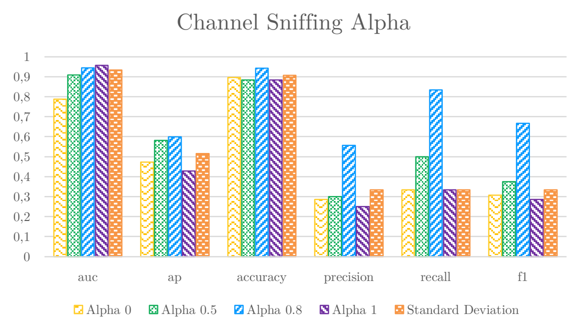
6.4.1 Impact of Structure and Attribute

To investigate the influence of node attributes and graph structure on our augmentation strategies **A3** against user impersonation and **A4** against channel sniffing in the context of the $\mathcal{P}_2$ view, we carry out experiments for each augmentation strategy using different α values. Additionally, we test the base CONAD model's approach of determining a meaningful α value based on standard deviation. Each experiment utilizes a batch size of 140 nodes. No other enhancement strategies are enabled except **A3** and **A4**.

In Figure 6.4, it is evident that a higher α value leads to a improved classification performance for both augmentation strategies **A3** and **A4**. However, for channel sniffing **A4**, the best overall performance is achieved when using a value between 0.5 and 1.0. Since **A4** introduces new edges between user and channel nodes (structure) and modifies the number of channels a user has joined to enhance the existing graph, this suggests that the model leverages both structure and attributes for learning. Notably, the recall and precision metrics profit from the structure, outperforming the experiments with different α values.
Regarding the user impersonation augmentation strategy **A3**, the highest overall performance is attained with a high α value. Considering that this strategy solely modifies the node attributes, this performance outcome is expected. Furthermore, as on the $\mathcal{P}_1$ message view (Section 6.3.1), it is evident that the proposed standard deviation method of the base CONAD model for determining the optimal α value does not apply to our customized augmentation strategies. Across all experiments, it significantly underperformed compared to experiments with the best α values.



(a) Impact of graph structure (low α) and node attributes (high α) on the user impersonation augmentation strategy **A3**. α has most impact on the precision value, with high α values performing best.



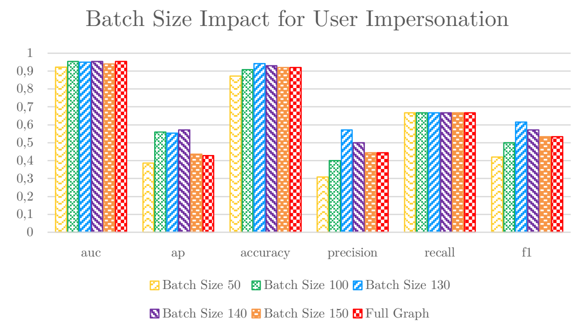
(b) Impact of structure (low α) and node attributes (high α) on the channel sniffing augmentation strategy **A4**. Compared to **A3**, α has most impact on precisison and recall. A value of $\alpha = 0.8$ performs best on all metrics, indicating that the model learns from combined structure and attributes.

Figure 6.4: Classification metrics of single applied augmentation strategies with a different focus on graph structure or node attributes. All experiments are conducted with a batch size of 140 nodes and no other augmentation strategies enabled. The standard deviation, used by CONAD as the default strategy to determine the α value, only produces moderate metrics compared to as best determined α value of 1 in Figure 6.4a and 0.8 in Figure 6.4b.

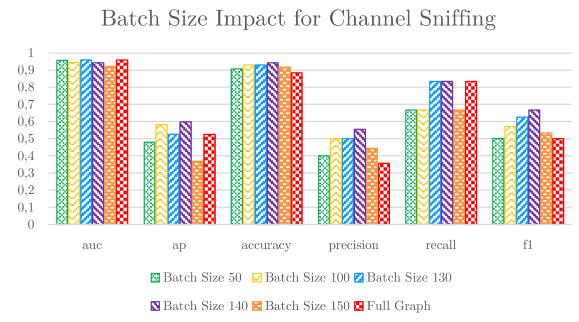
6.4.2 Impact of Batch Size

As described in Section 5.3, ECONAD employs a sampling strategy to divide the graph into smaller batches. In contrast to the previously tested $\mathcal{P}_1$ message view, where each batch represents a specific time period of message nodes sent within the team communication platform, the user and channel nodes in the $\mathcal{P}_2$ view remain unchanged between batches, with only their connecting edges being altered.
To evaluate the impact of batch size on classification performance, we test each augmentation strategy (**A3** and **A4**) on a sampled graph with batch sizes of 50, 100, 130, 140, and 150 nodes. Since there are only a total of approximately 280 nodes remaining after removing all message nodes in the $\mathcal{P}_2$ view, we are also able to train the model on the complete training graph without any applied sampling. In each experiment, we maintain an α value of 0.8, which was determined to be an optimal value for both augmentation strategies in the previous section.

For the User Impersonation augmentation strategy **A3**, Figure 6.5a shows minimal variations in metrics. It appears that the classification remains unaffected by the batch size for this strategy. However, the same cannot be said for the Channel Sniffing augmentation strategy, as shown in Figure 6.5b. In this case, a batch size of 140 nodes achieves the best overall performance, which is approximately half of the total graph nodes (280 nodes). Compared to



(a) Impact of the batch size on metrics for the user augmentation strategy **A3**. All metrics do not deviate much between applied batch sizes.



(b) Impact of the batch size on metrics for the channel sniffing augmentation strategy. Especially for recall and precision, a batch size of 140 nodes achieves the best overall results.

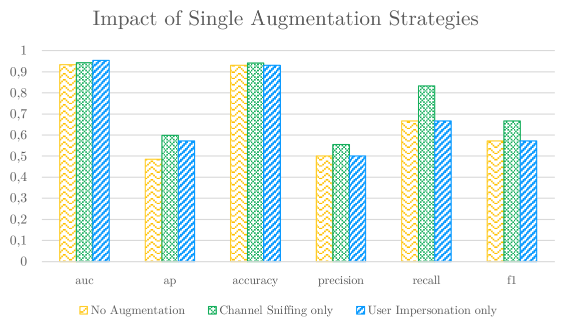
Figure 6.5: Impact of batch size on classification performance for single applied augmentation strategies. All experiments are conducted with an $\alpha = 0.8$. No other augmentation strategies are enabled.

full graph training, the batch size shows a significant impact on the precision metric.

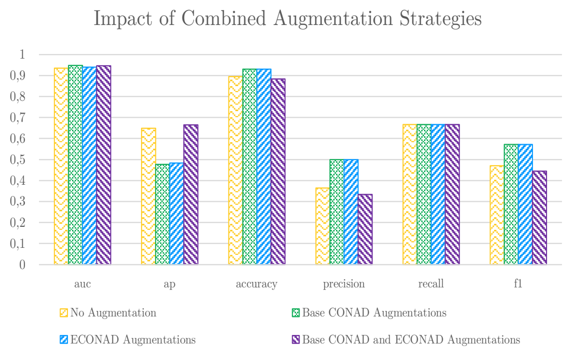
6.4.3 Comparing Augmentation Strategies

To test the impact of the augmentation method **A3** against user impersonation and **A4** against channel sniffing on the $\mathcal{P}_2$ user channel view, we first evaluate the classification performance of each augmentation strategy against a CONAD model without any applied augmentation in Figure 6.6. This step ensures that the strategies are effective and do not compromise the overall performance. Then we test the combined application of both strategies **A3** and **A4** against the base CONAD model and the combination of the base CONAD augmentations with all ECONAD augmentations, as illustrated in Figure 6.6b. All experiments are carried out using the optimized α value of 0.8 and a batch size of 140 nodes.

Figure 6.6a illustrates the positive impact of our introduced augmentation strategy **A4** to detect channel sniffing. It effectively improves the recall metric by 20%. However, the augmentation strategy **A3** to detect user impersonation appears to have minimal influence on the classification performance. It seems not possible to accurately detect user impersonation attacks solely by replicating the node attributes.
In Figure 6.6b, it is shown that the combination of both strategies (ECONAD augmentation) does not increase the recall metric. This lack of improvement may be attributed to a flawed user impersonation strategy. Compared to the model with only the **A4** augmentation applied, as shown in Figure 6.6a, the final F1 score of the model with applied **A3** augmentation decreases by almost 10%. Therefore, we conclude that the user impersonation strategy **A3** is ineffective and should not be employed. Without the user impersonation strategy, the



(a) Metrics for augmentation strategies for User Impersonation **A3** and Channel Sniffing **A4** compared to classification without any augmentation. Particularly in terms of the recall metric, **A4** outperforms the baseline by 20%.



(b) Performance of combined augmentation strategies **A3** and **A4** (ECONAD), compared to three other augmentation approaches: no augmentation, base CONAD augmentation, and combined CONAD and ECONAD augmentations. It is observed that the best overall performance is attained by either applying ECONAD or base CONAD. However, incorporating too many combined augmentation strategies appears to decrease the precision metric.

Figure 6.6: Impact of augmentation strategies on $\mathcal{P}_2$ user channel view. All experiments were conducted with the previously determined best batch size of 140 nodes and an α value of 0.8.

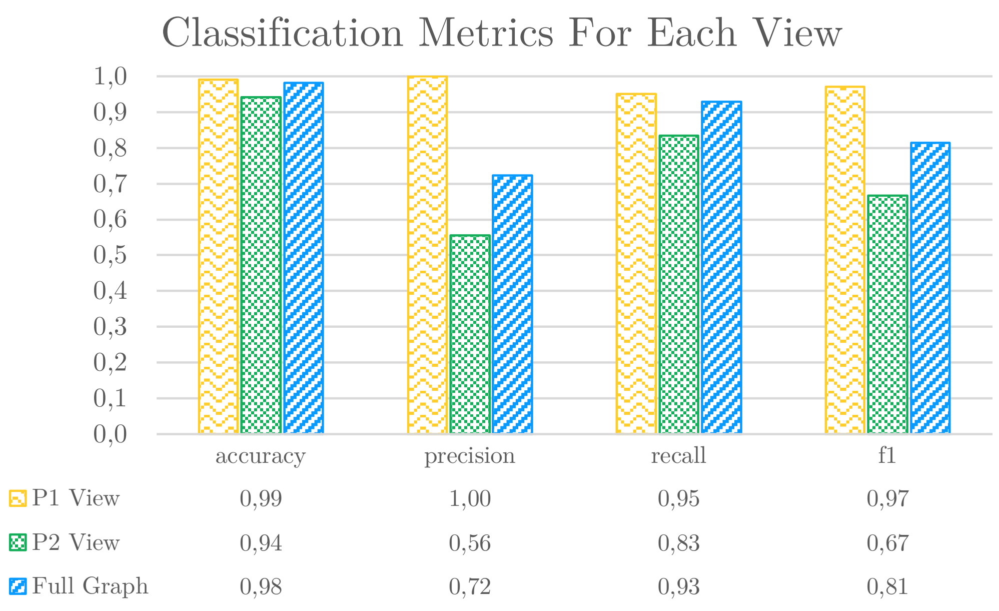
ECONAD model outperforms the base CONAD model by 20% in terms of recall and 10% in terms of the F1 score.

6.5 ECONAD Evaluation

To obtain the final classification from both views $\mathcal{P}_1$ and $\mathcal{P}_2$, the predictions of $\mathcal{P}_1$ and $\mathcal{P}_2$ are projected onto the dataset's original graph, as described in Section 5.5. The overall performance of each view and the final prediction on the graph is shown in Figure 6.7. It can be seen that the $\mathcal{P}_1$ message view prediction has a higher overall performance compared to the $\mathcal{P}_2$ user channel view prediction.

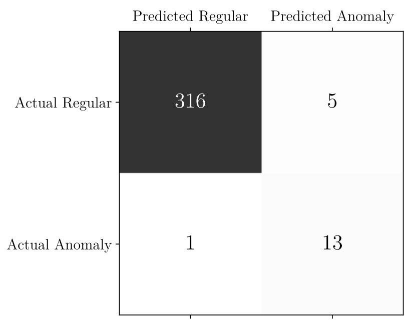
The final classification on the full graph is significantly influenced by the number of anomalies of each type in the dataset. The tested graph contains 7 lateral phishing attacks, 2 channel sniffing attacks, and 4 user impersonation attacks. Since the number of lateral phishing attacks is the highest among all, the correct classification of message attacks has the most impact on the classification metrics of the full graph. Overall, the final ECONAD model achieves an F1 score of 0.8 after combining both views.

Figure 6.8 shows the confusion matrix of the test set prediction from the final ECONAD model. Besides a single not found anomaly, all anomalies were correctly classified. However, the model classified 5 non-anomalous nodes as anomalous. Still, this is a significant improvement compared to the base CONAD model, which is only able to detect 6 out of 14 anomalies.



	accuracy	precision	recall	f1
P1 View	0,99	1,00	0,95	0,97
P2 View	0,94	0,56	0,83	0,67
Full Graph	0,98	0,72	0,93	0,81

Figure 6.7: Metrics for top performing models on the $\mathcal{P}_1$ and $\mathcal{P}_2$ view. Classification metrics on the $\mathcal{P}_1$ view are overall better than on the $\mathcal{P}_2$ view. The final classification for the full graph, is obtained by projecting the predictions of both views onto the graph.

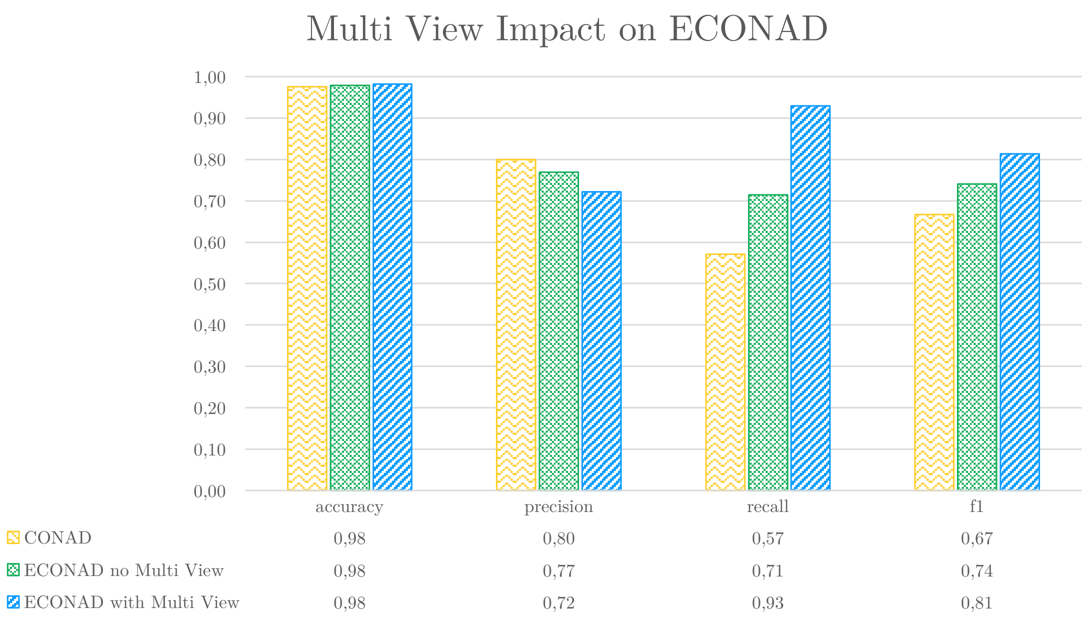


	Predicted Regular	Predicted Anomaly
Actual Regular	316	5
Actual Anomaly	1	13

Figure 6.8: Confusion matrix of the final ECONAD model. ECONAD identified 13 out of 14 injected anomalies.

6.5.1 Multi-View compared to Full Graph

In order to assess the impact of our applied multi-view approaches $\mathcal{P}_1$ and $\mathcal{P}_2$, we benchmark the ECONAD model's classification performance with and without the application of these views. To provide a comprehensive overview of the improvements, we also include the base CONAD model in the benchmark.



	accuracy	precision	recall	f1
CONAD	0,98	0,80	0,57	0,67
ECONAD no Multi View	0,98	0,77	0,71	0,74
ECONAD with Multi View	0,98	0,72	0,93	0,81

Figure 6.9: Improvements of the multi-view approach for ECONAD. When compared to ECONAD without multiple views, the multi-view approach demonstrates a 20% enhancement in the recall metric. Moreover, compared to the base CONAD model, the multi-view approach showcases an even more impressive 35% improvement in the recall metric.

In Figure 6.9, it is evident that the recall metric benefits the most from each modification made to the base model. When compared to ECONAD without multiple views, the multi-view approach demonstrates a 20% enhancement in the recall metric. Even without employing multiple views, ECONAD exhibits a 15% improvement in overall recall compared to the base CONAD model.

6.5.2 Comparison to Further Graph Anomaly Detection Algorithms

We conduct a comprehensive benchmark of the overall anomaly detection performance of our ECONAD model compared to other anomaly detection models implemented in the pyGOD graph outlier detection library [97]. In this evaluation, we compare our model's performance with CoLA [99], AnomalyDAE [100], DOMINANT [39], and the base CONAD model [45].

CoLA [99] (Contrastive self-supervised Learning framework for Anomaly detection on attributed networks) employs contrastive instance pair sampling to predict anomalies. The model measures the agreement of instance pairs using outputted scores, which are then utilized to evaluate the abnormality of each node. Like the CONAD base model, CoLA incorporates self-supervised learning to ease the reliance on labeled data.

AnomalyDAE [100] (Anomaly Detection through a Dual Autoencoder) uses two autoencoders (a structure autoencoder and an attribute autoencoder) for concurrent learning of node and attribute embeddings. It features an attention mechanism to detect crucial structural patterns. The model reconstructs node attributes using both node and attribute embeddings, enabling it to discern the interactions between network structure and node attributes.

As mentioned in Chapter 3, DOMINANT [39] was among the first GNN models to employ an autoencoder for computing an anomaly score. DOMINANT, served as the base for CONAD and ECONAD, and is therefore included in our benchmark.

The base CONAD model [45], as described in detail in Section 3.3, is the base model for our ECONAD model. It is specifically designed for social networks and incorporates prior human knowledge about attacks into its model.

All selected GNN models employ an autoencoder architecture [42] to detect anomalies in graphs and support graph sampling to split the full graph into smaller batches.
In our benchmark, each model is evaluated using a batch size of 30 and 100 nodes, respectively. The optimal performance achieved by each model is outlined in Table 6.1.

Beyond Accuracy and Precision metrics, ECONAD surpasses each model in terms of Recall and overall F1 score. In particular, ECONAD boasts a 20% improvement in the recall metric compared to the base CONAD model. CoLA and AnomalyDAE encounter challenges when distinguishing between anomalous and non-anomalous nodes, achieving only a precision of 0.05 and 0.06, and a recall of 0.29 and 0.43, respectively. These less-than-ideal metrics could potentially be attributed to the heterogeneous graph design of our dataset. In contrast, DOMINANT presents a precision of 0.4 but falls short with a recall of 0.14. Despite this, its overall accuracy is comparable to that of both CONAD and ECONAD.

We also attempted to benchmark our model against GAAN [101], a non-autoencoder Neural Network anomaly detection approach based on a Generative Adversarial Network [102]. However, GAAN does not support varying graph sizes, which are present in our training and test sets. Consequently, we are unable to include GAAN in our benchmark.

Model	Accuracy	Precision	Recall	F1	Confusion Matrix
CoLA [99]	0.72	0.05	0.29	0.08	$\begin{bmatrix} 237 & 84 \\ 10 & 4 \end{bmatrix}$
AnomalyDAE [100]	0.71	0.06	0.43	0.11	$\begin{bmatrix} 233 & 88 \\ 8 & 6 \end{bmatrix}$
DOMINANT [39]	0.96	0.40	0.14	0.21	$\begin{bmatrix} 318 & 3 \\ 12 & 2 \end{bmatrix}$
Base CONAD [45]	**0.98**	**0.80**	0.57	0.67	$\begin{bmatrix} 319 & 2 \\ 6 & 8 \end{bmatrix}$
ECONAD (Ours)	**0.98**	0.72	**0.93**	**0.81**	$\begin{bmatrix} \mathbf{316} & \mathbf{5} \\ \mathbf{1} & \mathbf{13} \end{bmatrix}$

Table 6.1: Benchmark of various graph anomaly detection models against our ECONAD model. The best performance of each metric is highlighted in **bold**.

7 Discussion

The main objective of this thesis is to develop an enhanced method for detecting anomalies and threats in team communication platforms. For this task, we delved into the field of Graph Neural Networks (GNNs) as we discovered that our data could be best represented using a graph structure. With the help of a GNN, we are able to detect anomalies in outlying graph structures and node attributes. We show that GNNs are a suitable method for detecting anomalies in team communication platforms and that they can identify new types of anomalies, such as channel sniffing, which was not possible to detect with previous methods.
In this chapter, we discuss the Masterclass dataset we created and the novel ECONAD model. Both, advantages and disadvantages, are highlighted, and each enhancement to ECONAD is discussed.

7.1 Dataset

In this thesis we demonstrate that the created Masterclass dataset can be used for graph-based machine learning tasks on team communication platforms. It offers a complete graph, representing the whole workspace, incorporating all users and their interactions. As outlined in Section 3.4.1 and Section 4.1, it is the first open and complete dataset that is suitable for anomaly detection tasks within team communication platforms. As demonstrated in Section 4.1.1, previous datasets lack the same type of user interactions and did not encompass the same level of message and user features found in private team communication platforms.

Furthermore, the dataset is highly extendible. Future researchers can incorporate additional nodes and edges, by modifying the dataframes mentioned in Section 4.3. This enables new research possibilities for Graph Neural Networks, which do not have to be necessarily related to the task of anomaly detection. Additionally, the provided source code for the dataset generation enables the creation of new datasets derived from different Slack workspaces, which is beneficial for organizations aiming to monitor their own Slack workspaces.

The underlying graph representation for the team communication platform proves to be a suitable method for understanding the user interactions and visualizing the data (see Section 4.6.3). This visual clarity enables an intuitive interpretation of the found anomalies within the dataset and helps to easily identify affected users of an attack.

7.1.1 Limitations

Despite its advantages, the Masterclass dataset presents certain limitations. As described in Section 4.2, the dataset generation relies on data exported from Slack workspaces, without support for other team communication platforms. Additionally, the richness of the graph is influenced by the type of Slack plan used during the data export. For instance, features like direct messages between users require a paid enterprise plan.

Furthermore, while using the OpenAI endpoint [81] to generate message embeddings has proven straightforward and efficient, the requirement for a paid OpenAI key poses a constraint. Alternative approaches, such as local text embedding creation, could bypass this issue. However, as outlined in Section 4.5, using the OpenAI endpoint demonstrates superior performance compared to other tested methods of embedding creation, such as Bert-as-a-Service [82].

We also observe that the effectiveness of detecting lateral phishing attacks depends on the number of representative phishing attacks provided. Providing an excess of similar attacks, or too many AI-created phishing messages, may compromise the model's ability to detect a range of phishing threats. A wider variety of different phishing attacks is needed to counterfeit this bias. However, this is not straightforward, as most phishing attacks are not publicly accessible. We reached out to the authors of the paper *Detecting and characterizing lateral phishing at scale*[28], who developed a phishing email classifier using the Enron[60] dataset. However, they could not provide us with their dataset of phishing emails due to proprietary restrictions.

7.1.2 Alternative Representations

As described in 4.6.3, we represent our dataset as a heterogeneous graph. Initially, we experimented with a homogeneous graph representation, where users are represented as nodes and messages as connecting edges. However, this is not our final design choice due to the limited available research that incorporates edge weights for anomaly detection. Additionally, this representation lacks flexibility, as it is impossible to add other node types, such as channels, to the graph.

Another approach is to represent the dataset as a dynamic temporal graph, since this representation might capture the evolving nature of a team communication platform more accurately. However, as noted by Ma et al. [10], the field of anomaly detection on temporal dynamic graphs is still emerging, and more research is needed for this specific task. Having selected the heterogeneous graph representation, we address this problem by partitioning our data for testing, validation, and training using message timestamps. The most recent message nodes are continuously incorporated into the test set, whereas the training and validation are composed of older messages.

7.2 ECONAD

This thesis introduces ECONAD, a novel graph neural network designed to detect anomalies in team communication platforms. Unlike existing methods, which focus primarily on attribute-based anomalies (as described in Section 3.1.2), ECONAD leverages graph structure for identifying anomalies. This approach distinguishes it from other techniques currently employed to detect anomalies within team communication platforms (see Section 3.5). Furthermore, it paves the way for identifying potential threats that cannot be detected by existing feature-based deep learning anomaly detection approaches.

7.2.1 Multi-View Splitting

One of the main challenges for applying a GNN to the domain of team communication platforms is the heterogeneous data structure. Existing models for classifying malicious nodes are primarily designed for homogeneous graph structures [10]. ECONAD addresses this issue by breaking down the graph into multiple views, each with a distinct focus on the anomalies being searched for, as described in Section 6.5.1. In Section 6.5, we evaluated the impact of the multi-view approach on the model's performance. We found that splitting the graph into multiple views significantly improves the model's performance and observe a 20% increase in recall and a 7% increase in the F1 score. However, it is worth noting that not all views contribute equally to this performance improvement. The user channel view $\mathcal{P}_2$ underperforms compared to the message view $\mathcal{P}_1$. We believe that the heterogeneity of $\mathcal{P}_2$, with two types of nodes (user, channel), is the main reason for this. The model struggles to differentiate between user and channel nodes. As a result, we prevent the model from flagging channels as anomalous by hard-coding the channel nodes as non-malicious.
Additional views, such as a user view that only considers user nodes, could improve the overall performance. This might be beneficial for the User Impersonation **A3** augmentation. However, we are uncertain how to model the relationship between users and channels for the Channel Sniffing augmentation **A4**. Transposing the graph to a new view without losing information presents a significant challenge, which leads us to incorporate both types of nodes in the user channel view $\mathcal{P}_2$.

7.2.2 Augmentation Strategies

Incorporating human knowledge of previous attacks from team communication into a model is not trivial. Evaluated breaches often use several attack vectors, which are team communication platform-specific and are not investigated by other GNN-based research. Therefore, we introduce the four custom augmentations *neighborhood phishing*, *poweruser phising*, *user impersonation*, and *channel sniffing* described in Section 5.1. These augmentations incorporate human knowledge to detect the attack vectors, as outlined in Section 2.2.
In Section 6.3.3 and Section 6.4.3, we tested the impact of each augmentation strategy on the model's performance. In both experiments, our custom augmentation strategies significantly

outperform the baseline without any applied augmentation strategies. However, compared to the predefined CONAD augmentations, our custom augmentations perform similarly in each view. Therefore, we combine our best-performing augmentation strategies with each view's best-performing base CONAD strategies. This strategy achieves the best overall performance for both views. Notably, we discovered that combining all augmentation strategies decreases the overall performance of each view. As this process also applies unsuitable augmentations for specific views, such as the base CONAD high-degree augmentation in the User Channel view, we assume this as the reason for the overall performance decrease.
Finding the best combination of augmentation strategies for each view turned out to be a challenging task. The number of possible combinations increases exponentially with the number of augmentation strategies, and we found that the combination also varies depending on the alpha value and batch size used. Therefore, we only tested the combinations for the previously determined best-performing alpha value and batch size. As the batch size depends on the number of messages sent within a workspace, the best-performing combination of augmentation strategies can vary for different workspaces. To investigate this hypothesis, we need to test our model's performance on different workspaces and see if a slightly different combination of augmentation strategies yields better results.

7.2.3 Impact of Node Attributes and Graph Structure

For each introduced augmentation strategy in Section 5.1, we evaluated the impact of node attributes and graph structure on the augmentation performance in Section 6.3.3 and Section 6.4.1. We found that node attributes have a greater impact on performance across all introduced augmentation strategies than graph structure. This is particularly evident for the message augmentation strategies, where the message embeddings stored in the node attributes prove to be the most influential feature. However, for the augmentation strategy Channel Sniffing **A4**, we discovered that the best performance is achieved when both node attributes and graph structure are considered.
Based on these findings, we can conclude that using a GNN for detecting message-based attacks in team communication platforms might not be the most effective approach, as no graph structure is needed to detect these attacks. However, in cases where identifying an attack solely based on anomalous node features (e.g. channel sniffing) is challenging, the graph structure significantly contributes to the detection of anomalies. In these scenarios, a GNN appears to be the most suitable approach.

7.2.4 Impact of Batch Size

For each augmentation strategy introduced in Section 5.1, we also evaluated the impact of batch size on augmentation performance in Section 6.3.2 and Section 6.4.2. Here, we found that the batch size significantly impacts the performance of each augmentation strategy. Specifically, for message-based augmentation strategies (**A1**, **A2**), we observe the best classification performance when the batch size closely matches that of the final test graph. As for

user channel-based augmentation strategies (**A3**, **A4**), the optimal batch size is around 140, roughly 50% of the size of all remaining nodes after removing the message nodes in the $\mathcal{P}_2$ view.
These observations suggest that the optimal batch size to achieve the best classification results is influenced by the workspace environment, which changes according to the number of users and their sent messages. Furthermore, our findings show that each augmentation strategy has its own optimal batch size. The difference in optimal batch sizes could contribute to why ECONAD's performance is superior on a graph divided into multiple views, as it allows for the application of different batch sizes for each view.

7.2.5 Benchmark with Further Anomaly GNN Models

Compared to other benchmarked GNN models in Section 6.5.2, ECONAD outperforms in the recall metric, a critical aspect for anomaly detection, as it helps identify as many true anomalies as possible while minimizing the number of false negatives. However, the precision metric of ECONAD is lower than that of CONAD, the model upon which ECONAD is developed. This suggests that ECONAD exhibits higher sensitivity to anomalies given its design purpose to detect as many true anomalies as possible.
In our benchmark, we opted for models that exclusively utilize auto-encoders for anomaly detection. However, as highlighted in Chapter 3, alternative approaches for anomaly detection on attributed graphs exist, which do not rely on autoencoder-based GNN networks. These approaches, however, were not considered in our benchmark.
A significant challenge in comparing different anomaly detection models is the variation in sample sizes used for training, validation, and testing. Numerous models provided in the pyGOD [97] library cannot handle graphs of differing sizes.

7.2.6 Limitations

Despite its capabilities, ECONAD has limitations that hinder its overall performance. The first limitation is observed in the effectiveness of the User Impersonation augmentation **A3**. It appears that this augmentation technique is not as effective as initially expected. It may be worth exploring the idea of copying specific features of the feature vector rather than all at once to enhance the detection of user impersonation attacks.

ECONAD incorporates time-based node relations by splitting the graph into smaller batches based on the timestamp of each message sent in the workspace. As evaluated in Section 6.3.2 and 6.4.2, this split dramatically impacts the overall performance of the augmentation strategies. However, other features that change over time, such as user features, are not processed by ECONAD. Incorporating changing node features in a GNN model is still an open research question, with little research currently available [10].

During the evaluation, we also observed that the performance of ECONAD in detecting lateral phishing attacks (**A1**, **A2**) heavily relies on the quality of the embeddings and the presence of a sufficient number of message anomalies. If the embeddings are of low quality or an insufficient number of message anomalies are provided, it compromises the model's overall performance.

Lastly, it is important to note that the performance of the base CONAD Triplet contrasting method, as mentioned in Section 3.3.3, cannot be tested due to limitations in the underlying implementation. However, the effectiveness of this contrasting method is uncertain, as the Triplet Contrast performs worse in the original paper than the Siamese Contrast used by ECONAD [45].

7.2.7 Alternative Design Approaches

ECONAD builds upon the foundation of the CONAD model, which demonstrates superior performance compared to other graph-based anomaly detection models on social network graphs [45]. As a result, ECONAD is specifically designed as a node classification model. However, with appropriate modifications to the data representation, the node classification problem could be reformulated as an anomalous link prediction problem.
Nevertheless, the main limitation lies in the insufficient extendibility of the graph within such a problem, where messages can only be represented as links. Furthermore, as highlighted in Section 4.6.3, not much research is available that incorporates attributed edges for the task of anomaly detection using Graph Neural Networks.

Moreover, our research indicates that node attributes play the most crucial role in enhancing the model's effectiveness in detecting anomalous messages. Given the recent emergence of large language models (LLMs) [80], it should be possible to perform this classification using LLMs instead of solely relying on embeddings. However, the behavior of LLMs in more complex graph structures involving direct messages or message replies remains unclear. In such cases, the graph structure itself might have a more significant impact on the model's performance. Further research is needed here to validate this assumption.

8 Conclusion

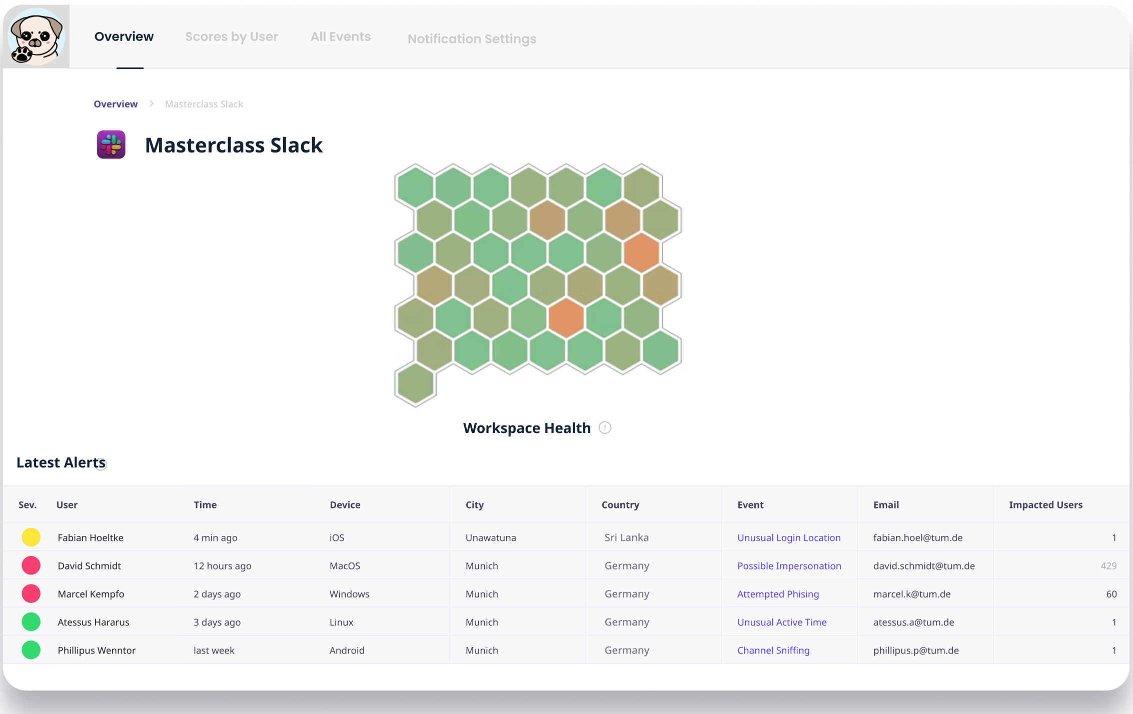
This thesis addressed the rising problem of effective anomaly detection in team communication platforms. As these platforms became increasingly integral to our professional and personal lives, the security of the data transmitted within is crucial. We introduced ECONAD, a Graph Neural Network (GNN) based model designed to detect anomalies and malicious actors within team communication platforms.
Throughout the literature research, we identified the limitations of existing anomaly detection approaches for team communication platforms, which do not incorporate graph structure for their predictions. Therefore we employed a GNN, as it can process a platform's full graph representation and also leverage its structure to detect anomalies.
By introducing multiple views to handle graph heterogeneity and custom augmentation strategies to incorporate human knowledge about previous attacks, ECONAD was fine-tuned to detect specific types of attacks, threats, and malicious activities within team communication platforms.
To evaluate the effectiveness of ECONAD, we created a custom team communication dataset, capturing the activity of 250 users over three years. This dataset enabled us to thoroughly assess the performance of ECONAD on a closed organization's team communication platform, with access to relevant user features and the entire message history of all public channels.
Initially, we tested the impact of batch size and graph structure on each augmentation strategy. Our findings revealed that incorporating graph structure enhances the detection of anomalies, especially when detecting specific attacks solely based on anomalous node features is challenging. Additionally, we observed that the batch size, used to split a graph into smaller subgraphs, has varying effects on each augmentation strategy, depending on the type of anomaly being sought.
Next, the effectiveness of each introduced view in ECONAD was evaluated. We found that a homogeneous view containing only the same types of nodes performed better than a heterogeneous view, which included two node types. Furthermore, transforming the graph into different views based on the searched anomaly significantly improved the overall performance compared to processing the full graph at once.
When benchmarked against other state-of-the-art GNN models with comparable design architecture, using our novel dataset for a classification test with 14 injected anomalies, we found that ECONAD outperformed the best-performing current GNN model by up to 36% in recall and 14% in the overall F1 score, demonstrating the effectiveness of our novel model and its ability to detect anomalies in team communication platforms.

Applying GNNs to team communication platforms is a promising approach for detecting anomalies and malicious actors. By incorporating the graph structure of team communication platforms, GNNs have the capability to uncover anomalies that traditional feature-based deep learning approaches are unable to detect. This research was potentially the first to apply GNNs to team communication platforms, and we hope that it will inspire future research in this field.

8.1 Future Work

The introduced dataset and ECONAD model serve as the foundation for numerous future experiments in the field of anomaly detection on team communication platforms. However, there is room for future improvements, which are outlined in the following:

- **Extending the dataset**: Our Masterclass dataset relied on the Slack pro plan and did not incorporate direct communication between users. It only included communication from users in public channels. This limitation could be addressed by upgrading to a more advanced Slack plan. Furthermore, user nodes could be extended with additional features, which could be utilized to detect new anomalies within the dataset.
- **Fine Tuning Augmentation strategies**: We tested ECONAD using four augmentation strategies: Neighborhood Phishing, Poweruser Phishing, User Impersonation, and Channel Sniffing. However, as outlined in 6.4.3 we found the Poweruser Phishing augmentation does not have a meaningful impact on the final classification. More fine-tuning is needed here. Furthermore, ECONAD could be enhanced by incorporating a new set of augmentation strategies to detect additional anomalies.
- **Support for More Team Communication Platforms**: Expanding the support for communication platforms beyond the current scope is essential. For instance, integrating with platforms such as Microsoft Teams would make our model accessible to a broader range of organizations interested in supervising their team communication platform.
- **Continuous Integration**: To ensure continuous and up-to-date anomaly detection, it is crucial to integrate the system for real-time data collection. Developing a new integration for team communication platforms would enable continuous data collection, facilitating timely anomaly detection and response.
- **Notifications**: By incorporating alarm systems that adjust based on a severity scale of the discovered anomalies, proactive measures can be taken quickly in response to potential threats.
- **User-Friendly Frontend**: Creating a user-friendly front-end to present the results of ECONAD would significantly enhance the analysis and interpretation of identified anomalies. By visually representing the connections and interdependencies among users and teams, administrators can promptly respond to found threats and identify impacted users. Figure 8.1 illustrates a draft version of a potential front-end.



Latest Alerts

Sev.	User	Time	Device	City	Country	Event	Email	Impacted Users
🟡	Fabian Hoeltke	4 min ago	iOS	Unawatuna	Sri Lanka	Unusual Login Location	fabian.hoel@tum.de	1
🔴	David Schmidt	12 hours ago	MacOS	Munich	Germany	Possible Impersonation	david.schmidt@tum.de	429
🔴	Marcel Kempfo	2 days ago	Windows	Munich	Germany	Attempted Phising	marcel.k@tum.de	60
🟢	Atessus Hararus	3 days ago	Linux	Munich	Germany	Unusual Active Time	atessus.a@tum.de	1
🟢	Phillipus Wenntor	last week	Android	Munich	Germany	Channel Sniffing	phillipus.p@tum.de	1

Figure 8.1: Draft for a possible frontend. On the main page, a quick overview of the overall workspace health is given using a honeycomb diagram. Here, anomalous users are highlighted in red, while normal users stay green. Each found anomaly is grouped by severity. A list of the latest alerts gives a brief overview of the types of anomalies that were found and how many users were impacted. When investigating a single alert, the graph of the team communication platform with all impacted users is shown.

ar

9 Appendix

```
"client_msg_id": "5b6bbd94-edee-4db6-b484-07ca26423afe",
    "type": "message",
    "text": "Dear alumni, as always, you’re warmly invited to join! :smiling_face..",
    "user": "U01KSJZHLAW",
    "ts": "1665122315.915109",
    "blocks": [
        {
            "type": "rich_text",
            "block_id": "tWzY",
            "elements": [
                {
                    "type": "rich_text_section",
                    "elements": [
                        {
                            "type": "text",
                            "text": "TUM EMC Offsite ",
                            "style": {
                                "bold": true
                            }
                        },
                        {
                            "type": "emoji",
                            "name": "mountain",
                            "unicode": "26f0-fe0f"
                        }
```

Figure 9.1: Slack message JSON export

List of Figures

List of Tables

Bibliography

[1] Alfresco. *Dimesional Research Collab Survey Findings Report*. https://www.alfresco.com/sites/www.alfresco.com/files/dimesional-research-collab-survey-findings-report-082415.pdf. [Accessed 11-Jul-2023].

[2] *Global Team Collaboration Software Industry — reportlinker.com*. https://www.reportlinker.com/p06050492/Global-Team-Collaboration-Software-Industry.html?utm_source=GNW. [Accessed 13-Jul-2023].

[3] O. Maor. *Are You Ready for a Breach in Your Organization's Slack Workspace? — darkreading.com*. https://www.darkreading.com/attacks-breaches/are-you-ready-for-a-breach-in-your-organization-s-slack-workspace-. [Accessed 18-Jun-2023].

[4] M. Große-Kampmann and M. Gruber. "Business Chat is Confused. It Hurt Itself in its Confusion-Chishing". In: ().

[5] M. Binder. *A teen hacked Uber and announced it in the company Slack. Employees thought it was a joke — mashable.com*. https://mashable.com/article/uber-teen-hacker-slack-joke. [Accessed 18-Jun-2023].

[6] *Hacker Breaches Activision Slack, Steals Call of Duty Info — vice.com*. https://www.vice.com/en/article/pkg7pn/hacker-breaches-activision-slack-steals-call-of-duty-info. [Accessed 18-Jun-2023].

[7] *How Hackers Used Slack to Break into EA Games — vice.com*. https://www.vice.com/en/article/7kvkqb/how-ea-games-was-hacked-slack. [Accessed 18-Jun-2023].

[8] K. MacDonald. *Grand Theft Auto 6 leak: who hacked Rockstar and what was stolen? — theguardian.com*. https://www.theguardian.com/games/2022/sep/19/grand-theft-auto-6-leak-who-hacked-rockstar-and-what-was-stolen. [Accessed 18-Jun-2023].

[9] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. "Graph neural networks: A review of methods and applications". In: *AI open* 1 (2020), pp. 57–81.

[10] X. Ma, J. Wu, S. Xue, J. Yang, C. Zhou, Q. Z. Sheng, H. Xiong, and L. Akoglu. "A comprehensive survey on graph anomaly detection with deep learning". In: *IEEE Transactions on Knowledge and Data Engineering* (2021).

[11] E. Müller, P. I. Sánchez, Y. Mülle, and K. Böhm. "Ranking outlier nodes in subspaces of attributed graphs". In: *2013 IEEE 29th international conference on data engineering workshops (ICDEW)*. IEEE. 2013, pp. 216–222.

[12] J. Li, H. Dani, X. Hu, and H. Liu. "Radar: Residual Analysis for Anomaly Detection in Attributed Networks." In: *IJCAI*. Vol. 17. 2017, pp. 2152–2158.

[13] A. Anders. "Team communication platforms and emergent social collaboration practices". In: *International Journal of Business Communication* 53.2 (2016), pp. 224–261.

[14] B. Lin, A. Zagalsky, M.-A. Storey, and A. Serebrenik. "Why developers are slacking off: Understanding how software teams use slack". In: *Proceedings of the 19th acm conference on computer supported cooperative work and social computing companion*. 2016, pp. 333–336.

[15] Y. Chen, Y. Gao, N. Ceccio, R. Chatterjee, K. Fawaz, and E. Fernandes. "Experimental Security Analysis of the App Model in Business Collaboration Platforms". In: *31st USENIX Security Symposium (USENIX Security 22)*. 2022, pp. 2011–2028.

[16] J. Edu, C. Mulligan, F. Pierazzi, J. Polakis, G. Suarez-Tangil, and J. Such. "Exploring the security and privacy risks of chatbots in messaging services". In: *Proceedings of the 22nd ACM Internet Measurement Conference*. 2022, pp. 581–588.

[17] *Uber*. `https://uber.com`. [Accessed 11-Jul-2023].

[18] *Activision Blizzard*. `https://www.activisionblizzard.com`. [Accessed 11-Jul-2023].

[19] *Slack*. `https://slack.com`. [Accessed 04-Jul-2023].

[20] J. Davalos. *Uber Slack Hacked*. `https://www.bloomberg.com/news/articles/2022-09-16/uber-says-it-s-investigating-extent-of-cybersecurity-incident`. [Accessed 04-Jul-2023].

[21] *Electronic Arts Home Page*. `https://www.ea.com`. [Accessed 11-Jul-2023].

[22] R. Games. *Rockstar Games*. `https://www.rockstargames.com`. [Accessed 11-Jul-2023].

[23] B. Perozzi and L. Akoglu. "Scalable anomaly ranking of attributed neighborhoods". In: *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM. 2016, pp. 207–215.

[24] B. Perozzi, L. Akoglu, P. Iglesias Sánchez, and E. Müller. "Focused clustering and outlier detection in large attributed graphs". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014, pp. 1346–1355.

[25] P. I. Sánchez, E. Müller, O. Irmler, and K. Böhm. "Local context selection for outlier ranking in graphs with multiple numeric node attributes". In: *Proceedings of the 26th international conference on scientific and statistical database management*. 2014, pp. 1–12.

[26] P. I. Sánchez, E. Müller, F. Laforet, F. Keller, and K. Böhm. "Statistical selection of congruent subspaces for mining attributed graphs". In: *2013 IEEE 13th international conference on data mining*. IEEE. 2013, pp. 647–656.

[27] R. Kawase, F. Diana, M. Czeladka, M. Schüler, and M. Faust. "Internet fraud: the case of account takeover in online marketplace". In: *Proceedings of the 30th ACM Conference on Hypertext and Social Media*. 2019, pp. 181–190.

[28] G. Ho, A. Cidon, L. Gavish, M. Schweighauser, V. Paxson, S. Savage, G. M. Voelker, and D. Wagner. "Detecting and characterizing lateral phishing at scale". In: *28th USENIX Security Symposium (USENIX Security 19)*. 2019, pp. 1273–1290.

[29] X. Shen, W. Lv, J. Qiu, A. Kaur, F. Xiao, and F. Xia. "Trust-Aware Detection of Malicious Users in Dating Social Networks". In: *IEEE Transactions on Computational Social Systems* (2022).

[30] X. He, Q. Gong, Y. Chen, Y. Zhang, X. Wang, and X. Fu. "DatingSec: Detecting malicious accounts in dating apps using a content-based attention network". In: *IEEE Transactions on Dependable and Secure Computing* 18.5 (2021), pp. 2193–2208.

[31] *Immomo*. `http://immomo.com`. [Accessed 03-Jul-2023].

[32] D. Seyler, L. Li, and C. Zhai. "Semantic text analysis for detection of compromised accounts on social networks". In: *2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE. 2020, pp. 417–424.

[33] J. Yang and J. Leskovec. "Patterns of temporal variation in online media". In: *Proceedings of the fourth ACM international conference on Web search and data mining*. 2011, pp. 177–186.

[34] L. Ilias and I. Roussaki. "Detecting malicious activity in Twitter using deep learning techniques". In: *Applied Soft Computing* 107 (2021), p. 107360.

[35] *Twitter*. `https://www.twitter.com`. [Accessed 11-Jul-2023].

[36] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. "A comprehensive survey on graph neural networks". In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 4–24.

[37] H. Kim, B. S. Lee, W.-Y. Shin, and S. Lim. "Graph Anomaly Detection with Graph Neural Networks: Current Status and Challenges". In: *IEEE Access* (2022).

[38] Z. Peng, M. Luo, J. Li, H. Liu, Q. Zheng, et al. "ANOMALOUS: A Joint Modeling Approach for Anomaly Detection on Attributed Networks." In: *IJCAI*. 2018, pp. 3513–3519.

[39] K. Ding, J. Li, R. Bhanushali, and H. Liu. "Deep anomaly detection on attributed networks". In: *Proceedings of the 2019 SIAM International Conference on Data Mining*. SIAM. 2019, pp. 594–602.

[40] M. W. Mahoney and P. Drineas. "CUR matrix decompositions for improved data analysis". In: *Proceedings of the National Academy of Sciences* 106.3 (2009), pp. 697–702.

[41] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger. "Simplifying graph convolutional networks". In: *International conference on machine learning*. PMLR. 2019, pp. 6861–6871.

[42] J. An and S. Cho. "Variational autoencoder based anomaly detection using reconstruction probability". In: *Special lecture on IE* 2.1 (2015), pp. 1–18.

[43] A. Chaudhary, H. Mittal, and A. Arora. “Anomaly detection using graph neural networks”. In: *2019 international conference on machine learning, big data, cloud and parallel computing (COMITCon)*. IEEE. 2019, pp. 346–350.

[44] Y. Dou, K. Shu, C. Xia, P. S. Yu, and L. Sun. “User preference-aware fake news detection”. In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2021, pp. 2051–2055.

[45] Z. Xu, X. Huang, Y. Zhao, Y. Dong, and J. Li. “Contrastive attributed network anomaly detection with data augmentation”. In: *Advances in Knowledge Discovery and Data Mining: 26th Pacific-Asia Conference, PAKDD 2022, Chengdu, China, May 16–19, 2022, Proceedings, Part II*. Springer. 2022, pp. 444–457.

[46] Y. Zhang, Y. Fan, Y. Ye, L. Zhao, and C. Shi. “Key player identification in underground forums over attributed heterogeneous information network embedding framework”. In: *Proceedings of the 28th ACM international conference on information and knowledge management*. 2019, pp. 549–558.

[47] Z. Peng, M. Luo, J. Li, L. Xue, and Q. Zheng. “A deep multi-view framework for anomaly detection on attributed networks”. In: *IEEE Transactions on Knowledge and Data Engineering* 34.6 (2020), pp. 2539–2552.

[48] *Facebook*. `https://www.facebook.com`. [Accessed 11-Jul-2023].

[49] D. Mesquita, A. Souza, and S. Kaski. “Rethinking pooling in graph neural networks”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 2220–2231.

[50] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. “Graph attention networks”. In: *arXiv preprint arXiv:1710.10903* (2017).

[51] V. Nair and G. E. Hinton. “Rectified linear units improve restricted boltzmann machines”. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814.

[52] P. Chatterjee, K. Damevski, N. A. Kraft, and L. Pollock. “Software-related slack chats with disentangled conversations”. In: *Proceedings of the 17th international conference on mining software repositories*. 2020, pp. 588–592.

[53] E. Parra, A. Ellis, and S. Haiduc. “Gittercom: A dataset of open source developer communications in gitter”. In: *Proceedings of the 17th International Conference on Mining Software Repositories*. 2020, pp. 563–567.

[54] *GitHub - freeCodeCamp/open-data — github.com*. `https://github.com/freeCodeCamp/open-data`. [Accessed 19-Jun-2023].

[55] *Gitter; Where developers come to talk. — gitter.im*. `https://gitter.im/`. [Accessed 30-Jun-2023].

[56] *Reddit*. `https://www.reddit.com`. [Accessed 11-Jul-2023].

[57] J. Leskovec and J. Mcauley. “Learning to discover social circles in ego networks”. In: *Advances in neural information processing systems* 25 (2012).

[58] K. Shu, D. Mahudeswaran, S. Wang, D. Lee, and H. Liu. "Fakenewsnet: A data repository with news content, social context, and spatiotemporal information for studying fake news on social media". In: *Big data* 8.3 (2020), pp. 171–188.

[59] W. Hamilton, Z. Ying, and J. Leskovec. "Inductive representation learning on large graphs". In: *Advances in neural information processing systems* 30 (2017).

[60] B. Klimt and Y. Yang. "Introducing the Enron corpus." In: *CEAS*. Vol. 45. 2004, pp. 92–96.

[61] *Enron*. `https://www.enron.ca`. [Accessed 11-Jul-2023].

[62] B. Klimt and Y. Yang. "The enron corpus: A new dataset for email classification research". In: *European conference on machine learning*. Springer. 2004, pp. 217–226.

[63] C. P. S. Technologies. *Slack Security | Avanan — avanan.com*. `https://www.avanan.com/slack-security`. [Accessed 19-Jun-2023].

[64] *Zerofox*. `https://www.zerofox.com/slack/`. [Accessed 19-Jun-2023].

[65] *Nightfall Cloud-Native DLP for Slack*. `https://www.nightfall.ai/integrations/slack`. [Accessed 19-Jun-2023].

[66] S. Alneyadi, E. Sithirasenan, and V. Muthukkumarasamy. "A survey on data leakage prevention systems". In: *Journal of Network and Computer Applications* 62 (2016), pp. 137–152.

[67] *Slack API*. `https://api.slack.com/admins/audit-logs-anomaly`. [Accessed 19-Jun-2023].

[68] *TUM Entrepreneurial Masterclass*. `https://masterclass.tum.de/`. [Accessed 14-Jun-2023].

[69] M. Bastian, S. Heymann, and M. Jacomy. *Gephi: An Open Source Software for Exploring and Manipulating Networks*. 2009. URL: `http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154`.

[70] T. M. Fruchterman and E. M. Reingold. "Graph drawing by force-directed placement". In: *Software: Practice and experience* 21.11 (1991), pp. 1129–1164.

[71] *Zentrum für Innovation & Gründung — unternehmertum.de*. `https://www.unternehmertum.de`. [Accessed 14-Jul-2023].

[72] Slack. *Pricing — slack.com*. `https://slack.com/intl/en-gb/pricing`. [Accessed 01-Jul-2023].

[73] *Export Slack Workspace*. `https://slack.com/help/articles/201658943-Export-your-workspace-data`. [Accessed 15-Jun-2023].

[74] Slack. *team.accessLogs API method — api.slack.com*. `https://api.slack.com/methods/team.accessLogs`. [Accessed 01-Jul-2023].

[75] *OpenAI API — platform.openai.com*. `https://platform.openai.com/docs/api-reference/completions`. [Accessed 15-Jun-2023].

[76] W. McKinney et al. “Data structures for statistical computing in python”. In: *Proceedings of the 9th Python in Science Conference*. Vol. 445. 1. Austin, TX. 2010, pp. 51–56.

[77] *ChatGPT — openai.com*. `https://openai.com/chatgpt`. [Accessed 15-Jun-2023].

[78] T. Mikolov, K. Chen, G. Corrado, and J. Dean. “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781* (2013).

[79] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[80] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, et al. “Emergent abilities of large language models”. In: *arXiv preprint arXiv:2206.07682* (2022).

[81] A. Neelakantan, T. Xu, R. Puri, A. Radford, J. M. Han, J. Tworek, Q. Yuan, N. Tezak, J. W. Kim, C. Hallacy, et al. “Text and code embeddings by contrastive pre-training”. In: *arXiv preprint arXiv:2201.10005* (2022).

[82] H. Xiao. *bert-as-service*. `https://github.com/hanxiao/bert-as-service`. 2018.

[83] N. Muennighoff. “Sgpt: Gpt sentence embeddings for semantic search”. In: *arXiv preprint arXiv:2202.08904* (2022).

[84] *openai Cookbook — github.com*. `https://github.com/openai/openai-cookbook/blob/main/examples/Customizing_embeddings.ipynb`. [Accessed 15-Jun-2023].

[85] G. Van Rossum and F. L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.

[86] A. Hagberg, P. Swart, and D. S Chult. *Exploring network structure, dynamics, and function using NetworkX*. Tech. rep. Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[87] M. Y. Wang. “Deep graph library: Towards efficient and scalable deep learning on graphs”. In: *ICLR workshop on representation learning on graphs and manifolds*. 2019.

[88] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. “Pytorch: An imperative style, high-performance deep learning library”. In: *Advances in neural information processing systems* 32 (2019).

[89] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang. “Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems”. In: *arXiv preprint arXiv:1512.01274* (2015).

[90] M. Abadi. “TensorFlow: learning functions at scale”. In: *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming*. 2016, pp. 1–1.

[91] M. Fey and J. E. Lenssen. “Fast graph representation learning with PyTorch Geometric”. In: *arXiv preprint arXiv:1903.02428* (2019).

[92] *torchGeometric Data*. `https://pytorch-geometric.readthedocs.io/en/latest/modules/data.html`. [Accessed 01-Jul-2023].

[93] B. Rozemberczki, P. Scherer, Y. He, G. Panagopoulos, A. Riedel, M. Astefanoaei, O. Kiss, F. Beres, G. López, N. Collignon, et al. "Pytorch geometric temporal: Spatiotemporal signal processing with neural machine learning models". In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2021, pp. 4564–4573.

[94] N. Shah, A. Beutel, B. Hooi, L. Akoglu, S. Gunnemann, D. Makhija, M. Kumar, and C. Faloutsos. "Edgecentric: Anomaly detection in edge-attributed networks". In: *2016 IEEE 16Th international conference on data mining workshops (ICDMW)*. IEEE. 2016, pp. 327–334.

[95] P. Hu and W. C. Lau. "A survey and taxonomy of graph sampling". In: *arXiv preprint arXiv:1308.5865* (2013).

[96] M. Xu. "Understanding graph embedding methods and their applications". In: *SIAM Review* 63.4 (2021), pp. 825–853.

[97] K. Liu, Y. Dou, Y. Zhao, X. Ding, X. Hu, R. Zhang, K. Ding, C. Chen, H. Peng, K. Shu, G. H. Chen, Z. Jia, and P. S. Yu. "PyGOD: A Python Library for Graph Outlier Detection". In: *arXiv preprint arXiv:2204.12095* (2022).

[98] A. Tharwat. "Classification assessment methods". In: *Applied computing and informatics* 17.1 (2020), pp. 168–192.

[99] Y. Liu, S. Pan, Y. G. Wang, F. Xiong, L. Wang, Q. Chen, and V. C. Lee. "Anomaly detection in dynamic graphs via transformer". In: *IEEE Transactions on Knowledge and Data Engineering* (2021).

[100] H. Fan, F. Zhang, and Z. Li. "Anomalydae: Dual autoencoder for anomaly detection on attributed networks". In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 5685–5689.

[101] Z. Chen, B. Liu, M. Wang, P. Dai, J. Lv, and L. Bo. "Generative adversarial attributed network anomaly detection". In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2020, pp. 1989–1992.

[102] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath. "Generative adversarial networks: An overview". In: *IEEE signal processing magazine* 35.1 (2018), pp. 53–65.