A Comprehensive Survey of Graph-based Deep Learning Approaches for Anomaly Detection in Complex Distributed Systems

Armin Danesh Pazho*, Ghazal Alinezhad Noghre*, Arnab A Purkayastha, Jagannadh Vempati, Otto Martin, and Hamed Tabkhi

Abstract—Anomaly detection is an important problem for complex distributed systems consisting of hardware and software components. A thorough understanding of the requirements and challenges of anomaly detection for such systems is pivotal to the security of a system, especially for real-world deployment. While there have been many diverse research areas and application domains that deal with the problem, few have attempted to provide an in-depth look at such systems. Most anomaly detection techniques have been specifically developed for certain application domains, while others are more generic. In this survey, we explore the significant potential of graph-based algorithms to identify and mitigate different types of anomalies in complex distributed heterogeneous systems. Our main focus is to provide an in-depth look at graphs when applied on heterogeneous computing devices spread across complex distributed systems. This study analyzes, compares, and contrasts the state-of-the-art research articles in the field. First, we describe the characteristics of the real-world distributed systems and their specific challenges of anomaly detection in such complex networks, such as data and evaluation, nature of the anomalies, and real-world requirements. Later, we discuss why graphs can be leveraged in such systems and the benefits of utilizing graphs. Then we will aptly delve into the state-of-the-art approaches and highlight their strength and weaknesses. Finally, we evaluate and compare these approaches and point out the areas for possible improvements.

Index Terms—Graphs, Anomaly Detection, Deep Learning, Heterogeneous Systems, Distributed Systems.

1 INTRODUCTION

Anomaly detection refers to finding abnormal behavior or patterns in the data or a system that does not match the expected behavior [1]. In other words, A non-benign change in the known and correct behavior of a system can be detected as an anomaly. Anomaly detection is critical in distributed heterogeneous systems because of their nature. In such systems, different components, from small variations like sensors, work all the way up to large components like control facilities to achieve the overall goal of the system. All these components are spread out in a big network; hence the word distributed, and they are dissimilar with respect to their structure and data production, hence the word heterogeneous.

Many distributed heterogeneous systems are operational in critical and/or important fields (e.g., Power Generation and Distribution systems). Thus, their security and correctness are critical. Vulnerabilities can be seen in every aspect of these systems and in different levels of abstraction; from problematic individual components within a system or a network to seeing abnormalities or irregularities where these components connect with each other to accomplish a more advanced task. Table 1 shows several distributed heterogeneous networks and possible sources of anomalies in them. Anomaly detection in these systems must meet many requirements, making the task even more complicated. Anomalies must be detected by processing large quantities of data, considering the nature of anomalies, which makes detecting them a challenging task and the real-world setup limitations.

In recent years, neural networks have shown a great potential and are being explored drastically. They have opened new avenues and increased interest in extending deep learning approaches for anomaly detection [2], [3]. Anomaly detection imposes several practical challenges that make adopting deep learning algorithms difficult. Among learning approaches for anomaly detection, there has been an increase in the number of applications where data is represented in the form of graphs to extract more information, such as relational features. However, graphs can be irregular; for example, a graph may have a variable size of unordered nodes, and/or nodes from a graph may have a different number of neighbors. Thus, as some important operations (e.g., regular convolutions) are easy to compute in the domains like image processing, they become difficult to apply to the graph domain [4]. Furthermore, a core assumption of existing machine learning algorithms is that instances are independent of each other. This assumption falls flat for graph data because each instance (node) is

* First two authors (A. Danesh Pazho and G. Alinezhad Noghre) have equal contribution.

- Armin Danesh Pazho, Ghazal Alinezhad Noghre, and Hamed Tabkhi are with the University of North Carolina at Charlotte, NC, USA. E-mail: adaneshp@uncc.edu, galinezh@uncc.edu, and htabkhiv@uncc.edu respectively.
- Arnab A Purkayastha is with Western New England University, MA USA. E-mail: arnab.purkayastha@wne.edu.
- Jagannadh Vempati and Martin Otto are with Siemens Technology, Princeton, NJ, USA. E-mail: jagannadh.vempati@siemens.com and m.otto@siemens.com respectively.
related to others by links and interactions. This has led to the whole research area exploring the use of graphs in this context.

Existing surveys mainly deal with either anomaly detection as a stand-alone problem or applied to a particular domain, for example, social media interactions, network traffic, recommendation systems, etc. Several works focus on anomaly detection based on graph learning [9]. Several surveys focus on specific applications of anomaly detection e.g., [10]. A few of the papers focus on anomaly detection on big data, and its challenges [11]. In addition, there are plenty of surveys that deal with general anomaly detection using deep learning techniques [12], [13], [14], [15]. However, none of these works explore the utilization of graphs for anomaly detection in heterogeneous distributed systems in the real-world as stand-alone research.

The major focus of this survey is to provide a comprehensive overview of the state-of-art graph-based techniques to solve the problem of anomaly detection. In particular, we look at real-time complex distributed systems and qualitatively model those to identify and analyze various methods in anomaly detection that utilize the benefits of graphs. In a nutshell, we make the following notable contributions in this survey:

- Modeling three conceptual use-cases and employing them for exploring the requirements, challenges, and benefits of anomaly detection algorithms.
- Identifying the logical, algorithmic, and implementation requirements and challenges of anomaly detection in real-world heterogeneous distributed systems.
- Comparing and contrasting the different graph-based anomaly detection algorithms in-depth for dealing with anomalies and qualitatively model them over the three conceptual use-cases that we introduce.
- Providing future research direction for graph-based anomaly detection in distributed heterogeneous systems.

The remaining of this paper is organized as follows: Section 2 describes three use-cases that are the representatives of real-world heterogeneous distributed systems. Section 3 identifies the challenges and requirements of anomaly detection in the context of real-world heterogeneous distributed systems. Next, Section 4 argues the benefits of moving toward using graphs in real-world heterogeneous distributed systems. Section 5 aptly delve into the graph-based approaches and the means of utilizing graphs for anomaly detection. Finally, Section 6 discusses the evaluation and complexity of these methods, the new challenges and requirements that graphs add, and the future directions that this research field can take.

2 Conceptual Use-cases

In order to motivate and put into perspective our research survey, we identify and present three conceptual use-cases of real-world applications here. These three conceptual use-cases are good samples of heterogeneous distributed systems, prone to showing anomalous behavior. In the following sections, for better understanding, we leverage these examples to demonstrate the limitations and benefits of the existing work.

2.1 Smart Surveillance System

Here we use an example from [12]. Real-time Edge Video Analytics for Multi-camera Privacy-aware Pedestrian Tracking or REVAMP[T] [12] is designed for tracking pedestrians across multiple cameras. The overview of the model is shown in Fig. 1. The installed edge nodes at each place of interest consist of a number of edge cameras empowered for analyzing streaming video. These cameras are connected to an edge server that acts as a database that contains information about seen objects. The video can be sent to a number of surveillance monitors with the extracted information if necessary and allowed. Also, the whole information can

| Network                  | Sources of Anomaly                                      |
|--------------------------|--------------------------------------------------------|
| Power Generation and Distribution | Anomalies in hardware, software, and the network |
| Smart Video Surveillance | Abnormal and dangerous events, hardware or software issues |
| Smart Cities             | Fault in infrastructure, safety and privacy issues       |
| Smart Grids              | Faults, cyber attacks, natural disturbances               |
| Social Network           | Illegal activities, bullying, spams, offensive content    |
| Telecommunication        | Malfunctions, hardware issues, intrusion                 |
| Company’s Internal Network | Hardware and software issues, unwanted access, intrusion |
| Factories’ Production Line | Fault in manufacturing, problem in manufacturing line equipment |
| Smart Transportation     | Technical issues, hardware and software issues, abnormal trajectory changes |
| Banking System           | Money laundering, fraud transactions, intrusion          |

TABLE 1
Example of distributed heterogeneous systems with sample source of anomalies

Fig. 1. REVAMP[T] setup. This setup can be expanded or shrunk to a new setup. Each edge server only sees the data of its own location and the cloud server is able to see all the locations connected to it. For privacy purposes a supervised amount of information is transmitted from each location to the cloud server.
be sent to a cloud server for further processing of less sensitive information. An anomaly can happen throughout this whole complex distributed heterogeneous system. For example, a camera might get broken, or even there can be a hacker trying to steal the extracted information from the edge server.

This heterogeneous distributed system is completely dynamic, leading to a dynamic graph representation. Benign changes like changing, removing, or updating the components can happen in this system, which should not be counted as anomalies. Other events such as receiving noisy video from a camera, information mismatch between the edge server and the cloud server, unwanted access to data at any point of the network can be seen as anomalies.

2.2 Sensor Network

We next look at the conceptual use-case of a heterogeneous Sensor Network that illustrates a complex distributed system. To accurately depict the security aspects of such a system, we take inspiration from the Purdue Model for ICS security [13]. The Purdue Enterprise Reference Architecture was created by mapping the interconnections and inter-dependencies of the high-level components of typical industrial control systems (ICS) to provide guidance on how to defend IT (Information Technology) and OT (Operations Technology) systems against malicious actors. Fig. 2 shows the model of such a system modified to incorporate all the possible components of a heterogeneous system. Here, Level 0 (L0) through Level 4 (L4) are the functional levels that span two broad zones, namely IT and OT. OT systems can be further classified into two sub-categories which will be discussed as follows.

- **L0**: Physical components that build products. Eg. motors, pumps, sensors, valves, etc
- **L1**: Systems that monitor and send commands to devices at L0. Eg. PLCs, RTUs, IEDs etc
- **L2**: Overall process controllers. Eg. HMIs, SCADA etc involves humans to manage and control the processes
- **L3**: Management of production workflows. Eg. batch management, operations management etc
- **L4**: ERP software, databases, email servers and other systems that manage the logistics of the manufacturing operations and provide communications and data storage

The anomaly tri-junction barrier (shown in red) serves as a demarcation of communication between the various levels and possible sources of anomalies.

2.3 Internal Local Area Network

An internal Local Area Network (LAN) is also a good sample model of a complex distributed heterogeneous system. The heterogeneity of the system and data is obvious. The graph for representing the system is dynamic and directed. It is dynamic because all the components in the network are subject to change. For example, clients can change their devices, the hardware of the server can change (e.g., adding more memory or upgrading the CPU), the mediums for connections can change, a client can get removed, or the framework of each platform can get updated.

Furthermore, all instances of anomalies can happen in this network. To emphasize, benign changes to the system must not be counted as an anomaly. We are dealing with a dynamic system that is subject to change, and it does not have a unique correct state. Thus, capturing anomalies is a more difficult task than when we are dealing with static systems.

3 CHALLENGES

When it comes to anomaly detection, especially in real-world systems, a number of requirements must be met. These requirements usually lead to challenging problems. In this section, we identify the generic requirements and challenges of anomaly detection in real-world distributed heterogeneous systems. We look at these challenges from four aspects, each of which examines anomaly detection from an alternative standpoint: Data, Anomaly Nature, Graph, and Real-world aspects. These viewpoints may overlap with each other at certain points. With that being said, we tried to distinguish them as much as possible.

Table 2 summarizes all the requirements and the challenges that follow them. In the rest of this section, we discuss each one of them in more detail.

3.1 Data and Evaluation Aspect

1. **Big Data**: When it comes to anomaly detection with learning approaches, the data becomes an immediate challenge. In many applications, the amount of data that has to be processed is immense, pushing the problem into the big data category. Big data meaning is best described through 5Vs of big data: Volume, Velocity, Variety, Veracity, and Value, each exhibiting a series of challenges. For anomaly detection, especially in distrusted heterogeneous systems, a huge amount of data (Volume) is being generated and processed at a very high speed (Velocity), containing multiple data types and formats coming from several sources (Variety), with uncertainty about the data and its quality (Veracity),
### Table 2
Challenges of Anomaly Detection.

| Category                  | Challenge                          | Short Description                                                                 |
|---------------------------|------------------------------------|-----------------------------------------------------------------------------------|
| Data and Evaluation       | Big Data                           | Volume, velocity, variety, veracity, and value of big data can be problematic.     |
|                           | High-Dimensionality                | Many numbers of features for each datapoint can cause sparsity of data which makes anomaly detection more arduous. |
|                           | Labeled Data                       | The number of labeled datasets are alarmingly low for anomaly detection purposes.   |
|                           | Unbalanced Data                    | Anomalies happen rarely, and training on such unbalanced data makes the model biased. |
|                           | Unclean Data                       | Many proposed models work based on the hypothesis that clean data is available which is not realistic. |
|                           | Metrics and Benchmarks             | Lack of Proper Metrics and benchmarks makes the assessment of proposed models difficult and impossible. |
| Nature of Anomaly         | Changing Nature                    | Novel and new types of anomalies can happen, and anomalies may adapt themselves with anomaly detection algorithms. |
|                           | Disparate Sources                  | Anomalies can happen in many different layers and parts of systems.                |
|                           | Obscured Anomalies                 | Outliers can be in disguise, especially the anomalies that are caused by a smart entity. |
|                           | Noise Resilience                   | In many cases, noises in the input data can mislead the models and be detected as false positives. |
| Real-World                | Inference Time                     | In real-world applications, it is vital to detect anomalies in a timely manner to take the appropriate action. |
|                           | Privacy                            | In many applications, the data of the users should be protected.                   |
|                           | Ciphered Data                      | In many environments, the exchanged data is ciphered.                              |
|                           | Dynamic Systems                    | Real-world systems often change over the time and have dynamic nature.             |
|                           | Interpretability                   | In real-world setups, the cause of the anomaly is important.                      |
|                           | Domain Shift                       | The model trained on one domain, cannot generalize even to a slightly different domain. |

2. **High Dimensionality**: This is another challenge that arises from the data. In many applications, the data that should be processed has a large number of features (dimensions). Huge dimensions of the dataset can cause numerous challenges in many applications. However, these challenges are exacerbated in the context of anomaly detection. The phrase “curse of dimensionality” generally refers to the problems that arise when the number of dimensions increases. A dataset is high-dimensional when we can see the curse of dimensionality. The growth of dimensions will increase the size of the data accordingly and causes sparsity which eventually results in the data points having relatively the same distance from each other. As a result, it would be more difficult to detect hidden anomalies in high-dimensional space. This issue is still a controversial topic in scientific society.

3. **Labeled Data**: In data-driven approaches, the quality of the data has a vital role in the outcome of the model. When it comes to anomaly detection, the number of labeled datasets is alarmingly low. This fact encourages the use of unsupervised or semi-supervised approaches. Thanks to the new technologies and the advancements in devices, a huge amount of data is available. However, in most cases, this data does not incorporate information regarding anomalous behavior. On the other hand, it is both expensive and time-consuming to obtain high-quality labels for available data. In many cases, anomalies must be discovered and labeled by experts who have field-specific knowledge, making the task harder and more demanding. Also, we should consider the noises added to the dataset by imprecise labeling. In many applications, this noise is not acceptable and can seriously degrade the performance of the model. The undetected anomalies can have a huge cost in critical situations of real-world applications. Thus, using supervised learning is somewhat problematic in the context of anomaly detection. This being said, labeled data can perform better in
certain tasks and should be explored. Several approaches try to overcome this challenge by taking advantage of synthetic labeled datasets, but many other approaches have moved toward using semi-supervised, unsupervised, and self-supervised algorithms.

4. **Unbalanced Data**: Another point to be mentioned is that anomalies are rare. Supervised learning approaches need to see enough examples while training to learn and understand anomalies. However, the number of anomalies is usually very subtle, causing the dataset to become unbalanced. Anomalies are out of the ordinary incidents, and it is not rational to expect enough samples of each type of anomaly to be available in the dataset. This challenge, requires the direction of solving the problem to adapt to it. The abundance of normal data points may result in neglecting the detection of anomalous data points which are valuable in anomaly detection. This problem arises from the fact that many machine learning algorithms are based on the hypothesis that classes of data have the same distributions which is not realistic in the case of anomaly detection.

5. **Unclean Data**: Many approaches, such as anomaly detection with autoencoders (discussed in Section 5) need datasets that only contain normal datapoints for training. These approaches learn the normal state and decide whether an anomaly has happened or not based on the divergence from features of normal datapoints. However, as discussed, anomalies can be rare and very subtle. There is a very high chance that an anomalous datapoint slips through the fingers of the dataset responsible individual or team, or it might not even be considered as an anomaly until later. As a result, often datasets include undesired outliers that make the dataset unclean. This unclean data can confuse the model about the features and representation of the normal datapoints and deteriorate the accuracy of the model.

6. **Metrics and Benchmarks**: On top of all the challenges, the most important one seems to be the fact that there is no unified metric or benchmark, where models can assess themselves and compare the results. Various kinds of metrics have been introduced and we discuss a number of important ones in Section 6, but almost none of the works assess those metrics on a single unique domain, to make the comparison between models possible.

### 3.2 Anomaly Nature

1. **Changing Nature**: Anomalies have a specific nature and a collection of characteristics that lead to a particular set of challenges. The first and the most important challenge is the changing nature of anomalies. Anomalies are not predefined static occurrences. Anomalies can transpire in different shapes and formats, not all of them can be considered prior to the occurrence of an anomaly. There can always be novel anomalies that have never been considered. They can adapt to anomaly detection mechanisms and evolve to pass through the defenses of anomaly detection algorithms. Because of this characteristic, most algorithms tend to skew toward online learning, unsupervised learning, or semi-supervised learning, where there are no predefined anomalies, and the algorithm learns based on what it sees at the moment on its own. In other words, most of the algorithms make an effort to eliminate external artificial biases and proceed toward generic solutions. This instigate the use of online learning, where the algorithm is constantly learning about the behaviour of the system.

2. **Disparate Sources**: Distributed heterogeneous systems typically incorporate multiple layers (resources), resulting in a complex and intricate system as a whole. The role of each of these layers, disregarding its majority or minority can not be neglected. Anomalies can penetrate each and every one of those components, making it very difficult to track them. Especially in large facilities, the number of components grows exponentially, and all of them are a potential point of anomaly. This disparate source of anomalies is one of the challenging complications that is troubling the anomaly detection algorithms.

3. **Obscured Anomalies**: In addition to these challenges, a considerable number of anomalies, predominantly the ones that came to existence because of a smart entity, do not like to be found. They might be disguised, or hidden. For example, a cyber-attack tries to be as masked as possible. On the other hand, there are anomalies that are not concealed and are just the result of a malfunctioning component in the system. Being conscious of both of these aspects, and prepared for obscured anomalies is another challenging task on the way to generic anomaly detection.

4. **Noise Resilience**: Systems are subject to experience a profuse number of internal or external noises. These noises are not anomalies until they pass a certain threshold. However, they are still an irregularity in the system. The anomaly detection algorithms must be capable of distinguishing between noises and anomalies. If not, they might raise too many false positive alarms, resulting in supererogatory attention and costs. These noises can have a wide variety of origins. Some of them are back-breaking to discover, and it is best to introduce a means of filtering those noises out inside the algorithm itself, instead of putting an effort to find the source of the noise.

### 3.3 Real-world Challenges

1. **Inference Time**: When it comes to using anomaly detection, time is of the essence, especially in real-world applications. It is expected, time-wise, that the detection of the anomaly does not sit far from the actual happening of the anomalous behavior. Particularly, in a critical environment such as health-care applications, a long latency in detecting a malfunctioning device might lead to catastrophic results, and it is impermissible. Thus, necessary actions must be taken to avoid such a latency. However, anomaly detection techniques based on deep learning are often a heavy task, requiring a colossal amount of computational power.

2. **Privacy**: Privacy is another important hindrance that must be considered in a real-world environment. Anomaly detection algorithms are involved with all the data circulation and the state of the system to the furthest extent. They know everything about a system, and if that kind of information leaks out of the environment, it can lead to cataclysmic undesirable results. Hence, an anomaly detection algorithm must be obliged to preserve privacy and be as secure as possible if it desires to be deployed in a real-world environment.

3. **Ciphered Data**: The fact that anomaly detection algorithms need to be aware of everything that is going on
related to a system brings about another challenge. Many systems, first and foremost large heterogeneous distributed systems like financial systems, social media, or facilities like power generation and distribution networks do not intercommunicate in plain, raw data. They encode the data and then transmit it. This ciphered data usually is not related to the original data in an obvious manner and that is the whole purpose of encoding the data. Hence, algorithms face yet another arduous challenge. For example, wireless communication between a sensor and the server contains an encoder at the edge transmitter, ciphering the data for privacy and security reasons, and a decoder at the receptor that deciphers the data for further processing. When an algorithm tries to discover anomalies on the medium (here the wireless connector) it cannot access the raw data before the transmission stage.

4. Dynamic Systems: Furthermore, static systems are extremely rare in a real-world environment. Real-world heterogeneous distributed systems are extremely dynamic. Components get removed, added, changed, or updated. These benign transformations must not be counted as anomalous behavior. When an algorithm learns about the normal state of a system, it might consider these transformations as anomalies. If so, the number of false-positive alarms might rise to a point that makes the algorithm non-functional and impractical for real-world adoption. On the other hand, the anomaly detection model should be flexible enough to learn about the updated features and structure of the network to fully incorporate existing information. Online learning is a tool for overcoming this challenge, however, this is still a very hot topic in research communities.

5. Interpretability: Another important challenge is the interpretability of the anomalies. Particularly, in real-world deployments the tendency to see the actual type of anomaly and not just the detection is more desired. One reason behind this is that recognizing the source of the anomaly, realizing the actual type of the anomaly (e.g., cyber attack on sensor 1A) and informing the responsible individuals is very important. Finding the root of the anomaly is important for maintaining the availability of the system. Furthermore, there is still the need to further investigate the abnormal behaviour to recover the system or for pulling it back to the normal situation.

6. Domain Shift: Last but not least, anomaly detection approaches become handy, when you are able to generalize over different domains. However, since we are dealing with learning-based approaches, this domain shift comes with a huge cost. Neural Networks learn about the environment and adapt to it, making it difficult to switch to another domain. This fact discourages the use of supervised learning and also pushes novel techniques toward online learning. This challenge is shared between graph-based anomaly detection and other approaches.

4 Motivation: Why Graph-based?

Non-deep learning algorithms for anomaly detection can be categorized into three general groups of methods: statistical methods, clustering methods, and nearest neighbors methods. In statistical algorithms, a statistical model that fits normal data is generated. Inference tests and data distribution is then carried out to calculate the anomaly scores and determine if there are outliers in the data. Statistics-based techniques have a significant advantage by reducing the dependency on labels. However, an increase of dimensionality in data leads to quadratic model complexity which may result in over-fitting. Clustering methods also are unsupervised techniques that try to group similar datapoints without any labels. As mentioned before, in real-world distributed systems, high-dimensionality of the data is inevitable which can seriously degenerate the performance of clustering methods. Finally, nearest neighborhood-based anomaly detection algorithms define the neighborhood viz., use distance or similarity measure. The distance or relative density between data and neighborhood is used as an anomaly score to judge anomalies. The benefit of the neighborhood-based approach is that it relaxes the assumptions on data distribution and prior knowledge. On the other hand, neighborhood-based approaches suffer from large computational and storage overhead.

Deep learning-based algorithms have proved to be more effective and generally show better results in terms of measurements such as accuracy or AUC (Area Under Curve) in comparison to traditional machine learning models. The general benefits of deep learning are applicable to deep anomaly detection. They are more suitable for learning complex information, their result is more generalizable and less prone to over-fitting, and they are model free. Especially when the number of datapoints increases, they are capable of learning complex relationships. However, the baseline version of all of these methods has a major drawback: they do not take advantage of the relationships between variables and parameters. This often results in the inadequate fitting of high-dimensional parameters leading to poor anomaly detection capabilities. Particularly in distributed heterogeneous systems, this relationship becomes even more important. Take the Smart Surveillance System shown in Fig. as an example. The anomaly can happen to Edge 0, the connection between Zone 0 and Edge Server 0, Zone 0 itself, or the combination of all these sections in this system. Here, graphs come to our aid.

Graphs are a common way to represent complicated relationships within a complex network. They enable us to store the data of every component and its relation with other components in the system. The use of graphs for anomaly detection is a very recent topic of interest in the machine learning and deep learning community. These approaches offer state-of-the-art performance on a varied nature of learning tasks that deal with complex distributed data. They suit well for both structured (symbols, images, grid-based data) and unstructured (knowledge graphs, social network data, distributed systems, citation networks, network traffic) that can be easily represented by regular and/or irregular graph structures. Moreover, the discriminative nature of graphs presents numerous opportunities to exploit the full potential of relationships between graph parameters and data. Graph-based anomaly detection advances our classification into four categories: Node, Edge, Sub-graph, and Graph. Referring back to the Smart Surveillance System Fig. an anomaly in Edge 0 is Node Anomaly, the connection of Zone 0 and Edge Server 0 is Edge Anomaly, Zone 0 itself is Sub-graph Anomaly, and finally, the problem in the whole system is a Graph Anomaly.
TABLE 3
Summary of Reviewed Approaches and Their Capabilities

| Approach                          | Model       | Attributed Graphs | Dynamic Graphs | Learning Adaptability | Scalability |
|----------------------------------|-------------|-------------------|----------------|-----------------------|-------------|
| Federated Learning               | DIO: [112]  | x                 | ✓              | S                     |             |
|                                 | VFL [113]   | x                 | ✓              | U                     | ✓           |
|                                 | [114]       | x                 | x              | U                     | ✓           |
|                                 | [115]       | x                 | x              | U                     |             |
|                                 | [116]       | x                 | x              | U                     |             |
| Autoencoders                     | Robust Deep Autoencoder (RDA) [117] | x | x | U | |
|                                 | Iterative Training Set Refinement (ITSR) [118] | ✓ | ✓ | U | |
|                                 | DONE [120]  | ✓                 | x              | U                     | ✓           |
|                                 | AdONE [120] | ✓                 | x              | U                     | ✓           |
|                                 | DeepSphere [121] | ✓ | ✓ | U | |
| Graph Embedding                  | DeepWalk [122] | x | x | U, O | ✓ |
|                                 | Node2Vec [123] | x | x | SM | ✓ |
|                                 | LINE [124]  | x                 | x              | -                     | ✓           |
|                                 | TADW [125]  | x                 | x              | -                     |             |
|                                 | NetWalk [126] | x | ✓ | U, O | |
|                                 | Graph Deviation Network (GDN) [98] | ✓ | x | U | |
|                                 | AddGraph [127] | ✓ | ✓ | SM | |
| Graph Transformers              | TADDY [129] | x                 | ✓              | U                     |             |
|                                 | GTA [130]   | x                 | ✓              | U                     |             |
| Graph Signal Processing         | GCCAD [132] | ✓                 | ✓              | U                     |             |
|                                 | CoLA [90]   | ✓                 | ✓              | SF                    | ✓           |
|                                 | SL-GAD [133] | ✓ | ✓ | SF | |

S: Supervised Learning, SM: Semi-supervised Learning, U: Unsupervised Learning, SF: Self-supervised Learning, O: Online Learning

Furthermore, graphs lead to the advantage of better visualization. Visualization is concerned with the visual representation of graphs that reveals structures and anomalies that may be present in the relational data and help the user to understand the graphs and the nature of anomalies.

In the next section, we discuss how we can utilize graphs for anomaly detection in detail.

5 GRAPH-BASED ALGORITHMS FOR ANOMALY DETECTION

In this section, we aim to introduce and compare techniques that combined with graphs have a great potential for solving the problem of anomaly detection in heterogeneous distributed systems. The summary of all reviewed works and their capabilities such as whether they can be used for dynamic or attributed graphs or not can be seen in Table 3.

5.1 Federated Learning (FL)

Federated Learning is a state-of-the-art technology that has recently emerged as an alternative to centralized systems. It focuses on collaboration while preserving the security aspect of client data information used for training machine learning algorithms [134], [135], [136], [137], [138]. It also greatly reduces communication overhead [139]. FL employs collaborative and experiential learning by training without the need to transfer data over to a centralized location. This feature has recently led to a rise in the applicability of FL in a variety of applications, ranging from medical to IoT, transportation, defense, and mobile apps.

Anomaly detection and prediction of static and dynamic time series data have been a popular topic, especially for IoT-based data [140], [141], [142]. These approaches look at multi-sensor systems as a collection of centralized sensors where abnormal sensor behavior is detected by a central model that runs on the server. Such systems are prone to failures, require longer access times, and are vulnerable to malicious invasion attacks leading to a potential data breach from clients to the server [143], [144]. It looks at the data collected by IoT sensors for energy-efficient applications like HVAC in smart buildings and proposes a federated stacked Long Short-time Memory model (LSTM) on time series data generated by IoT sensors for classification and regression tasks. [112], on the other hand, proposes a self-learning distributed system for security monitoring.

While such approaches deal with the heterogeneity (concerns data, privacy, and task requirements) and Autonomy
enough yet for real-world applications. Federated Graph Learning (FGL) addresses data diversity, data security and real-time continues learning but it is not mature shortcoming, although FGL addresses data diversity, data models such as Diot [112], a crucial requirement is available for aggregation. One consolidated model gets transferred to the Federated cloud server. Each network has its own testing and training dataset, local storage and loss function. The server receives individual weights from all the networks local models and then aggregates them based on their useable graph attributes while sending back the global weights.

A general structure of a complex distributed system can be easily depicted using the Federated Learning framework (Fig 3). Here we show two different network clusters that have heterogeneous nature of data as well as systems. In this framework, the models are trained at the device level or client-side, where they are brought over to the data sources or devices for training and prediction. The updated values are sent back to the federated cloud server for aggregation. One consolidated model gets transferred back to the devices to enable tracking and redistribution of each model to various devices. During the training phase, the input is reconstructed in the output until reconstruction error is minimized, which calculates the threshold value. This threshold value decides whether the observed patterns are anomalous or not.

Table 3 summarizes the strong points and weaknesses of two discussed works DloT [112] and VFL [113]. Both of these works are unable to process the additional information provided by attributed graphs which in real-world scenarios are common. It should be mentioned that for supervised models such as Dlot [112], a crucial requirement is available labeled data that is hard to provide. Based on mentioned shortcomings, although FGL addresses data diversity, data security and real-time continues learning it is not mature enough yet for real-world applications.

5.2 Autoencoders

Autoencoders are typically made of two main modules; encoders and decoders. The encoder part is responsible for mapping the input space to a bottleneck latent space, and the decoder reconstructs the original input based on the latent representation. In order to find the best possible latent representation, they try to minimize the reconstruction error of the original input. Thus, the network will learn to preserve the most informative parts of the input features in the latent representation. They were traditionally used for dimension reduction prior to feeding the data to the main network, but nowadays, they have vast applications in information retrieval, image processing, and anomaly detection.

In the context of anomaly detection, autoencoders can be used as a tool for calculating anomaly scores. After training the autoencoder with data that does not contain anomalous points, if any outlier data points are fed to the network, it will perform poorly. The reconstruction loss of that data point will be larger compared to normal ones since the network is not familiar with this type of data. As a result, the reconstruction error can be used as a measure of deviation from normal data points. This approach has been used for detecting anomalies in high-performance computing systems [114] and has shown promising results. Convolutional Autoencoders can improve the parameter efficiency and training time since they have shared parameters. Works such as [115], [116] have made use of convolutional autoencoders for anomaly detection and achieved significant improvements. DeepSphere [121] proposed a method for detecting anomalous snapshots in a dynamic network. This work adopts a LSTM autoencoder with an attention mechanism. In the constructed hidden space, DeepSphere [121] learns a spherically shaped boundary around the encoded normal representations. As a result, the encoded representation of an unseen anomalous snapshot of the network will fall outside the hyper-sphere and is detected as an anomaly.

All the aforementioned methods are developed based on the hypothesis that clean data (data that does not contain...
anomalous points) is available for training, but in many real-world cases, we do not have enough data points that satisfy these constrain. Thus, the reconstruction error for anomalous points will be lower, and the accuracy of finding outliers will degrade. To solve this issue, Robust Deep Autoencoder (RDA) [117] inspired by Robust Principal Component Analysis (RPCA) uses a filter layer that separates anomalous data points of input data. By removing these noisy and anomalous data points, the network will be able to better reconstruct the normal data points. On top of RDA [117], they add an anomaly detection algorithm to test the effectiveness of the proposed method. Iterative Training Set Refinement (ITSR) [118] adapts adversarial autoencoder network [149] architecture to add a prior distribution to constructed latent representation and places anomalies to the regions with lower likelihood. By this means, the model will be robust against noises and anomalies in the training data. DONE [120] is another network proposed for detecting anomalous nodes in attributed graphs. DONE makes use of two parallel autoencoders; one for encoding link structure and another one for attributes of nodes. These autoencoders are trained to preserve proximity and homophily in the network. The proposed loss function is designed to minimize the contribution of outliers, and by minimizing the loss equation, the anomaly scores for each node are calculated. Finally, the top k nodes are reported as anomalous points. They also propose AdONE [120] that makes use of adversarial learning to be able to construct an outlier resistant network embedding.

Traditional autoencoder networks and convolutional autoencoders are limited to a fixed input length which makes them unsuitable for detecting anomalies in dynamic networks. Also, they are limited to Euclidean data; thus, they are unable to model complex relationships that may occur in a dynamic heterogeneous network. Graph Neural Networks (GNN) can overcome these issues and generalize the network. They can also incorporate multi-dimensional edge and node attributes. Convolutional graph autoencoders have been used in [119] for detecting anomalies. This model uses both node features and edge features. Also, the decoder module has two separate branches for node reconstruction and edge reconstruction. The final reconstruction loss is the combination of edge reconstruction loss and node reconstruction loss and is used as a measure for finding anomalies.

Fig. 4 shows the workflow for detecting anomalies using autoencoder networks in the LAN Cluster example mentioned in Section 2.3. In the case of anomaly detection in a complex heterogeneous system, such as the three conceptual models that we introduced earlier, after training the model with normal data, the graph of the network is fed to the encoder module to be transformed into the latent space. In this case, since we are dealing with a heterogeneous network, the encoder and decoder modules should be able to incorporate the edge and node information and aggregate them to get the most of the available information. Also, at any time, the graph of the network can change. Thus, it is vital to employ a network architecture that can handle dynamic graphs, such as Graph Neural Networks. The decoder will try to reconstruct the original graph of the network from the latent representation. Anomalous edges, nodes, or sub-graphs of the network will have a high reconstruction error. In the most basic approach, only a fixed threshold can be used to decide whether the edge, node, or sub-graph should be considered anomalous or not.

For more clear scrutiny, Table 3 shows the different features of reviewed works. As mentioned before, one important characteristics of autoencoder structure is that it is trained in an unsupervised manner that directly solves the problem of availability of labeled data. Also, the changing nature of anomalies will not be a problem in these types of models since they will learn normal behaviour of the system and anything different from that will be counted as an outlier.

5.3 Graph Embedding
The first obstacle that we face in many tasks like anomaly detection in large networks is finding a way to map the data hidden in the network graph into a low-dimensional space. To do so, there are many different approaches. In general, we can separate these methods into two different classes. The first type of encoders is “Shallow Encoders” which will transform each node of a graph to exactly one vector in the latent space. On the other hand, there are methods called “Deep Encoders” which get the use of more complicated networks and are able to generate more complex embedding compared to “Shallow Encoders.” In the following, we will see details of each of these methods and their strengths and weaknesses. After obtaining the latent representations, an anomaly detection algorithm can be used on top of the embedding network in order to detect outliers and anomalies.
5.3.1 Shallow Embedding

Shallow Embedding methods try to find a unique latent representation for each vertex of a graph. The main difference between different approaches to shallow embedding is in the definition of similarity function. The similarity function basically describes how relationships in the latent space are mapped to the original input space. DeepWalk [122] is a random walk approach for node embedding that tries to find the best embedding that preserves similarity. Each random walk is an unbiased sequence of nodes with a fixed length. Authors claim that random walks can be treated as sentences of a text since the frequency of occurrence of the nodes in random walks follows the power law. DeepWalk [122] tries to find the feature representation for each node such that it maximizes the likelihood of visiting nodes seen in the random walks starting from that particular node. Another method to be mentioned is Node2Vec [123]. This algorithm makes it possible to have more flexible random walks in order to obtain richer latent representations. It combines breath-first sampling and depth-first sampling together to introduce a flexible biased sampling strategy that allows local and global views of the network.

Similar approaches such as LINE [124], and TADW [125] have been proposed. One might think of these algorithms as finding a simple look-up table for assigning each node in the original graph to a latent representation. As a result, these kinds of methods have limitations when we want to apply them to large heterogeneous networks. The first limitation comes from the fact that each node has to have its own unique embedding and there are no shared parameters in these kinds of networks. As a result, when the number of nodes grows, the number of parameters will grow respectively, and we will need V \times D number of parameters where D is the dimension of latent space. Also, they are unable to generalize to unseen nodes, and we cannot use them in dynamic networks. NetWalk [126] solves the problem with the changing networks and dynamically updates the representations as the network evolves. This model tries to satisfy two constraints: clique constraints which minimize the pairwise distances between representations of vertices in each random walk to preserve locality and autoencoder constrain, which serves as a global constrain and minimizes the reconstruction error of input using the output embedding. Still, there are more issues with these types of graph embedding algorithms; another problem is that in complex heterogeneous systems, each node may have its own specific features, but “Shallow Encoders” are unable to make use of these node-specific features.

5.3.2 Deep Embedding

Most of the methods that we have discussed until now are not capable of modeling more complicated dependencies such as complex inter-sensor relationships, but “Deep Encoders” have more capacity and are able to build richer latent representations. Graph Deviation Network (GDN) [98] introduced a structure learning approach in which the graph edges are initially unknown and have to be learned. GDN [98] consists of four important components. The first component is Sensor Embedding which captures the characteristics of each node or sensor. In the next step, Graph Structure Learning Learns the complicated relationships between pairs of sensors and models them to the edges of the graph. After the construction of the graph, Graph Attention-Based Forecasting predicts the behavior of sensors in the future time step. Finally, Graph Deviation Scoring will compare these predictions and the actual values in each time step and identify anomaly points that deviate from expected values. AddGraph [127] is able to aggregate more information such as structural, temporal, and content features to be able to build a more powerful embedding for anomaly detection in dynamic graphs. This model makes use of a graph convolutional neural network (GCN) for capturing structural and content features. By adding Gated
Recurrent Units (GRU) with attention modules, AddGraph [127] makes it possible to combine the long-term and short-term states of each node. Finally, based on the representation that contains structural, temporal, and content features, AddGraph [127] computes the anomalous score for edges using a single layer network.

Smart Video Surveillance System (Section 2.1), Sensor Network (Section 2.2), and LAN Network (Section 2.3) all contain many nodes. As a result, Shallow embedding techniques are not adequate for modeling these kinds of networks. We should also consider that all of these three conceptual use-cases are also dynamic (as an example adding a new camera to the Smart Video Surveillance System changes the network); thus, methods like DeepWalk [122] and Node2Vec [123] are not suitable since they are unable to generalize to unseen nodes. Another important issue is that nodes and edges can have features with different schemes that are useful for anomaly detection. Consider the Sensor Network; in each time step, a huge amount of data is being generated by a large number of interconnected sensors. The data from each sensor can be related to other sensors with a complex non-linear relationship. Shallow embedding techniques fail to take advantage of this kind of information. Thus, Deep Embedding methods that are more complicated such as GDN [98], and AddGraph [127] can be helpful in this case. Based on what we discussed, let’s use one of the introduced examples discussed in Section 2 for better understanding. In the LAN Cluster example, AddGraph seems to be a suitable choice.

Fig. 5 shows the structure of AddGraph which is applied to the LAN Cluster example. Previous hidden states and current snapshot of the network graph is used for constructing the node embedding. Also, an attention module and GRU combine the long-term and short term states to generate the current hidden state. In the final step a scoring function is responsible to assign anomaly score to the current hidden state. The main points of discussed techniques are available in Table 3.

5.4 Graph Transformers

The concept of attention mechanism introduced by [150] was firstly used for natural language processing. This mechanism aims to do one-step prediction instead of recurrent processing of the data. By this means, the attention mechanism reduces the path length of the computation, which means reducing the information loss and focusing on the most important features for predicting the output. Adapting this technique to graphs can improve the outcome in many applications, such as anomaly detection. Graph transformer networks are able to automatically generate meta-paths by learning and solving the aforementioned problem enabling many applications such as anomaly detection to work on complex heterogeneous systems [151]. In [128], the model takes advantage of relational graph transformers for finding anomalous nodes in a supervised manner. In this work, first, the heterogeneous graph of the network is extracted, and relational graph transformers and a semantic attention network are used for modeling the complex relationships between the nodes and encoding them, and then classifying them for finding anomalous nodes.

Supervised learning is not possible in many applications due to the lack of labeled data. Using an unsupervised setting, the transformer-based Anomaly Detection framework for Dynamic graphs (TADDY) [129] detects anomalous edges in dynamic graphs. TADDY [129] consists of four main modules as shown in Fig. 6. Edge-based substructure sampling captures spatial-temporal context of each target edge using graph diffusion method [152], [153]. For each edge, this module constructs a fixed-length importance-aware set of neighboring nodes. Then, TADDY [129] leverages a novel spatial-temporal node encoding for generating...
node embedding. After acquiring the embedding from the previous step, a transformer network is used as an encoder to capture spatial and temporal features, followed by a pooling module for aggregating the embedding of all nodes in the same neighboring set. At the final step, discriminative anomaly detector (which consists of a fully connected layer).

Graph Learning with Transformer for Anomaly detection (GTA) [88], leverages a modified transformer network. This model detects anomalies on multivariate time series generated from different sensors. These sensors can be related to each other in complicated unknown connections. GTA [88] first learns the dependencies using Gumbel-Softmax Sampling strategy. Once this topological structure is established, a graph convolution block updates each node representation by aggregating neighbors’ information and message passing to enrich the representations. Now, dilated convolution [130] is used for extracting temporal context, but with a novel modification. They have used a hierarchical scheme to make GTA capable of capturing temporal patterns with different lengths. On top of these, a more sophisticated and efficient version of transformers is introduced. Multi-branch attention module that is used in GTA [130] extracts long-distance temporal dependencies and neighboring nodes’ information. In the anomaly scoring module, the original input time series are divided into training sequences (for the encoder) and label sequences (for the decoder). The decoder predicts the behavior of time series in the target section, and by comparing the predicted output and the actual values, outliers can be detected.

Capabilities of all mentioned models are mentioned in Table 3. Unfortunately none of the anomaly detection models using graph transformers are compatible with attributed graphs which makes them less practical in real applications. Future research can move in the direction of solving this issue to overcome the complex nature of heterogeneous distributed systems.

5.5 Graph Signal Processing

Graph Signal Processing (GSP) is another technique that utilizes classical signal processing tools like Fourier transform, filtering, frequency response etc., to process data defined on both regular and irregular graph networks. 155 presents the recent advances in the currently developing GSP tools.

Further, the emphasis is made on graph datasets arising from data science, biological, and sensor networks that can be indexed by nodes of graph signals connected via weighted edges. Graph signals can be filtered and sampled to apply low-level processing techniques such as denoising and compression. One of the major differences between GSP and traditional Machine learning algorithms is that ML typically considers a graph as a discrete version of a complex network. However, this assumption falls flat for many real-world applications associated with graphs, GSP, on the other hand, look at existing problems from different perspectives. As an example, defining a graph for sensor-based networks involves choosing edge weights as a decreasing function of the distance between nodes represented by sensors. Observations from similar nodes can lead to a smooth graph function that can detect outliers or abnormal values through high-pass filtering or thresholding [156]. Moreover, a sparse set of sensor readings can also be used to build signal reconstruction methods that can be used to save on resources in sensor networks [157], [158].

Anomaly detection with GSP filtering on Wireless Sensor Networks (WSNs) has been a topic of interest in several works. [131] captures proximity information such as data between sensors to capture local anomalous behavior. They present three graph designs and use GSP filtering to find the cut-off frequency, Lambda for the filters. This is used to separate normal and anomalous sub-spaces for unsupervised detection. The anomalous space projections are finally utilized to generate anomaly scores as shown in Fig. 7. The main advantage of such a system is that in addition to raw sensor data, relational characteristics between the nodes, like proximity information between sensors and their environment, can be effectively captured. Graph-based filtering is found to be particularly useful for both regular and irregular graph structures for unsupervised anomaly detection. In Table 3, specification of the reviewed model is available. The ability of working on dynamic attributed graphs makes [131] aligned with the real-world requirements. On the other hand, leveraging GSP techniques can add deeper perspectives to anomaly detection problem and equip it with more relational information helpful for detecting outliers which is essential real-world domain.
5.6 Graph Contrastive Learning

In graph contrastive learning techniques, the objective is to construct a representation by contrasting pairs of data points. The loss function is designed in such a way that by optimizing it, the positive pairs (matching pairs) of data points are brought together, and the negative pairs will be separated. As a result, using this approach, the model is able to learn higher-level representations that are more powerful and distinguishable. In the context of anomaly detection, contrastive learning can be helpful since it is designed to provide a measure of similarity between data pairs. The main idea in GCCAD [132] is to detect anomalies using their distance from average normal points or global context. The key concept is that the outliers will have different features than most points; thus, by contrasting each node with the global context, we can define a measure for detecting anomalies.

GCCAD [132] has shown that the node embedding made by traditional GNN encoders is not able to discern anomalous nodes properly. However, GCCAD [132] is able to construct a more powerful embedding that highlights the differences between anomalous nodes and the global context. This work uses a context-aware loss function in a supervised manner. The loss function is optimized in the GNN encoder that consists of three modules: edge update, node update, and graph update module. The edge update module is responsible for calculating the likelihood of being a suspicious edge (an edge that connects a normal node to an anomalous one) and removing it, and updating the adjacency matrix. These modules are designed to conserve the homophily assumption between neighboring nodes. This assumption suggests that the neighboring nodes have the same labels. Still, homophily is violated in the case of a connection between a normal node and an anomalous one, and this violation has not been considered in traditional GNN networks. In the next step, the node embedding is updated using message passings by the node update module. Lastly, the graph update module updates the global context. The updated global context is the weighted aggregation of all the nodes. The authors also introduced a self-supervised version of this network named GCCAD-pre [132].

CoLA [90] focuses on anomaly detection in large-scale attributed graphs, which is applicable to real-world problems. This novel network has three main components: pair sampling, a GNN-based contrastive learning model, and the score computation module. The instance pair sampling is designed to generate the pairs of data for the training phase. In contrast with GCCAD [132] that contrast nodes with the global context, the definition of pairs in CoLA [90] focuses on the relationship between a node and its neighbors. Hence the strategy for defining the pairs is “target node versus local subgraph”. The GNN-based contrastive learning module consists of three sub-modules: (1) GNN model, (2) read-out module, and (3) discriminator module. After acquiring the pairs, the GNN module will extract embedding for the target node and the local subgraph. Then the readout module uses the average pooling function to create a vector of embedding of all nodes in the embedded subgraph. In the discriminator module, the positive and negative pairs are contrasted, and a score is generated. Finally, The score computation module measures the anomaly score for all nodes, and anomalous nodes can be picked by choosing the nodes with the highest scores. SL-GAD [133], introduced a novel generative and contrastive self-supervised model.

SL-GAD [133] is made of three major components similar to CoLA [90]: graph view sampling, contrastive self-
TABLE 4
Time complexity of reviewed algorithms. For symbol descriptions please refer to table 5.

| Approach          | Model | Time Complexity | Description |
|-------------------|-------|-----------------|-------------|
| Federated Learning| DIoT  [112] | - | - |
|                   | VFL   [113] | $\mathcal{O}(nm^2)$ | - |
|                   | [114] | - | - |
|                   | [115] | - | - |
|                   | [116] | - | - |
| Auto Encoders     | RDA   [117] | - | - |
|                   | TSR   [118] | - | - |
|                   | DONE  [120] | $\mathcal{O}(Nd)$ | - |
|                   | AdOne [120] | $\mathcal{O}(Nd)$ | - |
|                   | DeepSphere [121] | - | - |
| Graph Embedding   | Deep Walk [122] | - | - |
| Shallow Encoders  | Node2Vec [123] | $\mathcal{O}\left(\frac{1}{2(l-s)}\right)$ | Per sample complexity |
|                   | LINE  [124] | $\mathcal{O}(\text{mid})$ | - |
|                   | TADW  [125] | - | - |
|                   | NetWalk [126] | $\mathcal{O}(\text{nl}||\text{nl})$ | The complexity of walk generation |
|                   | - | $\mathcal{O}(\text{md})$ | The complexity of edge encoding |
|                   | - | $\mathcal{O}(\text{cd})$ | The complexity of anomaly detection on incoming data points |
| Deep Encoders     | GDN   [98] | - | - |
|                   | AddGraph [127] | - | - |
| Graph Transformers| TADDY [129] | $\mathcal{O}(\tau kmI + Tn^2)$ | - |
|                   | GTA   [130] | $\mathcal{O}(4\tau D^2 + 2\tau^2 D)$ | This is for the simplest attention module. More complex ones are available in the original paper. |
| Graph Signal Processing | [131] | - | - |
| Graph Contrastive Learning | GCCAD [132] | $\mathcal{O}(m)$ | - |
|                   | CoLA  [90] | $\mathcal{O}(knR(\eta + c))$ | - |
|                   | SL-GAD [133] | $\mathcal{O}(Rnk(\eta + K))$ | - |

supervised learning, and graph anomaly scoring. In the graph view sampling, for each target node, two local subgraphs are extracted. Next, in the contrastive self-supervised learning module, a GNN constructs the latent representation of the samples. In addition to contrastive scores, SL-GAD [133] also leverages a graph autoencoder network for reconstructing the feature vector of the target node in order to fully utilize contextual information of target nodes. Finally, graph anomaly scoring predicts the final scores based on both generative scores (from the graph autoencoder) and contrastive scores (from contrastive learning).

Fig. 8 shows the work flow of the SL-GAD [133] applied to the LAN Cluster example introduced in Section 2.3. The sampled subgraphs are fed to the GNN Encoder and the encoded representations are fed to two different branches, the discriminative and the generative modules. at the final step, anomaly scores from these two branches are combined to construct the final anomaly score.

Characteristics of all mentioned model can be seen in Table 3. As discussed, Graph Contrastive Learning approaches show promising results and are extremely adaptable to real-world scenarios. The discussed works are all able to overcome the dynamic complex nature of distributed heterogeneous systems and unsupervised/self-supervised learning adaptability of these models can alleviate the problem of available labelled data mentioned in Section 3.

6 EVALUATION, COMPARISON AND DISCUSSION

In this section, first, we discuss the metrics that are available for evaluation and then we try to compare the methods discussed in Section 5 from an algorithmic perspective. Then we discuss the requirements and challenges that utilizing graphs adds to previously mentioned requirements discussed in Section 3 and finally, we discuss the future directions that research in this field can take.

6.1 Metrics and Comparison

The performance and viability of any anomaly detection algorithm, graph-based or non-graph-based, can be eval-
TABLE 5
Table symbol description.

| Parameter | Description |
|-----------|-------------|
| n         | number of nodes in the graph |
| ˜n         | average number of nodes in a timestamp |
| m         | number of edges in the graph |
| k         | number of nodes in a subgraphs |
| ℓ         | number of nodes in a subgraphs |
| T         | number of timestamps |
| τ         | size of the time window |
| ˜n         | number of negative samples |
| l         | length of a walk |
| | number of walks |
| s         | number of samples |
| D         | number of input dimension |
| d         | latent dimension of node representation |
| d         | degree of a node |
| c         | number of clusters |
| R         | number of evaluation rounds |
| I         | number of iterations |
| ˜R         | number of sampling rounds for each node |

utilized by Precision, Recall, Accuracy, Receiver Operating Characteristic (ROC) curve, and Area Under the ROC Curve (AUC). While Precision, Recall, and Accuracy are a measure of the true positives, false positives, true negatives, and false negatives, AUC, on the other hand, summarizes the information contained in the ROC curve. Larger AUC values indicate better performance at distinguishing between anomalous and normal observations.

A major challenge while reporting the above-mentioned metrics is the absence of unified evaluation criteria amongst the diversity of algorithms, the application type, and the heterogeneous nature of data and devices involved. As an example, if we want to compare the latency of the two algorithms, we have to make sure that they are both tested on particular hardware, they are getting tested on a specific job, and we have to eliminate different variables that have an effect on that metric.

In contrast, Time Complexity is the only metric that can give us some insight into the algorithm’s performance. Table 4 shows the time complexity of reviewed algorithms. In order to assess how well an algorithm performs compared to other algorithms, and also with respect to points in Section 3, we can refer to Time Complexity. It gives us a good understanding of how fast and efficient the method is, a general idea of the algorithm’s complexity, and whether it is suitable for specific tasks or hardware based on the power it needs.

6.2 Graph-specific Challenges

Graphs add an enormous number of benefits to our systems. They are extremely powerful for catching relational information, especially in distributed heterogeneous systems. They help with addressing challenges and meeting the requirements discussed in Section 5 as well as capturing important features of the data like relational features and information. However, there is a cost to all these benefits. While utilizing graphs these challenges should be addressed:

**Types of Graphs:** Each system can be represented by a specific type of graph. Each type of graph has its features and is suitable for a group of domains. For example, attributed graphs add more complexity by introducing the features of the connection between the nodes, and the features of the nodes themselves. Also, a graph representing a system might be dynamic or static, depending on the nature of the original system. All these sub-classes of graphs add to the complexity of graph-based learning techniques. Subsequently, the introduced models should be able to handle these diverse types of network graphs, but as we discussed in previous sections, many of the models can just work on plain static graphs which is not satisfactory in many applications. [159], [160], [161], [162], [163], [164]

**Graph Anomalies:** By introducing graphs, we get a better representation of a system. Graphs and Graph Neural Networks make us capable of modeling complicated dependencies in real networks. But if we look at it from another perspective, this adaptation of graph representation introduces different places for anomalies to take place. In graphs, outliers can occur in nodes, edges, subgraphs, and even the whole graph. But models are usually only able to detect one kind of foresaid anomalies, and there is no unique framework to address them all together. This means that the approaches must specifically determine what is the objective of their detection, which types they handle, and where they show the best performance. [95], [165], [166]

6.3 Future Research Directions

Graph-based anomaly detection is a relatively new topic with many opportunities drawing a fair amount of attention. Here we discuss a few excellent opportunities and novel directions that can be the subject of future research independently or in conjunction.

The most important research direction at the moment seems to be finding a means of quantifying the results of anomaly detection algorithms in a unified manner, thereby enabling comparison in various domains and fields. A great share of all the advancements in technology originates from quantifying and comparing new findings with already existing works. However, considering the variety in this field, comparing algorithms to each other is nearly impossible. First, the best metric for evaluating a model must be identified. Alongside that, an evaluation mechanism, general enough to be applicable to disparate anomaly detection algorithms are also needed. As an example, in computer vision, COCO [167] provides such an environment for emerging algorithms. A framework of testing can also be another fruitful tool. In this direction, creating a simulator, or a system that can serve as a toy sample of a large distributed system can be beneficial as well. To recapitulate, there is a need for a unified evaluation of graph-based and non-graph-based anomaly detection algorithms and techniques.

As discussed in the Challenges Section, the second-largest issue in the way of anomaly detection is the lack of a suitable dataset. Generating or preparing a proper dataset can be a substantial contribution to the field of anomaly detection. However, in the context of anomaly detection,
usually, an expert must supervise the process of labeling and collecting data and this fact makes the generated dataset prone to human errors. All of these are added to the actual cost of creating such a dataset. Synthetic datasets are another option, and creating them can be explored through a whole line of research.

Numerous methods of anomaly detection are designed to discover whether an anomaly has happened or not. They are able to notify a responsible person or group to look more for further investigation. However, they are not able to identify what sort of anomaly has occurred. A major subject for further work and research is the interpretability of anomalies. Down the road, there is a tendency to distinguish between anomalies using the algorithms at hand. For example, in the Sensor Network mentioned in [2], we like the model to be able to distinguish between a cyber attack and power shortage and raise the correct alarm. If we go further than that, the ways that the model can restore the system or stop the anomaly can be investigated.

7 Conclusion

In this survey, we have discussed several state-of-the-art graph-based approaches for anomaly detection in details. We introduce three unique conceptual use-cases based on real-time complex distributed systems. Our work builds on the existing traditional machine learning and deep learning paradigms used for anomaly detection and goes on to introduce four separate categories of graph neural network algorithms. We further apply our formulated models to each of the methods and provide a thorough review and summarization of the categories. At last, we comprehensively look at graph specific big-data challenges and provide a couple of future research directions in this field.

Acknowledgments

This work is part of a research collaboration with Siemens Inc., USA.

References

[1] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” ACM computing surveys (CSUR), vol. 41, no. 3, pp. 1-58, 2009.
[2] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, “Deep learning for anomaly detection: A review,” ACM Computing Surveys (CSUR), vol. 54, no. 2, pp. 1-38, 2021.
[3] G. Pang, L. Cao, and C. Aggarwal, “Deep learning for anomaly detection: Challenges, methods, and opportunities,” in Proceedings of the 14th ACM International Conference on Web Search and Data Mining, ser. WSDM ’21. New York, NY, USA: Association for Computing Machinery, 2021, p. 1127–1130. [Online]. Available: https://doi.org/10.1145/3437963.3441659
[4] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in International Conference on Learning Representations (ICLR), 2017.
[5] 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,” IEEE Transactions on Knowledge and Data Engineering, 2021.
[6] S. Ahmed, K. Hinkelmans, and F. Cerradini, “Combining machine learning with knowledge engineering to detect fake news in social networks—a survey,” arXiv preprint arXiv:2201.08032, 2022.
[7] R. A. A. Habeeb, F. Nasaruddin, A. Gani, I. A. T. Hashem, E. Ahmed, and M. Imran, “Real-time big data processing for anomaly detection: A survey,” International Journal of Information Management, vol. 45, pp. 289–307, 2019.
[8] S. Thudumu, P. Branch, J. Jin, and J. J. Singh, “A comprehensive survey of anomaly detection techniques for high dimensional big data,” Journal of Big Data, vol. 7, no. 1, pp. 1–30, 2020.
[9] R. Chalapathy and S. Chawla, “Deep learning for anomaly detection: A survey,” arXiv preprint arXiv:1901.03467, 2019.
[10] T. Ruff, J. R. Kaufmann, R. A. W. Vlaselemu, C. Monatova, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Müller, “A unifying review of deep and shallow anomaly detection,” Proceedings of the IEEE, 2021.
[11] G. Fernandez, J. J. Rodrigues, L. F. Carvalho, J. F. Al-Muhtadi, and M. L. Proença, “A comprehensive survey on network anomaly detection,” Telecommunication Systems, vol. 70, no. 3, pp. 447–489, 2019.
[12] C. Neff, M. Mendieta, S. Mohan, M. Baharani, S. Rogers, and H. Tabki, “Revamp 2 t: real-time edge video analytics for multicamera privacy-aware pedestrian tracking,” IEEE Internet of Things Journal, vol. 7, no. 4, pp. 2591–2602, 2019.
[13] T. J. Williams, “The purdue enterprise reference architecture,” Comput. Ind., vol. 24, no. 2–3, p. 141–158, sep 1994. [Online]. Available: https://doi.org/10.1016/0166-3315(94)90017-5
[14] E.-Y. Song and Y. Zhu, “Big data and data science: what should we teach?,” Expert Systems, vol. 33, no. 4, pp. 364–376, 2016.
[15] A. Katal, M. Wazid, and R. H. Goudar, “Big data: issues, challenges, tools and good practices,” in 2013 Sixth international conference on contemporary computing (IC3). IEEE, 2013, pp. 404–409.
[16] K. Shin, B. Hooi, J. Kim, and C. Faloutsos, “D-cube: Dense-block detection in terabyte-scale tensors,” in Proceedings of the Twenty-Fourth ACM International Conference on Information and Data Mining, ser. WSDM ’17. New York, NY, USA: Association for Computing Machinery, 2017, p. 681–689. [Online]. Available: https://doi.org/10.1145/3018661.3018870
[17] E. M. Knorr and R. T. Ng, “Algorithms for mining distance-based outliers in large datasets,” in Proceedings of the 23rd International Conference on Very Large Data Bases, ser. VLDB ’98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, p. 392–403.
[18] S. Ramaswamy, R. Rastogi, and K. Shim, “Efficient algorithms for mining outliers from large data sets,” in Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, ser. SIGMOD ’00. New York, NY, USA: Association for Computing Machinery, 2000, p. 427–438. [Online]. Available: https://doi.org/10.1145/342009.335457
[19] A. Angiulli and F. Fassetti, “Very efficient mining of distance-based outliers,” in Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management, ser. CIKM ’07. New York, NY, USA: Association for Computing Machinery, 2007, p. 791–800. [Online]. Available: https://doi.org/10.1145/1321440.1321550
[20] A. Arning, R. Agrawal, and P. Raghavan, “A linear method for deviation detection in large databases,” in Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, ser. KDD’96. AAAI Press, 1996, p. 164–169.
[21] J. Gama, “A survey on learning from data streams: current and future trends,” Progress in Artificial Intelligence, vol. 1, no. 1, pp. 45–55, Apr 2012. [Online]. Available: https://doi.org/10.1007/s13748-011-0002-6
[22] K. W. Pettis, T. A. Bailey, A. M. Jain, and R. C. Dubes, “An intrinsic dimensionality estimation from near-neighbor information,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-1, no. 1, pp. 25–37, 1979.
[23] Q. Yu, K.-M. Tang, S.-X. Tang, and X. Lv, “Uncertain frequent itemsets mining algorithm on data streams with constraints,” in Intelligent Data Engineering and Automated Learning – IDEAL 2016, H. Yin, Y. Gao, B. Li, D. Zhang, M. Yong, Y. Li, F. Klavon, and A. J. Tallon-Ballesteros, Eds. Cham: Springer International Publishing, 2016, pp. 192–201.
[24] M. Kontaki, A. Gounaris, A. N. Papadopoulos, K. Tsichlas, and Y. Manolopoulos, “Continuous monitoring of distance-based outliers over data streams,” in 2011 IEEE 27th International Conference on Data Engineering, 2011, pp. 135–146.
[25] L. Breiman, “Bias, variance, and arcing classifiers,” Tech. Rep. 465, Statistics Department, University of California, Berkeley . . . , Tech. Rep., 1996.
[26] R. E. Schapire, “The strength of weak learnability,” Machine Learning, vol. 5, no. 2, pp. 197–227, Jun 1990. [Online]. Available: https://doi.org/10.1007/BF01160357
A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà, “New ensemble methods for evolving data streams,” in Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ser. KDD ’09. New York, NY, USA: Association for Computing Machinery, 2009, p. 135–144. [Online]. Available: https://doi.org/10.1145/1557019.1557041

N. Tatbul, “Streaming data integration: Challenges and opportunities,” in 2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW 2010), 2010, pp. 155–158.

R. Bellman, Dynamic programming. Chelmsford: Courier Corporation, 2013.

J. Ahn, J. S. Marron, K. M. Muller, and Y.-Y. Chi, “The high-dimension, low-sample-size geometric representation holds under mild conditions,” *Biometrika*, vol. 94, no. 3, pp. 760–766, 2007. [Online]. Available: http://www.jstor.org/stable/2044111

C. C. Aggarwal, A. Hinneburg, and D. A. Keim, “On the surprising behavior of distance metrics in high dimensional space,” in *Database Theory — ICDT’01*, J. Van den Bussche and V. Vianu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 217–235.

P. Hall, J. Marron, and A. Neeman, “Geometric representation of high dimension, low sample size data,” *Journal of the Royal Statistical Society Series B*, vol. 67, pp. 427–444, 02 2005.

A. Koufakou and M. Georgiopoulos, “A fast outlier detection algorithm for high-dimensional datasets,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 3, pp. 739–751, 2014.

R. Evans and E. Grefenstette, “Learning explanatory rules from flagged fraudsters,” in *Database Theory — ICDT’99*, C. Beerli and P. Buneman, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 217–235.

J. Fan and Y. Fan, “High dimensional classification using features mixed attributes,” *Data Mining and Knowledge Discovery*, vol. 20, no. 2, pp. 259–289, Mar. 2010. [Online]. Available: https://doi.org/10.1007/s10618-009-0148-z

J. Fan and Y. Fan, “High dimensional classification using features annealed independence rules,” *Annals of Statistics*, vol. 36, no. 6, pp. 2655–2678, 2008, 19169416[pmid]. [Online]. Available: https://publications.arpitjournals.com/doi/pdf/10.1214/08-AOS5941

F. Keller, E. Muller, and K. Bohm, “Hics: High contrast subspaces for density-based outlier ranking,” in 2012 IEEE 28th International Conference on Data Engineering, 2012, pp. 1037–1048.

N. Görnitz, M. Kloft, K. Rieck, and U. Brefeld, “Toward supervised anomaly detection,” *Journal of Artificial Intelligence Research*, vol. 46, pp. 235–262, 2013.

S. Suthaharan, M. Alazahrawi, R. Rajasegarar, C. Leckie, and M. Palaniswami, “Labelled data collection for anomaly detection in wireless sensor networks,” in 2010 sixth international conference on intelligent sensors, sensor networks and information processing. IEEE, 2010, pp. 269–274.

M. Luo, K. Wang, Z. Cai, A. Liu, Y. Li, and C. F. Cheang, “Using imbalanced triangle synthetic data for machine learning anomaly detection,” *Comput.* , *Mater. Continua*, vol. 58, no. 1, pp. 15–26, 2019.

W. Lin, J. Gao, Q. Wang, and X. Li, “Learning to detect anomaly events in crowd scenes from synthetic data,” *Neurocomputing*, vol. 436, pp. 248–259, 2021.

M. Frasca, A. Bertoni, M. Re, and G. Valentini, “A neural network algorithm for semi-supervised node label learning from unbalanced data,” *Neural Networks*, vol. 43, pp. 84–98, 2013.

S. N. Kalid, K.-H. Ng, G.-X. Tong, and K.-C. Khor, “A multiple classifiers system for anomaly detection in credit card data with unbalanced and overlapped classes,” *IEEE Access*, vol. 8, pp. 28 210–28 221, 2020.

H. H. Pajouh, G. Dastghaibyfard, and S. Hashemi, “Two-tier network anomaly detection model: a machine learning approach,” *Journal of Intelligent Information Systems*, vol. 48, no. 1, pp. 61–74, 2017.

S. El Hajjami, J. Malki, M. Berrada, and B. Fourka, “Machine learning for anomaly detection. performance study considering anomaly distribution in an unbalanced dataset,” in *2020 5th International Conference on Cloud Computing and Artificial Intelligence: Technologies and Applications (CloudTec)*. IEEE, 2020, pp. 1–8.

Z. Kang, H. Pan, S. C. Hoi, and Z. Xu, “Robust graph learning from noisy data,” *IEEE transactions on cybernetics*, vol. 50, no. 5, pp. 1833–1843, 2019.

R. Evans and E. Grefenstette, “Learning explanatory rules from noisy data,” *Journal of Artificial Intelligence Research*, vol. 61, pp. 64–183, 2018.

P. Niyogi, S. Smale, and S. Weinberger, “A topological view of unsupervised learning from noisy data,” *SIAM Journal on Computing*, vol. 40, no. 3, pp. 646–663, 2011.

N. Cesa-Bianchi, S. Shalev-Shwartz, and O. Shamir, “Online learning of noisy data,” *IEEE Transactions on Information Theory*, vol. 57, no. 12, pp. 7907–7931, 2011.

Y. Kim, J. Yim, J. Yun, and J. Kim, “Nlnl: Negative learning for noisy labels,” in *IEEE/CVF International Conference on Computer Vision*, 2019, pp. 101–110.

M. Frasca, A. Bertoni, M. Re, and G. Valentini, “Intrusion detection systems in the cicsid2017 dataset,” *IEEE access*, vol. 9, pp. 22 351–22 370, 2021.

M. Z. Zaheer, J.-h. Lee, M. Astrid, A. Mahmood, and S.-I. Lee, “Cleaning label noise with clusters for minimally supervised anomaly detection,” *arXiv preprint arXiv:2104.14770*, 2021.

S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, “Unsupervised real-time anomaly detection for streaming data,” *Neurocomputing*, vol. 262, pp. 134–147, 2017.

Z. K. Maseer, R. Yusof, N. Bahaman, S. A. Mostafa, and C. F. M. Foozy, “Benchmarking of machine learning for anomaly based intrusion detection systems in the cicsid2017 dataset,” *IEEE access*, vol. 9, pp. 22 351–22 370, 2021.

A. F. Emmott, S. Das, T. Dietterich, A. Fern, and W.-K. Wong, “Noisier2noise: A systematic construction of noyite detection benchmarks from real data,” in *Proceedings of the ACM SIGKDD workshop on outlier detection and description*, 2013, pp. 16–21.

C. R. Banbury, V. J. Reddi, M. Lam, W. Fu, A. Fazel, J. Holleman, X. Huang, R. Hurtado, D. Kanter, A. Lohkhomtov et al., “Benchmarking tinyml systems: Challenges and direction,” *arXiv preprint arXiv:2003.04821*, 2020.

G. Almezahed-Noghre, A. Danesh Pazho, J. Sanchez, N. Hewitt, C. Neff, and H. Tabluk, “Adg-Pose: Automated dataset generation for real-world human pose estimation,” in *International Conference on Pattern Recognition and Artificial Intelligence*. Springer, 2022, pp. 258–270.

G. Yu, Z. Cai, S. Wang, H. Chen, F. Liu, and A. Liu, “Unsupervised online anomaly detection with parameter adaptation for kpi abrupt changes,” *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1294–1308, 2020.

W. Wang, Q. Chen, T. Liu, X. He, and L. Tang, “A distributed online learning approach to detect anomalies for virtualized network slicing,” in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.

A. Guezzaz, Y. Asimi, M. Azrour, and A. Asimi, “Mathematical validation of proposed machine learning classifier for heterogeneous traffic and anomaly detection,” *Big Data Mining and Analytics*, vol. 4, no. 1, pp. 18–24, 2021.

S. Li, Y. Cheng, Y. Liu, W. Wang, and T. Chen, “Abnormal client behavior detection in federated learning,” 2019.

L. Erhan, M. Ndubuaku, M. Di Mauro, W. Song, M. Chen, G. Fortino, O. Bagdasar, and A. Liotto, “Smart anomaly detection in sensor systems: A multi- perspective review,” *Information Fusion*, vol. 67, pp. 29–79, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566235220307177

D. Pou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu, “Enhancing graph neural network-based fraud detectors against camouflaged fraudsters,” in *Proceedings of the 29th ACM International
V. Duddu, A. Boutet, and V. Shejwalkar, "Quantifying privacy leakage in databases," in Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, 2021, pp. 7084–7096, 2020.

Y. Cheng, M. Jin, Y. Liu, L. Chi, K. T. Phan, and Y.-P. C. Chen, "Generative and contrastive self-supervised learning for graph anomaly detection," IEEE Transactions on Knowledge and Data Engineering, 2021.

Y. Liu, T. Dillon, W. Yu, W. Rahayu, and F. Mostafa, "Noise removal in the presence of significant anomalies for industrial internet of things data in manufacturing," IEEE Internet of Things Journal, vol. 7, no. 8, pp. 7084–7096, 2020.

Y. Tang, Z. Liu, M. Pan, Q. Zhang, C. Wan, F. Guan, F. Wu, and D. Chen, "Detection of magnetic anomaly signal based on information entropy of differential signal," IEEE Geoscience and Remote Sensing Letters, vol. 15, no. 4, pp. 512–516, 2018.

W. Sultani, C. Chen, and M. Shah, "Real-world anomaly detection in surveillance videos," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2020, pp. 7679–7685, 2016. [Online]. Available: https://www.mdpi.com/1424-8220/15/4/8764

X. Liu, L. Xie, Y. Wang, J. Zou, J. Xiong, Z. Ying, and A. Vasi lakos, "Privacy and security issues in deep learning: A survey," IEEE Access, vol. 9, pp. 4566–4593, 2020.

Y. Shen, G. Leus, and G. B. Giannakis, "Online graph-adaptive learning with scalability and privacy," IEEE Transactions on Signal Processing, vol. 67, no. 9, pp. 2471–2483, 2019.

B. Wang, J. Jia, and N. Z. Gong, "Graph-based security and privacy analytics via collective classification with joint weight learning and propagation," arXiv preprint arXiv:1812.01661, 2018.

C. Yang, H. Wang, K. Zhang, L. Chen, and L. Sun, "Secure deep graph generation with link differential privacy," arXiv preprint arXiv:2005.00435, 2020.

V. Vukadinovic, A. Boyent, and V. Shejwalkar, "Quantifying privacy leakage in graph embedding," in Mobiqitous 2020-17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, 2020, pp. 76–85.

J. Zhou, C. Chen, L. Zheng, H. Wu, J. Wu, X. Zheng, B. Wu, Z. Liu, and L. Wang, "Vertically federated graph neural network for privacy-preserving node classification," arXiv preprint arXiv:2102.13517, 2021.

S. Sajadmanesh and D. Gatica-Perez, "Locally private graph neural networks for anomaly detection in graph embedding," in arXiv preprint arXiv:2005.00455, 2020.

C. Zhang, S. Zhang, J. James, and S. Yu, "Fastgcn: A topological information protected federated learning approach for traffic speed forecasting," IEEE Transactions on Industrial Informatics, vol. 17, no. 12, pp. 8464–8474, 2021.

Q. Li, M. Coutino, G. Leus, and M. G. Christensen, "Privacy-preserving distributed graph filtering," in 2020 28th European Signal Processing Conference (EUSIPCO). IEEE, 2021, pp. 2155–2159.

T. Bakhshi and B. Ghiya, "Anomaly detection in encrypted internet traffic using hybrid deep learning," Security and Communication Networks, vol. 2021, 2021.

T.-D. Pham, T.-L. Ho, T. Truong-Huu, T.-D. Cao, and H.-L. Truong, "Mappgraph: Mobile-app classification on encrypted network traffic using deep graph convolution neural networks," in Annual Computer Security Applications Conference, 2021, pp. 1025–1038.
