Trustworthy anomaly detection for smart manufacturing

Li, Z.

Trustworthy Anomaly Detection for Smart Manufacturing

Proefschrift

ter verkrijging van
de graad van doctor aan de Universiteit Leiden,
op gezag van rector magnificus prof.dr.ir. H. Bijl,
volgens besluit van het college voor promoties
te verdedigen op donderdag 1 mei 2025
klokke 11:30 uur

door

Zhong Li 李忠

geboren te Yunnan, China

in 1994

Promotores:

Dr. M. van Leeuwen

Prof.dr. T.H.W. Bäck

Promotiecommissie:

Prof.dr. J. Davis	Katholieke Universiteit Leuven
Prof.dr. M. Pechenizkiy	Eindhoven University of Technology
Prof.dr. M.M. Bonsangue	
Prof.dr. K.J. Batenburg	
Dr. M. Baratchi	

This PhD project was mainly conducted in the following research group:

- Explanatory Data Analysis, Leiden Institute of Advanced Computer Science, Leiden University, Leiden, The Netherlands

SIKS Dissertation Series No. 2025-24. The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

Research presented in this dissertation was supported by Project 4 of the Digital Twin research programme, a TTW Perspectief programme with project number P18-03 that is primarily financed by the Dutch Research Council (NWO).

ISBN: 978-94-6522-222-6

Printed by: Ridderprint

In loving memory of my paternal grandfather, 李廷友 *(1954–2021),*
my maternal grandfather, 明真貴 *(a.k.a.* 明紹康*, 1933–2022),*
and my maternal grandmother, 李宗蓮 *(1939–2023).*

Abstract

The widespread integration of artificial intelligence (AI) technology into manufacturing processes has fostered the evolution of what is now termed *smart manufacturing*. Smart manufacturing leverages data from various sources, including sensors, event logs, and more, to optimize operations through the use of advanced machine learning techniques. A critical component of smart manufacturing is anomaly detection, which aims to identify abnormal patterns in data that may indicate machine malfunctions or system faults, facilitating predictive maintenance. However, the application of anomaly detection to real-world manufacturing environments presents challenges, such as complex data types, high-dimensionality, explainability, generalizability, and automatability. This dissertation addresses these challenges by focusing on *trustworthy anomaly detection*, approached from both *data-centric AI* and *model-centric AI* perspectives.

Data-centric AI emphasizes the importance of data quality, consistency, and richness, while model-centric AI focuses on developing effective algorithms with existing, often fixed, datasets. This dissertation argues that these two paradigms are complementary. While recent trends have shifted towards data-centric AI, we contend that both approaches are crucial for advancing smart manufacturing. Solving the challenges associated with anomaly detection requires understanding both the algorithms used and the insights extracted from complex, large datasets. Therefore, this work adopts a holistic approach, addressing the challenges from both perspectives to enhance the reliability and efficiency of smart manufacturing systems.

This dissertation is structured around two primary research questions, each targeting different aspects of anomaly detection in smart manufacturing. The first research question addresses how to develop and improve anomaly detection from a data-centric AI perspective. In real-world manufacturing, data can take various forms, such as event logs, time series, and graphs. To address this complexity, the dissertation pro-

poses a novel graph-based anomaly detection method that converts event logs into directed, weighted graphs. This method enables unsupervised anomaly detection and provides explanations for each detected anomaly (Chapter 2). Furthermore, a feature selection method is developed to handle high-dimensional data, improving the accuracy of traditional log anomaly detection methods by identifying and reducing redundant log events (Chapter 3).

The second research question examines anomaly detection from a model-centric AI perspective, addressing issues related to explainability, generalizability, and automatability. The dissertation introduces a novel approach using Quantile Regression Forests for inherently interpretable contextual anomaly detection (Chapter 4). It also explores the vulnerability of post-hoc explanation methods for graph neural networks (GNNs) to adversarial attacks, developing an optimization-based adversarial attack method to test the robustness of post-hoc GNN explanations (Chapter 5). Additionally, a cross-domain anomaly detection method is proposed to improve the generalizability of models across different datasets when encountering concept drift (Chapter 6), and a novel strategy for automated hyperparameter tuning in unsupervised anomaly detection systems is presented to enhance their reliability and performance without requiring labeled data (Chapter 7).

Through these contributions, the dissertation provides a comprehensive framework for advancing trustworthy anomaly detection in smart manufacturing. By integrating data-centric and model-centric AI approaches, it offers novel solutions that improve anomaly detection accuracy, interpretability, robustness of explanations to adversarial attacks, generalizability across domains, and automatability in deployment. These advancements have the potential to substantially enhance the reliability and efficiency of smart manufacturing systems, while the insights gained from this research are also applicable to other domains where anomaly detection plays a critical role.

In conclusion, this dissertation advances the state-of-the-art in trustworthy anomaly detection for smart manufacturing by addressing key challenges through a dual focus on data-centric and model-centric AI. The proposed methods contribute to improved accuracy, explainability, robustness of post-hoc explanations, and automatability of anomaly detection systems, ensuring their applicability in complex, high-dimensional, and dynamic real-world manufacturing environments.

Contents

Chapter 1

Introduction

The utilization of artificial intelligence in manufacturing processes has attracted considerable interest from both academic researchers and industry practitioners [1, 2]. This growing field is commonly known as *smart manufacturing* [2]. Anomaly detection, in particular, is a key area of research with numerous successful and potential applications in smart manufacturing [3], where machine learning algorithms analyze data collected from sensors and other sources to identify abnormal patterns and facilitate predictive maintenance. However, applying anomaly detection techniques to real-world use-cases in smart manufacturing often presents obstacles, motivating the development of this dissertation.

In this chapter, we will first provide an overview of smart manufacturing and related concepts, setting the stage for the broader context of this dissertation and outlining the current research landscape. We will then introduce the concept of trustworthy anomaly detection, which is central to this work. This section includes a review of fundamental concepts and relevant literature on anomaly detection and trustworthy AI. Building on this, we will present trustworthy anomaly detection in the context of smart manufacturing, in which we summarize the main contributions of this dissertation. Furthermore, we will explore the concepts of model-centric AI and data-centric AI, highlighting the necessity of integrating both perspectives for data-driven smart manufacturing. Following this, we will formally present the research questions addressed in this dissertation and the corresponding contributions made to answer them. Finally, we will outline the structure of this dissertation and provide an overview of the remaining chapters.

This chapter is partially based on the following publication:

- Zhong Li, Yuxuan Zhu, and Matthijs van Leeuwen. "A survey on explainable anomaly detection." ACM Transactions on Knowledge Discovery from Data 18, no. 1 (2023): 1-54.

1.1 Smart Manufacturing

This section introduces the concepts of Industry 4.0, Smart Manufacturing, Cyber-Physical Systems, Digital Twin, and Predictive Maintenance.

1.1.1 Industry 4.0 and Smart Manufacturing

The evolution of mass industries is generally divided into four stages: 1) Industry 1.0, which relied on steam and water-powered machinery; 2) Industry 2.0, characterized by the use of electricity and mass production techniques; 3) Industry 3.0, which introduced digitization and automation; and 4) Industry 4.0, which focuses on cyber-physical systems [4]. Specifically, with the rapid development of information, communication, and other technologies—such as the Internet of Things, Cloud Computing, Big Data, Robotics, and Artificial Intelligence—the manufacturing industry has entered the fourth stage of industrial production, known as Industry 4.0, since the early 2010s [5]. However, the concept of Industry 4.0 does not have a universally accepted definition. A commonly recognized definition is presented below.

Definition 1.1 (Industry 4.0)**.** Industry 4.0 is commonly viewed as the convergence of various advanced technologies, such as Internet of Things, Cloud Computing, Big Data, Robotics, and Artificial Intelligence, aimed at integrating the physical and virtual worlds through cyber-physical systems (CPS). [4]

Particularly, Smart Manufacturing is a subset within the broader Industry 4.0 framework. It focuses specifically on applying these advanced technologies to improve manufacturing processes. Smart manufacturing has garnered significant attention from industry, government organizations, and academia. Despite this interest, a universally accepted definition of *smart manufacturing* has yet to be established [6, 7]. We present its more commonly used definition in Definition 1.2.

Definition 1.2 (Smart Manufacturing)**.** According to the National Institute of Standards and Technology, "smart manufacturing is fully integrated, collaborative manufacturing system that respond in real time to meet changing demands and conditions in the factory, in the supply network and in customer needs." [6]

Besides, smart manufacturing is also commonly known as intelligent manufacturing. However, Wang et al. [2] performed detailed comparisons to show the differences between these two concepts. In brief, intelligent manufacturing is defined as the intersection of artificial intelligence and manufacturing [8], and this term has been used since 1980s. Zhong et al. [8] pointed out that the manufacturing has been shifted from knowledge-based intelligent manufacturing to knowledge-enabled and data-driven smart manufacturing, where the term "smart" refers to the creation and usage of data. Therefore, Toben et al. [9] considered smart manufacturing as a new version of intelligent manufacturing that highlights the use of advanced technologies and analytics methods. Specifically, Zheng et al. [5] proposed a conceptual framework of smart manufacturing systems for Industry 4.0, and this framework consists of the following dimensions: smart design, smart machining, smart monitoring, smart control, and smart scheduling.

1.1.2 Cyber-Physical Systems and Digital Twin

Tao et al. [10] indicated that cyber-physical integration is a crucial prerequisite for and the core of smart manufacturing. Particularly, Cyber-Physical Systems (CPS) and Digital Twins (DTs) are two preferred means to achieve such an integration. The formal definition for CPS is given in Definition 1.3, while the formal definition for DT is given in Definition 1.4.

Definition 1.3 (Cyber-Physical Systems)**.** "Cyber-Physical Systems are multidimensional and complex system that integrate the cyber world and the dynamic physical world. Through the integration and collaboration of computing, communication, and control, CPS provide real-time sensing, information feedback, dynamic control, and other services." [10, 11, 12]

Definition 1.4 (Digital Twin)**.** A Digital Twin is a high-fidelity digital replica of a physical asset, process, or system in virtual space. It uses real-time data and simulations to mirror the physical counterpart's state, performance, and behavior. [10, 13]

Tao et al. systematically analyzed and compared Cyber-Physical Systems (CPS) and Digital Twins (DTs) across several aspects: origin, cyber-physical mapping, hierarchical modeling, and core elements [10]. The relationship between CPS and DTs can be summarized as follows: CPS are foundational systems that integrate physical processes with digital controls and computations [14], while DTs are specific digital

replicas of physical entities or processes [15]. DTs rely on CPS for real-time data acquisition and interaction, as CPS emphasize the powerful computing and communication capabilities of the cyber world more strongly when compared to DTs [16]. Conversely, DTs focus on creating high-fidelity virtual models that replicate the physical system's geometry, structure, behavior, rules, functionality, and other dynamics [10]. Consequently, DTs can provide detailed insights into operations, enabling predictive maintenance, optimization of manufacturing processes, and what-if analysis through simulations [10].

1.1.3 Predictive Maintenance

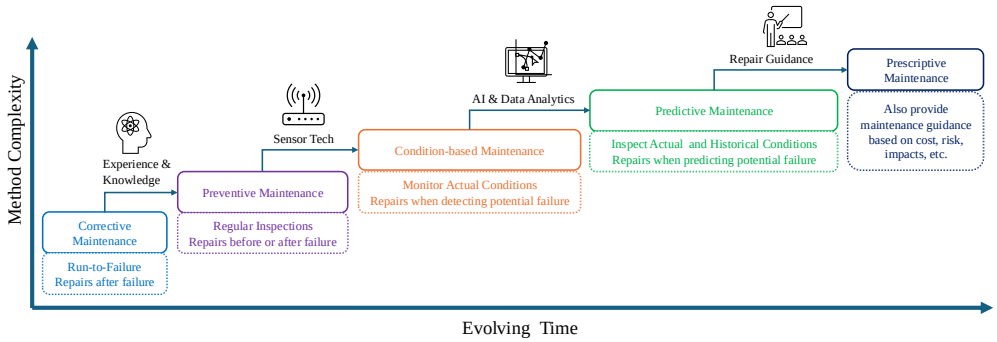


Figure 1.1: Maintenance strategies can be divided into five main stages based on their evolution over time. (Inspired by Figure 1 in [17])

In this dissertation, we focus on the *predictive maintenance* aspect, which is one of the major goals of Industry 4.0 [18]. Specifically, predictive maintenance is the application that can leverage data and simulations from DTs to anticipate and plan maintenance activities before failures occur. On a broader level, maintenance plays a pivotal role in industrial applications, as effective strategies prevent unexpected downtimes, lower operational costs, and potentially extend the remaining useful lifetime of machinery [17]. Consequently, maintenance practices have attracted substantial attention from both industry and academia [19, 20, 17], showing continuous evolution over recent years. According to [17], these practices can broadly be categorized into the following five stages, as shown in Figure 1.1:

1) **Corrective Maintenance**: Also known as reactive maintenance, this run-to-

failure strategy involves performing repairs or replacements only after equipment has failed. It focuses on addressing issues after they occur, rather than preventing them in advance.

2) **Preventive Maintenance**: Also known as planned maintenance, this approach involves scheduling regular inspections and maintenance activities to prevent equipment failures. It is carried out proactively, even when the equipment is operating normally.

3) **Condition-based Maintenance**: It involves monitoring the actual condition of equipment based on real-time data, aiming to determine when maintenance is needed. Maintenance actions are performed only when specific indicators show that performance is deteriorating or a failure is likely to occur.

4) **Predictive Maintenance**: It builds upon Condition-based Maintenance by using advanced analytics (such as machine learning and data mining techniques) to predict when equipment will fail based on its current condition and historical data. It not only monitors equipment conditions but also forecasts future failures with a higher degree of accuracy, enabling more efficient scheduling of maintenance activities.

5) **Prescriptive Maintenance**: It refers to more advanced strategies that not only predict but also recommend optimal maintenance actions to prevent or mitigate equipment failures. It provides guidance on the best interventions, considering factors like cost, risk, and operational impact.

While the more recent prescriptive maintenance strategy offers several advantages over predictive maintenance, its implementation in real-world applications remains challenging without a robust foundation in predictive maintenance. Moreover, predictive maintenance itself faces numerous unresolved issues [17], such as: 1) industrial data is susceptible to errors due to harsh environmental conditions, sensor faults, or transmission errors; 2) the volume of data is large and growing exponentially; 3) data types can vary in actual industrial applications, resulting in different modalities of data such as videos, audios, texts, images, time series, graphs, logs, and tabular data; and 4) industrial environments and production systems can differ widely between manufacturers. Additionally, Digital Twins (DTs), which can be modeled in the virtual world for each critical component in the physical world, provide a promising approach to enhancing predictive maintenance [5]. For these reasons, our focus is on predictive maintenance. Formally, *predictive maintenance* can be defined as follows.

Definition 1.5 (Predictive Maintenance)**.** "Predictive maintenance is a set of activities that detect changes in the physical condition of equipment (signs of failure) in order to carry out the appropriate maintenance work for maximizing the service life of equipment without increasing the risk of failure." [21]

However, with the rapid advancement of technology and increasing demands in industry, the traditional definition of predictive maintenance (namely Definition 1.5) has become somewhat overly generalized and even outdated. To address evolving needs, Predictive Maintenance 4.0 (PdM 4.0) has emerged, aligned with the principles of Industry 4.0, providing a blueprint for more intelligent and efficient predictive maintenance systems [22, 23, 24, 21]. PdM 4.0 leverages advanced technologies such as the Internet of Things (IoT) for data collection, Big Data techniques for data preprocessing, and Data Mining and Machine Learning techniques for in-depth data analysis. These technologies collectively enable decision support systems that accurately predict when maintenance should be performed, thereby preventing unexpected breakdowns and minimizing downtime. As shown in Figure 1.2, the system architecture for PdM 4.0 generally consists of the following components [22, 23, 24, 21]:

- **Data Acquisition:** Sensors are employed to collect data such as temperature, humidity, and vibration from physical assets. This data is then transmitted through networks using Internet of Things (IoT) technologies.
- **Data Pre-processing:** The collected data is stored in data warehouses, which can be in the Cloud, where Big Data techniques are used for data cleaning, integration, feature extraction, and transformation.
- **Data Analysis:** Data mining and machine learning techniques are leveraged to perform two main tasks:
  - *Diagnosis:* Involves anomaly detection (unsupervised or semi-supervised) and anomaly classification (supervised) in the data.
  - *Prognosis:* Focuses on predicting future anomalies and potential failures.
- **Decision Support:** Utilizes the results from data analysis to conduct fault detection, fault isolation, fault prediction, and degradation assessment, thereby supporting maintenance decision-making.
- **Maintenance Implementation:** Implements maintenance activities in the physical world according to the maintenance decisions generated in the cyber world.

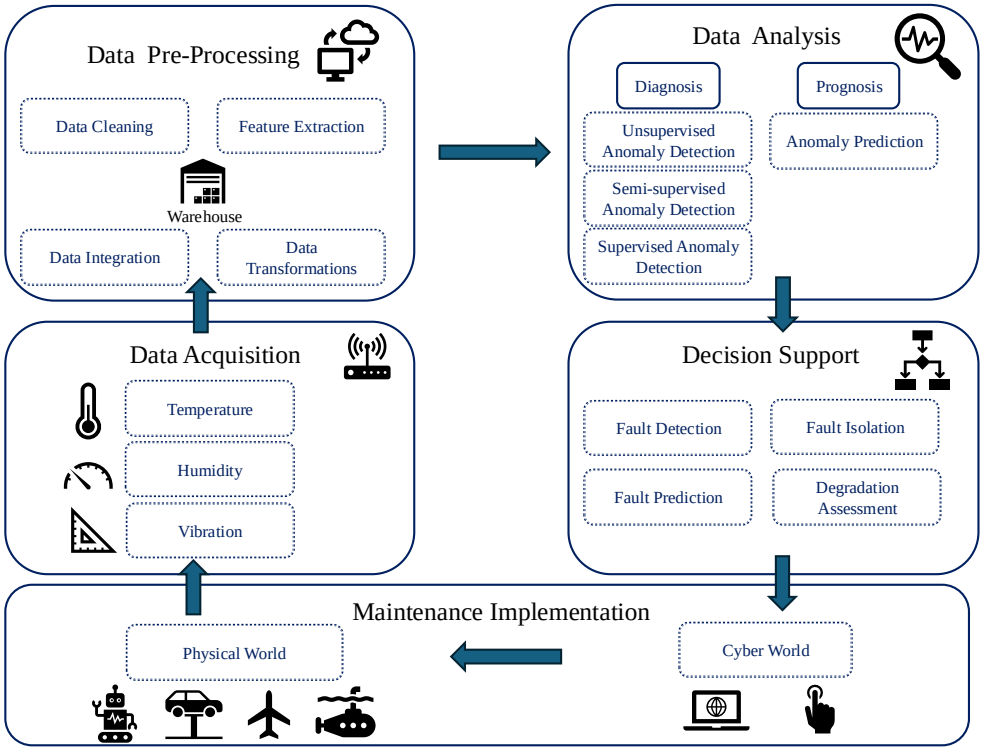


Figure 1.2: The system architecture for PdM 4.0 generally consists of five components. (Inspired by Figure 6 in [24])

As emphasized in [18], anomaly detection is central to Predictive Maintenance, with a primary focus on identifying anomalies in equipment at early stages and alerting the manufacturing technicians to initiate maintenance activities. Moreover, the abnormal behavior of data can stem from various causes. As Nunes et al. [17] highlighted, the detected anomalies can be explored for decision support in two main ways based on their causes of anomalies: 1) if anomalies are due to noise from sensor malfunctions, low battery, or other external disturbances, they are considered irrelevant and should be removed to prevent misinterpretation; 2) if anomalies result from relevant events, such as potential equipment failures or process issues, they should be automatically detected from sensor data and utilized by models to predict the remaining useful lifetime, prompting further analysis and potentially leading to maintenance actions. The main contributions of this dissertation center around anomaly detection, which will be introduced in the next section. However, rather than limiting our scope to

anomaly detection in the specific context of predictive maintenance and sensor data, we emphasize that many of our developed approaches and findings are applicable to more generic scenarios.

1.2 Trustworthy Anomaly Detection

In this section, we will begin by reviewing the fundamental concepts, terminologies, and relevant literature in anomaly detection. Next, we will provide a brief overview of trustworthy artificial intelligence. Building on this foundation, we will highlight the intersection of these two fields, where we will position the contributions of the various research papers included in this dissertation.

1.2.1 Anomaly Detection

We first introduce the concepts of *anomaly* and *anomaly detection*. Then, we present a categorization of anomaly detection techniques based on the availability of labels.

Definition 1.6 (Anomaly)**.** An anomaly is an object that is notably different from the majority of the remaining objects.

Depending on the specific application domain, an anomaly can also be called an outlier or a novelty. Moreover, it may also be known as an unusual, irregular, atypical, inconsistent, unexpected, rare, erroneous, faulty, fraudulent, malicious, unnatural, or strange object [25]. Except for a few works such as [25], the term *outlier* is used as a synonym for *anomaly* in most research. For consistency, we will use the term *anomaly* in this dissertation.

Definition 1.7 (Anomaly Detection)**.** The process of identifying anomalies is called *anomaly detection*.

Since the seminal work in [26], anomaly detection has been well studied and there exists a plethora of comprehensive surveys and reviews on it, including but not limited to [27, 28, 29, 30, 31, 32, 33, 34, 35, 36]. Particularly, depending on the specific type of data and the application domain, the techniques used to perform anomaly detection can be different. The commonly seen data types include tabular data, time series, text, image, video, audio, graph, and log data. In this dissertation, we primarily focus on the following three types of data:

- Tabular data, which refers to data that is organized into a table, where information consists of rows (observations) and columns (features or attributes);

- Log data, which refers to the detailed records generated by software or devices during their operation. These logs capture information about events that occur within a system, providing valuable insights into its performance, security, and usage patterns.
- Graph data, which represents information that is structured as a collection of nodes and edges that connect these nodes. This structure is used to model relationships between different entities in a way that makes the connections and associations between them easy to understand and analyze.

Anomaly detection techniques can be categorized into three types based on the availability of labels, regardless of the data type: supervised, unsupervised, and semi-supervised [31, 34]. Before introducing these categories, it is essential to understand two key concepts: 1) *Transductive learning*, which makes predictions directly for specific test instances without learning a general model for unseen data, thus lacking distinct training and test stages; and 2) *Inductive learning*, which involves training a model on labeled data to learn general rules that are then applied to unseen test instances [37], thus containing separate training and test stages.

- **Supervised anomaly detection** requires labeled instances of both normal and abnormal data during training, treating the task as an *imbalanced* binary classification problem due to the typically low ratio of anomalies (e.g., less than 5%). It is an inductive learning approach, where the model learns general rules from specific training examples and applies these rules to unseen test instances.
- **Unsupervised anomaly detection**, in contrast, does not require labeled training data and is usually performed in a transductive learning manner, without a separate training phase. These techniques are generally based on the assumption that normal instances occur far more frequently than abnormal ones in the dataset.
- **Semi-supervised anomaly detection** operates with partially labeled data, where the labeled instances may include both normal and abnormal examples, or more commonly, only normal examples. In this common scenario, most semi-supervised methods train on datasets exclusively consisting of normal data, aiming to model the normal data distribution during training. Any instances that significantly deviate from this distribution during inference are considered anomalies. Thus, it is also an inductive learning approach. Notably, many semi-supervised methods can function in an unsupervised manner while still being

inductive rather than transductive: given the typically low anomaly ratio, these methods use a subset of samples from the data to train the model, assuming that the training data contains few anomalies and the model can robustly learn the normal distribution.

Particularly, semi-supervised and unsupervised anomaly detection are the most commonly used techniques in manufacturing due to the scarcity of labels [3]. In this dissertation, we will primarily consider semi-supervised and unsupervised anomaly detection for the same reason.

1.2.2 Trustworthy Artificial Intelligence

Definition 1.8 (Trustworthy AI)**.** "Trustworthy AI is a framework to ensure that a system is worthy of being trusted based on the evidence concerning its stated requirements. It makes sure that the users' and stakeholders' expectations are met in a verifiable way." [38]

As highlighted in [38], with the growing volume of research on trustworthy AI, reaching a consensus on a common set of principles to ensure AI trustworthiness is challenging. However, we argue that at least the following five principles should be considered [38, 39]:

- **Effectiveness**: AI systems must provide accurate predictions, as it is crucial for their use in replacing or assisting human decision-makers. Without accuracy, the utility of AI systems is compromised.
- **Explainability**: Stakeholders involved with the AI system should be able to understand the reasoning behind its decisions, ensuring transparency and trust in the system's outputs.
- **Generalizability/Robustness**: AI systems should function reliably across various conditions. Robustness refers to the ability of an AI system to handle execution errors, erroneous inputs, or unseen data, while generalizability pertains to the system's capability to make accurate predictions on unseen data [39]. It is important to note that the relationship between robustness and generalizability can be complex [40, 39]; for simplicity, they are treated as a unified concept in this dissertation.
- **Fairness**: AI systems should avoid introducing or perpetuating biases and discrimination against any individual or group within society.

- **Privacy**: AI systems must ensure that sensitive information is protected throughout its entire lifecycle. In particular, unauthorized use of data that can identify individuals or households should be strictly avoided [39].

1.2.3 Trustworthy Anomaly Detection in the Context of Smart Manufacturing

Anomaly detection has achieved a wide range of successful applications in many decision-critical domains. For example, anomaly detection algorithms are being used to diagnose diseases in healthcare [41]. In financial services, many banks use anomaly detection methods to detect abnormal behavior in credit card transactions [42]. In addition, the self-driving car manufacturing industry applies anomaly detection algorithms on camera data to detect corner cases [43]. In other decision-critical areas—such as spacecraft design—anomaly detection algorithms are used to detect sensor faults [44]. As we can see, anomaly detection systems for high-stakes decisions are deeply impacting our daily lives and society.

Despite significant successes, anomaly detection still faces many limitations. Particularly, given that these tasks are often safety-critical and can have life-changing consequences, a major concern is whether we can truly trust the results provided by these anomaly detection systems. Therefore, ensuring that anomaly detection systems operate in a trustworthy manner is essential when deploying them in real-world applications [45].

Anomaly detection is an important subfield of machine learning and data mining, both of which are critical components of the broader field of AI. Therefore, to assess the trustworthiness of an anomaly detection system, we should consider the five key principles established for trustworthy AI systems, as outlined in Chapter 1.2.2: effectiveness, explainability, generalizability, fairness, and privacy. However, this dissertation focuses specifically on anomaly detection in smart manufacturing, which typically involves data from machines rather than humans. Consequently, the principles of fairness and privacy will not be addressed. In the following, we will briefly introduce the principles of effectiveness, explainability, and generalizability to achieve trustworthy anomaly detection in the context of smart manufacturing.

Principle 1: Effectiveness of Anomaly Detection. Effectively identifying anomalies is the fundamental requirement for all anomaly detection systems [45]. Given a dataset, high accuracy in an anomaly detection system is typically achieved by selecting and deploying an appropriate detection model. However, in real-world

industrial applications, we encounter two key challenges: 1) Most anomaly detection research, since the seminal work in [26], has focused on simple tabular data, whereas real-world scenarios in smart manufacturing often involve complex data such as log events, time series, or graphs; 2) Even after converting complex data into simple form (e.g., through feature extraction), the resulting high dimensionality can reduce the effectiveness of many traditional anomaly detection methods in smart manufacturing. Particularly, Chapters 2 (to Challenge 1) and 3 (to Challenge 2) mainly make contributions in this aspect.

Principle 2: Explainability of Anomaly Detection. According to [46], we can define eXplainable Anomaly Detection (XAD) as *the extraction of relevant knowledge from an anomaly detection model concerning relationships either contained in data or learned by the model*, where the knowledge is considered relevant if it can provide insight into the anomaly detection problem investigated by the end-user. Particularly, providing anomaly detection results with corresponding explanations can help gain the trust of end-users in anomaly detection systems. Moreover, the explanations can also assist end-users to validate the anomaly detection results in unsupervised settings. Even more, explanations can potentially enable end-users to find the root causes of anomalies and thereby take remedial or preventive actions. For a long time, however, the anomaly detection community has mainly focused on detection accuracy, largely ignoring the interpretation of corresponding outcomes. More importantly, there is an increasing demand for explainability when deploying anomaly detection systems in industrial manufacturing. Chapters 4 and 5 mainly make contributions in this aspect.

Principle 3: Generalizability of Anomaly Detection. We define generalizability as the ability of an anomaly detection system to operate reliably under various conditions, which includes two main aspects: First, the anomaly detection system should perform reliably on unseen data, especially when there is concept drift in the data. Second, given that anomaly detection tasks are often unsupervised or semi-supervised, the system should perform consistently across different datasets, requiring the capability to automatically tune model hyperparameters without relying on data labels. Chapter 6 contributes to the first aspect, focusing on generalizability to unseen data, while Chapter 7 addresses the second aspect, emphasizing adaptability across various datasets.

1.3 Model-Centric AI and Data-Centric AI

In this section, we begin by introducing concepts in model-centric AI (and we will discuss how the included papers can be interpreted from this perspective in Chapter 1.4). Next, we explore concepts in data-centric AI (and we will illustrate how the papers presented in this dissertation relate to them in Chapter 1.4). Finally, we explain the necessity of integrating these two complementary approaches to further advance data-driven smart manufacturing.

Definition 1.9 (Model-centric AI)**.** "Model-centric AI is the paradigm emphasizing the choice of the suitable model type, architecture, and hyperparameters from a wide range of possibilities for building effective and efficient AI systems." [47]

More specifically, model-centric AI focuses on enhancing the performance of AI systems by improving algorithms to work more effectively with existing, often fixed, datasets [48]. This approach emphasizes designing better model architectures, fine-tuning hyperparameters, developing more effective and efficient optimization techniques, and proposing new models types or learning paradigms [47]. Typically, the data is created once, and its quality and quantity remain consistent throughout the development cycle of the AI system, while substantial efforts are directed toward building more advanced learning models. Despite the huge successes of Model-centric AI in the past decades, it still suffers from some notable limitations [48]:

- Vulnerability to adversarial samples,
- Low generalization capacity.

Definition 1.10 (Data-centric AI)**.** "Data-centric AI is the paradigm emphasizing that systematic design and engineering of data are essential for building effective and efficient AI systems." [47]

Data is an essential element in AI systems, which include anomaly detection systems. Recently, the significance of data has been greatly amplified by the rise of data-centric AI [49, 50]. Although data-centric AI is a relatively new concept [49, 50, 47, 51, 52, 53], many classic research topics such as data augmentation and feature selection can be considered as subfields of data-centric AI. Data augmentation aims to enrich the training dataset by adding slightly modified data instances, while feature selection attempts to reduce data complexity by keeping only relevant features. According to [49], there are three main data-centric AI objectives (the underlined parts are involved in this dissertation):

- **Training data development**, which includes *data collection*, *data labeling*, *data preparation* (such as data cleaning, feature extraction, and data transformation), *data reduction* (such as feature selection, dimension reduction, instance selection), and *data augmentation* (e.g., basic manipulation, deep learning approaches); Particularly, data collection and data labeling are considered data creation, and the rest is data processing;
- **Inference data development**, which consists of *in-distribution evaluation* (such as data slicing, algorithmic recourse), *out-of-distribution evaluation* (such as adversarial perturbation, distribution shift), and *prompt engineering*;
- **Data maintenance**, which contains *data understanding* (such as data visualization, data valuation), *data quality assurance* (such as quality assessment, quality improvement), and *data storage & retrieval* (such as resource allocation, query acceleration).

In summary, Data-centric AI focus on improving the quality, consistency, and richness of the data used to train and test AI models. In this paradigm, the focus shifts from constantly refining the model to ensuring that the data is clean, well-labeled, diverse, and representative of the problem domain. The idea is that with high-quality data, even simpler models can perform exceptionally well. Data-centric AI is often advocated for in situations where models are already highly optimized, and further gains can be achieved by addressing data issues rather than the models.

Complementary views of model-centic AI and data-centric AI. While there has been a recent shift in focus from model-centric AI to data-centric AI, we argue that these two paradigms are inherently complementary. In other words, although we emphasize the importance of data-centric AI, this should not diminish the role of model-centric AI. Successfully addressing challenges in smart manufacturing requires considering both the methods of action (namely algorithms on "how-to") and the insights hidden within the data (namely knowledge on "what-is") [48]. In this dissertation, we will approach the trustworthy anomaly detection problem from both data-centric and model-centric AI perspectives, recognizing them as twin drivers for advancing smart manufacturing.

1.4 Research Questions and Contributions

In this dissertation, we develop anomaly detection approaches primarily with the aim to enhance the manufacturing process of high-tech systems. However, it is important to

note that many of our developed approaches and findings are applicable to (and often even were designed for) more generic scenarios. As achieving trustworthy anomaly detection for smart manufacturing is non-trivial, we tackle them from two different and complementary perspectives (i.e., data-centic AI and model-centric AI), leading to two primary research questions, each with several sub-questions as outlined below:

Q1 **How to develop and/or improve anomaly detection for smart manufacturing from a data-centric AI perspective?**

First, we face challenges from a data-centric AI perspective (which systematically engineers the data used as input for the anomaly detection system [54]):

- **Complex data**. Not limited to simple tabular datasets, a real-world use case could involve complex data such as log events, time series, graphs, etc. We propose the following research question and make corresponding contributions to answer it:
  - *Q1.1*: How to deal with complex data in system logs to effectively detect and explain anomalies?
  - *Answer to Q1.1* (Contribution 1): *Graph Neural Network based Log Anomaly Detection and Explanation* [55], which will be presented in Chapter 2. In this chapter, we propose a graph-based method for unsupervised log anomaly detection, dubbed *Logs2Graphs*, which first converts event logs into attributed, directed, and weighted graphs, and then leverages graph neural networks to perform graph-level anomaly detection. Specifically, we introduce One-Class Digraph Inception Convolutional Networks, abbreviated as OCDiGCN, a novel graph neural network model for detecting graph-level anomalies in a collection of attributed, directed, and weighted graphs. Crucially, we furnish a concise set of nodes pivotal in OCDiGCN's prediction as explanations for each detected anomaly, offering valuable insights for subsequent root cause analysis.
- **High-dimensional data**. Even after transforming complex data to simple data (e.g., via feature extraction), the dimensionality of the resulting data can be very high, rendering many traditional anomaly detection approaches less effective. We propose the following research question and make corresponding contributions to answer it:

- *Q1.2*: How to handle high-dimensionality in system logs to effectively detect anomalies?
- *Answer to Q1.2* (Contribution 2): *Feature Selection for Fault Detection and Prediction based on Event Log Analysis* [56], which will be presented in Chapter 3. In this chapter, we develop a feature selection method for log-based anomaly detection and prediction. Specifically, our method consists of three main modules: the *Log Event Vectorization* module that converts semi-structured log texts into time series; the *Selection of Relevant Features* module that leverages Kendall rank correlation and Granger causality test to select log events for fault detection and prediction; and the *Removal of Redundant Features* module that utilizes Kendall rank correlation to reduce redundant log events.

Q2 **How to develop and/or improve anomaly detection for smart manufacturing from a model-centric AI perspective?**

On the other hand, we may face challenges from the model-centric AI perspective (which aims to produce the best anomaly detection system for a given dataset):

- **Explainability.** Many algorithms, especially those based on neural networks, lack transparency and are therefore not easily understandable to end-users. We propose the following research questions and make corresponding contributions to answer these questions:
  - *Q2.1*: How to achieve intrinsic explainability of anomaly detection systems?
  - *Q2.2*: Are post-doc explanations for Graph Neural Networks robust to adversarial attacks?
  - *Answer to Q2.1* (Contribution 3): *A Survey on Explainable Anomaly Detection* [46], on which Chapter 1 (namely this chapter) is partially based. Specifically, this work provides a comprehensive and structured survey on state-of-the-art explainable anomaly detection techniques. We propose a taxonomy based on the main aspects that characterize each explainable anomaly detection technique, aiming to help practitioners and researchers find the explainable anomaly detection method that best suits their needs.
  - *Answer to Q2.1* (Contribution 4): *Explainable Contextual Anomaly Detection Using Quantile Regression Forests* [57], which will be presented in Chapter 4. In this chapter, we develop connections between

dependency-based traditional anomaly detection methods and contextual anomaly detection methods. Based on resulting insights, we propose a novel approach to inherently interpretable contextual anomaly detection that uses Quantile Regression Forests to model dependencies between features.

  - *Answer to Q2.2* (Contribution 5): *Explainable Graph Neural Networks under Fire* [58], which will be presented in Chapter 5. In this chapter, we demonstrate that post-hoc Graph Neural Networks (GNNs) explanations cannot be trusted, as common GNN explanation methods turn out to be highly susceptible to adversarial perturbations. That is, even small perturbations of the original graph structure that preserve the model's predictions may yield drastically different explanations. This calls into question the trustworthiness and practical utility of post-hoc explanation methods for GNNs. To be able to attack GNN explanation models, we devise a novel attack method dubbed *GXAttack*, the first *optimization-based* adversarial white-box attack method for post-hoc GNN explanations under such settings.

- **Generalizability.** Algorithms developed based on specific datasets suffer from performance degradation when deployed on new datasets that are generated by new but similar mechanisms (e.g., due to robot, or software upgrade). We propose the following research question and make corresponding contributions to answer it:
  - *Q2.3*: How to detect graph-level anomalies in an unseen target domain with the help of labeled normal graphs from a different but related source domain?
  - *Answer to Q2.3* (Contribution 6): *Cross-domain Graph Level Anomaly Detection* [59], which will be presented in Chapter 6. In this chapter, we propose a *cross-domain graph level anomaly detection method*, aiming to identify anomalous graphs from a set of unlabeled graphs (*target domain*) by using easily accessible normal graphs from a different but related domain (*source domain*). Our method consists of four components: a feature extractor that preserves semantic and topological information of individual graphs while incorporating the distance between different graphs; an adversarial domain classifier to make graph level representations domain-invariant; a one-class classifier to exploit label

information in the source domain; and a class aligner to align classes from both domains based on pseudolabels.

- **Automatability**. Due to the lack of labels, it is challenging to perform model selection (e.g., hyper-parameter optimization) for unsupervised anomaly detection algorithms. This is a critical part of achieving an automated anomaly detection system. We propose the following research question and make corresponding contributions to answer it:
  - *Q2.4*: How to automatically tune hyperparamaters in anomaly detection systems without relying on labels?
  - *Answer to Q2.4* (Contribution 7): *Towards Automated Self-Supervised Learning for Truly Unsupervised Graph Anomaly Detection*, which will be presented in Chapter 7. In this chapter, we empirically demonstrate that three important factors can substantially impact detection performance of self-supervised learning (SSL) based graph anomaly detection methods across datasets: 1) the specific SSL strategy employed; 2) the tuning of the strategy's hyperparameters; and 3) the allocation of combination weights when using multiple strategies. Most SSL-based graph anomaly detection methods circumvent these issues by arbitrarily or selectively choosing SSL strategies, hyperparameter settings, and combination weights. To mitigate this issue, we propose to use an internal evaluation strategy (with theoretical analysis) to select hyperparameters in SSL for unsupervised anomaly detection.

1.5 Outline of This Dissertation

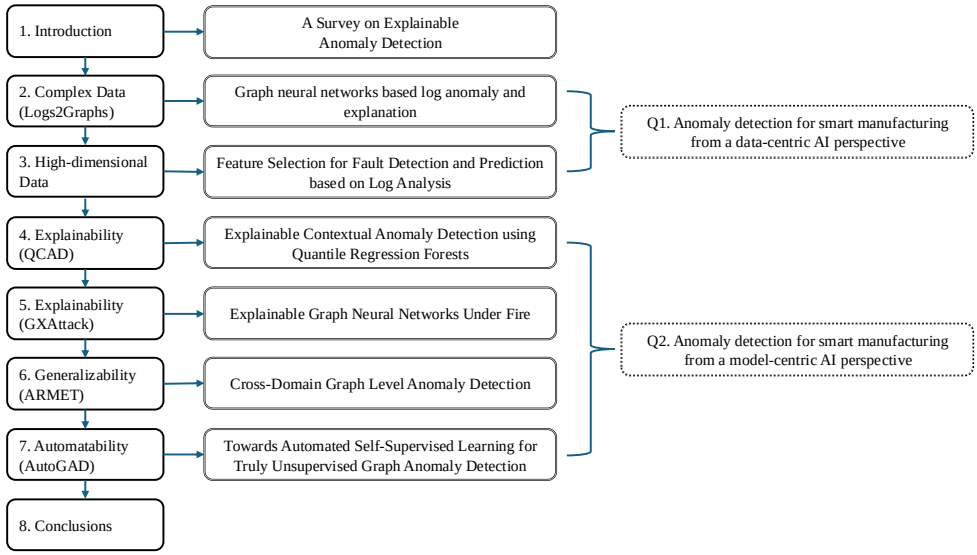


Figure 1.3: Organization of this dissertation

As shown in Figure 1.3, this *Introduction* chapter provides necessary background information for understanding this dissertation, including the introduction of Smart Manufacturing, Trustworthy Anomaly Detection, Model-Centric AI and Data-Centric AI. Moreover, we also present the research questions and contributions, and the outline of this dissertation in this chapter.

Chapters 2 through 7 contain the research papers as published or submitted, where Chapters 2 and 3 aim to answer research question Q1 (including sub-questions *Q1.1* and *Q1.2*) and Chapters 4, 5, 6, and 7 attempt to answer research question Q2 (including sub-questions *Q2.1*, *Q2.2*, *Q2.3* and *Q2.4*).

Chapter 8 –*Conclusions and Future Directions* concludes the dissertation by summarizing the findings and answers to research questions from Chapters 2 to 7. Moreover, we also discuss the limitations of current work, point out challenges and outline possible future directions.

1.6 List of Publications

The chapters in this dissertation are based on the following publications and manuscripts. Chapter 1 is partially based on publication [46], while the remaining chapters use the publications without any changes to their content, edited only for style cohesion.

Chapter	Publication
1	**Zhong Li**, Yuxuan Zhu, & Matthijs van Leeuwen (2024). *A Survey on Explainable Anomaly Detection.* ACM Transactions on Knowledge Discovery from Data 18(1), 1-54. [46]
2	**Zhong Li**, Jiayang Shi, & Matthijs van Leeuwen (2024). *Graph Neural Networks based Log Anomaly Detection and Explanation.* In Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings, 306-307. (Extended version submitted to Engineering Applications of Artificial Intelligence) [55]
3	**Zhong Li** & Matthijs van Leeuwen (2022). *Feature Selection for Fault Detection and Prediction based on Log Analysis.* SIGKDD Explorations Newsletter 24(2), 96–104. [56]
4	**Zhong Li** & Matthijs van Leeuwen (2023). *Explainable Contextual Anomaly Detection using Quantile Regression Forests.* Data Mining and Knowledge Discovery 37(6), 2517-2563. [57]
5	**Zhong Li**, Simon Geisler, Yuhang Wang, Stephan Günnemann, & Matthijs van Leeuwen (2024). *Explainable Graph Neural Networks Under Fire.* Manuscript submitted to IEEE Transactions on Knowledge and Data Engineering.
6	**Zhong Li**, Sheng Liang, Jiayang Shi, & Matthijs van Leeuwen (2024). *Cross-Domain Graph Level Anomaly Detection.* IEEE Transactions on Knowledge and Data Engineering 36(12), 7839–7850. [59]
7	**Zhong Li**, Yuhang Wang, & Matthijs van Leeuwen (2024). *Towards Automated Self-Supervised Learning for Truly Unsupervised Graph Anomaly Detection.* Manuscript submitted to Data Mining and Knowledge Discovery.

Particularly, the following publications or manuscripts are not used in this dissertation.

Chapter	Publication
—	**Zhong Li**, Matteo Quartagno, Stefan Böhringer, & Nan van Geloven (2022). *Choosing and changing the analysis scale in non-inferiority trials with a binary outcome.* Clinical Trials, 19(1), 14-21. [60]
—	Yuhang Wang*, **Zhong Li***, Shujian Yu, & Matthijs van Leeuwen (2024). *Labels Are Not All You Need: Evaluating Node Embedding Quality without Relying on Labels.* Manuscript submitted to NeurIPS 2025. ('*' means equal contributions) [61]
—	**Zhong Li**, Qi Huang*, Lincen Yang*, Jiayang Shi, Zhao Yang, Niki van Stein, Thomas Bäck, & Matthijs van Leeuwen. *Diffusion Models for Tabular Data: Challenges, Current Progress, and Future Directions.* Manuscript submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence. ('*' means equal contributions) [62]

Chapter 2

Graph Neural Networks based Log Anomaly Detection and Explanation

Authors: **Zhong Li**, Jiayang Shi, Matthijs van Leeuwen

A short version of this chapter was published in Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings. The extended version (the version presented here) was submitted to Engineering Applications of Artificial Intelligence journal for possible publication.

Abstract

Event logs are widely used to record the status of high-tech systems, making log anomaly detection important for monitoring those systems. Most existing log anomaly detection methods take a log event count matrix or log event sequences as input, exploiting quantitative and/or sequential relationships between log events to detect anomalies. However, only considering quantitative or sequential relationships may result in low detection accuracy. To alleviate this problem, we propose a graph-based method for unsupervised log anomaly detection, dubbed *Logs2Graphs*, which first converts event logs into attributed, directed, and weighted graphs, and then leverages graph neural networks to perform graph-level anomaly detection. Specifically, we introduce One-Class Digraph Inception Convolutional Networks, abbreviated as OCDiGCN, a novel graph neural network model for detecting graph-level anomalies in a collection of attributed, directed, and weighted graphs. By integrating graph representation and anomaly detection, OCDiGCN learns a specialized representation that leads to high detection accuracy. Crucially, we furnish a concise set of nodes pivotal in OCDiGCN's prediction as explanations for each detected anomaly, offering valuable insights for subsequent root cause analysis. Experiments on five benchmark datasets show that *Logs2Graphs* exhibits comparable or superior performance when compared to state-of-the-art log anomaly detection methods. Moreover, we introduce an ongoing case study wherein we are deploying *Logs2Graphs* to lithography systems, aiming to enhance fault detection and analysis in the wafer transfer subsystem.

2.1 Introduction

Modern high-tech systems, such as cloud computers or lithography machines, typically consist of a large number of components. Over time these systems have become larger and more complex, making manual system operation and maintenance hard or even infeasible [56]. Therefore, automated system operation and maintenance is highly desirable. To achieve this, system logs are universally used to record system states and important events. By analyzing these logs, faults and potential risks can be identified, and remedial actions may be taken to prevent severe problems. System logs are usually semi-structured texts though, and identifying anomalies through log anomaly detection is often challenging.

Since both industry and academia show great interest in identifying anomalies from logs, a plethora of log anomaly detection methods have been proposed. Existing log anomaly detection methods can be roughly divided into three categories: quantitative-based, sequence-based, and graph-based methods. Specifically, quantitative-based methods, such as OCSVM [63] and PCA [64], utilize a log event count matrix to detect anomalies, and are therefore unable to capture semantic information of and sequential information between log events. Meanwhile, sequence-based methods, including DeepLog [65] and LogAnomaly [66], aim to detect anomalies by taking sequential (and sometimes semantic) information into account. They cannot consider the full structure among log events though. In contrast, graph-based methods, such as GLAD-PAW [67] and GLAD [68], convert logs to graphs and exploit semantic information as well as the structure among log events, exhibiting the following three advantages over the former two categories of methods [69]: 1) they are able to identify problems for which the structure among events is crucial, such as performance degradation; 2) they are capable of providing contextual log messages corresponding to the identified problems; and 3) they can provide the 'normal' operation process in the form of a graph, helping end-users find root-causes and take remedial actions. However, graph-based methods like GLAD transform log events into *undirected* graphs though, which may fail to capture important information on the order among log events. Moreover, most existing graph-based methods perform graph representation and anomaly detection separately, leading to suboptimal detection accuracy.

Moreover, as highlighted by Li et al. [70], with the growing adoption of anomaly detection algorithms in safety-critical domains, there is a rising demand for providing explanations for the decisions made within those domains. This requirement, driven by ethical considerations and regulatory mandates, underscores the importance of

accountability and transparency in such contexts. Moreover, in practical applications, the attainment of precise anomaly explanations contributes to the timely isolation and diagnosis of anomalies, which can mitigate the impact of anomalies by facilitating early intervention [71]. However, to our knowledge, most existing log anomaly detection methods focus exclusively on accurate detection without giving explanations.

To overcome these limitations, we propose *Logs2Graphs*, a graph-based unsupervised log anomaly detection approach for which we design a novel one-class graph neural network. Specifically, *Logs2Graphs* first utilizes off-the-shelf methods to learn a semantic embedding for each log event, and then assigns log messages to different groups. Second, *Logs2Graphs* converts each group of log messages into an attributed, directed, and weighted graph, with each node representing a log event, the node attributes containing its semantic embedding, directed edges representing how an event is followed by other events, and the corresponding edge weights indicating the number of times the events follow each other. Third, by coupling the graph representation learning and anomaly detection objectives, we introduce One-Class Digraph Inception Convolutional Networks (OCDiGCN) as a novel method to detect anomalous graphs from a set of graphs. As a result, *Logs2Graphs* leverages the rich and expressive power of attributed, directed, and edge-weighted graphs to represent logs, followed by using graph neural networks to effectively detect graph-level anomalies, taking into account both semantic information of log events and structure information (including sequential information as a special case) among log events. Importantly, by decomposing the anomaly score of a graph into individual nodes and visualizing these nodes based on their contributions, we provide understandable explanations for identified anomalies.

Overall, our contributions can be summarized as follows: (1) We introduce *Logs2Graphs*, which formalizes log anomaly detection as a graph-level anomaly detection problem and represents log sequences as directed graphs to capture more structure information than previous approaches; (2) We introduce OCDiGCN, a general end-to-end unsupervised graph-level anomaly detection method for attributed, directed and edge-weighted graphs. By coupling the graph representation and anomaly detection objectives, we improve the potential for accurate anomaly detection over existing approaches; (3) For each detected anomaly, we identify important nodes as explanations, offering cues for subsequent root cause diagnosis; (4) We empirically compare our approach to eight state-of-the-art log anomaly detection methods on five benchmark datasets, showing that *Logs2Graphs* performs at least on par with and often better than its competitors.

Comparison with our previous work. This paper has been published as a two-pages short paper [55], in which we only presented the problem addressed, a brief

summary of the algorithm, and the achieved main results. In contrast, in this paper we give the motivations of designing algorithms, present related work and the design details of algorithms, elucidate the experiment setting, and present full experiment results with detailed interpretation. Importantly, we also present an ongoing case study wherein we are applying our algorithm to lithography systems.

Organization of the paper. The remainder of this paper is organized as follows. Chapter 2.2 revisits related work, after which Chapter 2.3 formalizes the problem. Chapter 2.4 describes Digraph Inception Convolutional Networks [72], which are used for *Logs2Graphs* in Chapter 2.5. We then evaluate *Logs2Graphs* in Chapter 2.6 with publicly available datasets, and present an ongoing case-study in Chapter 2.7. We analyze the threats to validity in Chapter 2.8, and conclude in Chapter 2.9.

2.2 Related Work

Graph-based log anomaly detection methods usually comprise five steps: log parsing, log grouping, graph construction, graph representation learning, and anomaly detection. In this paper we focus on graph representation learning, log anomaly detection, and explanation, thus only revisiting related work in these fields.

2.2.1 Graph Representation Learning

Graph-level representation learning methods, such as GIN [73] and Graph2Vec [74], are able to learn a mapping from graphs to vectors. Further, graph kernel methods, including Weisfeiler-Lehman (WL) [75] and Propagation Kernels (PK) [76], can directly provide pairwise distances between graphs. Both types of methods can be combined with off-the-shelf anomaly detectors, such as OCSVM [63] and iForest [77], to perform graph-level anomaly detection.

To improve on these naïve approaches, efforts have been made to develop graph representation learning methods especially for anomaly detection. For instance, OCGIN [78] and GLAM [79] combine the GIN [73] representation learning objective with the SVDD objective [80] to perform graph-level representation learning and anomaly detection in an end-to-end manner. GLocalKD [81] performs random distillation of graph and node representations to learn 'normal' graph patterns. Further, OCGTL [82] combines neural transformation learning and one-class classification to learn graph representations for anomaly detection. Although these methods are unsupervised or semi-supervised, they can only deal with attributed, undirected, and unweighted

graphs.

iGAD [83] considers graph-level anomaly detection as a graph classification problem and combines attribute-aware graph convolution and substructure-aware deep random walks to learn graph representations. However, iGAD is a supervised method, and can only handle attributed, undirected, and unweighted graphs. CODEtect [84] takes a pattern-based modeling approach using the minimum description length principle and identifies anomalous graphs based on *motifs*. CODEtect can (only) deal with labeled, directed, and edge-weighted graphs, but is computationally very expensive. In contrast, we introduce a general unsupervised method for graph-level anomaly detection that can handle attributed, directed and edge-weighted graphs.

2.2.2 Log Anomaly Detection and Explanation

Log anomaly detection methods can be roughly divided into: 1) traditional, 'shallow' methods, such as principal component analysis (PCA) [64], one-class SVM (OCSVM) [63], isolation forest (iForest) [77], and histogram-based outlier score (HBOS) [85], which take a log event count matrix as input and analyze quantitative relationships; 2) deep learning based methods, such as DeepLog [65], LogAnomaly [66], and AutoEncoder [86], which employ sequences of log events (and sometimes their semantic embeddings) as input, analyzing sequential information and possibly semantic information of log events to identify anomalies; and 3) graph-based methods, such as TCFG [69] and GLAD-PAW [67], which first convert logs into graphs and then perform graph-level anomaly detection.

To our knowledge, only a few works [67, 87, 88, 68] have capitalized on the powerful learning capabilities of graph neural networks for log anomaly detection. GLAD [68] transforms logs into undirected, weighted, and attributed heterogeneous graph, and propose a temporal-attentive graph edge anomaly detection model to detect anomalous relations. However, converting logs into undirected graphs may result in loss of important sequential information. Further, DeepTraLog [87] combines traces and logs to generate a so-called Trace Event Graph, which is an attributed and directed graph. On this basis, they train a Gated Graph Neural Networks based Deep Support Vector Data Description model to identify anomalies. However, their approach requires the availability of both traces and logs, and is unable to handle edge weights. In contrast, like LogGD [88] and GLAD-PAW [67], our proposed *Logs2Graphs* approach is applicable to generic logs by converting logs into attributed, directed, and edge-weighted graphs. However, LogGD is a supervised method that requires fully labeled train-

ing data, which is usually impractical and even impossible. Moreover, LogGD and GLAD-PAW [67] are not capable of providing explanations for identified anomalies. In contrast, our proposed algorithm OCDiGCN is a general *unsupervised* graph-level anomaly detection method for attributed, directed, and edge-weighted graphs, able to provide explanations for identified anomalies.

Although anomaly explanation has received much attention in traditional anomaly detection [70, 89], only a few studies [90] considered log anomaly explanation. Specifically, PLELog [90] offers explanations by quantifying the significance of individual log events within an anomalous log sequence, thereby facilitating improved identification of relevant log events by operators. Similarly, our method provides explanations for anomalous log groups by identifying and visualizing a small subset of important nodes.

2.3 Problem Statement

Before we state the log anomaly detection problem, we first introduce notation and definitions regarding event logs and graphs.

Event logs. *Logs* are used to record system status and important events, and are usually collected and stored centrally as log files. A *log file* typically consists of many *log messages*. Each *log message* is composed of three components: a timestamp, an event type (*log event* or *log template*), and additional information (*log parameters*). *Log parsers* are used to extract log events from log messages.

Further, log messages can be grouped into *log groups* (a.k.a. *log sequences*) using certain criteria. Specifically, if a *log identifier* is available for each log message, one can group log messages based on such identifiers. Otherwise, one can use a *fixed* or *sliding window* to group log messages. Besides, counting the occurrences of each log event within a log group yields an *event count vector*. Consequently, for a log file consisting of many log groups, one can obtain an *event count matrix*. The process of generating an *event count matrix* (or other feature matrix) is known as *feature extraction*. Extracted features are often used as input to an anomaly detection algorithm to identify *log anomalies*, i.e., log messages or log groups that deviate from what is considered 'normal'.

Graphs. We consider an attributed, directed, and edge-weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{Y})$, where $\mathcal{V} = \{v_1, ..., v_{|\mathcal{V}|}\}$ denotes the set of *nodes* and $\mathcal{E} = \{e_1, ..., e_{|\mathcal{E}|}\} \subseteq \mathcal{V} \times \mathcal{V}$ represents the set of edges. If $(v_i, v_j) \in \mathcal{E}$, then there is an edge from node v_i to node v_j. Moreover, $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$ is the node attribute matrix, with the i-th row representing the

attributes of node v_i, and d is the number of attributes. Besides, $\mathbf{Y} \in \mathbb{N}^{|\mathcal{E}|\times|\mathcal{E}|}$ is the edge-weight matrix, where $\mathbf{Y}_{ij}$ represents the weight of the edge from node v_i to v_j.

Equivalently, $\mathcal{G}$ can be described as $(\mathbf{A}, \mathbf{X}, \mathbf{Y})$, with adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, where $\mathbf{A}_{ij} = \mathbb{I}[(v_i, v_j) \in \mathcal{E}]$ indicates whether there is an edge from node v_i to node v_j, for $i, j \in \{1, ..., |\mathcal{V}|\}$.

2.3.1 Graph-based Log Anomaly Detection

Given a set of log files, we let $\mathcal{L} = \{L_1, ..., L_{|\mathcal{L}|}\}$ denote the set of unique log events. We divide the log messages into M log groups $\mathbf{Q} = \{\mathbf{q}_1, ..., \mathbf{q}_m, ..., \mathbf{q}_M\}$, where $\mathbf{q}_m = \{\mathbf{q}_{m1}, ..., \mathbf{q}_{mn}, ..., \mathbf{q}_{mN}\}$ is a log group and $\mathbf{q}_{mn}$ a log message.

For each log group $\mathbf{q}_m$, we construct an attributed, directed, and edge-weighted graph $\mathcal{G}_m = (\mathcal{V}_m, \mathcal{E}_m, \mathbf{X}_m, \mathbf{Y}_m)$ to represent the log messages and their relationships. Specifically, each node $v_i \in \mathcal{V}_m$ corresponds to exactly one log event $L \in \mathcal{L}$ (and vice versa). Further, an edge $e_{ij} \in \mathcal{E}_m$ indicates that log event i is at least once immediately followed by log event j in $\mathbf{q}_m$. Attributes $\mathbf{x}_i \in \mathbf{X}_m$ represent the semantic embedding of log event i, and $y_{ij} \in \mathbf{Y}_m$ is the weight of edge e_{ij}, representing the number of times event i was immediately followed by event j. In this manner, we construct a set of log graphs $\{\mathcal{G}_1, ..., \mathcal{G}_m, ..., \mathcal{G}_M\}$.

We use these definitions to define graph-based log anomaly detection:

Problem 1 (Graph-based Log Anomaly Detection). *Given a set of attributed, directed, and weighted graphs that represent logs, find those graphs that are notably different from the majority of graphs.*

What we mean by 'notably different' will have to be made more specific when we define our method, but we can already discuss what types of anomalies can potentially be detected. Most methods aim to detect two types of anomalies:

- A log group (here a graph) is considered a *quantitative anomaly* if the occurrence frequencies of some events in the group are higher or lower than expected from what is commonly observed. For example, if a file is opened (event A) twice, it should normally also be closed (event B) twice. In other words, the number of event occurrences $\#A = \#B$ in a normal pattern and an anomaly is detected if $\#A \neq \#B$.

- A log group is considered to contain *sequential anomalies* if the order of certain events violates the normal order pattern. For instance, a file can be closed only

after it has been opened in a normal workflow. In other words, the order of event occurrences $A \rightarrow B$ is considered normal while $B \rightarrow A$ is considered anomalous.

An advantage of graph-based anomaly detection is that it can detect these two types of anomalies, but also anomalies reflected in the structures of the graphs. Moreover, no *unsupervised* log anomaly detection approaches represent event logs as attributed, directed, weighted graphs, which allow for even higher expressiveness than undirected graphs (and thus limiting the information loss resulting from the representation of the log files as graphs).

2.4 Preliminaries: Digraph Inception Convolutional Nets

To learn node representations for attributed, directed, and edge-weighted graphs, Tong et al. [72] proposed Digraph Inception Convolutional Networks (DiGCN).

Specifically, given a graph $\mathcal{G}$ described by an adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, a node attribute matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}|\times d}$, and an edge-weight matrix $\mathbf{Y} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, DiGCN defines the k-th order digraph convolution as

$$\mathbf{Z}^{(k)} = \begin{cases} \mathbf{X}\Theta^{(0)} & k = 0 \\ \Psi\mathbf{X}\Theta^{(1)} & k = 1 \\ \Phi\mathbf{X}\Theta^{(k)} & k \geq 2, \end{cases} \tag{2.1}$$

where $\Psi = \frac{1}{2}\left(\Pi^{(1)\frac{1}{2}}\mathbf{P}^{(1)}\Pi^{(1)\frac{-1}{2}} + \Pi^{(1)\frac{-1}{2}}\mathbf{P}^{(1)T}\Pi^{(1)\frac{1}{2}}\right)$ and $\Phi = \mathbf{W}^{(k)\frac{-1}{2}}\mathbf{P}^{(k)}\mathbf{W}^{(k)\frac{-1}{2}}$. Particularly, $\mathbf{Z}^{(k)} \in \mathcal{R}^{|\mathcal{V}|\times f}$ denotes the convolved output with f output dimension, and $\Theta^{(0)}, \Theta^{(1)}, \Theta^{(k)}$ represent the trainable parameter matrices.

Moreover, $\mathbf{P}^{(k)}$ is the k-th order proximity matrix defined as

$$\mathbf{P}^{(k)} = \begin{cases} \mathbf{I} & k = 0 \\ \tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}} & k = 1 \\ Ins\left((\mathbf{P}^{(1)})^{(k-1)}(\mathbf{P}^{(1)T})^{(k-1)}\right) & k \geq 2, \end{cases} \tag{2.2}$$

where $\mathbf{I} \in \mathcal{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is an identity matrix, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, and $\tilde{\mathbf{D}}$ denotes the diagonal degree matrix with $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$. Besides, $Ins\left((\mathbf{P}^{(1)})^{(k-1)}(\mathbf{P}^{(1)T})^{(k-1)}\right)$ is defined

as

$$\frac{1}{2}Intersect\left((\mathbf{P}^{(1)})^{(k-1)}(\mathbf{P}^{(1)T})^{(k-1)},(\mathbf{P}^{(1)T})^{(k-1)}(\mathbf{P}^{(1)})^{(k-1)}\right)$$

, with $Intersect(\cdot)$ denoting the element-wise intersection of two matrices (see [72] for computation details). In addition, $\mathbf{W}^{(k)}$ is the diagonalized weight matrix of $\mathbf{P}^{(k)}$, and $\Pi^{(1)}$ is the approximate diagonalized eigenvector of $\mathbf{P}^{(1)}$. Particularly, the approximate diagonalized eigenvector is calculated based on personalized PageRank [91], with a parameter α to control the degree of conversion from a digraph to an undirected graph. We omit the details to conserve space, and refer to [72] for more details.

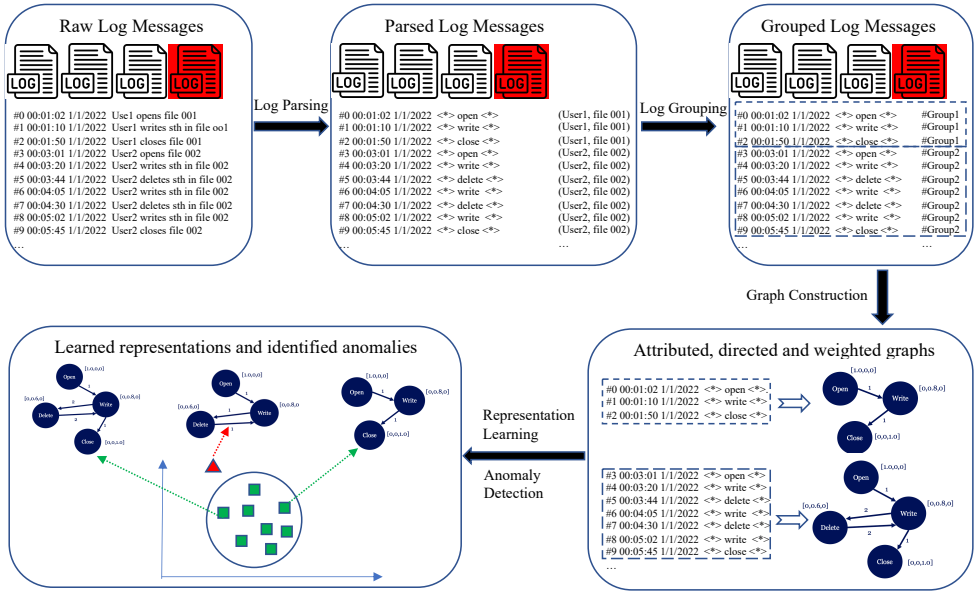


Figure 2.1: The Logs2Graphs pipeline. We use attributed, directed, and weighted graphs for representing the log files with high expressiveness, and integrate representation learning and anomaly detection for accurate anomaly detection. We use off-the-shelf methods for log parsing, log grouping, and graph construction.

After obtaining the multi-scale features $\{\mathbf{Z}^{(0)},\mathbf{Z}^{(1)},...,\mathbf{Z}^{(k)}\}$, DiGCN defines an Inception block as

$$\mathbf{Z}=\sigma\left(\Gamma\left(\mathbf{Z}^{(0)},\mathbf{Z}^{(1)},...,\mathbf{Z}^{(k)}\right)\right), \tag{2.3}$$

where σ represents an activation function, and $\Gamma(\cdot)$ denotes a fusion operation, which can be summation, normalization, and concatenation. In practice, we often adapt a fusion operation that keeps the output dimension unchanged, namely $\mathbf{Z}\in\mathcal{R}^{|\mathcal{V}|\times f}$. As

a result, the i-th row of $\mathbf{Z}$ (namely $\mathbf{Z}_i$) denotes the learned vector representation for node v_i in a certain layer.

2.5 Graph-Based Anomaly Detection for Event Logs

We propose *Logs2Graphs*, a graph-based log anomaly detection method tailored to event logs. The overall pipeline consists of the usual main steps, i.e., log parsing, log grouping, graph construction, graph representation learning, and anomaly detection, and is illustrated in Figure 2.1. Note that we couple the graph representation learning and anomaly detection steps to accomplish end-to-end learning once the graphs have constructed.

First, after collecting logs from a system, the *log parsing* step extracts log events and log parameters from raw log messages. Since log parsing is not the primary focus of this article, we use Drain [92] for this task. Drain is a log parsing technique with fixed depth tree, and has been shown to generally outperform its competitors [93]. We make the following assumptions on the log files:

- Logs files are written in English;
- Each log message contains at least the following information: date, time, operation detail, and log identifier;
- The logs contain enough events to make the mined relationships (quantitative, sequential, structural) statistically meaningful, i.e., it must be possible to learn from the logs what the 'normal' behavior of the system is.

Second, the *log grouping* step uses the log identifiers to divide the parsed log messages into log groups. Third, for each resulting group of log messages, the *graph construction* steps builds an attributed, directed, and edge-weighted graph, as described in more detail in Chapter 2.5.1. Fourth and last, in an integrated step for *graph representation learning and anomaly detection*, we learn a One-Class Digraph Inception Convolutional Network (OCDiGCN) based on the obtained set of log graphs. The resulting model can be used for graph-level anomaly detection. This model couples the graph representation learning objective and anomaly detection objective, and is thus trained in an end-to-end manner. The model, its training, and its use for graph-level anomaly detection are explained in detail in Chapter 2.5.2.

2.5.1 Graph Construction

We next explain how to construct an attributed, directed, and edge-weighted graph given a group of parsed log messages, and illustrate this in Figure 2.2.

First, we utilize nodes to represent different log events. As a result, the number of nodes depends on the number of unique log events that occur within the log group. Second, starting from the first line of log messages in chronological order, we add a directed edge from log event L_i to L_j and set its edge-weight to 1 if the next event after L_i is L_j. If the corresponding edge already exists, we increase its edge-weight by 1. In this manner, we obtain a labeled, directed, and edge-weighted graph.

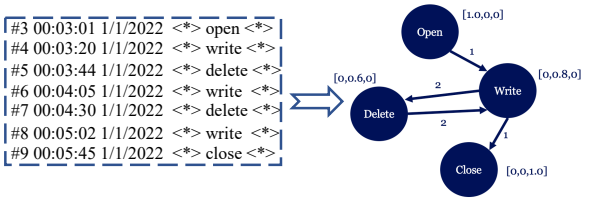


Figure 2.2: The construction of an attributed, directed, and edge-weighted graph from a group of log messages.

However, using only the labels (e.g., *open* or *write*) of log events for graph construction may lead to missing important information. That is, we can improve on this by explicitly taking the semantic information of log events into account, by which we mean that we should look at the text of the log event in entirety. Specifically, we generate a vector representation for each log event as follows:

1. *Preprocessing*: for each log event, we first remove non-character words and stop words, and split compound words into separate words;

2. *Word embedding*: we use Glove [94], a pre-trained word embedding model with 200 embedding dimensions to generate a vector representation for each word in a log event;

3. *Sentence embedding*: we generate a vector representation for each log event. Since the words in a sentence are usually not of equal importance, we use Term Frequency-Inverse Document frequency (TF-IDF) [95] to measure the importance of words. As a result, the weighted sum of word embedding vectors composes the vector representation of a log event.

By augmenting the nodes with the vector representations of the log events as attributes, we obtain an attributed, directed, and edge-weighted graph.

2.5.2 OCDiGCN: One-Class Digraph Inception Convolutional Nets

We next describe One-Class Digraph Inception Convolutional Networks, abbreviated as OCDiGCN, a novel method for end-to-end graph-level anomaly detection. We chose to build on Digraph Inception Convolutional Networks (DiGCN) [72] for their capability to handle directed graphs, which we argued previously is an advantage in graph-based log anomaly detection.

Given that DiGCN was designed for node representation learning, we repurpose it for graph representation learning as follows:

$$\mathbf{z} = \text{Readout}(\mathbf{Z}_i \mid i \in \{1, 2, ..., |\mathcal{V}|\}). \tag{2.4}$$

That is, at the final iteration layer, we utilize a so-called Readout$(\cdot)$ function to aggregate node vector representations to obtain a graph vector representation. Importantly, Readout$(\cdot)$ can be a simple permutation-invariant function such as maximum, sum or mean, or a more advanced graph-level pooling function [96].

Next, note that DiGCN work did not explicitly enable learning edge features (i.e., $\mathbf{Y}$). However, as DiGCN follows the Message Passing Neural Network (MPNN) framework [97], incorporating $\mathbf{Y}$ into Equation (1) and conducting computations in Equations (2-4) analogously enables learning edge features.

Now, given a set of graphs $\{\mathcal{G}_1, ..., \mathcal{G}_m, ..., \mathcal{G}_M\}$, we can use Equation (2.4) to obtain an explicit vector representation for each graph, respectively. We denote the vector presentation of $\mathcal{G}_m$ learned by the DiGCN model as DiGCN$(\mathcal{G}_\text{m}; \mathcal{H})$.

In graph anomaly detection, anomalies are typically identified based on a reconstruction or distance loss [98]. In particular, the One-Class Deep SVDD objective [99] is commonly used for two reasons: it can be easily combined with other neural networks, and more importantly, it generally achieves a state-of-the-art performance [36]. To detect anomalies, we thus train a one-class classifier by optimizing the following One-Class Deep SVDD objective:

$$\min_{\mathcal{H}} \frac{1}{M} \sum_{m=1}^{M} \|\text{DiGCN}(\mathcal{G}_\text{m}; \mathcal{H}) - \mathbf{o}\|_2^2 + \frac{\lambda}{2} \sum_{l=1}^{L} \|\mathbf{H}^{(l)}\|_F^2, \tag{2.5}$$

Table 2.1: Summary of datasets. #Events refers to the number of log event templates obtained using log parser Drain [92]. #Groups means the number of generated graphs. #Anomalies represents the number of anomalous graphs. #Nodes denotes the average number of nodes in generated graphs. #Edges indicates the average number of edges in the generated graphs.

Name	#Events	#Graphs	#Anomalies	#Nodes	#Edges
HDFS	48	575,061	16,838	7	20
Hadoop	683	978	811	34	120
BGL	1848	69,251	31,374	10	30
Spirit	834	10,155	4,432	6	24
Thunderbird	1013	52,160	6,814	16	52

where $\mathbf{H}^{(l)}$ represents the trainable parameters of DiGCN at the l-th layer, namely $(\Theta^{(0)(l)}, \Theta^{(1)(l)}, ..., \Theta^{(k)(l)})^T$, $\mathcal{H}$ denotes $\{\mathbf{H}^{(1)}, ..., \mathbf{H}^{(L)}\}$, $\lambda > 0$ represents the weight-decay hyperparameter, $\|\cdot\|_2$ is the Euclidean norm, and $\|\cdot\|_F$ denotes the Frobenius norm. Moreover, $\mathbf{o}$ is the center of the hypersphere in the learned representation space. Ruff et al. [99] empirically found that setting $\mathbf{o}$ to the average of the network representations (i.e., graph representations in our case) obtained by performing an initial forward pass is a good strategy.

Ruff et al. [99] also pointed out, however, that One-Class Deep SVDD classification may suffer from a hypersphere collapse, which will yield trivial solutions, namely mapping all graphs to a fixed center in the representation space. To avoid a hypersphere collapse, the hypersphere center $\mathbf{o}$ is set to the average of the network representations, the bias terms in the neural networks are removed, and unbounded activation functions such as ReLU are preferred.

After training the model on a set of non-anomalous graphs (or with a very low proportion of anomalies), given a test graph $\mathcal{G}_m$, we define its distance to the center in the representation space as its anomaly score, namely

$$score(\mathcal{G}_m) = \|\mathrm{DiGCN}(\mathcal{G}_\mathrm{m}; \mathcal{H}) - \mathbf{o}\|_2. \tag{2.6}$$

Training and hyperparameters: In summary, OCDiGCN is composed of an L-layer DiGCN architecture to learn node representations, plus a Readout$(\cdot)$ function to obtain the graph representation. It is trained in an end-to-end manner via optimizing the SVDD objective, which can be optimized using stochastic optimization techniques such as Adam [100]. Overall, OCDiGCN takes a collection of non-anomalous graphs

Table 2.2: Description of hyperparameters involved in OCDiGCN. **Range** indicates the values that we have tried on validation data, and boldfaced values are the values suggested to use in experiments. Particularly, for the embedding dimensions: 300 is suggested for BGL and 128 for others. For the batch sizes: 32 is suggested for HDFS and 128 for others. For the training epochs: 100 for BGL and Thunderbird, 200 for HDFS, 300 for Hadoop and 500 for Spirit are suggested.

Symbol	Meaning	Range
Bs	Batch size	{16, 32, 64, **128**, 256, 512, 1024, 1536, 2048, 2560}
Op	optimization method	Adam, **SGD**
L	number of layers	$\{\mathbf{1}, 2, 3, 4, 5\}$
λ	weight decay parameter	$\{\mathbf{0.0001}, 0.001, 0.01, 0.1\}$
η	learning rate	$\{0.0001, 0.001, \mathbf{0.01}\}$
k	proximity parameter	$\{\mathbf{1}, 2\}$
α	teleport probability	$\{0.05, \mathbf{0.1}, 0.2\}$
Γ	fusion operation if $k \geq 2$	sum, concatenation
Re	readout function	**mean**, sum, max
d	embedding dimension	$\{32, 64, \mathbf{128}, 256, 300\}$
Ep	Epochs for training	range(100,1000,50)

and a set of hyperparameters, which are outlined in Table 2.2, as inputs. The pseudo-code for Logs2Graphs is given in Algorithm 1.

Algorithm 1 Pseudo-code of Logs2Graphs

Input: Training dataset D_{tr}, testing dataset D_{ts}, model θ
Output: Predicted labels and explanations for D_{ts}

1: $\hat{D}_{tr}, \hat{D}_{ts} \leftarrow Drain\_parse(D_{tr}), Drain\_parse(D_{ts})$
2: Group $\hat{D}_{tr}$ and $\hat{D}_{ts}$ based on log identifier $\rightarrow$ Obtain grouped dataset $\tilde{D}_{tr}$ and $\tilde{D}_{ts}$
3: Construct graphs using $\tilde{D}_{tr}$ and $\tilde{D}_{ts}$ $\rightarrow$ Obtain graph sets $\mathbf{Q}_{tr}$ and $\mathbf{Q}_{ts}$
4: Train the OCDiGCN model using Equation (2.5) with $\mathbf{Q}_{tr}$ $\rightarrow$ Obtain trained model $\hat{\theta}$
5: Use $\hat{\theta}$ to predict anomalies in $\mathbf{Q}_{ts}$ $\rightarrow$ Obtain a set of anomalies $\{Q_1, ..., Q_n\}$
6: Generate explanations for $Q_i \in \{Q_1, ..., Q_n\}$

2.5.3 Anomaly Explanation

Our anomaly explanation method can be regarded as a decomposition method [101], that is, we build a score decomposition rule to distribute the prediction anomaly score to the input space. Concretely, a graph $\mathcal{G}_m$ is identified as anomalous if and only if its graph-level representation has a large distance to the hyper-sphere center (Equation

2.6). Further, the graph-level representation is obtained via a Readout(·) function applied on the node-level representations (Equation 2.4). Therefore, if the Readout(·) function is attributable (such as the sum or the mean), we can easily obtain the a small subset of important nodes (in the penultimate layer) whose node embeddings contribute the most to the distance. Specifically, the importance score of node v_j (in the penultimate layer) in a graph $\mathcal{G}_m$ is defined as

$$\frac{|score(\mathcal{G}_m) - score(\mathcal{G}_m \setminus \{\mathbf{Z}_j\})|}{score(\mathcal{G}_m)} \tag{2.7}$$

where $score(\mathcal{G}_m)$ is defined in Equation 2.6 and $score(\mathcal{G}_m \setminus \{\mathbf{Z}_j\})$ is the anomaly score by removing the embedding vector of v_j (namely $\mathbf{Z}_j$) when applying the Readout function to obtain the graph-level representation.

Next, for each important node (with a high importance score) in the penultimate layer, we extend the LRP (Layerwise Relevance Propagation) algorithm [102] to obtain a minor set of important nodes in the input layer (this is not the contribution of our paper and we simply follow the practice in [103, 104]). If certain of these nodes are connected by edges, the resulting subgraphs can provide more meaningful explanations. As the LRP method generates explanations utilizing the hidden features and model weights directly, its explanation outcomes are deemed reliable and trustworthy [70].

2.6 Experiments

We perform extensive experiments to answer the following questions:

1. **Detection accuracy:** How effective is *Logs2Graphs* at identifying log anomalies when compared to state-of-the-art methods?

2. **Directed vs. undirected graphs**: Is the directed log graph representation better than the undirected version for detecting log anomalies?

3. **Node Labels vs. Node Attributes**: How important is it to use semantic embeddings of log events as node attributes?

4. **Robustness analysis:** To what extent is Logs2Graphs robust to contamination of the training data?

5. **Ability to detect structural anomalies:** Can Logs2Graphs better capture

structural anomalies and identify structurally equivalent normal instances than its contenders?

6. **Explainability Analysis:** How understandable are the anomaly explanations given by Logs2Graphs?

7. **Sensitivity analysis:** How do the values of the hyperparameters influence the detection accuracy?

8. **Runtime analysis:** What are the runtimes for the different methods?

2.6.1 Experiment Setup

Datasets

The five datasets that we use, summarized in Table 2.1, were chosen for three reasons: 1) they are commonly used for the evaluation of log anomaly detection methods; 2) they contain ground truth labels that can be used to calculate evaluation metrics; and 3) they include log identifiers that can be used for partitioning log messages into groups. For each group of log messages in a dataset, we label the group as anomalous if it contains at least one anomaly. More details are given as follows:

- HDFS [64] consists of Hadoop Distributed File System logs obtained by running 200 Amazon EC2 nodes. These logs contain *block_id*, which can be used to group log events into different groups. Moreover, these logs are manually labeled by Hadoop experts.

- Hadoop [105] was collected from a Hadoop cluster consisting of 46 cores over 5 machines. The *ContainerID* variable is used to divide log messages into different groups.

- BGL, Spirit, and Thunderbird contain system logs collected from the Blue-Gene/L (BGL), Spirit, and Thunderbird supercomputing systems located at Sandia National Labs, respectively. For those datasets, each log message was manually inspected by engineers and labeled as normal or anomalous. For BGL, we use all log messages, and group log messages based on the *Node* variable. For Spirit and Thunderbird, we only use the first 1 million and first 5 million log messages for evaluation, respectively. Furthermore, for these two datasets, the *User* is used as log identifier to group log messages. However, considering that an ordinary user may generate hundreds of thousands of logs, we regard every

100 consecutive logs of each user as a group. If the number of logs is less than 100, we also consider it as a group.

Baselines

To investigate the performance of *Logs2Graphs*, we compare it with the following seven log anomaly detection methods: Principal Component Analysis (PCA) [64], One-Class SVM (OCSVM) [63], Isolation Forest (iForest) [77], HBOS [85], DeepLog [65], LogAnomaly [66], and AutoEncoder [86], and one state-of-the-art graph level anomaly detection method: GLAM [79].

We choose these methods as baselines because they are often regarded to be representatives of traditional machine learning-based (PCA, OCSVM, IForest, HBOS) and deep learning-based approaches (DeepLog, LogAnomaly and AutoEncoder), respectively. All methods are unsupervised or semi-supervised methods that do not require labeled anomalous samples for training the models.

Evaluation Metrics

The Area Under Receiver Operating Characteristics Curve (ROC AUC) and the Area Under the Precision-Recall Curve (PRC AUC) are widely used to quantify the detection accuracy of anomaly detection [106]. This is mainly because they can provide a single value that summarizes the overall performance of the anomaly detection model across various thresholds. In contrast, other metrics such as Precision, Recall and F1-score depend on choosing a threshold to determine whether an instance is anomalous or normal. Consequently, different thresholds can result in different values. Therefore, we employ ROC AUC and PRC AUC to evaluate and compare the different log anomaly detection methods. PRC AUC is also known as Average Precision (AP). For both ROC AUC and PRC AUC, values closer to 1 indicate better performance.

2.6.2 Model Implementation and Configuration

Traditional machine learning based approaches—such as PCA, OCSVM, iForest, and HBOS—usually first transform logs into log event count vectors, and then apply traditional anomaly detection techniques to identify anomalies. For these methods, we utilize their open-source implementations provided in PyOD [107]. Meanwhile, for deep learning methods DeepLog, LogAnomaly, and AutoEncoder, we use their open-source implementations in Deep-Loglizer [108]. For these methods, we use their default hyperparameter values.

Table 2.3: Anomaly detection accuracy on five benchmark datasets for *Logs2Graphs* and its eight competitors. AP and RC denote Average Precision and ROC AUC, respectively. HDFS, BGL, and Thunderbird have been downsampled to 10,000 graphs each while maintaining the original anomaly rates. For each method on each dataset, to mitigate potential biases arising from randomness, we conducted ten experimental runs with varying random seeds and report the average values along with standard deviations of AP and RC. Moreover, we highlight the best results with **bold** and the runner-up with underline.

Method	HDFS		Hadoop		BGL		Spirit		Thunderbird	
	AP	RC	AP	RC	AP	RC	AP	RC	AP	RC
PCA	0.91±0.03	**1.0**±0.00	0.84±0.00	0.52±0.00	0.73±0.01	0.82±0.00	0.31±0.00	0.19±0.00	0.11±0.00	0.34±0.01
OCSVM	0.18±0.01	0.88±0.01	0.83±0.00	0.45±0.00	0.47±0.00	0.47±0.01	0.34±0.00	0.29±0.00	0.12±0.00	0.45±0.01
IForest	0.73±0.04	0.97±0.01	0.85±0.01	0.55±0.01	0.79±0.01	0.83±0.01	0.32±0.03	0.23±0.02	0.11±0.01	0.24±0.10
HBOS	0.74±0.04	0.99±0.00	0.84±0.00	0.50±0.00	0.84±0.02	0.87±0.03	0.35±0.00	0.22±0.00	0.15±0.01	0.29±0.05
DeepLog	**0.92**±0.07	0.97±0.04	**0.96**±0.00	0.47±0.00	0.89±0.00	0.72±0.00	0.99±0.00	0.97±0.00	0.91±0.01	0.96±0.00
LogAnomaly	0.89±0.09	0.95±0.05	**0.96**±0.00	0.47±0.00	0.89±0.00	0.72±0.00	0.99±0.00	0.97±0.00	0.90±0.01	0.96±0.00
AutoEncoder	0.71±0.03	0.84±0.01	**0.96**±0.00	0.52±0.00	0.91±0.01	0.79±0.02	0.96±0.00	0.92±0.01	0.44±0.02	0.46±0.05
GLAM	0.78±0.08	0.89±0.04	0.95±0.00	**0.61**±0.00	0.94±0.02	0.90±0.03	0.93±0.00	0.91±0.00	0.75±0.02	0.85±0.01
Logs2Graphs	0.87±0.04	0.91±0.02	0.95±0.00	0.59±0.00	**0.96**±0.01	**0.93**±0.01	**1.0**±0.00	**1.0**±0.00	**0.99**±0.00	**1.0**±0.00

For all deep learning based methods, the experimental design adopted in this study follows a train/validation/test strategy with a distribution of 70% : 5% : 25% for normal instances. Specifically, the model was trained using 70% of normal instances, while 5% of normal instances and an equal number of abnormal instances were employed for validation (i.e., hyperparameter tuning). The remaining 25% of normal instances and the remaining abnormal instances were used for testing. Table 2.2 summarizes the hyperparameters involved in OCDiGCN as well as their recommended values.

We implemented and ran all algorithms in Python 3.8 (using PyTorch [109] and PyTorch Geometric [110] libraries when applicable), on a workstation equipped with an Intel i7-11700KF CPU and Nvidia RTX3070 GPU.

For reproducibility, all code and datasets are released on GitHub.[1]

2.6.3 Comparison to the State Of The Art

We first compare *Logs*2*Graphs* to the state of the art. We have the following main observations according to the results in Table 2.3:

- In terms of ROC AUC, *Logs2Graphs* achieves the best performance against its competitors on three out of five datasets. Particularly, *Logs2Graphs* outperforms the closet competitor on BGL with 9.6% and delivers remarkable results (i.e., an ROC AUC larger than 0.99) on Spirit and Thunderbird. Similar observations can be made for Average Precision.

[1] https://github.com/ZhongLIFR/Logs2Graph

- Deep learning based methods generally outperform the traditional machine learning based methods. One possible reason is that traditional machine learning based methods only leverage log event count vectors as input, which makes them unable to capture and exploit sequential relationships between log events and the semantics of the log templates.

- The performance of (not-graph-based) deep learning methods is often inferior to that of *Log2Graphs* on the more complex datasets, i.e., BGL, Spirit, and Thunderbird, which all contain hundreds or even thousands of log templates. This suggests that LSTM-based models may not be well suited for logs with a large number of log templates. One possible reason is that the test dataset contains many unprecedented log templates, namely log templates that are not present in the training dataset.

- In terms of ROC AUC score, all methods except for OCSVM and AutoEncoder achieve impressive results (with $RC > 0.91$) on HDFS. One possible reason is that HDFS is a relatively simple log dataset that contains only 48 log templates. Concerning AP, PCA and LSTM-based DeepLog achieve impressive results (with $AP > 0.89$) on HDFS. Meanwhile, *Logs2Graphs* obtains a competitive performance (with $AP = 0.87$) on HDFS.

2.6.4 Directed vs. Undirected Graphs

To investigate the practical added value of using *directed* log graphs as opposed to *undirected* log graphs, we convert the logs to attributed, undirected, and edge-weighted graphs, and apply GLAM [79], a graph-level anomaly detection method for undirected graphs. We use the same graph construction method as for *Logs2Graphs*, except that we use undirected edges. Similar to our method, GLAM also couples the graph representation learning and anomaly detection objectives by optimizing a single SVDD objective. The key difference with OCDiGCN is that GLAM leverages GIN [73], which can only tackle undirected graphs, while OCDiGCN utilizes DiGCN [72] that is especially designed for directed graphs.

The results in Table 2.3 indicate that GLAM's detection performance is comparable to that of most competitors. However, it consistently underperforms on all datasets, except for Hadoop, when compared to *Logs2Graphs*. Given that the directed vs undirected representation of the log graphs is the key difference between the methods, a plausible explanation is that directed graphs have the capability to retain the

temporal sequencing of log events, whereas undirected graphs lack this ability. Consequently, GLAM may encounter difficulties in detecting sequential anomalies and is outperformed by *Logs2Graphs*.

2.6.5 Node Labels vs. Node Attributes

To investigate the importance of using semantic embeddings of log events as node attributes, we replace the node semantic attributes with one-hot-encoding of node labels (i.e., using an integer to represent a log event). The performance comparisons in terms of ROC AUC for Logs2Graphs are depicted in Figure 2.3, which shows that using semantic embeddings is always superior to using node labels. Particularly, it can lead to a substantial performance improvement on the Hadoop, Spirit and HDFS datasets. The PRC AUC results show a similar behavior and thus are omitted.

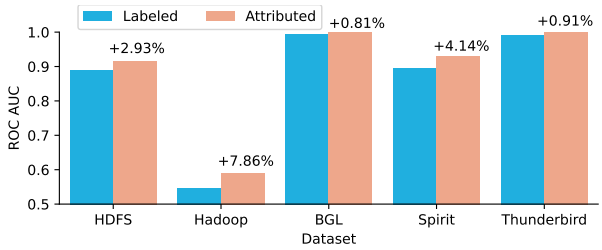


Figure 2.3: The comparative performance analysis of Logs2Graphs, measured by ROC AUC, demonstrating the distinction between utilizing node semantic attributes and node labels.

2.6.6 Robustness to Contamination

To investigate the robustness of Logs2Graphs when the training dataset is contaminated (namely integrating anomalous graphs in training data), we report its performance in terms of ROC AUC under a wide range of contamination levels. Figure 2.4 shows that the performance of Logs2Graphs decreases with an increase of contamination in the training data. The PRC AUC results show a similar behavior and thus are omitted. Hence, it is important to ensure that the training data contains only normal graphs (or with a very low proportion of anomalies).

2.6.7 Ability to Detect Structural Anomalies and Recognize Unseen Normal Instances

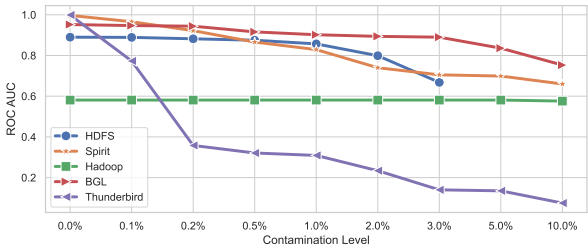


Figure 2.4: ROC AUC results of Logs2Graphs w.r.t. a wide range of contamination levels. Results are averaged over 10 runs. Particularly, HDFS contains only 3% anomalies and thus results at 5% and 10% are not available.

To showcase the effectiveness of different neural networks in detecting structural anomalies, we synthetically generate normal and anomalous directed graphs as shown in Figure 2.5. As Deeplog, LogAnomaly and AutoEncoder require log sequences as inputs, we convert directed graphs into sequences by sequentially presenting the endpoints pair of each edge. Moreover, for GLAM we convert directed graphs into undirected graphs by turning each directed edge into an undirected edge.

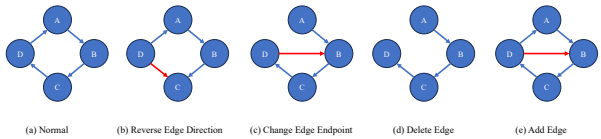


Figure 2.5: Synthetic generation of normal (10000) and structurally anomalous (200 each) graphs.

Moreover, to investigate their capability of recognising unseen but structurally equivalent normal instances, we generate the following normal log sequences based on the synthetic normal graph as training data: $A \to B \to C \to D \to A$ (1000), $B \to C \to D \to A \to B$ (1000) and $C \to D \to A \to B \to C$ (1000), and the following as test dataset: $D \to A \to B \to C \to D$ (1000).

The results in Table 2.4 indicate that Logs2Graphs, Deeplog and LogAnomaly can effectively detect structural anomalies, while AutoEncoder and GLAM fail in some cases. However, log sequences based methods, namely Deeplog, LogAnomaly and AutoEncoder, can lead to high false positive rates due to their inability of recognizing unseen but structurally equivalent normal instances.

Table 2.4: ROC AUC results (higher is better) of detecting structural anomalies and False Positive Rate (lower is better) of recognizing unseen normal instances. S1: Reverse Edge Direction; S2: Change Edge Endpoint; S3: Delete Edge; S4: Add Edge; N1: Unseen normal instances.

Case	Deeplog	LogAnomaly	AutoEncoder	GLAM	Ours
S1 (ROC)	1.0	1.0	0.0	0.0	1.0
S2 (ROC)	1.0	1.0	0.50	1.0	1.0
S3 (ROC)	1.0	1.0	1.0	1.0	1.0
S4 (ROC)	1.0	1.0	1.0	1.0	1.0
N1 (FPR)	100%	100%	100%	0%	0%

2.6.8 Anomaly Explanation

Figure 2.6 provides an example of log anomaly explanation with the HDFS dataset. For each detected anomalous log graph (namely a group of logs), we first quantify the importance of nodes according to the description in Chapter 2.5.3. Next, we visualize the anomalous graph by assigning darker shade of red to more important nodes. In this example, the node "WriteBlock(WithException)" contributes the most to the anomaly score of an anomalous log group and thus is highlighted in red.

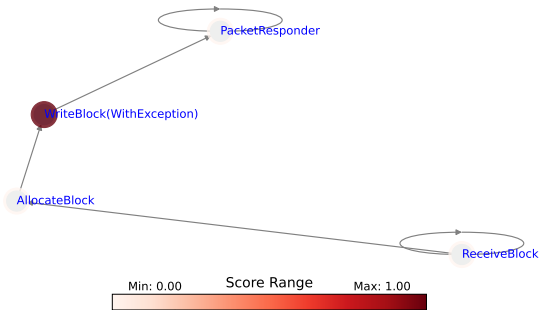


Figure 2.6: Example of anomaly explanation with HDFS (the log event templates are simplified for better visualization).

2.6.9 Sensitivity Analysis

We examine the effects of three hyperparameters in OCDiGCN on the detection performance.

The Number of Convolutional Layers: L is a potentially important parameter as it determines how many convolutional layers to use in OCDiGCN. Figure 2.7 (top

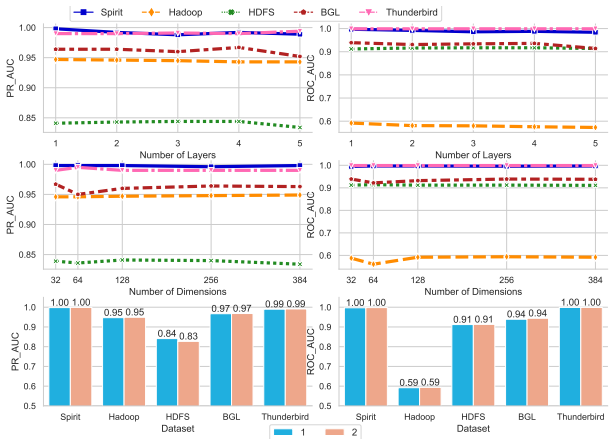


Figure 2.7: The effects of the number of layers (top row), the embedding dimensions (middle row) and the proximity parameter (bottom row) on AP (left column) and ROC AUC (right column).

row) depicts PRC AUC and ROC AUC for the five benchmark datasets when L is varied from 1 to 5. We found that $L = 1$ yields consistently good performance. As the value of L is increased, there is only a slight enhancement in the resulting performance or even degradation, while the associated computational burden increases substantially. We thus recommend to set $L = 1$.

The Embedding Dimension d: From Table 2.7 (middle row) , one can see that $d = 128$ yields good performance on Spirit, Hadoop, HDFS and Thunderbird, while further increasing d obtains negligible performance improvement or even degradation. However, an increase of d on BGL leads to substantially better performance. One possible reason is that BGL is a complex dataset wherein anomalies and normal instances are not easily separable on lower dimensions.

The Proximity Parameter k: As this parameter increases, a node can gain more information from its further neighbors. Figure 2.7 (bottom row) contrasts the detection performance when k is set to 1 and 2, respectively. Particularly, we construct one Inception Block when $k = 2$, using concatenation to fuse the results.

We observe that there is no large improvement in performance when using a value of $k = 2$ in comparison to $k = 1$. It is important to recognize that a node exhibits 0th-order proximity with itself and 1st-order proximity with its immediately connected neighbors. If $k = 2$, a node can directly aggregate information from its 2nd-order neighbors. As described in Table 1, graphs generated from logs usually contain a limited number of nodes, varying to 6 to 34. Therefore, there is no need to utilize the

Inception Block, which was originally designed to handle large graphs in [72].

2.6.10 Runtime Analysis

Traditional machine learning methods, including PCA, OCSVM, IForest and HBOS, usually perform log anomaly detection in a transductive way. In other words, they require the complete dataset beforehand and do not follow a train-and-test strategy. In contrast, neural network based methods, such as DeepLog, LogAnomaly, AutoEncoder, and Logs2Graphs, perform log anomaly detection in an inductive manner, namely following a train-and-test strategy.

Figure 2.8 shows that most computational time demanded by *Logs2Graphs* is allocated towards the graph generation phase. In contrast, the training and testing phases require a minimal time budget. The graph generation phase can be amenable to parallelization though, thereby potentially reducing the overall processing time. As a result, *Logs2Graphs* shows great promise in performing online log anomaly detection. Meanwhile, other neural networks based models—such as DeepLog, LogAnomaly, and AutoEncoder—demand considerably more time for the training and testing phases.

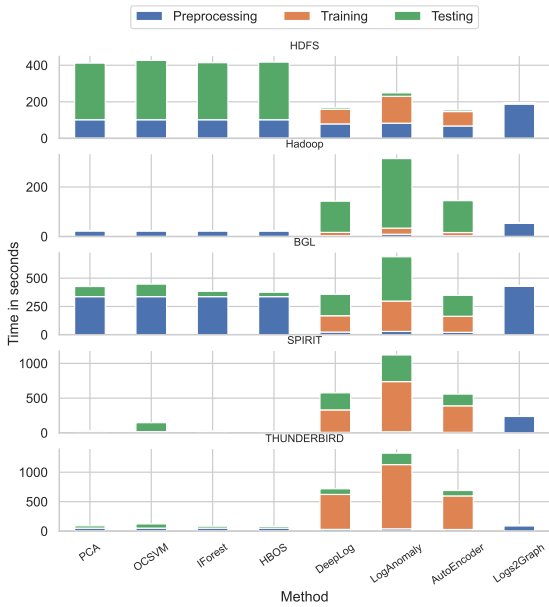


Figure 2.8: Runtime for all eight methods on all datasets, wherein HDFS, BGL, and Thunderbird have been downsampled to 10,000 graphs. Runetimes are averaged over 10 repetitions. We report the training time per epoch for neural network based methods.

Table 2.5: An example event log of lithography systems. All values are fictional. This table is taken from [56], and please refer to the original paper for more information.

M	Code	Level	Detail	DateTime
1	AA-BBBB	Low	descr	2020-01-01 00:00:01
1	CC-DDDD	Med	descr	2020-01-01 00:00:01
1	AA-BBBB	Low	descr	2020-01-01 00:01:00
1	AA-BBBB	Low	descr	2020-01-01 00:02:03
1	EE-FFFF	High	descr	2020-01-01 00:05:00
⋮	⋮	⋮	⋮	⋮

2.7 An Ongoing Case Study to Lithography Systems

In this section, we present an ongoing case study in which we are applying *Logs2Graphs* to lithography systems, aiming to enhance fault detection and analysis in the wafer transfer subsystem.

2.7.1 Background

A lithography system is an intricate apparatus designed for the fabrication of chips. Specifically, a lithography system usually consists of several main subsystems, including the objective lens subsystem, the table subsystem, the mask table subsystem, the mask transfer subsystem, the light source subsystem, the exposure subsystem, and the wafer transfer subsystem [111]. Particularly, the wafer transfer subsystem, which plays a crucial role in the precision of semiconductor fabrication, facilitates the movement of silicon wafers between the track and the wafer stage. This subsystem is essential for operational efficiency during lithography processes, and it typically comprises a load robot and an unload robot. However, these robots are susceptible to malfunctions during production, potentially impacting the fabrication precision.

To optimize productivity by reducing machine downtime, it is essential to detect and analyze the robot faults in an automated and intelligent manner. As shown in Table 2.5, a lithography machine is equipped with an information system that can record all critical events activated during its operation into logs. The event logs contain critical information for fault detection and analysis, motivating us to apply log-based anomaly detection methods to detect faults.

2.7.2 Deployment of *Logs2Graphs*

As shown in Figure 2.1, the deployment of *Logs2Graphs* consists of the following main steps:

1) Log Collection: we use 25 real-world datasets provided by our industrial partners to test our method. Specifically, the 25 datasets were generated by 20 different test machines, of which 16 machines each generated one dataset and the remaining 4 machines each generated at least 2 datasets. Please refer to Table 3 in [56] for more detail. Besides, more data will be collected by our industrial partners;

2) Log Parsing: we use Drain [92] for this task;

3) Log Grouping (Ongoing): we dedicate much efforts in this step as lithography system is very complicated in the following aspects: first, the information system records event logs from all subsystems while some events are irrelevant to the wafer transfer subsystem; second, rather than simply using time as log identifier, we must use domain knowledge to construct more advanced log identifiers using the information provided in the "Detail" column;

4) Graph Construction (Ongoing): we will construct directed, attribute and edge-weighted graphs;

5) Graph Representation Learning and Anomaly Detection (Ongoing): we will utilize OCDiGCN;

6) Anomaly Explanation and Analysis (Ongoing): we follow the methods given in Chapter 2.5.3.

At the request of our industrial partners, the datasets, code, and detailed results of this case-study cannot be published. Despite this, *Logs2Graphs* is a promising tool for intelligent fault detection and analysis when combined with domain expertise.

2.8 Threats to Validity

We have discerned several factors that may pose a threat to the validity of our findings.

Limited Datasets. Our experimental protocol entails utilizing five publicly available log datasets, which have been commonly employed in prior research on log-based anomaly detection. However, it is important to acknowledge that these datasets may not fully encapsulate the entirety of log data characteristics. To address this limitation,

our future work will conduct experiments on additional datasets, particularly those derived from industrial settings, in order to encompass a broader range of real-world scenarios.

Limited Competitors. This study focuses solely on the experimental evaluation of eight competing models, which are considered representative and possess publicly accessible source code. However, it is worth noting that certain models such as GLAD-PAW did not disclose their source code and it requires non-trivial efforts to re-implement these models. Moreover, certain models such as CODEtect require several months to conduct the experiments on our limited computing resources. For these reasons, we exclude them from our present evaluation. In subsequent endeavors, we intend to re-implement certain models and attain more computing resources to test more models.

Purity of Training Data. The purity of training data is usually hard to guarantee in practical scenarios. Although Logs2Graphs is shown to be robust to very small contamination in the training data, it is critical to improve the model robustness by using techniques such as adversarial training [112] in the future.

Graph Construction. The graph construction process, especially regarding the establishment of edges and assigning edge weights, adheres to a rule based on connecting consecutive log events. However, this rule may be considered overly simplistic in certain scenarios. Therefore, application-specific techniques will be explored to construct graphs in the future.

2.9 Conclusions

We introduced *Logs2Graphs*, a new approach for unsupervised log anomaly detection. It first converts log files to attributed, directed, and edge-weighted graphs, translating the problem to an instance of graph-level anomaly detection. Next, this problem is solved by OCDiGCN, a novel method based on graph neural networks that performs graph representation learning and graph-level anomaly detection in an end-to-end manner. Important properties of OCDiGCN include that it can deal with directed graphs and do unsupervised learning.

Extensive results on five benchmark datasets reveal that *Logs2Graphs* is at least comparable to and often outperforms state-of-the-art log anomaly detection methods such as DeepLog and LogAnomaly. Furthermore, a comparison to a similar method for graph-level anomaly detection on *undirected* graphs demonstrates that using directed log graphs leads to better detection accuracy in practice.

Chapter 3

Feature Selection for Fault Detection and Prediction based on Event Log Analysis

Authors: **Zhong Li**, Matthijs van Leeuwen
Published in ACM SIGKDD Explorations Newsletter 24.2 (2022): 96-104.

Abstract

Event logs are widely used for anomaly detection and prediction in complex systems. Existing log-based anomaly detection methods usually consist of four main steps: log collection, log parsing, feature extraction, and anomaly detection, wherein the feature extraction step extracts useful features for anomaly detection by counting log events. For a complex system, such as a lithography machine consisting of a large number of subsystems, its log may contain thousands of different events, resulting in abounding extracted features. However, when anomaly detection is performed at the subsystem level, analyzing all features becomes expensive and unnecessary. To mitigate this problem, we develop a feature selection method for log-based anomaly detection and prediction. Specifically, our method consists of three main modules: the *Log Event Vectorization* module that converts semi-structured log texts into time series; the *Selection of Relevant Features* module that leverages Kendall rank correlation and Granger causality test to select log events for fault detection and prediction; and the *Removal of Redundant Features* module that utilizes Kendall rank correlation to reduce redundant log events. Results on 25 real-world datasets show that our method can detect and predict faults more accurately by selecting a small proportion of log events, thereby improving the effectiveness and efficiency.

3.1 Introduction

A lithography machine is a piece of complex structural equipment used to manufacture chips. Typically, it consists of the following main subsystems: the light source subsystem, the objective lens subsystem, the table subsystem, the mask table subsystem, the mask transfer subsystem, the wafer transfer subsystem, and the exposure subsystem [111]. Particularly, the wafer transfer subsystem serves to transfer silicon wafers between the track and wafer stage, having a great impact on the precision of chip fabrication. A wafer transfer subsystem usually contains two robots, namely a load robot and an unload robot. When the lithography machine goes into production, these two robots may encounter faults. We assume there are $L + M$ types of faults, viz. GF_1, GF_2 , ..., GF_L and SF_1, SF_2, ..., SF_M. Specifically, GF_1, GF_2, ..., GF_L represent faults that occur gradually and thus can be detected in an early stage (i.e., they are typically predictable). In contrast, SF_1, SF_2, ..., SF_M denote faults that generally occur suddenly and are often hard to predict.

To minimize machine downtime and thus maximize productivity, the possible faults of load and unload robots should be detected and predicted (if possible) in an automated way. To this end, the wafer transfer subsystem usually uses sensors to measure the position of the two robots in real time, collecting time series data that can be used for data-driven fault detection and prediction. However, due to the limited information contained in sensor data, it is challenging to detect all possible faults based on time series data alone. Meanwhile, as shown in Figure 3.1 and Table 3.1, a lithography machine also has an information system that records all triggered events—in the form of logs—when the machine is working. Since the different subsystems in a lithography machine are interconnected, a fault incurred in one subsystem (e.g., the wafer transfer system) may trigger events not only in that subsystem, but also in other subsystems. Besides, the components in the same subsystem are usually closely interconnected. Therefore, the fault of one component is very likely to cause faults of other components. Therefore, the event logs contain important information for fault detection and prediction. Traditional log-based anomaly detection methods can be used to detect such faults [113].

Due to the complexity of the lithography machine, there can be thousands of unique log events, resulting in millions of log events in a relatively short working time of the machine. Moreover, when attempting to detect faults of certain components in a specific subsystem (e.g., the load and unload robots in the wafer transfer subsystem), many of these log events are irrelevant or abundant. Hence, a direct application of

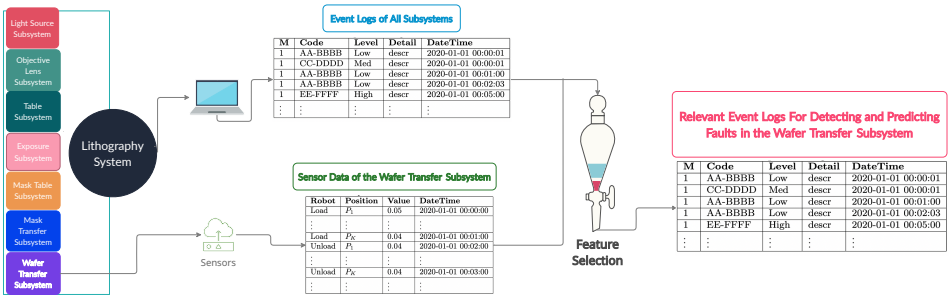


Figure 3.1: The composition of the lithography system, and an illustration of the feature selection problem addressed in this work.

existing log-based anomaly detection methods on all log events can be computationally prohibitive and may also produce misleading detection results due to the inclusion of irrelevant log events. To mitigate this problem, we regard each log event as a feature and develop a feature selection method that aims to select relevant features for log-based fault detection and prediction.

In brief, our method consists of three main modules, namely *Log Event Vectorization*, *Selection of Relevant Features* and *Removal of Redundant Features*. Specifically, the *Log Event Vectorization* module aims at converting unstructured log events into time series data; the *Selection of Relevant Features* module attempts to select relevant features for fault detection and prediction by using the variables measured by sensors as target; and the *Removal of Redundant Features* module focuses on eliminating redundant features to further reduce the number of selected features. The main contributions of this paper can be summarized as follows:

1) We formalize the problem of feature selection for fault detection and prediction based on event log analysis. To our knowledge, this problem is not uncommon in the industry, but has not yet been studied.

2) To address this problem, we further propose a novel approach using two types of feature selection.

3) We demonstrate the effectiveness of our approach by conducting extensive experiments on 25 real-world datasets gathered from lithography systems.

The remainder of this paper is organized as follows. Chapter 3.2 introduces related work. Next, Chapter 3.3 introduces terminology, formalizes the problem, and presents our proposed method in detail. Chapter 3.4 empirically evaluates the performance of

Table 3.1: An example event log. All values are fictional.

M	Code	Level	Detail	DateTime
1	AA-BBBB	Low	descr	2020-01-01 00:00:01
1	CC-DDDD	Med	descr	2020-01-01 00:00:01
1	AA-BBBB	Low	descr	2020-01-01 00:01:00
1	AA-BBBB	Low	descr	2020-01-01 00:02:03
1	EE-FFFF	High	descr	2020-01-01 00:05:00
⋮	⋮	⋮	⋮	⋮

our method. Finally, Chapter 3.5 concludes the paper and discusses possible future directions.

3.2 Related Work

Existing log-based anomaly detection methods usually consist of four main steps: *Log Collection*, *Log Parsing*, *Feature Extraction* and *Anomaly Detection* [114]. First, the *Log Collection* step is responsible for recording triggered events in the form of logs. A record is called a *log message*, which usually contains the date and time of occurrence and the detailed description of event. More concretely, detailed descriptions are typically presented in predefined templates, and may also include parameters. Second, the *Log Parsing* step aims at converting each log message into a specific *log event* template [93]. Usually, a *log event* corresponds to a unique template. Third, based on derived log events, the *Feature Extraction* step attempts to convert each *log sequence* into a *log count vector*. Specifically, a *log sequence* is composed of multiple *log events*. In general, a *log count vector* is a vector with each entry indicating the number of times that the corresponding *log event* was triggered. Note that the entries of the *log count vector* can be computed in another refined way [115]. Finally, the *Anomaly Detection* step performs anomaly detection based on the extracted *log count vectors*.

There exist many log parsing methods based on clustering [116], frequent pattern mining [117], or heuristic techniques [92]. Since our work centers around feature selection (e.g., log event selection) for fault detection and prediction, we will only consider the *Feature Extraction* and *Anomaly Detection* steps.

He et al. [118] have performed a systematic comparison of five state-of-the-art log-based anomaly detection methods, including DeepLog [65], LogAnomaly [66], PLELog [71], LogRobust [119], CNN [120]. Their findings show that log-based anomaly detec-

tion methods are often not as good as expected (i.e., as claimed by the original authors of each method) in real-world datasets. Importantly, they found that some log-based anomaly detectors, such as LogAnomaly, perform poorly on datasets with numerous log events. Therefore, feature selection can be exploited to improve the performance of these anomaly detection methods. However, we are not aware of any existing publications that attempt to address this problem.

3.3 Method

In the following section, we first introduce the terminology used in this work and then formalize the problem. Next, we elucidate our proposed method that consists of three modules: *Log Event Vectorization*, *Selection of Relevant Features*, and *Removal of Redundant Features.*

3.3.1 Terminology and Problem Formulation

We assume that we have access to two types of data. First, as shown in Table 3.1, we assume that the log data in a lithography machine, denoted by **X**, has been collected and accurately parsed. Without loss of generality, we suppose there is a **Code** as the unique identifier for each *log event*, a **Level** roughly indicating the severity level of triggered *log event*, a **Detail** describing the detail of each *log message* that is an instantiation of a *log event* using a predefined template, and a **DateTime** containing the corresponding date and time. Hereinafter, we also call each *log event* a *log feature*. In addition, we may have log data for multiple lithography machines, and we use **M** to represent the corresponding name of the machine.

Ideally, applying an existing log-based anomaly detection method on **X** can detect most faults related to load and unload robots. However, due to the large number of *log features*, it is computationally prohibitive to directly use most existing anomaly detection methods. Furthermore, the presence of irrelevant *log features* may significantly degrade detection performance and even lead to misleading detection results.

Second, we also assume the availability of sensor data, denoted by **Y**, that measures the positions of robots. As shown in Table 3.2, there are measurements and corresponding timestamps from K different positions for the load robot and the unload robot, respectively. By using the *Log Event Vectorization* module in our proposed method (as will be explained in the sequel), **Y** can be rewritten as ($\mathbf{LP_1}, ..., \mathbf{LP_K}$, $\mathbf{UP_1}, ..., \mathbf{UP_K}$). Specifically, for $k \in \{1, ..., K\}$, $\mathbf{LP_k} = \{(Value_t, DateTime_t)\}_{t \in \mathbf{T}}$ de-

Table 3.2: Example sensor data. All values are fictional.

Robot	Position	Value	DateTime
Load	P_1	0.05	2020-01-01 00:00:00
⋮	⋮	⋮	⋮
Load	P_K	0.04	2020-01-01 00:01:00
Unload	P_1	0.04	2020-01-01 00:02:00
⋮	⋮	⋮	⋮
Unload	P_K	0.04	2020-01-01 00:03:00
⋮	⋮	⋮	⋮

notes the corresponding time series of load robot from position k, and at the same time $\mathbf{UP_k} = \{(Value_t, DateTime_t)\}_{t\in\mathbf{T}}$ denotes the corresponding time series of unload robot from position k, where $\mathbf{T}$ denotes the timestamp when the time series was sampled.

By applying an appropriate time series anomaly detection method on $(\mathbf{LP_1}, ..., \mathbf{LP_K})$ and $(\mathbf{UP_1}, ..., \mathbf{UP_K})$, we can detect certain faults (especially gradual faults) of the load and unload robot, respectively. However, due to the limited fault information contained in $\mathbf{Y}$, these faults are difficult to predict using sensor data only.

Therefore, we aim to address the following problem: *Suppose there is a complex system Δ that is composed of several interconnected subsystems $\{\Gamma, \Lambda, ..., \Theta\}$. Given a database of logs $\mathbf{X}_\Delta$ generated by the entire system Δ and a database $\mathbf{Y}_\Theta$ consisting of time series measured by sensors from a certain subsystem Θ, we assume the set of unique log events in $\mathbf{X}_\Delta$ is $\mathbf{C}_\Delta = \{C_1, C_2, ..., C_N\}$. Based on $\mathbf{Y}_\Theta$, we attempt to select a subset of $\mathbf{C}_\Delta$, denoted by $\mathbf{C}_\Theta = \{C_1^{'}, C_2^{'}, ..., C_L^{'}\}$ with $L \ll N$ and resulting in a new database $\mathbf{X}_\Theta \subset \mathbf{X}_\Delta$, that mainly keeps relevant log events for detecting and predicting faults that occurred in subsystem Θ.*

Specifically, as shown in Figure 3.1, system Δ is a lithography machine and subsystem Θ is the wafer transfer subsystem in this work. To solve the above problem, we propose a method consisting of three modules, namely *Log Event Vectorization*, *Selection of Relevant Features* and *Remove of Redundant Features*, each of which will be separately described next.

3.3.2 Log Event Vectorization (Module 1)

The first module, namely *Log Event Vectorization*, mainly aims at converting unstructured or semi-structured log events into time series data. Considering the log messages

given in Table 3.1, it is straightforward to generate a time series for each **Code** per machine by keeping only the corresponding records. Without loss of generality, we assume that the smallest unit of time in the original data is seconds. Accordingly, for each log event in a certain machine, we can obtain a time series in the form of $\{(Value_t, DateTime_t)\}_{t\in\mathbf{T}}$, meaning that this log event is triggered $Value_t$ times at the timestamp $DateTime_t$ for $t \in \mathbf{T}$, where $\mathbf{T}$ represent all time points (an ordered list) when this log event was triggered.

After preliminary results and based on domain knowledge, we have decided to take each day as an interval to count the number of times a log event is triggered. Besides, we take the start point of this time interval to represent the time point when this log event is triggered.

3.3.3 Selection of Relevant Features (Module 2)

Given that the load robot and unload robots are very similar, for simplicity, we only consider the load robot when elucidating the proposed method. That is, we assume a multivariate time series database $\mathbf{Y} = (\mathbf{Y_1}, ..., \mathbf{Y_K})$. Note that all these time series are of equal length. Therefore, we assume $\mathbf{Y_k} = \{y_{kt}\}_{t\in\mathbf{T}}$ for $k \in \{1, ..., K\}$, with $\mathbf{T}$ denoting the timestamp when $\mathbf{Y_k}$ was sampled. Meanwhile, after applying the *Log Event Vectorization* module on the log event database $\mathbf{X}$, we can obtain another multivariate time series database $\mathbf{Z} = (\mathbf{Z_1}, ..., \mathbf{Z_n}, ..., \mathbf{Z_N})$, where N represents the number of unique log features (i.e., log events). Although different log events are usually triggered at different time points, we have taken each day as an interval to count the number of times that each log event is triggered. As a result, for $n \in \{1, 2, ..., N\}$, $\mathbf{Z_n}$ has a fixed length and thus we assume $\mathbf{Z_n} = \{z_{ns}\}_{s\in\mathbf{S}}$, with $\mathbf{S}$ denoting the timestamp when $\mathbf{Z_n}$ was sampled.

To detect faults, for $k \in \{1, ..., K\}$, we can apply a univariate time series anomaly detector $\phi(\cdot)$ on $\mathbf{Y_k}$, resulting in $\phi(\mathbf{Y_k})$. Alternatively, we can apply a multivariate time series anomaly detector $\Phi(\cdot)$ on $\mathbf{Y}$ by jointly considering all $\mathbf{Y_k}$ for $k \in \{1, ..., K\}$, leading to $\Phi(\mathbf{Y_1}, ..., \mathbf{Y_K})$. As shown in Figure 3.2 (the two subplots at the bottom), the identification of faults is feasible by applying time series anomaly detectors on $\mathbf{Y}$. However, for gradual faults, just identifying them is not enough. It is also necessary to predict them accurately. Since there is only limited fault information in $\mathbf{Y}$, it is difficult to predict these faults based on $\mathbf{Y}$ alone. Therefore, we attempt to select relevant log features from $\mathbf{Z}$ to better detect and predict faults. For $k \in \{1, ..., K\}$, by considering $\mathbf{Y_k}$ as the target variable and $\mathbf{Z_1}, ..., \mathbf{Z_n}, ..., \mathbf{Z_N}$ as the prediction variables, it

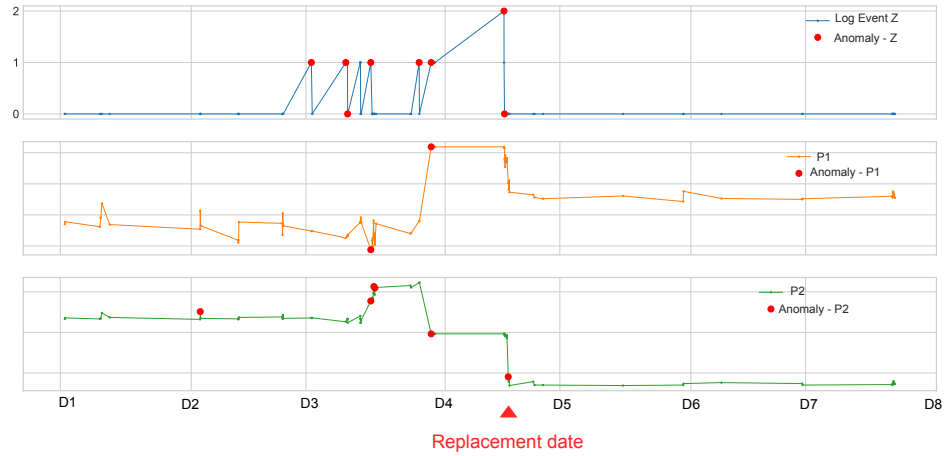


Figure 3.2: An example showing the relevance of a specific log event (upper plot) to detecting and predicting faults of a load robot based on sensor data (lower two plots). The shared x-axis represents the timestamp and the y-axes represent the measured values of each feature. Note that the y-axes of P_1 and P_2 are intentionally hidden and their timestamps are anonymized.

becomes a supervised feature selection problem. Compared to traditional supervised feature selection problems, however, there are three novel challenges:

1) The features considered are time series rather than numeric tabular data;

2) The target and prediction variables are not of equal length, and their timestamps are also different;

3) Traditional similarity metrics (e.g. Euclidean distance, dynamic time warping) do not give meaningful results when measuring the relevance/similarity of the predictor variable to the target variable.

We now detail why traditional similarity metrics fail to provide meaningful results when trying to find relevant log features. As shown in Figure 3.2, we can see that from 'D3' some faults started to appear in the robot and became detectable after a certain time based on **P1** and **P2**. These faults disappeared after the replacement of specific components on 'Replacement Date'. Meanwhile, we can observe that log event **Z** was triggered several times before the replacement date. More importantly, it was triggered several times even before the faults became detectable based on **P1** and **P2**. At other times, this log event was not triggered. In other words, the log event **Z** could

potentially be used to detect and predict these faults. However, if we consider their shapes (wrapped or not) or values (normalized or not), we can see that log feature **Z** is not similar to **P1** or **P2**. To address this problem, we propose a novel similarity metric that consists of three steps, as follows.

As summarized in Algorithm 2, for $k \in \{1, ..., K\}$, we first apply an appropriate univariate time series anomaly detector $\phi(\cdot)$ on $\mathbf{Y_k}$, resulting in a time series of anomaly scores $\mathbf{U_k} = \{u_{kt}\}_{t \in \mathbf{T}}$ (Lines 5-8). Second, for $n \in \{1, 2, ..., N\}$, we apply an appropriate univariate time series anomaly detector $\varphi(\cdot)$ on $\mathbf{Z_n}$, resulting in a time series of anomaly scores $\mathbf{V_n} = \{v_{ns}\}_{s \in \mathbf{S}}$ (Lines 9-12). Note that $\phi(\cdot)$ and $\varphi(\cdot)$ can be different considering that $\mathbf{Z_n}$ is sampled at a regular frequency (i.e., an observation per day) but $\mathbf{Y_k}$ is sampled at an irregular frequency (e.g., multiple observations in day A but no observation in day B). Third, we can select relevant log events by comparing $\mathbf{U_k}$ with $\mathbf{V_n}$ (Lines 13-23). Note that the lengths and scales of $\mathbf{U_k}$ and $\mathbf{V_n}$ may be different, but their peaks should overlap (for detection) or preferably the peak of $\mathbf{V_n}$ precedes the corresponding peak of $\mathbf{U_k}$ (for prediction) if we compare them using the same timeline. A peak here means a relatively high degree of outlyingness.

Time series anomaly detector $\phi(\cdot)$

Since $\mathbf{Y_k}$ is sampled at an irregular frequency, it may have multiple observations on a given day, but no observations in the following days. However, traditional time series anomaly detection methods usually assume that the input time series is regularly sampled. To circumvent this limitation, we adapt a simple yet effective anomaly detection strategy, which is called *persistence checking*. Specifically, for each time series value Y_{kt} in $\mathbf{Y_k}$, we compare this value with its previous value to obtain an anomaly score, defined as $U_{kt} = \phi(Y_{kt}) = |Y_{kt} - Y_{kh}|$ with $t = h + 1$. Particularly, this anomaly detector can deal with time series sampled at irregular frequencies.

Time series anomaly detector $\varphi(\cdot)$

Although $\mathbf{Z_n}$ is in the form of a time series, we are not concerned about the temporal order of observations when detecting anomalies. This is because if the lithography machine is working under a normal production environment, the occurrence of each log event should remain stable. Therefore, we can apply a traditional anomaly detector designed for tabular data on it. Specifically, we define $V_{ns} = \varphi(Z_{ns}) = \frac{|Z_{ns} - med(\mathbf{Z_n})|}{std(\mathbf{Z_n})}$ as the anomaly score for the sample point Z_{ns} in the time series $\mathbf{Z_n}$, where *med* and *std* denote the median and standard deviation of all sample points in $\mathbf{Z_n}$, respectively.

Feature selection for fault detection

We perform feature selection for fault detection by comparing obtained anomaly scores $\mathbf{U_k}$ and $\mathbf{V_n}$. Assuming that the robot works continuously for T days, for $\mathbf{V_n}$ we have a value per day. However, for $\mathbf{U_k}$, we may have multiple values on some days, but no values on most days. To make $\mathbf{U_k}$ and $\mathbf{V_n}$ comparable, we modify $\mathbf{U_k}$ as follows: for each day, if there are multiple anomaly scores, we take the maximum of these scores as the final anomaly score; if there is no anomaly score, we set the final anomaly score as zero. We denote the modified $\mathbf{U_k}$ as $\widehat{\mathbf{U}}_\mathbf{k}$, which has the same length as $\mathbf{V_n}$.

On this basis, we compute the Kendall's τ coefficient [121] between $\widehat{\mathbf{U}}_\mathbf{k}$ and $\mathbf{V_n}$ for feature selection. We prefer Kendall's rank correlation measure over other correlation measures for several reasons:

- it measures monotonicity relationships and has a straightforward interpretation;
- it does not require the tested features to follow a normal distribution, as anomaly scores usually do not follow a normal distribution;
- it is robust to noise and can be calculated easily.

Suppose $\mathbf{V_n} = (\alpha_1, ..., \alpha_T)$ and $\widehat{\mathbf{U}}_\mathbf{k} = (\beta_1, ..., \beta_T)$, then Kendall's τ coefficient measures the similarity of orderings of elements in $\widehat{\mathbf{U}}_\mathbf{k}$ and $\mathbf{V_n}$. Specifically, we let (α_1, β_1) , ..., (α_T, β_T) be paired observations. Furthermore, (α_i, β_i) and (α_j, β_j) are regarded as concordant if and only if one of the following conditions are fulfilled:

- $\alpha_i < \alpha_j$ and $\beta_i < \beta_j$;
- $\alpha_i > \alpha_j$ and $\beta_i > \beta_j$.

Otherwise, (α_j, β_j) are considered discordant. Kendall's τ coefficient is then defined as:

$$\tau = \frac{\#concordant - \#discordant}{\#concordant + \#discordant}, \tag{3.1}$$

where $\#concordant$ and $\#discordant$ represent the number of concordant pairs and the number of discordant pairs, respectively. Hence, τ can take a value in $[-1, 1]$, with a high absolute value indicating a high relevancy of log event $\mathbf{Z_n}$ to $\mathbf{Y_k}$ for detecting faults. By setting a threshold τ_0, we can select a subset of log events that are considered to be the most relevant for fault detection.

Feature selection for fault prediction based on Kendall rank correlation test

For $k \in \{1, ..., K\}$ and $n \in \{1, ..., N\}$, after obtaining $\mathbf{V_n} = (\alpha_1, ..., \alpha_T)$ and $\widehat{\mathbf{U}}_\mathbf{k} = (\beta_1, ..., \beta_T)$ as described in Chapter 3.3.3, we achieve feature selection for fault prediction by comparing $\mathbf{V_n}$ and the lagged values of $\widehat{\mathbf{U}}_\mathbf{k}$. The underlying rational is that if $\mathbf{Z_n}$ is useful for predicting faults in $\mathbf{Y_k}$, then the anomaly score vector $\mathbf{V_n}$ should be correlated with the lagged values of anomaly score vector $\widehat{\mathbf{U}}_\mathbf{k}$. In other words, the value of Kendall's τ coefficient between $\mathbf{V_n}$ and $Lag(\widehat{\mathbf{U}}_\mathbf{k}, l)$ should be high for some $l \in \{1, 2, ..., T\}$, where $Lag(\cdot, l)$ denotes the lag operator and l represents the the number of lagged times. For instance,

$$Lag((\beta_1, \beta_2, \beta_3, ..., \beta_{T-2}, \beta_{T-1}, \beta_T), 2) = (\beta_3, \beta_4, \beta_5, ..., \beta_T).$$

Hence, the resulting vector is shorter than the original vector.

For a specific l, we calculate the Kendall's τ coefficient between $\mathbf{V_n}$ and $Lag(\widehat{\mathbf{U}}_\mathbf{k}, l)$ as follows. We let (α_1, β_{1+l}) , ..., (α_{T-l}, β_T) be paired observations, and then count the number of concordant pairs and discordant pairs, followed by using Equation (3.1) to obtain the corresponding value of τ. A high value of τ indicates a high relevancy of log event $\mathbf{Z_n}$ to predict faults in $\mathbf{Y_k}$. Particularly, although l can take a value in $\{1, 2, ..., T\}$, an overly large value will cause few observations, and may produce misleading results. Therefore, we only consider $l \in \{1, 2, 3, 4, 5\}$ in this work. Similarly, this coefficient can take a value in $[-1, 1]$, with a high absolute value indicating a high relevancy of log event $\mathbf{Z_n}$ to predict $\mathbf{Y_k}$. By setting a threshold τ_1, we can select a subset of log events that are considered to be the most relevant for fault prediction.

Our preliminary results suggest that only applying Kendall rank correlation to select features for fault prediction may not be sufficient. Given a value for l, we only consider a single and equal number of lag operations at each time point. Therefore, it may result in few features selected for prediction. To mitigate this problem, we utilize Granger causality test to select more features for fault prediction, as follows.

Feature selection for fault prediction based on Granger causality test

The Granger causality test is widely used to determine whether a time series is useful for predicting another time series [122]. Formally, given an information set $\Omega_t = (\alpha, \beta)$ with $\alpha = (\alpha_1, ..., \alpha_t)$ and $\beta = (\beta_1, ..., \beta_t)$, α is said to Granger cause β if and only if $\delta_f^2(\beta_t - P(\beta_t|\beta_{j:j<t}, \alpha_{i:i<t})) < \delta_r^2(\beta_t - P(\beta_t|\beta_{j:j<t}))$. Specifically, δ_f^2

represents the variance of prediction errors generated by the optimal linear predictor $P(\beta_t|\beta_{j:j<t}, \alpha_{i:i<t})$ based on full information $(\alpha_{i:i<t}, \beta_{j:j<t})$. Meanwhile, δ_r^2 denotes the variance of prediction errors generated by the optimal linear predictor $P(\beta_t|\beta_{j:j<t})$ based on reduced information $(\beta_{j:j<t})$. In other words, the 'usefulness' for prediction in Granger causality test means the ability to increase the prediction accuracy. Therefore, the Granger causality test is suitable for selecting log features that can be used to more accurately predict faults.

More concretely, we specify the following linear prediction equation:

$$\beta_t = c + \sum_{i=1}^{I} a_i \alpha_{t-i} + \sum_{j=1}^{J} b_j \beta_{t-j} + \mu_t, \tag{3.2}$$

where I and J represent the number of lagged operations considered for α and β, respectively, while μ_t denotes the corresponding residual. Intuitively, if $\sum_{i=1}^{I} |a_i| \neq 0$, we can say α is useful for predicting β. Accordingly, we call Equation (3.2) the full or unrestricted regression model. Meanwhile, we call the following equation the reduced or restricted regression model:

$$\beta_t = c + \sum_{j=1}^{J} b_j \beta_{t-j} + \mu_t, \tag{3.3}$$

Specifically, given two time series $\alpha = (\alpha_1, ..., \alpha_t)$ and $\beta = (\beta_1, ..., \beta_t)$, the Granger causality test calculates the following F-statistic:

$$F = \frac{\frac{SSR_r - SSR_f}{J}}{\frac{SSR_f}{D-(I+J+1)}}, \tag{3.4}$$

where SSR_r is the sum of squared residuals on the reduced regression model and SSR_f is the sum of squared residuals on the full regression model. Besides, D denotes the degrees of freedom (i.e., the number of observations), while I and J have the same meaning as in Equation (3.2). Consequently, a higher value of F indicates that α is more useful for predicting β. As a result, α is declared Granger causal for β if the F-statistic is larger than the $(1 - Sig)\%$ quantile of an $F(J, D - (I + J + 1))$ distribution, where Sig denotes the significance level.

We should be aware that the Equation (3.4) requires several explicit and implicit assumptions to effectively identify Granger causality [122]:

- Linearity: the generating processes of the two time series are linear, and their

causal effects are also linear;

- Continuous-valued series: the two time series are supposed to have continuous-valued observations;
- Discrete-time: the two time series are sampled on a discrete and regular manner;
- Stationarity: the statistical properties of these two time series do not change over time;
- Known lag: the linear dependency on past values are assumed to have a known order;
- Perfectly observed: the two time series do not have measure errors;
- Complete system: there are no unmeasured confounders.

Overall, this subsection is summarized on Lines 5-23 in Algorithm 2.

3.3.4 Removal of Redundant Features (Module 3)

After selecting a subset of relevant log events for detecting and predicting faults in Module 2, we can further reduce the number of selected log events by computing the correlation between them. Specifically, we compute the pairwise Kendall's τ coefficient between selected log events and remove the redundant ones by setting a threshold τ_2 for the τ coefficient. More concretely, given two time series α and β, if the absolute value of their pairwise τ coefficient exceeds the threshold, we randomly remove one feature. This subsection is summarized on Lines 24-27 in Algorithm 2.

3.4 Experiments and Results

To demonstrate the effectiveness and efficiency of our method, we perform a series of experiments. The experimental setup, results, and corresponding analysis are described as follows.

3.4.1 Data Description

Given the novelty of the problems faced by this work, we are not aware of any publicly available datasets that can be used. The traditional benchmark datasets for

Table 3.3: Summary of datasets and experiment results. *Replacement* indicates the date when some components of the robot are replaced (faults always happen on or earlier than this date). *#Messages* represents the number of log messages. *#Raw* denotes the number of log features in the original dataset and *#Selected* denotes the number of selected log features. Besides, AD indicates whether the fault is detected based on the corresponding log features, where 'Yes' means the failure has been successfully detected or predicted and 'No' otherwise. If a fault is detected early (i.e., predicted), we further use 'P' to highlight it. Moreover, if a fault is only detected or predicted with the top 3 points but not the top 1 point, we denote it with an asterisk. Note that the value of *Fault* is anonymized, where GF, SF and $Unknown$ represent gradual faults, sudden faults or unknown types of faults, respectively.

Machine	Robot	Fault	Replacement	#Messages	#Raw(AD)	#Selected(AD)
1	Unload	GF_1	2021-01-05	103294	301(Yes)*	34(Yes)
2	Unload	GF_1	2021-01-06	90974	246(No)	47(Yes/P)
3	Unknown (Load)	GF_1	2021-01-10	90783	276(Yes)*	37(Yes)*
4	Unload	GF_1	2021-02-08	93729	437(No)	48(Yes)*
5	Unload	GF_1	2021-02-08	76797	245(No)	16(Yes)*
6	Load	SF_1	2019-06-24	86988	303(Yes)	38(Yes)
7	Unload	SF_1	2020-03-10	95513	376(Yes/P)	49(Yes/P)
8	Load	SF_1	2020-07-21	23262	404(Yes)	47(Yes)
9	Unload	SF_1	2020-07-24	34158	366(Yes)	69(Yes)
11	Unknown (Load)	SF_2	2019-01-24	13173	434(No)	27(Yes)
12 (1)	Load	SF_1	2019-03-22	51877	358(Yes/P)	95(Yes/P)
12 (2)	Unload	GF_1	2020-10-06	100223	337(Yes)	56(Yes)
13	Load	SF_3	2019-12-04	94426	294(Yes)	51(Yes)
14 (1)	Unload	$Unknown$	2020-01-28	116177	246(Yes)	15(Yes)
14 (2)	Unload	GF_1	2021-03-11	117146	327(Yes)*	72(Yes/P)
15 (1)	Load	$Unknown$	2020-03-24	101482	238(Yes)	9(Yes)
15 (2)	Unload	$Unknown$	2020-11-18	96719	293(Yes)*	39(Yes)
17	Unload	GF_1	2020-06-03	107414	217(No)	39(Yes)
18	Load	$Unknown$	2020-06-09	93875	402(Yes/P)	21(Yes/P)
19 (1)	Load	GF_1	2020-06-14	94125	259(No)	14(Yes)*
19 (2)	Load	GF_1	2020-10-19	106073	310(No)	19(No)
19 (3)	Load	$Unknown$	2021-03-29	105027	227(Yes)*	12(Yes)*
20	Unload	GF_1	2020-06-22	106070	215(Yes)*	21(Yes)*
21	Load	$Unknown$	2020-08-23	28414	345(Yes/P)	76(Yes/P)
22	Load	GF_1	2021-03-29	89350	187(No)	15(Yes)*

log anomaly detection, including HDFS [64], BGL [123], Spirit [123] and Thunderbird [123], are mainly produced by complex systems such as cloud service providers or supercomputers. As a result, sensor data for monitoring hardware is often unavailable.

Therefore, we use 25 real-world datasets provided by our industrial partners to test our method. Specifically, the 25 datasets were generated by 20 different test machines, of which 16 machines each generated one dataset and the remaining 4 machines each generated at least 2 datasets. A summary of the datasets is given in Table 3.3. These datasets contain multiple types of faults, including gradual faults, sudden faults, and some unknown types of faults. Moreover, these faults may occur in the loading and/or unloading robot. In particular, for some machines it is unknown whether the fault

occurred in the unloading or loading robot. Therefore, we examined the datasets of both robots to find the robot most likely to have failed.

3.4.2 Baseline, Hyperparameter settings and Performance metrics

Based on domain knowledge and preliminary experiments, in module 2 we set the threshold $\tau_0 = 0.6$ for the similarity coefficient to select log events for fault detection. Furthermore, we set $\tau_1 = 0.6$ in the Kendall rank correlation and $Sig = 0.05$ in the Granger causality test to select features for fault prediction. In module 3, we further remove some highly correlated log features by setting $\tau_2 = 0.95$.

After constructing the event count matrix based on the selected features, we apply a commonly used unsupervised anomaly detector on it: KNN [124]. To demonstrate the necessity of feature selection, we also build an event count matrix from all original features and then apply KNN on it as a baseline. Specifically, a fault is considered detected if at least one of the points with the highest anomaly scores (we consider the top 3 points in our experiments) is on the date of the known fault. Moreover, a fault is considered predicted if one of the points with the highest anomaly scores is located slightly earlier (we consider 1-7 days) than the date when the fault is known to occur. On this basis, as a performance metric, we count the number of datasets in which the faults are accurately detected and/or predicted.

3.4.3 Results and Analysis

As shown in Table 3.3, our proposed feature selection method can help improve log-based anomaly detection performance. Specifically, based on the selected log features, KNN was able to accurately detect or predict faults in 24 out of 25 machines. In contrast, KNN can only accurately detect or predict faults in 17 out of 25 datasets based on all log features. One possible reason is that logs from all subsystems are entangled and the inclusion of many irrelevant log events renders the detection/prediction of gradual faults difficult.

3.5 Conclusion and Future Work

In this work we have proposed a simple yet effective feature selection method for log based anomaly detection. This method has been empirically proven to be effective on 25 real-world datasets. In the future, we plan to include more datasets for testing,

evaluating whether our approach generalizes to other settings, and investigating hyperparameter tuning. More importantly, we will try more anomaly detection methods when defining $\phi(\cdot)$ and $\varphi(\cdot)$. Particularly, the Equation (3.4) used in Granger causality requires several explicit and implicit assumptions to effectively identify Granger causal effects. Some of these assumptions may not be fulfilled by our use-case though. Therefore, we will further explore and improve Granger causality test [125] or other similar techniques to find log events that can be used to predict sensor time series anomalies. Furthermore, we have not fully explored the causal relationships between different log events. In the future, by constructing a causality graph using techniques such as the PC-algorithm [126] on log events, we can investigate the causal relationships between log events. As a result, it might be possible to pinpoint the root causes of anomalies.

Algorithm 2 Feature Selection for Fault Detection and Prediction (FS4FDP)

Input: Event log database $\mathbf{Z} = (\mathbf{Z}_1, ..., \mathbf{Z}_n, ..., \mathbf{Z}_N)$; Sensor database $\mathbf{Y} = (\mathbf{Y}_1, ..., \mathbf{Y}_k, ..., \mathbf{Y}_K)$; Anomaly detector $\varphi(\cdot)$ for $\mathbf{Z}_n$; Anomaly detector $\phi(\cdot)$ for $\mathbf{Y}_k$; Threshold parameters τ_0, τ_1, τ_2 and Sig with $\tau_0 = \tau_1 = 0.6$, $\tau_2 = 0.95$ and $Sig = 0.05$ by default.

Output: Subset of log features $\mathbf{F}$

procedure FS4FDP($\mathbf{Z}, \mathbf{Y}, \varphi(\cdot), \phi(\cdot), \tau_0, \tau_1, \tau_2, Sig$)
  $\mathbf{F} \Leftarrow \{\}$
  $\mathbf{U} \Leftarrow \{\}$
  $\mathbf{V} \Leftarrow \{\}$
  for $k \in \{1, ..., K\}$ **do** ▷ Obtaining anomaly score for each sensor time series
    $\mathbf{U}_k = \varphi(\mathbf{Y}_k)$
    $\mathbf{U}.append(\mathbf{U}_k)$
  end for
  for $n \in \{1, ..., N\}$ **do** ▷ Obtaining anomaly score for each log event time series
    $\mathbf{V}_n = \varphi(\mathbf{Z}_n)$
    $\mathbf{V}.append(\mathbf{V}_n)$
  end for
  for $\mathbf{U}_k \in \mathbf{U}$ and $\mathbf{V}_n \in \mathbf{V}$ **do**
    $\widehat{\mathbf{U}}_k \Leftarrow Modify(\mathbf{U}_k)$ ▷ Modifying the length of anomaly score vector of each sensor time series
    if $|\tau(\widehat{\mathbf{U}}_k, \mathbf{V}_n)| > \tau_0$ **then** $\mathbf{F}.append(\mathbf{Z}_n)$ ▷ Feature selection for fault detection using Kendall rank correlation
    end if
    for $l \in \{1, 2, 3, 4, 5\}$ **do** ▷ Feature selection for fault prediction using Kendall rank correlation on lagged values
      if $|\tau(Lag(\widehat{\mathbf{U}}_k, l), \mathbf{V}_n)| > \tau_1$ **then** $\mathbf{F}.append(\mathbf{Z}_n)$
      end if
    end for
    if $Pvalue(GrangerCausalityTest(\mathbf{V}_n, \widehat{\mathbf{U}}_k)) < Sig$ **then** $\mathbf{F}.append(\mathbf{Z}_n)$ ▷ Feature selection for fault prediction using Granger causality test
    end if
  end for
  for $\mathbf{F}_i, \mathbf{F}_j \in \mathbf{F}$ with $i \neq j$ **do** ▷ Removing redundant features using Kendall rank correlation
    if $|\tau(\mathbf{F}_i, \mathbf{F}_j)| > \tau_2$ **then** $\mathbf{F}.delete(\mathbf{F}_i)$
    end if
  end for
  return F
end procedure

Chapter 4

Explainable Contextual Anomaly Detection using Quantile Regression Forests

Authors: **Zhong Li**, Matthijs van Leeuwen
Published in Data Mining and Knowledge Discovery, 37(6), 2517-2563.

Abstract

Traditional anomaly detection methods aim to identify objects that deviate from most other objects by treating all features equally. In contrast, contextual anomaly detection methods aim to detect objects that deviate from other objects within a *context* of similar objects by dividing the features into contextual features and behavioral features. In this paper, we develop connections between dependency-based traditional anomaly detection methods and contextual anomaly detection methods. Based on resulting insights, we propose a novel approach to inherently interpretable contextual anomaly detection that uses Quantile Regression Forests to model dependencies between features. Extensive experiments on various synthetic and real-world datasets demonstrate that our method outperforms state-of-the-art anomaly detection methods in identifying contextual anomalies in terms of accuracy and interpretability.

4.1 Introduction

According to the well-known definition of [127], an anomaly is an object that is notably different from most remaining objects. [31] subdivided anomalies into three types: point anomalies (an object is considered anomalous when compared against the rest of objects), contextual anomalies (an object is anomalous in a specific context), and collective anomalies (a collection of objects is anomalous with respect to the entire dataset). The analysis of anomalies has a wide range of applications, such as in network security [128], bioinformatics [129], fraud detection [130], and fault detection and isolation [131].

Anomaly analysis consists of two equally important tasks: anomaly detection and anomaly explanation. A wealth of 'shallow' machine learning based methods, i.e., not based on deep learning, have been proposed to detect anomalies [31]. More recently, many deep learning based anomaly detection methods have also been developed [36]. However, deep learning based anomaly detection methods are notoriously known as not being interpretable, in the sense that generally both the model itself is non-transparent and the resulting anomaly scores are challenging to interpret without the use of a post-hoc explainer. In this paper it is especially the latter that we consider to be problematic, as post-hoc explanations often rely not only on the model but also on the specific explainer used. In addition, deep learning methods typically require large amounts of data and training the models is a time-consuming process. In many real-world applications, however, 'native' interpretability (i.e., without posthoc explainer) may be required, and limited data and/or computation time may be available. For these reasons, in this paper we restrict our focus to 'shallow' machine learning based methods. In correspondence with this choice, we focus on settings where the amount of data is smaller than is typically required to learn accurate deep models. Most existing shallow methods only consider point anomaly detection, largely ignoring contextual anomaly detection. Moreover, anomaly explanation has received very limited attention. In this paper, we address both the problem of contextual anomaly detection and that of anomaly explanation, for small to moderately sized tabular data having categorical and/or quantitative features.

Shallow anomaly detectors are typically categorized into distance-based, density-based, and distribution-based approaches [132]. Distance-based and density-based methods use knowledge about the spatial proximity of objects to identify anomalies, while distribution-based methods use knowledge about the distribution of the data to detect anomalies. These methods work under the assumption that *objects having a*

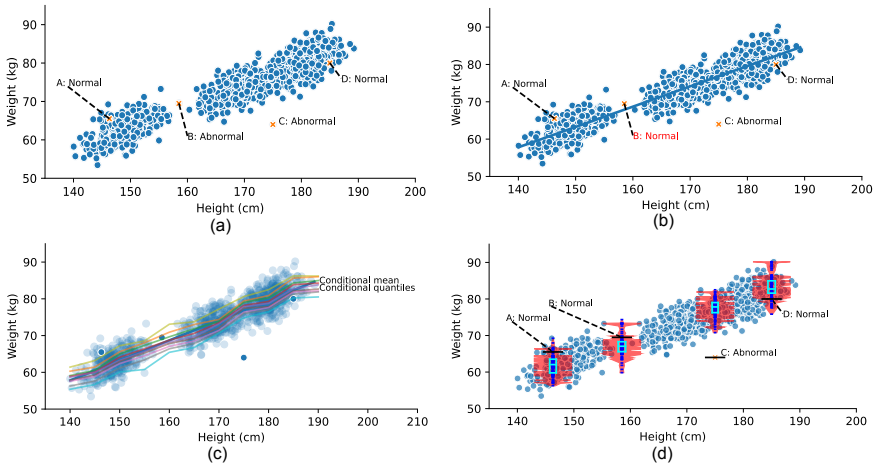


Figure 4.1: Different types of anomaly detection. (a) Distance- and density-based methods both consider object B to be abnormal, because it is far away from other objects and in a low-density region. (b) Dependency-based methods model the relationship between height and weight, and consequently consider object B to be normal. (c) Conditional quantiles provide more information about the conditional distribution than just the mean, and (d) can be used to visualize—using beanplots (or, rather, a variation of a beanplot combined with boxplots, see Chapter 4.5.2 for details)—why a certain object is considered (ab)normal.

large distance or different density from their spatial neighbors are anomalous or *objects that take rare values under a marginal or joint distribution of features are anomalous*, respectively. Using either of these assumptions may lead to false positives though. For example, as shown in Figure 4.1(a), these methods will mistakenly identify object B as an anomaly if the aim is to detect people that are over- or underweight.

As this is often undesirable, we aim to define and detect anomalies from another perspective, that is, we assume that *objects that violate a dependency, i.e., a relationship between features, are anomalous*. This is the idea behind dependency-based anomaly detection [133], which is not new but has received limited attention compared to other types of anomalies. It leverages the intrinsic structure and properties of the data to find potentially more relevant anomalies. For example, Figure 4.1(b) shows that this approach will identify object B as normal by modeling the dependency between height and weight.

Traditional anomaly detection techniques—including distance-, density-, and dependency-based methods—treat all features equally when identifying anomalies. However, in domains such as healthcare, sensor networks, and environment protection, some features should never be used directly to quantify anomalousness. For instance, forest

fire detection systems should not treat 'deviating' values of latitude, longitude, date, and time as an indication of an anomaly. We should not simply discard these features either though, as they may contain relevant information. For example, the 'normal' temperature may be higher for certain regions than for others. This motivates us to investigate *contextual* anomaly detection, which can take such extra features into account.

Contextual anomaly detection assumes that *an object is anomalous if it strongly deviates from objects within its 'context' of similar objects.* The apparent contradiction in this assumption is explained by the division of the features into two disjoint subsets, i.e., contextual features and behavioral features. The contextual features are only used to define the contexts, using some similarity measure, while the behavioral features are only used to determine whether an object deviates from other objects within its context. Domain knowledge often leads to a natural division between contextual and behavioral features.

We observe that both contextual and dependency-based anomaly detection methods identify anomalies by explicitly or implicitly exploring dependencies between features. Concretely, dependency-based anomaly detection methods model dependency relationships between all features explicitly, while contextual anomaly detection methods model dependency relationships between behavioral and contextual features implicitly or explicitly. As far as we know this connection has not yet been pointed out in the literature.

Approach and contributions. In this paper, we introduce an approach for contextual anomaly detection and explanation that integrates the core principle of dependency-based anomaly detection into contextual anomaly detection to obtain a very accurate approach. As is common, we use regression analysis to model dependencies. Existing methods for dependency-based and contextual anomaly detection that use regression, however, typically only estimate the conditional mean of the response variable and directly interpret that as 'normal' value. This strongly limits how well anomalies can be detected, as the conditional mean provides very limited information about the conditional distribution. We therefore use *quantile regression*, which can model a conditional distribution in much more detail by estimating conditional quantiles. Figure 4.1(c) shows how conditional quantiles provide more information about the relationship between weight and height than a conditional mean could.

More specifically, a subset of the features, dubbed contextual features, are used to define the context of an object, while the remaining features, dubbed behavioral features, are used for detecting deviations within a context. In this paper we assume

that the contextual features can be mixed but all behavioral features are numerical. Given this context and our aims, we use Quantile Regression Forests [134] to perform predictions for each behavioral feature and obtain corresponding uncertainty quantifications. By summing the quantified uncertainties (with a wider quantile interval representing a higher level of uncertainty) for all individual behavioral features, we obtain the anomaly score for a data instance. By attributing parts of the anomaly score to individual behavioral features, the approach intrinsically provides explanations in the sense that it can convey to which extent which features contributed to making the instance an anomaly. This offers advantages to post-hoc explanation methods such as SHAP [135], which we do not consider 'attributable' for the following two reasons. First, the post-hoc explanation may not match the information/rationale used by the model to detect an anomaly [70]. Second, it has recently been shown [136] that attribute-based explanations cannot be both recourse sensitive and robust, which is a good reason to avoid such explainers when possible and use a 'native' attribution-based method instead.

As far as we are aware, we are the first to use quantile regression for dependency-based or contextual anomaly detection. Specifically, we choose to employ Quantile Regression Forests for two reasons. First, it can model both linear and non-linear dependency between features. Second, in the paper that introduced the method it was empirically shown to outperform other conditional quantile estimators in most cases. Figure 4.1(d) shows how the estimated conditional quantiles can be used to approximate the conditional probability density at different locations, which we can use to accurately detect anomalies. Moreover, the quantiles are helpful to explain why an object is considered an anomaly without having to explicitly refer to other objects.

The main contributions of our work can be summarized as follows: (1) We identify a connection between dependency-based traditional anomaly detection methods and contextual anomaly detection methods, and exploit this observation to introduce a novel high-level approach to contextual anomaly detection and explanation; (2) We instantiate this generic approach using quantile regression (and Quantile Regression Forests specifically) for anomaly detection and a beanplot-based visualization for anomaly explanation; and (3) We perform extensive experiments on synthetic and real-world datasets to empirically demonstrate the effectiveness, and interpretability of the proposed method when compared to state-of-the-art methods.

The remainder of this paper is organized as follows. Chapter 4.2 discusses related work, both in contextual and traditional anomaly detection. Chapter 4.3 introduces notation, formalizes the problem, and presents the high-level approach that we pro-

pose. Chapter 4.4 then describes some technical preliminaries, most notably Quantile Regression Forests. Chapter 4.5 introduces QCAD, our proposed method for Quantile-based Contextual Anomaly Detection and Explanation that instantiates the high-level approach. Chapter 4.6 empirically compares QCAD to its competitors, and provides a case study that investigates the use of QCAD to find exceptional football players from data. Chapter 4.7 concludes the paper.

4.2 Related Work

We first discuss related work on contextual anomaly detection and explanation, and then proceed with the two most closely related types of traditional anomaly detection: dependency-based and subspace-based anomaly detection.

4.2.1 Contextual Anomaly Detection and Explanation

Contextual anomaly detection has received particular attention in spatial data [137], temporal data [138], and spatio-temporal data [139], where spatial and/or temporal features are used to define contexts. These methods are not directly applicable to other domains, where the contexts are defined by other types of features; 'generic' contextual anomaly detection has received limited attention in the community.

CAD [140] is a seminal work that introduced generic contextual anomaly detection. It assumes a user-specified partition of features into contextual (called 'environmental') and behavioral (called 'indicator') features, and uses Gaussian Mixture Models to fit the distributions of the contextual and behavioral feature spaces. Dependencies between contextual features and behavioral features are then learned by means of 'mapping functions', and an object is considered anomalous if it violates the learned functions. For this to work CAD assumes that both the contextual and behavioral features consist of an unknown number of multiple Gaussian components, which may be a strong assumption in practice. Further, CAD can only handle numerical features and is computationally very expensive. Our proposed method makes no assumptions about the distribution of the features, can deal with mixed contextual features and numerical behavioral features, and we will show empirically that it is computationally more efficient than CAD while achieving a higher detection accuracy.

ROCOD [141] is also closely related, and uses local and global models of expected behavior to describe the dependencies between contextual and behavioral features. Concretely, standard regression models such as CART are used to learn global pat-

terns, with contextual features as predictor variables and behavioral features as response variables. Local patterns are computed based on the means of behavioral feature values of an object's neighbors. An object's actual value is compared to the local and global pattern, and the weighted average of these differences forms the anomaly score. As the conditional mean describes only one aspect of a conditional distribution, and is not necessarily the point with the highest probability of occurrence (i.e., the mode). To address this, our method employs quantile regression analysis to estimate conditional quantiles, which provide a much more complete description of the conditional distribution. As a result, our method empirically outperforms ROCOD in terms of accuracy.

With the increasing use of anomaly detection algorithms in safety-critical domains, such as healthcare and manufacturing, the ethical and regulatory obligations to provide explanations for the high-stakes decisions made by these algorithms has become more pronounced [70]. Existing contextual anomaly detection do not provide such explanations though, and non-trivial modifications would be needed for them to do so. Specifically, CAD [140] computes the anomaly score of a data instance by measuring its deviation from the mapping functions that are learned from the majority of data instances. This approach poses a challenge in generating intrinsic explanations, as it does not allow for the attribution of the anomaly score to individual features. Further, ROCOD [141], the other known method for contextual anomaly detection, calculates the weighted average of an instance' s differences to the learned local and global patterns as its anomaly score. The local pattern is obtained using its neighbors, while the global pattern is learned using all instances, making it difficult to associate the anomaly score with individual features.

Despite the existence of numerous methods for explaining anomalies, such as those outlined in recent (survey) papers [142, 70, 143], there has been very limited research on contextual anomaly explanation. Particularly, COIN [144] explains outliers by reporting their outlierness score, the features that contribute to its abnormality, and a contextual description of its neighborhoods. COIN treats all features equally though, while our method divides features into contextual and behavioral features. Further, we develop a visualization that helps explain contextual anomalies in addition to reporting the overall anomaly score, feature importance, contextual neighbors, and individual anomaly score for each behavioral feature.

The following methods are less relevant because they consider slightly different problems. [145] construct a non-parametric graph-based method for conditional anomaly detection, which only addresses the problem of a single categorical behavioral fea-

ture. [146] first identify anomalies on behavioral features, and then refine the detected anomalies by clustering all objects on contextual features. [147] detect group anomalies from multidimensional categorical data. [148] present a contextual anomaly detection framework dedicated to categorical behavioral features which also considers the dependencies between behavioral features. Moreover, [149] apply robust metric learning on contextual features to find more meaningful contextual neighbors and then leverage k-NN kernel regression to predict the behavioral feature values. [150] develop ConOut to automatically find and incorporate multiple contexts to identify and interpret outliers.

4.2.2 Traditional Anomaly Detection

Although traditional anomaly detection considers a problem that is different from contextual anomaly detection, dependency-based and subspace-based anomaly detection leverage techniques that are related to our method.

Dependency-based Anomaly Detection

Dependency-based anomaly detection aims to identify anomalies by exploring dependencies between features. [151] explores the dependency between non-target and target features to identify and correct possible noisy data points. To detect networking intrusions, [152] present Cross-Feature Analysis to capture the dependency patterns between features in normal networking traffic. To detect disease outbreaks, [153] propose to explore the dependency between features using a Bayesian network. [154] propose to detect anomalies by using an ensemble of models, with each model exploring the dependency between a response feature and other features. [155] use Linear Gaussian Bayesian networks to detect anomalies that violate causal relationships.

LoPAD [156] first uses a Bayesian network to find the Markov Blankets of each feature. Then, a predictive model (e.g., CART) is learned for each individual feature as response variable, with its Markov Blankets as predictor variables. Given an object, LoPAD computes the Euclidean distance between its actual value and predicted value (i.e., conditional mean) as its anomaly score, for each feature. The resulting anomaly scores are normalized and summed to obtain the final anomaly score for an object. The method does not distinguish contextual and behavioral features and—like ROCOD—uses conditional means to represent the conditional distribution. Consequently, LoPAD cannot (accurately) detect contextual anomalies.

Subspace-based Anomaly Detection

Subspace-based anomaly detection seeks to find anomalies in part of the feature space. Specifically, [157] propose SOD to identify outliers in varying subspaces. Concretely, they construct axis-parallel subspaces spanned by the neighbors of a given object. On this basis, they investigate whether this object deviates significantly from its neighbors on any of these subspaces. Furthermore, [158] extend this work to determine whether an object is anomalous on arbitrarily oriented subspaces spanned by its neighbors. [159] propose to find subspaces with strong mutual correlations and then identify anomalies on these subspaces. Finally, [160] use archetype analysis to project the feature space into various subspaces with linear correlations based on nearest neighbors. On this basis, they explore outliers by ensembling the results obtained on relevant subspaces. Overall, these methods pursue to identify anomalies in a subset of features, but treat all features equally and are thus not suitable to identify contextual anomalies.

4.3 Contextual Anomaly Detection and Explanation

We first introduce the necessary terminology and notations, and illustrate this with the running example depicted in Table 4.1. A dataset $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_N\}$ contains N instances (or data points) over the set of features (a.k.a. attributes or variables) denoted by $\mathbf{F} = \{\mathbf{f}^1, \ldots, \mathbf{f}^j, \ldots, \mathbf{f}^D\}$. x_i^j denotes the value of the i-th object, $\mathbf{x}_i$, for the j-th feature, $\mathbf{f}^j$. In the running example, we have $N = 16$, $D = 6$, $\mathbf{F} = \{Latitude, Longitude, Season, Temperature, Rain, Wind\}$, and $x_1^{Rain} = 69$.

We assume that feature set $\mathbf{F}$ is divided into two disjoint feature sets (typically using domain knowledge): a contextual feature set $\mathbf{C} = \{\mathbf{c}^1, ..., \mathbf{c}^p, \ldots, \mathbf{c}^P\}$ and a behavioral feature set $\mathbf{B} = \{\mathbf{b}^1, \ldots, \mathbf{b}^q, \ldots, \mathbf{b}^Q\}$, such that $D = P + Q$, $\mathbf{F} = \mathbf{C} \cup \mathbf{B}$, and $\mathbf{C} \cap \mathbf{B} = \emptyset$. Without loss of generality, we can rearrange the features of an object $\mathbf{x}_i = (x_i^1, ..., x_i^j, ..., x_i^D)$ and represent it as $\mathbf{x}_i = (x_i^1, ..., x_i^P, x_i^{P+1}, ..., x_i^{P+Q}) = (c_i^1, ..., c_i^p, ..., c_i^P, b_i^1, ..., b_i^q, ..., b_i^Q) = (\mathbf{c}_i, \mathbf{b}_i)$, so that $\mathbf{c}_i$ and $\mathbf{b}_i$ denote its contextual and behavioral feature values, respectively. Accordingly, we refer to the space spanned by $\mathbf{C}$ as *contextual space*, i.e., $\mathcal{C} = \mathbf{c}^1 \times ... \times \mathbf{c}^p \times ... \times \mathbf{c}^P$, and to the space spanned by $\mathbf{B}$ as *behavioral space*, i.e., $\mathcal{B} = \mathbf{b}^1 \times ... \times \mathbf{b}^q \times ... \times \mathbf{b}^Q$.

Finally, let Pow denote the powerset, i.e., $\text{Pow}(\mathbf{X}) = \{X \subseteq \mathbf{X}\}$.

Table 4.1: Running example: fictional climate data for Dutch cities. Temperature is measured in degrees Celsius, rain in mm, and wind in miles per hour. The city is not used for anomaly detection. *Latitude*, *Longitude* and *Season* are treated as contextual features, while **Temperature**, **Rain** and **Wind** are considered the behavioral features.

City	*Latitude*	*Longitude*	*Season*	**Temperature**	**Rain**	**Wind**
Leiden	52.16	4.49	Winter	3.0	69	16
Amsterdam	52.37	4.89	Winter	2.9	55	17
Rotterdam	51.92	4.46	Winter	2.7	60	15
Oss	51.45	5.31	Winter	1.1	58	25
Eindhoven	51.44	5.46	Autumn	16.6	62	25
Delft	52.00	4.21	Autumn	16.1	45	19
Utrecht	52.09	5.10	Autumn	18.3	42	20
The Hague	52.07	4.28	Autumn	18.5	49	18
Tilburg	51.33	5.52	Summer	22.1	39	22
Middelburg	51.49	3.61	Summer	20.3	41	23
Arnhem	51.98	5.89	Summer	19.6	48	17
Venlo	51.37	6.17	Summer	21.8	43	35
Emmen	52.46	6.55	Spring	8.3	80	10
Meppel	52.69	6.19	Spring	7.1	27	13
Groningen	53.13	6.34	Spring	4.2	17	19
Leeuwarden	53.10	5.80	Spring	7.2	17	19

4.3.1 Problem Statement

In contextual anomaly detection, contextual features are used to determine the so-called *context* of an object. An object's context is used to estimate whether it is anomalous. The latter is achieved by comparing the object's values for the behavioral features to what is 'normal' within the object's context—if the object's behavioral values strongly deviate, it is flagged as an anomaly.

For contextual anomaly detection to be meaningful, we must assume that *there exist dependencies between the contextual and behavioral data.* If such a relationship does not exist, there is no need to use contextual anomaly detection; one could simply remove the contextual features and reduce the problem to a traditional anomaly detection problem.

We illustrate this using the running example in Table 4.1. We can detect anomalous weather taking into account different regions and seasons by specifying contextual feature set $\mathbf{C} = \{Latitude, Longitude, Season\}$ and behavioral feature set $\mathbf{B} = \{Temperature, Rain, Wind\}$. Each city is now only compared to cities with a similar latitude, longitude, and season, i.e., its context. If a city has values for temperature, rain, and/or wind that strongly deviate from those of the cities in its context, it is marked as anomalous.

For example, Amsterdam and Rotterdam could form the context of Leiden; they are both nearby and we have measurements for the same season. Temperature and wind are similar for all three cities, but there was substantially more rain in Leiden than in Amsterdam and Rotterdam. Hence, for that reason Leiden could be flagged as an anomaly.

To formalize the problem, we introduce a generic 'context function' that maps each possible data point to a subset of the dataset, i.e., its context, based on the data point's contextual features.

Problem 1: Contextual Anomaly Detection Given a dataset $\mathbf{X}$ with a feature set $\mathbf{F} = (\mathbf{C}, \mathbf{B})$, a context function $Context : \mathcal{C} \rightarrow \text{Pow}(\mathbf{X})$, an anomaly detector $Anomaly : \mathcal{B} \times \text{Pow}(\mathbf{X}) \rightarrow \mathbb{R}^+$, and a threshold ϕ, find all data points for which the anomaly scores exceed ϕ and are thereby flagged as anomalous—based on the behavioral features—within their individual contexts, i.e., $\{(\mathbf{c}, \mathbf{b}) \in \mathbf{X} \mid Anomaly(\mathbf{b}, Context(\mathbf{c})) \geq \phi\}$.

Note that the context is determined based only on contextual feature values, and that the anomaly detector may only use the behavioral feature values of the data points in the given context when establishing if the given data point is anomalous or

not.

In practice, analysts are not only interested in identifying anomalies, but also need to know the underlying reasons for why a specific object is reported as anomaly. This leads to the second problem that we consider.

Problem 2: Contextual Anomaly Explanation Given an anomalous object $\mathbf{x} \in \mathbf{X}$ and the context function *Context* and anomaly detector *Anomaly* that were used to detect it, find the behavioral features $\mathbf{B}' \subseteq \mathbf{B}$ for which $\mathbf{x} = (\mathbf{c}, \mathbf{b})$ substantially deviates from $Context(\mathbf{c})$.

4.3.2 Overall Approach

The problem statement in the Chapter 4.3.1 suggests a three-pronged approach based on 1) context generation, 2) anomaly detection, and 3) anomaly explanation. In this subsection we explain and illustrate the overall approach that we propose, using the running example from Table 4.1.

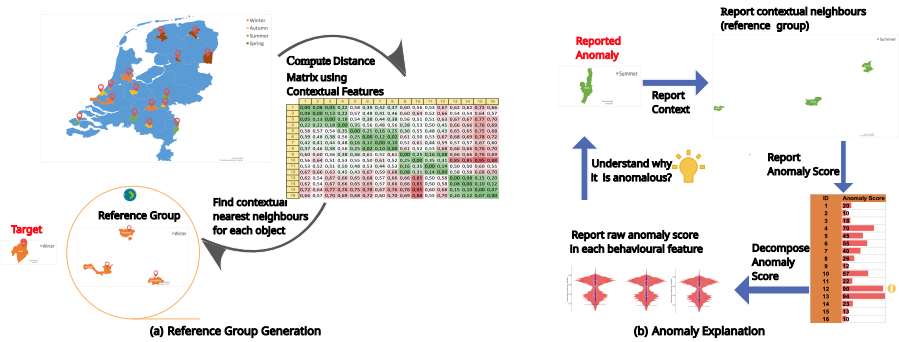


Figure 4.2: Reference Group Generation and Anomaly Explanation. The Reference Group Generation phase computes the distance matrix using only the contextual features, and finds a reference group for each object on this basis. The Anomaly Explanation phase produces an explanation for each identified anomaly by reporting its contextual neighbors, final anomaly score, and raw anomaly scores in each behavioral feature.

Phase 1: Reference Group Generation Many choices are possible for the context function; in this manuscript we choose to use an object's k nearest neighbors, which we refer to as *reference group*. The most important reasons for this choice are that 1) once a global contextual distance matrix has been computed the nearest neighbors of any object can be found relatively quickly; 2) this approach only requires a distance metric to be chosen, which can be defined for any type of data and be adapted to the

problem at hand; and 3) it is generic enough to allow for different uses of the resulting contexts.

Given a dataset $\mathbf{X}$ with contextual feature set $\mathbf{C}$, we first compute the distance matrix $\mathbf{M}$ between all objects using only the contextual features. Second, for any object $\mathbf{x} \in \mathbf{X}$, we find its k nearest neighbors in *contextual space* based on $\mathbf{M}$. As a result, the k nearest neighbors of $\mathbf{x}$ form a reference group, denoted as $R(\mathbf{x}, k)$, which serves as context in Problems 1 and 2.

For instance, as shown in Figure 4.2 (a), given $\{Latitude, Longitude, Season\}$ as the contextual feature set, we first calculate the distance matrix in the contextual space $Latitude \times Longitude \times Season$. Second, based on the distance matrix, we can find the three nearest neighbors of any object as its reference group[1]. Concretely, the three nearest neighbors for $\mathbf{x}_1$ are $\{\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$, which forms a reference group for $\mathbf{x}_1$, denoted as $R(\mathbf{x}_1, 3)$.

Phase 2: Anomaly Detection Given an object $\mathbf{x} \in \mathbf{X}$ with its reference group $R(\mathbf{x}, k)$, we apply an anomaly detector *Anomaly* to obtain an anomaly score *based only on the behavioral attribute values*. We repeat the above process for all objects, leading to N anomaly scores $S = \{s_1, s_2, ..., s_N\}$. We sort S and use threshold ϕ to obtain a ranked list of contextual anomalies $A = \{\mathbf{a}_1, ..., \mathbf{a}_m, ..., \mathbf{a}_M\}$, with $M \ll N$.

For example, we can apply an anomaly detector on the *behavioral space* $Temperature \times Rain \times Wind$ of $\mathbf{x}_1$ and its reference group $R(\mathbf{x}_1, 3) = \{\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$. As a result, we obtain anomaly score s_1 for $\mathbf{x}_1$. Accordingly, repeating this process leads to a set of anomaly scores $S = \{s_1, ..., s_{16}\}$. In our running example we find $\{\mathbf{x}_4, \mathbf{x}_{12}, \mathbf{x}_{13}\}$ as contextual anomalies, as they behave differently in the behavioral space $Temperature \times Rain \times Wind$ when compared to their corresponding neighbors defined in the contextual space $Latitude \times Longitude \times Season$. For example, Oss ($\mathbf{x}_4$) has relatively stronger winds in winter when compared to its nearest cities Leiden, Amsterdam and Rotterdam.

We require anomaly detectors to be *attributable*, meaning that an anomaly score s generated by an anomaly detector must be decomposable into individual contributions towards anomalousness for each behavioral feature using the anomaly detector's native structure.

[1] We use $k = 3$ for illustrative purposes; in practice, we would have a dataset with far more than 16 records and k would also be much larger.

Phase 3: Anomaly Explanation For each anomalous object $\mathbf{a}_m$, we first report its reference group $R(\mathbf{a}_m, k)$ using the distance matrix obtained in Phase 1. Second, we report its anomaly score s_m, as obtained in Phase 2. Moreover, we decompose s_m into individual contributions from each behavioral feature $\mathbf{b}^q \in \mathbf{B}$, resulting in a list of raw anomaly scores $\mathbf{s}_m = \{s_{m1}, ..., s_{mq}, ..., s_{mQ}\}$. This is possible because s_m is *attributable*. Next, we report the top-h raw anomaly scores in $\mathbf{s}_m$ with their corresponding behavioral features, where h can be specified by the analyst. The top-h behavioral features and raw scores enable analysts to better understand why a specific object is flagged as a contextual anomaly.

For example, Figure 4.2 (b) reports object $\mathbf{x}_{12}$ as an anomaly. To explain why, we first inspect its reference group $R(\mathbf{x}_{12}, 3) = \{\mathbf{x}_{10}, \mathbf{x}_{11}, \mathbf{x}_{12}\}$, which contains the three objects most similar to $\mathbf{x}_{12}$. Second, we report its anomaly score, i.e., 90. Next, we decompose the score and report the behavioral features with the highest deviations, e.g., *Wind*. We can interpret the result as: $\mathbf{x}_{12}$ deviates substantially in *Wind* when compared to $\{\mathbf{x}_{10}, \mathbf{x}_{11}, \mathbf{x}_{12}\}$, which are all similar in terms of *Latitude*, *Longitude* and *Season*.

4.4 Preliminaries

We first define the conditional mean and conditional quantiles, which we use to motivate the use of conditional quantiles for anomaly detection. Next, we detail quantile regression forests, a model class that can robustly estimate conditional quantiles. Readers familiar with these concepts can skip this section.

4.4.1 From Conditional Mean to Conditional Quantiles

Given a real-valued variable V and a set of variables $\mathbf{U}$, regression analysis aims to model the distribution of V conditioned on $\mathbf{U}$. Specifically, it takes V as the response variable and $\mathbf{U}$ as the predictor variables to construct a model that can be used to predict V based on $\mathbf{U}$. Standard regression uses training data $\{(\mathbf{U}_1, V_1), ..., (\mathbf{U}_n, V_n)\}$ to learn a model that estimates the conditional mean $E(V \mid \mathbf{U} = \mathbf{u})$ and uses that as prediction for V when $\mathbf{U} = \mathbf{u}$ is given. For example, Least Squares Regression fits a model $\hat{\theta}$ by minimizing the expected squared error loss, namely $\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\, E\{(V - \hat{V}(\theta))^2) \mid \mathbf{U} = \mathbf{u}\}$.

The conditional mean, however, is only a single statistic of the conditional distribution and is thereby limited in what it can capture. For example, if the conditional

distribution is a multi-modal distribution, the mean is insufficient to describe it (regardless of whether we also consider the standard deviation).

To allow for more comprehensive descriptions of conditional distributions, [161] proposed to estimate *conditional quantiles*. As usual, quantiles are splitting points that divide the range of the probability distribution into consecutive intervals having equal probability. Given a continuous variable V, its conditional α-quantile given $\mathbf{U} = \mathbf{u}$ is defined by $Q_\alpha(\mathbf{u}) = \inf\{v : F(v \mid \mathbf{U} = \mathbf{u}) \geq \alpha\}$, where $F(v \mid \mathbf{U} = \mathbf{u}) = P(V \leq v \mid \mathbf{U} = \mathbf{u})$ is the cumulative distribution function. Conditional quantiles have the potential to describe the full conditional distribution of response variable V.

Existing regression-based anomaly detection methods typically estimate the conditional mean of V by its expected value $\hat{v}$, and then use the Euclidean distance between its actual value and the expected value, i.e., $dist(v, \hat{v})$, as anomaly score. It is hard to interpret this distance though, as the shape and range of the conditional distribution are unknown. In this paper we address this by estimating conditional quantiles instead. In particular, we will show that we can use conditional quantiles to approximate the probability of observing a certain data point in its context, which is then used for the anomaly score.

4.4.2 Quantile Regression Forests

Tree-based regression approaches—such as CART, M5, and Random Forests—are often used to learn both linear and non-linear dependencies. [134] extended the Random Forest to Quantile Regression Forest, which estimates and predicts conditional quantiles instead of means.

Specifically, a quantile regression forest (QRF) is constructed by building an ensemble of K independent decision trees to estimate the full conditional cumulative distribution function of V given $\mathbf{U} = \mathbf{u}$, based on n independent observations $\{(\mathbf{U}_1, V_1), ..., (\mathbf{U}_i, V_i), ..., (\mathbf{U}_n, V_n)\}$. Each $\mathbf{U}_i$ consists of d dimensions. The estimated full conditional distribution can be written as

$$\hat{F}(v|\mathbf{U} = \mathbf{u}) = \hat{P}(V \leq v|\mathbf{U} = \mathbf{u}) = \hat{E}(\mathbb{I}(V \leq v)|\mathbf{U} = \mathbf{u}) = \sum_{i=1}^{n} \omega_i(\mathbf{u})\mathbb{I}(V_i \leq v), \quad (4.1)$$

where $\omega_i(\mathbf{u})$ denotes the weight assigned to observation $(\mathbf{U}_i, V_i)$.

The decision trees that make up a quantile regression forest are constructed similarly to how a random forest is learned, i.e., for each individual tree $m \leq n$ data points are sampled (with replacement), $d' \ll d$ features are randomly selected, and a

criterion such as information gain is used to recursively split the data and tree. Each leaf node keeps all its observations though. K decision trees, namely $T_1(\theta), ..., T_K(\theta)$, are independently grown to form a forest.

Once the forest has been constructed, for a given $\mathbf{U} = \mathbf{u}$ each decision tree $T_j(\theta)$ is traversed to find the leaf node that $\mathbf{u}$ resides in. A weight $\omega_i(\mathbf{u}, T_j(\theta))$ is then computed for each observation $\mathbf{U}_i$, with $i \in \{1, ..., n\}$: if observation $\mathbf{U}_i$ and $\mathbf{u}$ reside in the same leaf node, then $\omega_i(\mathbf{u}, T_j(\theta))$ is defined as 1 divided by the number of samples residing in the leaf node. Otherwise, $\omega_i(\mathbf{u}, T_j(\theta))$ is 0. Next, it takes the average of $\omega_i(\mathbf{u}, T_j(\theta))$ over all decision trees, i.e.,

$$\omega_i(\mathbf{u}) = \frac{1}{K} \sum_{j=1}^{K} \omega_i(\mathbf{u}, T_j(\theta)), \tag{4.2}$$

which is the weight assigned to an observation $\mathbf{U}_i$. Finally, conditional quantile $Q_\alpha(\mathbf{u}) = \inf\{v : F(v \mid \mathbf{U} = \mathbf{u}) \geq \alpha\}$ can be estimated by $\hat{Q}_\alpha(\mathbf{u}) = \inf\{v : \hat{F}(v \mid \mathbf{U} = \mathbf{u}) \geq \alpha\}$. Under reasonable assumptions, [134] proved quantile regression forests to be consistent, i.e.,

$$|\hat{F}(v \mid \mathbf{U} = \mathbf{u}) - F(v \mid \mathbf{U} = \mathbf{u})| \xrightarrow{p} 0, \text{with } n \longrightarrow \infty \tag{4.3}$$

holds pointwise for every $\mathbf{u}$.

4.5 Quantile-based Contextual Anomaly Detection and Explanation

We present an instance of the generic approach for contextual anomaly detection presented in Chapter 4.3.2 that is based on quantile regression forests.

The main idea of our method is to estimate the deviation of an object ' s behavioral values within a given context using uncertainty quantification around predictions, where the predictions are assumed to capture 'normal' behavior. Then, a higher uncertainty implies a higher deviation within the context and thus a higher degree of anomalousness. To this end several approaches could be explored. For example, one might use multi-target regression models—such as Multivariate Random Forests [162]—on all behavioral features, or single-target regression models—such as Random Forests [163]—on individual behavioral features followed by aggregation and conformal inference [164]. In this paper we instead opt to use Quantile Regression Forests [134],

a single-target regression model, because it is (relatively) simple, offering advantages with regard to interpretability, and directly provides uncertainty quantifications. More concretely, we derive intervals for the behavioral features from the underlying quantile regression forests as inherent uncertainty quantifications around predictions, resulting in statistically sound and interpretable measures of degree of anomaly.

In the first phase, we generate reference groups using a distance matrix computed on the *contextual space* of all data points. Specifically, we use Gower's distance [165] (see Chapter 4.6.1 for more detail), to be able to deal with both quantitative and categorical features, and select the k objects having the smallest distances to an object $\mathbf{x}$ as its reference group $R(\mathbf{x}, k)$.

Next, in the second phase, an anomaly score is computed for each individual data point, based on the values in the *behavioral space* of the data point and its reference group. The algorithm is dubbed QCAD—for Quantile-based Contextual Anomaly Detection—and forms the core of our approach; it is introduced in Chapter 4.5.1. Finally, Chapter 4.5.2 describes how the found anomalies are explained by decomposing the anomaly score in the third phase.

Algorithm 3 Quantile-based Contextual Anomaly Detection (QCAD)

Input: Dataset $\mathbf{X}$; contextual feature set $\mathbf{C}$; behavioral feature set $\mathbf{B}$; number of behavioral features Q; number of nearest neighbors k; number of conditional quantiles to estimate n_q; number of trees n_t; maximum number of features used in a tree n_f; minimum number of samples to split a node n_s.
Output: Anomaly score list $\mathbf{S}$

procedure QCAD($\mathbf{X}, \mathbf{C}, \mathbf{B}, k, n_q, n_t, n_f, n_s$)
  $\mathbf{S} \Leftarrow \{\}$
  for $\mathbf{x} \in \mathbf{X}$ **do**
    $R(\mathbf{x}, k) \Leftarrow GetReferenceGroup(\mathbf{X}, \mathbf{C}, \mathbf{x}, k)$
    for $q \in \{1, 2, ..., Q\}$ **do**
      $QRF \Leftarrow LearnQRF(R(\mathbf{x}, k), \mathbf{C}, \mathbf{b}^q, n_q, n_t, n_f, n_s)$
      $s(\mathbf{x}|\mathbf{b}^q) \Leftarrow AnomalyScore(QRF, \mathbf{C}, \mathbf{b}^q, \mathbf{x})$
    end for
    $s(\mathbf{x}) = \frac{1}{Q} \sum_{q=1}^{Q} s(\mathbf{x}|\mathbf{b}^q)$
    $\mathbf{S}.append((\mathbf{x}, R(\mathbf{x}, k), s(\mathbf{x})))$
  end for
  return S
end procedure

4.5.1 Detecting Anomalies with Quantile Regression Forests

Algorithm 3 outlines the QCAD algorithm, which takes a dataset and a number of hyperparameters as input and outputs a list of all data points together with their reference groups and computed anomaly scores. Specifically, we assume that the contextual features can be mixed but all behavioral features are numerical. We will first describe the overall algorithm, and then go into the specifics of the score computation.

Algorithm After initializing the empty score list (Line 2), the algorithm iterates over all data points in dataset $\mathbf{X}$ (Ln 3–11). For each object $\mathbf{x} = (c^1, ..., c^p, ..., c^P, b^1, ..., b^q, ...b^Q) = (\mathbf{c}, \mathbf{b})$ we first obtain the reference group that was computed in the first phase (Ln 4). We then iterate over all features in behavioral feature set $\mathbf{X}$ in order to compute a partial anomaly score for each behavioral feature (Ln 5–8). These partial anomaly scores are summed to obtain the anomaly score for $\mathbf{x}$ (Ln 9), i.e., we assume the behavioral features to all have equal potential to contribute to the overall anomaly score. After this, the data point, its reference group, and its anomaly score are appended to the anomaly score list (Ln 10). Finally, the anomaly score list is returned as output (Ln 12).

Within the inner `for` loop, we first learn a quantile regression forest using behavioral feature $\mathbf{b}^q$ as response variable and the data point's reference group $R(\mathbf{x}, k)$ as training data (Ln 6). All contextual features $\mathbf{C}$ are used as predictor variables for every constructed QRF. We then use the learned quantile regression forest, the features, and the data point to compute the raw partial anomaly score (Ln 7), which we will motivate and explain in detail next.

QRF-based anomaly score As argued in the Introduction and Chapter 4.4.1, taking the distance between a data point and a conditional mean as basis for a contextual anomaly score may be too limiting: this only works if all 'normal' data points reside close to the mean. Instead, we aim to—conceptually—consider the entire conditional probability density function and use the local density of a given data point as a proxy for anomalousness: the lower the density, the higher the anomaly score. Directly accurately estimating the density function is hard though, especially in areas of low density, which are of particular importance to us.

This is where the quantile regression forests come in: given sufficient training data, they accurately learn the conditional cumulative distribution function, which can be queried in inverse form, i.e., through conditional quantiles. When querying a quantile regression forest, this can be done at different granularities. For example, Figure 4.3(a)

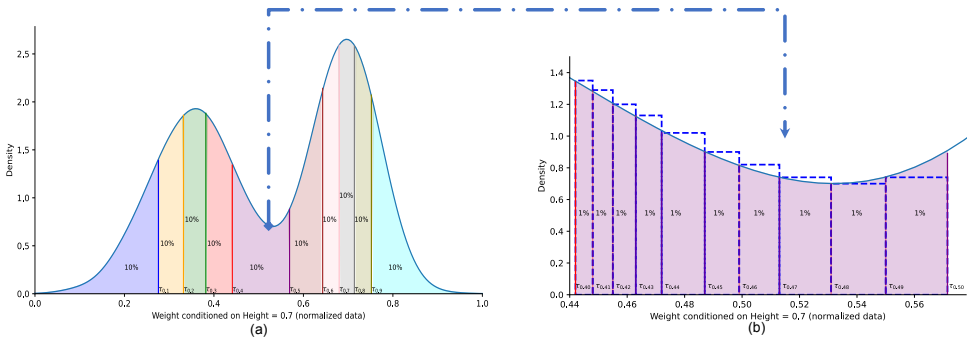


Figure 4.3: (a) Estimated conditional quantiles $\{Q_{0.1}, Q_{0.2}, \ldots, Q_{0.9}\}$ for a behavioral feature conditioned on contextual features. (b) Zooming in on the area between $Q_{0.40}$ and $Q_{0.50}$, we see that percentiles are more likely to be sufficiently detailed than the quantiles in (a). A high-resolution online version improves readability of small fonts.

shows that conditional quantiles $\{Q_{0.1}, Q_{0.2}, \ldots, Q_{0.9}\}$ may very well be insufficient to accurately describe a conditional distribution as they overly smooth the underlying distribution and thus fail to capture the nuances accurately, while Figure 4.3(b) show that conditional *percentiles*, i.e., $\{Q_{0.01}, Q_{0.02}, \ldots, Q_{0.99}\}$, are much more likely to provide sufficient detail.

We use conditional percentiles for our anomaly score, because they provide a high level of granularity while not requiring very large amounts of data to be estimated accurately. As additional benefit, the difference between each two consecutive percentiles is always assessed by a weighted combination of a comparable number of training data points, i.e., at the cost of some smoothing we do not suffer from extremely poor local density estimates in low density areas.

To formally develop our anomaly score, we first define τ_i to be the ith percentile, i.e., $\tau_i = Q_{i/100}, \forall i \in [0, 100]$. For any *percentile interval*, i.e., an interval $[\tau_i, \tau_{i+1}]$ defined by two consecutive percentiles i and $i+1$, we have by definition that it spans exactly 0.01 probability (see Figure 4.3(b)). We could estimate the local density of a percentile interval and use that for our anomaly score, but we aim for a score that becomes larger when a data point is deemed to be more anomalousness.

To this end we define *percentile interval width* w_i as the difference between two consecutive percentiles i and $i+1$, i.e., $\tau_{i+1} - \tau_i$. As such, interval width can be regarded as 'inverse density', meaning that width will increase as the local density decreases. For example, in Figure 4.3(b) we have $\tau_{46} = 0.5$ and $\tau_{47} = 0.513$, which gives $w_{46} = 0.013$. Data points that fall in percentile intervals having relatively large

widths are more likely to be contextual anomalies, as they reside in low-density areas of the conditional distribution.

The basic idea is thus to define the anomaly score for an object $\mathbf{x}$ and behavioral feature $\mathbf{b}^q$ as $w(\mathbf{x}|\mathbf{b}^q)$, i.e., the width of the (QRF-predicted) percentile interval in which the behavioral value of the data point falls. We need to consider a special case though: the actual behavioral feature value b^q may be less than the smallest estimated conditional quantile, i.e., τ_0^q, or greater than the largest estimated conditional quantile, i.e., τ_{100}^q. That is, the actual value may not fall in any estimated conditional percentile interval. To address this we extrapolate beyond τ_0^q and τ_{100}^q, leading to intermediate anomaly score

$$is(\mathbf{x}|\mathbf{b}^q) = \begin{cases} \left(1 + \dfrac{\tau_0^q - b^q}{\tau_{75}^q - \tau_{25}^q}\right) \max(w(\mathbf{x}|\mathbf{b}^q)), \text{ if } b^q < \tau_0^q; \\ \left(1 + \dfrac{b^q - \tau_{100}^q}{\tau_{75}^q - \tau_{25}^q}\right) \max(w(\mathbf{x}|\mathbf{b}^q)), \text{ if } b^q > \tau_{100}^q, \\ w(\mathbf{x}|\mathbf{b}^q), \text{ otherwise,} \end{cases} \tag{4.4}$$

where $\max(w(\mathbf{x}|\mathbf{b}^q))$ represents the maximum interval width of all conditional percentile intervals for the behavioral feature $\mathbf{b}^q$.

Unfortunately, directly using Equation (4.4) as a partial anomaly score would make our approach prone to the so-called **dictator effect**: summing such partial scores for a data point that *strongly* deviates in only a *few* behavioral features would lead to a larger anomaly score than anomalous data points deviating moderately in *many* behavioral features. As a result, data points with few, strong deviations would 'dictate' highest scores; this is undesirable.

To avoid the dictator effect, we truncate the partial anomaly scores to a predefined maximum and arrive at the final partial anomaly score as

$$s(\mathbf{x}|\mathbf{b}^q) = \begin{cases} \dfrac{\eta}{100}, \text{ if } is(\mathbf{x}|\mathbf{b}^q) > \dfrac{\eta}{100}; \\ is(\mathbf{x}|\mathbf{b}^q), \text{ otherwise,} \end{cases} \tag{4.5}$$

where η is a hyperparameter. The rationale for η is as follows: if the conditional distribution is uniformly distributed and the behavioral feature has range $[0, 1]$, then the expected width of any percentile interval width is 0.01 and η can be interpreted as the maximum number of *expected percentile interval widths*. In the experiments we use $\eta = 10$ because of its strong empirical performance (also shown in the ablation study in Appendix 4.7).

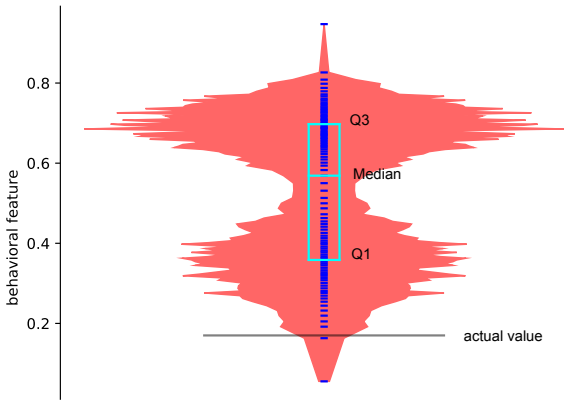


Figure 4.4: Anomaly beanplot giving insight in what the quantile regression forest learned for a particular data point and behavioral feature, and why the data point is (not) considered an anomaly. The short blue lines indicate the conditional percentiles $\tau_0, \tau_1, \ldots, \tau_{100}$ as learned by the QRF (top to bottom), for the given data point and behavioral feature. As in a box plot, the (cyan) box indicates first quartile (τ_{25}), median (τ_{50}), and third quartile (τ_{75}). The wider red area represents probability densities as estimated based on the conditional percentiles. Finally, the black line indicates the actual value that the data point has.

By scaling the behavioral features to $[0, 1]$ before anomaly detection, e.g., using min-max normalization, the estimated conditional percentiles should also lie in this interval and the interval widths obtained for the individual behavioral features should thus be comparable. In turn, this implies they can be summed to obtain the final anomaly score for a data point (Algorithm 3, Line 9).

4.5.2 Explaining Anomalies with Anomaly Beanplots

In the third phase, we provide explanations for the reported anomalies. From the definition of our anomaly score it is clear that it is *attributable*: the overall score can be decomposed into partial scores for individual behavioral features.

For each identified anomaly $\mathbf{a}$, we report its contextual neighbors $R(\mathbf{a})$ and the final anomaly score as computed on Line 9 of Algorithm 3. Further, we report the partial anomaly scores corresponding to the behavioral features, i.e., we report $s(\mathbf{x}|\mathbf{b}^q)$ for all q, ranked from highest to lowest to indicate in which behavioral features the anomaly deviates most.

Since visualization often helps to quickly provide valuable insight, we propose the *anomaly beanplot*, a variant of the beanplot by [166]. Figure 4.4 shows an example, visually depicting the learned conditional percentiles, their interval widths, and the probability densities that can be estimated from those, all for a particular data point and behavioral feature. (Remember that a quantile regression forest is learned on the reference group of a data point, hence the anomaly beanplot for each data point may be different.)

By including the actual value of the data point in the beanplot (as a horizontal black line), the analyst can easily see how its behavioral feature value is positioned relative to those of its reference group, and why and to what extent it contributes to the its anomalousness.

4.6 Experiments

To demonstrate the effectiveness of our overall approach and proposed method QCAD, we conduct experiments on a wide range of synthetic and real-world datasets. We will first explain the choices regarding datasets, baseline algorithms, and evaluation criteria, after which we present both quantitative results and a case study that investigates interpretability and practical utility.

In addition, we demonstrate the robustness of QCAD with regard to the 'number of nearest neighbors' hyperparameter by means of a sensitivity analysis, and conduct several ablation studies to investigate the impact of the hyperparameters. For reasons of space and brevity, the sensitivity analysis and ablation studies are given in Appendices 4.7 and 4.7.

4.6.1 Data

It is challenging to evaluate unsupervised anomaly detection algorithms due to the lack of commonly agreed-upon benchmark data, and down-sampling classification datasets has been criticized for its variation in the nature of the resulting outliers [167, 168].

When evaluating unsupervised *contextual* anomaly detection algorithms, this problem is further compounded by the requirement to have both contextual and behavioral features, and—more importantly—treating those differently [141]. Consequently, a generally accepted approach is to artificially inject contextual anomalies into existing datasets using a perturbation scheme.

Data Preprocessing

To make the datasets suitable to all anomaly detection methods, we need to preprocess them before injecting contextual anomalies. First, we leverage Label Encoding [169] to transform categorical contextual features to numerical form. Second, we employ Min-Max normalization to scale all behavioral features to $[0, 1]$. Min-Max normalization is routinely used in many anomaly detection and generally improves performance [170].

Gower's Distance To be able to calculate the similarity between two data points containing both categorical and numerical features, we can utilize Gower's distance [165]. Specifically, the Gower's distance between data points $\mathbf{c}_i = (c_i^1, ..., c_i^p, ...c_i^P)$ and $\mathbf{c}_j = (c_j^1, ..., c_j^p, ...c_j^P)$ is defined as $1 - \frac{1}{P}\sum_{p=1}^{P} ps_{ij}^p$, where ps_{ij}^p represents the partial similarity between data instances $\mathbf{c}_i$ and $\mathbf{c}_j$ in the p-th dimension. For a numerical feature, $ps_{ij}^p = 1 - \frac{|c_i^p - c_j^p|}{\max(\mathbf{c}^p) - \min(\mathbf{c}^p)}$, with $\max(\mathbf{c}^p)$ and $\min(\mathbf{c}^p)$ denoting the maximum and minimum value of all data points for the p-th feature, respectively. For a categorical feature, $ps_{ij}^p = \mathbb{I}(c_i^p - c_j^p)$, where $\mathbb{I}(\cdot)$ represents the indicator function. Consequently, Gower's distance between two data points is always in $[0, 1]$, with a lower value indicating a larger similarity.

Perturbation Scheme for Outlier Injection

[140] proposed a perturbation scheme to inject contextual anomalies in datasets without ground-truth anomalies, which has become a de-facto standard for the evaluation of contextual anomaly detection [140, 141, 149, 171].

This perturbation scheme, however, has also been criticized for two reasons [140, 172]. First, the objects obtained by simply swapping the feature values are still likely to be contextually normal. Second, some very common types of anomalies cannot be yielded through this perturbation scheme. For example, one cannot obtain extreme values by swapping features values in a clean dataset, whereas most anomaly detection methods assume the training dataset is uncontaminated. To avoid these problems, [172] introduced another perturbation scheme to inject anomalies. We develop a new perturbation scheme by refining this scheme, as follows.

Given a dataset $\mathbf{X}$ containing N data instances with contextual feature set $\mathbf{C} = \{\mathbf{c}^1, ..., \mathbf{c}^P\}$ and behavioral feature set $\mathbf{B} = \{\mathbf{b}^1, ..., \mathbf{b}^Q\}$, we first use Min-Max normalization to scale the behavioral features of all objects to $[0, 1]$ (keeping the contextual features intact), resulting in a new dataset $\tilde{\mathbf{X}}$. Second, to inject m anomalies into $\tilde{\mathbf{X}}$, with $0 < m \ll N$, we select m objects $\mathbf{x}_1, \ldots, \mathbf{x}_m$ uniformly at random from $\tilde{\mathbf{X}}$. For each $\mathbf{x} = (\mathbf{c}, \mathbf{b})$ in $\tilde{\mathbf{X}}$, $\mathbf{c}$ represents the contextual feature values and $\mathbf{b}$

denotes the behavioral feature values. Third, for a selected object $\mathbf{x}_i = (\mathbf{c}_i, \mathbf{b}_i) = (c_i^1, ..., c_i^p, ..., c_i^P, b_i^1, ..., b_i^q, ..., b_i^Q)$ and behavioral feature $\mathbf{b}^q$, we sample a number uniformly at random from $[-0.5, -0.1] \bigcup [0.1, 0.5]$, and then add this random number to the behavioral feature value of $\mathbf{x}_i$, namely b_i^q, resulting in $\hat{b}_i^q$. In the same manner, we repeat this process for each behavioral feature, resulting in $(\hat{b}_i^1, ..., \hat{b}_i^Q)$, or $\hat{\mathbf{b}}_i$. Accordingly, we generate a new object $\tilde{\mathbf{x}} = (\mathbf{c}, \hat{\mathbf{b}})$ as contextual anomaly. Fourth, we repeat the third step for each selected object, leading to m perturbed objects. Fifth and final, we replace the selected objects in the original dataset with their corresponding perturbed objects. To allow extreme values to be injected, we deliberately do not truncate the values outside $[0, 1]$ after adding a random number in each behavioral feature.

Our perturbation scheme has the following advantages. When compared to the swapping perturbation scheme proposed by [140], the objects obtained by our perturbation scheme are very unlikely to remain contextually normal. In addition, our perturbation scheme can—but does not always—lead to extreme values. Note that we do not strictly follow the perturbation scheme proposed by [172] because their method only adds a non-negative number to the behavioral features. On the one hand, sometimes this non-negative number is zero, leading to injecting a normal object as 'anomaly'. On the other hand, sometimes this non-negative number is huge, which makes the injected object (too) easy to detect. Our perturbation scheme avoids these problems by firstly normalizing the behavioral feature values, and then setting more reasonable lower and upper bounds for the perturbation.

Datasets

To evaluate and compare our method on a diverse range of datasets, we generate 10 synthetic datasets and select 20 real-world datasets; their properties are summarized in Tables 4.2 and 4.3 respectively.

We first discuss the synthetic data. To be able to produce data with various forms and degrees of dependencies between behavioral and contextual features, we propose the following generation schemes. For $q \in \{1, ..., Q\}$, we have

(S1) $\mathbf{b}^q = \sum_{p=1}^{Q} (\alpha_{qp} \cdot \mathbf{c}^p) + \boldsymbol{\epsilon}$;

(S2) $\mathbf{b}^q = \sum_{p=1}^{Q} \left(\alpha_{qp} \cdot (\mathbf{c}^p)^3\right) + \boldsymbol{\epsilon}$,

(S3) $\mathbf{b}^q = \sum_{p=1}^{Q} (\alpha_{qp} \cdot \sin(\mathbf{c}^p)) + \boldsymbol{\epsilon}$,

(S4) $\mathbf{b}^q = \sum_{p=1}^{Q} (\alpha_{qp} \cdot \log(\mathbf{1} + |\mathbf{c}^p|)) + \boldsymbol{\epsilon}$,

Table 4.2: Summary of synthetic datasets. $\#Num, \#Cat, \#\mathbf{C}$ and $\#\mathbf{B}$ represent the number of numerical features, the number of categorical/nominal features, the number of contextual features, and the number of behavioral features, respectively. All behavioral features are numerical, the contextual features can be mixed.

Dataset	Scheme	#Num	#Cat	#**C**	#**B**
Syn1	S1	8	2	5	5
Syn2	S2	8	2	5	5
Syn3	S3	8	2	5	5
Syn4	S4	8	2	5	5
Syn5	S5	8	2	5	5
Syn6	S1	19	6	20	5
Syn7	S2	19	6	20	5
Syn8	S3	19	6	20	5
Syn9	S4	19	6	20	5
Syn10	S5	19	6	20	5

(S5) $\mathbf{b}^q = \sum_{p=1}^{Q} \left(\alpha_{qp} \cdot \mathbf{c}^p + \beta_{qp} \cdot (\mathbf{c}^p)^3 + \gamma_{qp} \cdot \sin(\mathbf{c}^p) + \delta_{qp} \cdot \log(\mathbf{1} + |\mathbf{c}^p|)\right) + \boldsymbol{\epsilon}$,

where $\alpha_{qp}, \beta_{qp}, \gamma_{qp}, \delta_{qp} \overset{i.i.d.}{\sim} \mathcal{U}(0,1)$ and are replaced by zero with a probability of 1/3. Further, $\boldsymbol{\epsilon} = (\epsilon_1, ..., \epsilon_n, ..., \epsilon_N)^T$, with $\epsilon_n \overset{i.i.d.}{\sim} \mathcal{U}(0, 0.05)$.

In addition, for $p \in \{1, ..., P\}$, $\mathbf{c}^p$ is generated from a Gaussian mixture model with five clusters. If it is numerical, each of the Gaussian centroids is sampled uniformly at random from $[0, 1]$ and the diagonal element of the covariance matrix is fixed at 1/4 of the average pairwise distance between the centroids in each behavioral feature. Otherwise, the centroids are sampled uniformly at random from $\{0, 1, ..., 10\}$ with the same covariance setting. Moreover, every generated number is rounded to an integer to ensure that it is categorical. On this basis, we generate a wide collection of synthetic datasets by varying the generation scheme and the number of contextual features and behavioral features. Sample size is always set to 2000, and the rate of injected anomalies is fixed at 2.5%. See Table 4.2 for an overview.

Next, we employ the above-mentioned perturbation scheme to inject contextual anomalies into 20 real-world datasets. We use these datasets because they are representative of the potential application areas of our QCAD framework. That is, they stem from healthcare & life sciences (e.g., Bodyfat, Heart Failure, Indian River Patient, Hepatitis, Parkinson Telemonitoring, Abalone, Fish Weight, QSAR Fish Toxi-

Table 4.3: Summary of real-world datasets. $\#Num, \#Cat, \#\mathbf{C}$ and $\#\mathbf{B}$ represent the number of numerical features, the number of categorical/nominal features, the number of contextual features, and the number of behavioral features, respectively. All behavioral features are numerical, the contextual features can be mixed.

Dataset	#Num	#Cat	#Samples	#Anomalies (Ratio)	#**C**	#**B**
Abalone	8	1	4177	100 (2.4%)	4	5
AirFoil	6	0	1503	70 (4.7%)	5	1
BodyFat	15	0	252	20 (7.9%)	13	2
Boston	12	2	506	40 (7.9%)	13	1
Concrete	9	0	1030	50 (4.8%)	8	1
ElNino	3	8	20000	200 (1%)	6	5
Energy	10	0	768	50 (6.5%)	8	2
FishWeight	6	1	157	15 (9.5%)	6	1
ForestFires	11	2	517	50 (9.7%)	4	9
GasEmission	11	0	7384	100 (1.4%)	8	3
HeartFailure	7	5	299	30 (10%)	6	6
Hepatitis	11	2	615	30 (4.9%)	3	10
LiverPatient	9	2	579	30 (5.2%)	3	8
Maintenance	18	0	11934	100 (0.8%)	15	3
Parkinson	21	1	5875	100 (1.7%)	20	2
PowerPlant	5	0	9568	100 (1%)	4	1
QSRanking	12	1	475	40 (8.4%)	7	6
SynMachine	5	0	557	50 (8.9%)	4	1
Toxicity	7	0	908	50 (5.5%)	6	1
Yacht	7	0	308	30 (9.7%)	6	1

city), social sciences (e.g., Boston House Price, University Ranking), environmental-protection area (e.g., El Nino, Forest Fires), and engineering (e.g., Gas Turbine CO and NOx Emission, Yacht Hydrodynamics, Condition Based Maintenance of Naval Propulsion Plants, Synchronous Machine, Airfoil Self-Noise, Concrete Compressive Strength, Combined Cycle Power Plant). A summary is given in Table 4.3; each dataset is described in more detail in Appendix 4.7.

4.6.2 Baseline Algorithms and Implementations

We empirically compare our method to state-of-the-art algorithms, including traditional anomaly detection methods (distance-based, density-based, dependency-based, etc.) and contextual anomaly detection methods. For a fair comparison, we select the following anomaly detectors, which all return an anomaly score—rather than a binary outcome—for each data instance.

- Local Prediction Approach to Anomaly Detection (LoPAD) by [156], which is the state-of-the-art dependency-based traditional anomaly detector;
- Conditional Outlier Detection (CAD) by [140], which was the first anomaly detector dedicated to identify contextual anomalies;
- Robust Contextual Outlier Detection (ROCOD) by [141], which is the state-of-the-art contextual anomaly detector;
- Isolation Forest (IForest) by [77], which is one of the state-of-the-art isolation-based traditional anomaly detectors;
- Local Outlier Factor (LOF) by [173], which is one of the state-of-the-art density-based traditional anomaly detectors;
- k-NN anomaly detector by [174], which is one of the state-of-the-art distance-based traditional anomaly detectors;
- Anomaly detector using axis-parallel subspaces (SOD) by [157], which is one of the state-of-the-art subspace-based traditional anomaly detectors; and
- Histogram-Based Outlier Score (HBOS) by [85], which is one of the state-of-the-art histogram-based traditional anomaly detectors.

We implemented and ran all algorithms in Python 3.8 on a computer with Apple M1 chip 8-core CPU and 8GB unified memory. For classical algorithms such as IForest,

LOF, k-NN, SOD, and HBOS, we use their publicly available implementations in PyOD [107] with their default settings. Unfortunately, for LoPAD, CAD and ROCOD no implementation was publicly available. For LoPAD, we first use the 'bnlearn' package [175] in R to find the Markov Blankets of each dataset based on Fast-IAMB [176]. Next, we implement the LoPAD algorithm in Python using CART [177] as the prediction model with bagging size 200. For CAD, we implement the CAD-GMM-Full algorithm as recommended by [140], with default settings except for parameter 'number of Gaussian component'; if this parameter would be set to its default 30, it would take more than one month to finish all the experiments on our computer. In addition, preliminary experiments show that the difference in results when this parameter is set to 30 and 5 respectively is negligible for most datasets. We therefore set this parameter to 5 in all experiments. We implemented ROCOD in Python with its default settings. One important parameter, namely the distance threshold used to find neighbors, is not discussed in [141] though. For different datasets and distance metrics, it is hard to obtain a single best value for this parameter and preliminary experiments revealed that ROCOD is sensitive to this parameter. Nevertheless, we also observed that it often achieves relatively good results when the distance threshold used to find neighbors is set to 0.9; hence, we decided to set this parameter to 0.9 by default.

QCAD Parameters Setting

As summarized in Table 4.4, the QCAD algorithm also requires several parameters to be set. First, in our contextual anomaly framework, we need to set the number of nearest neighbors (k) used to generate reference group. Second, when creating quantile regression forests, we need to specify the following parameters: the number of trees, the number of maximal features used to construct a tree, and the number of minimal sample size to split in a node of tree. Third, we need to specify the number of conditional quantiles (l) to estimate for an object in each behavioral feature. Last, we also need to set the number of features (h) used to generate explanations. We set these parameters as follows.

- The number of nearest neighbors (k): the sensitivity analysis (see appendix) indicates that our approach is robust with respect to this parameter as long as its value is not overly small. By default, we set this parameter to $\min(N/2, 500)$, where N is the sample size.
- The number of conditional quantiles to estimate (n_q): theoretically, an increase

Table 4.4: Summary of parameters involved in QCAD. Particularly, **Value** represents the values that we recommend to use in experiments.

Symbol	Meaning	Value
k	number of nearest neighbors	$\min(N/2, 500)$
n_q	number of conditional quantiles to estimate	100
n_t	number of trees used to construct a QRF	100 or 10
n_f	number of maximal features used to construct a tree	$\lvert\mathbf{C}\rvert$
n_s	minimum number of samples to split a node	10
h	number of features used to generate explanations	$\min(\lvert\mathbf{B}\rvert, 3)$

in this number will result in better performance in terms of accuracy, at the expense of a larger running time. However, preliminary experiments show that increasing this number beyond 100 will only produce slightly better results. Therefore, we set it to 100 by default.

- The number of trees used to construct a quantile regression forest (n_t): in theory, a larger number of trees will produce better performance in terms of accuracy, but at the cost of a larger running time. We empirically found that 100 trees usually gives good results and further increasing the number of trees leads to negligible improvement. Due to time constraints, we set this number to 10 in all experiments in this paper.

- The number of maximal features used to construct a tree in a quantile regression forest (n_f): [134] demonstrated the stability of quantile regression forest on this parameter, and set this parameter to 1/3 of the number of variables in their experiments. However, in our experiments, sometimes the number of contextual features is less than 3. To render quantile regression forests applicable on various datasets, we set this parameter to the number of all variables by default.

- The minimal sample size for the node of a tree to be split (n_s): as indicated in [134], different values of this parameter do not seem to have much effect on the results, and our preliminary experiments are also in line with this statement. Therefore, we set this number to 10 by default, as also used in [134].

- The number of features used to generate explanations (h): This parameter is set to $\min(|\mathbf{B}|, 3)$ by default. However, the end-users can set this parameter according to their preferences as long as its value is between 0 and $|\mathbf{B}|$, where $|\mathbf{B}|$ represents the number of behavioral features.

For reproducibility, we make all code and datasets publicly available[2].

4.6.3 Evaluation Criteria

PRC AUC [141, 172], ROC AUC [178, 179, 180], and Precision@n [181] are widely used for the evaluation and comparison of anomaly detection methods that generate a full list of anomaly scores for all observations. They are defined as follows:

- Receiver Operating Characteristic (ROC), which is obtained by plotting the true positive rate (y-axis) versus the false positive rate (x-axis) at various threshold settings. The area under this curve, namely ROC AUC, is a threshold-agnostic performance measure widely used in anomaly detection;
- Precision-Recall Curve (PRC), which is created by plotting precision (y-axis) against recall (x-axis) at various threshold settings. The area under this curve, namely PRC AUC, is another widely-used, threshold-agnostic performance measure, and is also called Average Precision;
- Precision at n, or P@n, is defined as the precision of the observations ranked among the top-n, where $n \in \{1, 2, ..., N\}$. In our experiments, we set n to the number of injected contextual anomalies.

For completeness: precision is defined as $\frac{\#\{\text{Real anomalies}\}\cap\{\text{Reported anomalies}\}}{\#\{\text{Reported anomalies}\}}$, while recall is defined as $\frac{\#\{\text{Real anomalies}\}\cap\{\text{Reported anomalies}\}}{\#\{\text{Real anomalies}\}}$. We perform ten independent trials of injecting contextual anomalies on each dataset and report the means and standard deviations of each of the three evaluation criteria.

4.6.4 Anomaly Detection Performance

Results on 10 synthetic datasets and 20 real-world datasets are presented in Tables 4.5, 4.6, and 4.7, where the first table concerns synthetic data, and the latter two tables concern real-world data.

From Table 4.5, we observe that QCAD generally dominates other methods in terms of anomaly detection accuracy according to the average ranks. More specifically, QCAD achieved the best results on 9 out of 10 synthetic datasets in terms of PRC AUC and ROC AUC, and on all datasets in terms of Precision@n. On Syn8, QCAD is on par with its best competitor (i.e., CAD) in terms of ROC AUC, whereas

[2]https://github.com/ZhongLIFR/QCAD

Table 4.5: Performance in terms of PRC AUC, ROC AUC, and P@n, on synthetic data, with 10 independent runs of injecting contextual anomalies into each dataset. For each anomaly detector on each dataset, the mean value and standard deviation of each evaluation criterion is presented. The best results obtained on each dataset are highlighted in bold.

		QCAD	LoPAD	ROCOD	CAD	IForest	LOF	k-NN	SOD	HBOS
Syn1	PRC AUC	**1.00**±0.00	0.03±0.00	0.88±0.03	0.94±0.15	0.32±0.08	0.04±0.01	0.05±0.01	0.29±0.03	0.18±0.03
	ROC AUC	**1.00**±0.00	0.49±0.04	0.99±0.00	0.94±0.16	0.95±0.01	0.64±0.05	0.72±0.03	0.97±0.00	0.90±0.02
	P@n	**0.99**±0.01	0.04±0.00	0.79±0.03	0.89±0.31	0.37±0.08	0.05±0.03	0.05±0.02	0.20±0.07	0.22±0.06
Syn2	PRC AUC	**1.00**±0.00	0.03±0.00	0.22±0.02	0.94±0.03	0.15±0.03	0.04±0.01	0.05±0.01	0.29±0.02	0.38±0.08
	ROC AUC	**1.00**±0.00	0.49±0.02	0.95±0.00	**1.00**±0.00	0.93±0.01	0.66±0.03	0.71±0.02	0.97±0.00	0.95±0.01
	P@n	**0.98**±0.02	0.03±0.01	0.13±0.03	0.94±0.02	0.09±0.05	0.04±0.02	0.04±0.02	0.27±0.06	0.36±0.08
Syn3	PRC AUC	**1.00**±0.00	0.02±0.00	0.69±0.04	0.96±0.02	0.31±0.05	0.07±0.01	0.06±0.01	0.34±0.04	0.26±0.05
	ROC AUC	**1.00**±0.00	0.47±0.06	0.97±0.01	0.99±0.01	0.93±0.02	0.77±0.02	0.76±0.02	0.97±0.00	0.92±0.01
	P@n	**0.97**±0.01	0.02±0.02	0.60±0.04	0.94±0.02	0.34±0.06	0.09±0.03	0.06±0.02	0.34±0.06	0.29±0.05
Syn4	PRC AUC	**1.00**±0.00	0.03±0.01	0.90±0.03	0.99±0.01	0.38±0.08	0.05±0.01	0.06±0.01	0.44±0.04	0.32±0.04
	ROC AUC	**1.00**±0.00	0.52±0.02	**1.00**±0.00	**1.00**±0.00	0.95±0.01	0.72±0.03	0.78±0.03	0.99±0.00	0.93±0.01
	P@n	**1.00**±0.00	0.03±0.03	0.82±0.04	0.98±0.02	0.43±0.06	0.05±0.03	0.08±0.03	0.49±0.07	0.37±0.05
Syn5	PRC AUC	**1.00**±0.00	0.02±0.00	0.26±0.03	**1.00**±0.00	0.15±0.02	0.04±0.00	0.05±0.00	0.22±0.01	0.21±0.03
	ROC AUC	**1.00**±0.00	0.52±0.03	0.96±0.00	**1.00**±0.00	0.93±0.01	0.61±0.03	0.70±0.02	0.96±0.00	0.94±0.01
	P@n	0.99±0.01	0.02±0.02	0.21±0.04	**1.00**±0.00	0.11±0.06	0.04±0.02	0.05±0.03	0.12±0.03	0.25±0.05
Syn6	PRC AUC	**0.99**±0.01	0.03±0.01	0.97±0.01	0.91±0.14	0.18±0.04	0.03±0.00	0.03±0.00	0.03±0.00	0.40±0.07
	ROC AUC	**1.00**±0.00	0.49±0.06	**1.00**±0.00	0.94±0.16	0.88±0.02	0.52±0.03	0.51±0.04	0.53±0.05	0.93±0.01
	P@n	**0.95**±0.02	0.03±0.03	0.91±0.03	0.83±0.29	0.26±0.06	0.03±0.03	0.04±0.03	0.02±0.02	0.42±0.06
Syn7	PRC AUC	**1.00**±0.00	0.02±0.00	0.68±0.07	0.98±0.02	0.10±0.03	0.03±0.00	0.03±0.00	0.03±0.01	0.28±0.05
	ROC AUC	**1.00**±0.00	0.48±0.05	0.98±0.01	**1.00**±0.00	0.84±0.03	0.49±0.04	0.50±0.03	0.55±0.03	0.94±0.01
	P@n	**0.99**±0.01	0.02±0.02	0.64±0.05	0.97±0.03	0.12±0.05	0.02±0.02	0.01±0.02	0.02±0.03	0.32±0.04
Syn8	PRC AUC	0.77±0.04	0.02±0.00	0.81±0.04	**0.95**±0.03	0.17±0.06	0.03±0.00	0.03±0.00	0.03±0.01	0.27±0.06
	ROC AUC	0.98±0.01	0.49±0.04	0.98±0.01	**0.99**±0.01	0.84±0.02	0.50±0.03	0.52±0.03	0.54±0.03	0.91±0.02
	P@n	**0.91**±0.04	0.02±0.02	0.76±0.04	0.90±0.03	0.22±0.07	0.02±0.02	0.02±0.02	0.02±0.02	0.32±0.05
Syn9	PRC AUC	**0.98**±0.02	0.02±0.00	0.92±0.03	0.96±0.03	0.21±0.03	0.03±0.00	0.03±0.00	0.03±0.00	0.36±0.04
	ROC AUC	**1.00**±0.00	0.49±0.05	**1.00**±0.00	**1.00**±0.00	0.88±0.02	0.50±0.04	0.51±0.05	0.56±0.04	0.94±0.02
	P@n	**0.93**±0.03	0.01±0.01	0.87±0.05	0.92±0.03	0.27±0.07	0.02±0.02	0.01±0.02	0.02±0.02	0.41±0.05
Syn10	PRC AUC	**1.00**±0.00	0.03±0.01	0.49±0.06	0.88±0.04	0.11±0.02	0.03±0.01	0.03±0.01	0.03±0.01	0.28±0.05
	ROC AUC	**1.00**±0.00	0.50±0.04	0.98±0.01	0.99±0.01	0.87±0.02	0.51±0.03	0.51±0.04	0.54±0.05	0.94±0.01
	P@n	**0.99**±0.01	0.04±0.03	0.45±0.05	0.78±0.05	0.12±0.04	0.02±0.02	0.03±0.02	0.03±0.01	0.32±0.04
Ranking[1]	PRC AUC	1.2	8.6	3.0	1.9	5.1	7.4	7.1	5.6	4.6
	ROC AUC	1.3	9.0	2.5	2.0	5.1	7.7	7.2	4.6	4.7
	P@n	1.1	8.0	3.2	2.0	5.0	7.3	7.3	5.9	4.3

[1] This is the average ranking of each anomaly detector on 10 synthetic datasets in terms of PRC AUC, ROC AUC and P@n, respectively.

QCAD is slightly worse than CAD and ROCOD in terms of PRC AUC. This demonstrates the effectiveness of QCAD in identifying contextual anomalies for different forms and degrees of dependencies between the behavioral features and contextual features. More importantly, the poor performance of LoPAD indicates the importance of distinguishing behavioral features from contextual features. Additionally, other traditional anomaly detectors—including IForest, LOF, k-NN, SOD, and HBOS—perform poorly on most datasets because they treat all features equally.

Another important observation is that QCAD is generally superior to other methods in terms of robustness. That is, QCAD attains high PRC AUC, ROC AUC, and Precision@n values with small standard deviations on Syn1, Syn2, Syn3, Syn4 and Syn5, indicating that it is robust to different forms and degrees of dependency relationships, including linearity and non-linearity. In contrast, its strongest contender, CAD, has high standard deviations on Syn1 and Syn6. One possible reason is that CAD gets trapped in a bad local minimum when using expectation-maximization algorithm to learn parameters. Despite being a contextual anomaly detector, ROCOD performs poorly on datasets Syn2 and Syn5 in terms of PRC AUC and Precision@n. From the results on Syn6, Syn7, Syn8, Syn9 and Syn10, it appears that a larger number of contextual features does not substantially affect the performance of QCAD. Compared to other contextual anomaly detectors, and specifically CAD, QCAD does not perform particularly better on these synthetic datasets.

The results on real-world datasets presented in Tables 4.6 and 4.7 show that QCAD performs best overall when compared to its contenders, as witnessed by its average ranking results. Concretely, QCAD outperforms the other methods on 13 out of 20 real-world datasets in terms of PRC AUC and ROC AUC, and on 11 out of 20 datasets in terms of P@n. On most of the remaining datasets, QCAD is on par with its strongest competitors. For instance, HBOS achieves the best performance on Forest Fires, Heart Failures, Hepatitis, and Indian Liver Patient; QCAD's performance on these datasets is comparable. The number of contextual features in these datasets is often substantially less than the number of behavioral features, reducing the contextual anomaly detection problem to a traditional anomaly detection problem. QCAD is only slightly worse than ROCOD on Power Plant and Toxicity in terms of PRC AUC. Note that CAD produces surprisingly good results on the Bodyfat dataset, and surpasses other methods including QCAD. One possible explanation is that the contexts and behaviors are both composed of well-separated Gaussian components, and that the association between contextual and behavioral components is strong.

We also observe that QCAD performs well on datasets with varying sample size,

Table 4.6: Performance in terms of PRC AUC, ROC AUC, and P@n, on real-world data, with 5 or 10 independent runs of injecting contextual anomalies into each dataset[1]. For each anomaly detector on each dataset, the mean value and standard deviation of each evaluation criterion is presented. The best results obtained on each dataset are highlighted in bold.

		QCAD	LoPAD	ROCOD	CAD[2]	IForest	LOF	k-NN	SOD	HBOS
Abalone	PRC AUC	**0.96**±0.01	0.02±0.00	0.55±0.04	0.27	0.39±0.05	0.92±0.01	0.94±0.00	0.75±0.02	0.19±0.03
	ROC AUC	**1.00**±0.00	0.36±0.05	0.98±0.00	0.93	0.98±0.00	**1.00**±0.00	**1.00**±0.00	0.96±0.01	0.91±0.01
	P@n	0.90±0.02	0.02±0.00	0.58±0.01	0.40	0.41±0.07	0.92±0.02	**0.93**±0.00	0.74±0.02	0.19±0.06
Airfoil	PRC AUC	**0.69**±0.05	0.05±0.01	0.41±0.05	0.46±0.04	0.10±0.02	0.05±0.01	0.05±0.01	0.14±0.02	0.09±0.02
	ROC AUC	**0.91**±0.03	0.51±0.03	0.79±0.02	0.76±0.03	0.72±0.03	0.53±0.04	0.52±0.02	0.78±0.02	0.66±0.02
	P@n	**0.66**±0.05	0.04±0.03	0.40±0.04	0.45±0.02	0.13±0.04	0.05±0.03	0.06±0.03	0.14±0.03	0.12±0.02
BodyFat	PRC AUC	0.60±0.11	0.08±0.02	0.54±0.00	**0.92**±0.08	0.16±0.03	0.09±0.02	0.09±0.02	0.10±0.04	0.18±0.04
	ROC AUC	0.91±0.05	0.48±0.04	0.50±0.00	**0.99**±0.00	0.73±0.04	0.51±0.06	0.52±0.05	0.53±0.08	0.76±0.04
	P@n	0.58±0.12	0.06±0.06	0.08±0.03	**0.88**±0.05	0.16±0.06	0.08±0.08	0.10±0.06	0.10±0.04	0.23±0.07
Boston	PRC AUC	**0.72**±0.05	0.08±0.01	0.30±0.06	0.33±0.13	0.11±0.02	0.08±0.01	0.08±0.02	0.08±0.01	0.13±0.03
	ROC AUC	**0.92**±0.03	0.48±0.03	0.83±0.04	0.70±0.11	0.60±0.04	0.48±0.04	0.49±0.04	0.51±0.04	0.66±0.04
	P@n	**0.68**±0.04	0.07±0.02	0.37±0.05	0.25±0.10	0.13±0.03	0.07±0.04	0.07±0.04	0.06±0.03	0.15±0.06
Concrete	PRC AUC	**0.63**±0.06	0.09±0.01	0.33±0.06	0.33±0.07	0.08±0.02	0.06±0.01	0.06±0.01	0.06±0.01	0.25±0.06
	ROC AUC	**0.93**±0.03	0.62±0.03	0.77±0.04	0.70±0.04	0.61±0.03	0.49±0.06	0.50±0.05	0.50±0.03	0.72±0.05
	P@n	**0.62**±0.05	0.1±0.03	0.33±0.04	0.33±0.06	0.08±0.03	0.06±0.03	0.06±0.02	0.05±0.02	0.29±0.04
El Nino	PRC AUC	**0.98**±0.01	0.11±0.01	0.40±0.02	0.57	0.53±0.08	0.01±0.00	0.01±0.00	0.04±0.00	0.77±0.03
	ROC AUC	**1.00**±0.00	0.91±0.01	0.98±0.01	0.91	0.97±0.01	0.50±0.02	0.50±0.03	0.90±0.00	0.99±0.00
	P@n	**0.93**±0.02	0.13±0.03	0.42±0.03	0.55	0.50±0.06	0.01±0.01	0.01±0.01	0.01±0.01	0.70±0.02
Energy	PRC AUC	**0.92**±0.02	0.05±0.01	0.42±0.06	0.32±0.05	0.35±0.07	0.10±0.03	0.37±0.02	0.89±0.04	0.31±0.05
	ROC AUC	**0.99**±0.00	0.40±0.05	0.82±0.03	0.62±0.04	0.88±0.03	0.55±0.04	0.95±0.00	0.98±0.00	0.79±0.02
	P@n	**0.81**±0.04	0.02±0.01	0.42±0.04	0.32±0.06	0.43±0.07	0.11±0.04	0.30±0.03	0.79±0.06	0.35±0.06
FishWeight	PRC AUC	**0.81**±0.05	0.13±0.06	0.46±0.10	0.41±0.12	0.23±0.07	0.12±0.04	0.12±0.04	0.11±0.03	0.21±0.06
	ROC AUC	**0.96**±0.02	0.52±0.05	0.82±0.07	0.69±0.11	0.77±0.06	0.51±0.10	0.52±0.09	0.56±0.08	0.69±0.06
	P@n	**0.71**±0.10	0.13±0.09	0.44±0.08	0.45±0.12	0.31±0.11	0.14±0.11	0.13±0.10	0.09±0.06	0.30±0.13
ForestFires	PRC AUC	0.99±0.00	0.10±0.01	0.48±0.05	0.29±0.07	0.97±0.02	0.17±0.02	0.18±0.02	0.58±0.04	**1.00**±0.00
	ROC AUC	**1.00**±0.00	0.50±0.04	0.88±0.02	0.85±0.05	**1.00**±0.00	0.71±0.04	0.74±0.01	0.96±0.01)	**1.00**±0.00
	P@n	0.96±0.02	0.09±0.04	0.41±0.04	0.28±0.12	0.94±0.02	0.16±0.05	0.12±0.04	0.60±0.04	**0.99**±0.01
GasEmission	PRC AUC	**0.94**±0.02	0.01±0.00	0.86±0.02	0.63	0.10±0.01	0.01±0.00	0.01±0.00	0.03±0.00	0.27±0.07
	ROC AUC	**1.00**±0.00	0.47±0.01	**1.00**±0.00	0.99	0.93±0.01	0.52±0.04	0.52±0.01	0.78±0.01)	0.97 ±0.00
	P@n	**0.89**±0.02	0.00±0.00	0.83±0.02	0.62	0.05±0.02	0.02±0.01	0.02±0.02	0.02±0.02	0.24±0.05
HeartFailure	PRC AUC	0.90±0.03	0.05±0.00	0.53±0.04	0.82±0.06	0.86±0.04	0.13±0.02	0.18±0.04	0.38±0.07	**0.97**±0.01
	ROC AUC	0.98±0.01	0.10±0.03	0.52±0.05	0.98±0.01	0.98±0.01	0.57±0.06	0.68±0.05	0.88±0.02	**1.00**±0.00
	P@n	0.83±0.05	0.00±0.00	0.16±0.14	0.78±0.05	0.80±0.05	0.12±0.06	0.20±0.07	0.34±0.07	**0.92**±0.03
Hepatitis	PRC AUC	0.98±0.01	0.05±0.01	0.55±0.03	0.79±0.07	0.92±0.03	0.17±0.02	0.14±0.02	0.35±0.01	**0.99**±0.01
	ROC AUC	**1.00**±0.00	0.49±0.06	0.98±0.00	0.99±0.00	**1.00**±0.00	0.87±0.02	0.84±0.01	0.96±0.00	**1.00**±0.00
	P@n	0.92±0.02	0.05±0.02	0.65±0.04	0.78±0.06	0.89±0.02	0.06±0.04	0.07±0.04	0.33±0.03	**0.96**±0.01

Table 4.7: (Table 4.6 Continued) Performance in terms of PRC AUC, ROC AUC, and P@n, on real-world data, with 5 or 10 independent runs of injecting contextual anomalies into each dataset[1]. For each anomaly detector on each dataset, the mean value and standard deviation of each evaluation criterion is presented. The best results obtained on each dataset are highlighted in bold.

		QCAD	LoPAD	ROCOD	CAD[2]	IForest	LOF	k-NN	SOD	HBOS
IndianLiver	PRC AUC	0.90±0.05	0.05±0.00	0.48±0.06	0.84±0.12	0.93±0.02	0.11±0.02	0.17±0.04	0.44±0.06	**0.99**±0.01
	ROC AUC	**1.00**±0.00	0.48±0.05	0.97±0.01	0.94±0.16	**1.00**±0.00	0.72±0.05	0.81±0.04	0.96±0.01	**1.00**±0.00
	P@n	0.85±0.03	0.05±0.02	0.47±0.07	0.74±0.25	0.90±0.02	0.07±0.03	0.20±0.10	0.40±0.07	**0.95**±0.02
Maintenance	PRC AUC	**0.91**±0.01	0.01±0.00	0.50±0.00	0.50	0.02±0.01	0.01±0.00	0.01±0.00	0.02±0.00	0.19±0.08
	ROC AUC	**1.00**±0.00	0.48±0.05	0.50±0.00	0.50	0.70±0.02	0.49±0.01	0.55±0.02	0.79±0.01	0.67±0.04
	P@n	**0.84**±0.02	0.01±0.02	0.04±0.01	0.00	0.03±0.02	0.00±0.00	0.01±0.00	0.01±0.01	0.32±0.09
Parkinson	PRC AUC	**0.81**±0.01	0.02±0.01	0.58±0.06	0.68	0.02±0.00	0.02±0.00	0.02±0.00	0.04±0.00	0.02±0.00
	ROC AUC	**0.99**±0.00	0.43±0.03	0.94±0.02	0.90	0.56±0.03	0.48±0.03	0.48±0.02	0.75±0.02	0.59±0.03
	P@n	**0.85**±0.01	0.02±0.01	0.54±0.05	0.62	0.01±0.01	0.04±0.01	0.06±0.01	0.02±0.01	0.01±0.01
PowerPlant	PRC AUC	0.56±0.03	0.02±0.00	**0.65**±0.05	0.32	0.04±0.00	0.01±0.00	0.01±0.00	0.01±0.00	0.22±0.03
	ROC AUC	**0.98**±0.01	0.45±0.01	0.97±0.01	0.64	0.78±0.03	0.50±0.03	0.49±0.02	0.63±0.03	0.71±0.04
	P@n	0.58±0.05	0.01±0.01	**0.68**±0.01	0.31	0.07±0.02	0.01±0.01	0.02±0.02	0.01±0.01	0.27±0.04
QSRanking	PRC AUC	**0.79**±0.04	0.06±0.00	0.57±0.08	0.24±0.07	0.09±0.03	0.09±0.03	0.09±0.02	0.09±0.01	0.47±0.08
	ROC AUC	**0.96**±0.01	0.37±0.02	0.89±0.03	0.74±0.04	0.48±0.07	0.48±0.07	0.48±0.04	0.51±0.04	0.87±0.03
	P@n	**0.72**±0.05	0.02±0.02	0.53±0.08	0.26±0.09	0.08±0.05	0.08±0.04	0.08±0.05	0.08±0.03	0.48±0.07
SynMachine	PRC AUC	**0.96**±0.03	0.34±0.04	0.79±0.06	0.42±0.07	0.34±0.05	0.80±0.04	0.91±0.00	0.71±0.06	0.26±0.03
	ROC AUC	0.98±0.01	0.78±0.03	0.89±0.03	0.65±0.07	0.84±0.03	0.92±0.03	**0.99**±0.01	0.85±0.03	0.68±0.02
	P@n	**0.94**±0.03	0.39±0.05	0.73±0.07	0.34±0.11	0.39±0.05	0.72±0.04	0.78±0.03	0.65±0.04	0.27±0.05
Toxicity	PRC AUC	**0.46**±0.09	0.10±0.01	**0.46**±0.06	0.50±0.06	0.10±0.01	0.08±0.02	0.07±0.01	0.12±0.02	0.10±0.01
	ROC AUC	**0.88**±0.03	0.57±0.03	0.84±0.03	0.56±0.13	0.71±0.04	0.60±0.04	0.60±0.03	0.73±0.06	0.69±0.03
	P@n	**0.49**±0.07	0.18±0.01	0.47±0.06	0.12±0.15	0.09±0.04	0.06±0.03	0.05±0.01	0.11±0.04	0.09±0.04
Yacht	PRC AUC	**0.90**±0.04	0.53±0.02	0.29±0.05	0.55±0.00	0.28±0.08	0.20±0.04	0.25±0.03	0.24±0.05	0.32±0.08
	ROC AUC	**0.98**±0.02	0.96±0.00	0.78±0.03	0.50±0.00	0.78±0.06	0.72±0.05	0.83±0.02	0.59±0.07	0.76±0.05
	P@n	**0.80**±0.06	0.58±0.06	0.30±0.06	0.08±0.04	0.32±0.08	0.19±0.05	0.24±0.04	0.25±0.05	0.30±0.07
Ranking [3]	PRC AUC	1.40	7.45	3.30	3.45	4.90	7.20	6.75	5.75	4.25
	ROC AUC	1.40	7.95	3.70	5.00	4.15	7.00	6.05	5.00	4.10
	P@n	1.45	7.35	3.65	4.00	4.60	6.95	6.25	6.05	4.10

[1] Due to computation time limitations, we perform 5 independent trials for datasets with a sample size greater than 2000. For smaller datasets, we conduct 10 independent trials.

[2] We only conduct an independent experiment using CAD on Maintenance, Gas Emission, Parkinson Telemonitoring, Power Plant, and Elnino. It would take more than 1 week for CAD to complete 5 independent trials on each of these individual datasets.

[3] This is the average ranking of each anomaly detector on 20 real-world datasets in terms of PRC AUC, ROC AUC, or P@n, respectively.

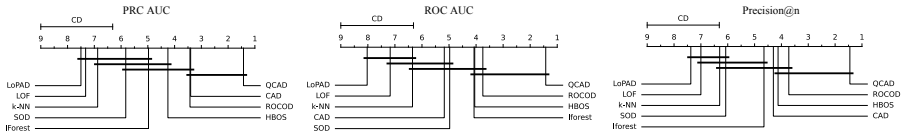


Figure 4.5: Critical difference diagram showing statistical difference comparisons between QCAD and its contenders in terms of PRC AUC, ROC AUC and Precision@n. To achieve this, we use Friedman tests [182] followed by Nemenyi post hoc analysis [183] with a significance level of 0.05. The post-hoc Nemenyi test indicates there are no significant differences within QCAD, CAD and ROCOD in terms of PRC AUC; there are no significant differences within QCAD, ROCOD, HBOS and IForest in terms of ROC AUC; there are no significant differences within QCAD, ROCOD, HBOS and CAD in terms of Precision@n. However, one can see that QCAD consistently outperforms its contenders by a large margin in three metrics.

dimensionality, and rate of injected anomalies. Specifically, QCAD achieved a high ROC AUC (≥ 0.85) on all datasets. This is much better than the ROC AUC values close to 0.50 obtained by ROCOD, CAD, HBOS, and IForest on some datasets, implying they are random guessing. In addition, QCAD attained high PRC AUC (≥ 0.8) and Precision@n values (≥ 0.7) on most datasets. The lowest PRC AUC and Precision@n value are is 0.46 and 0.49, respectively, both obtained on the Toxicity dataset. Possible reasons for QCAD's moderate performance on Airfoil, BodyFat, Power plant, and Toxicity are: the dependency relationship between behavioral features and contextual features is not strong, or the dependency relationship is too complex for the Quantile Regression Forests to capture. ROCOD, COD, HBOS, and IForest, however, obtain PRC AUC values less than 0.70 and Precision@n values less than 0.60 on most datasets. Moreover, HBOS and IForest sometimes attain PRC AUC and Precision@n lower than 0.10, which is extremely poor. This implies that traditional anomaly detection methods are not suitable for identifying contextual anomalies, as they treat all features equally. Furthermore, the relatively poor performance of ROCOD and CAD—compared to our method—demonstrates the importance of modeling more properties of the conditional distribution than just the mean.

4.6.5 Runtime Analysis

To investigate the scalability and efficiency of QCAD, we perform a runtime analysis by varying the sample size, the number of contextual features, and the number of behavioral features. As shown in Figure 4.6, we can empirically observe that the runtime of QCAD scales linearly with respect to the sample size, the number of contextual features, and the number of behavioral features on small and medium datasets. In

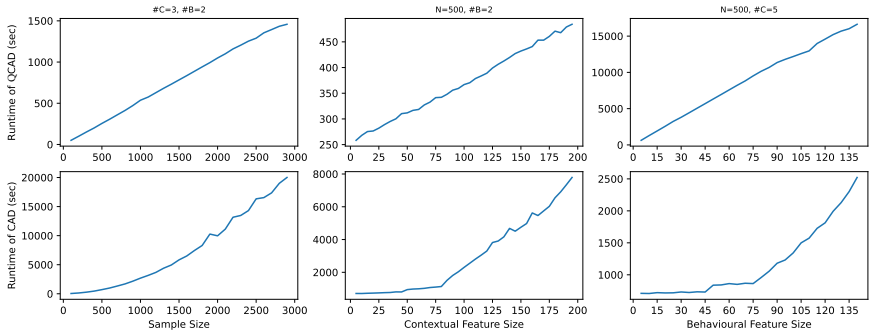


Figure 4.6: Runtime analysis by varying the sample size, the number of contextual features (#**C**), and the number of behavioral features (#**B**). The results are obtained with 5 independent trials. The data was synthesized using scheme S1. Note the varying y-axis scales.

contrast, CAD is computationally prohibitively expensive even on small datasets. To further understand the complexity of QCAD, we perform a time complexity analysis in Appendix 4.7. The theoretical analysis is in line with our empirical results and analysis.

4.6.6 Using QCAD to Find Promising Football Players

In this section, we illustrate the capability of QCAD on identifying potentially meaningful contextual anomalies and providing intuitive and understandable explanations for reported anomalies when applied to a real-world problem.

As use case, we consider the problem of finding exceptional players in the English Premier League. The dataset[3] describes the background information and performance statistics of 532 football players from 2020 to 2021. We use the variables *Position1,Position2* (positions for which the player plays), *Age* (age of the player), *Matches* (number of matches played), *Starts* (number of matches that the player was in the starting lineup), and *Mins* (number of minutes the player played overall) as contextual features. Two performance statistics, *Goals* (number of goals scored by the player) and *Assists* (number of assists given by player), are regarded as behavioral features.

Figure 4.7 depicts the distribution of all data objects in behavioral space *Goals* × *Assists*. Without considering contextual information, traditional anomaly detectors

[3]https://www.kaggle.com/rajatrc1705/english-premier-league202021

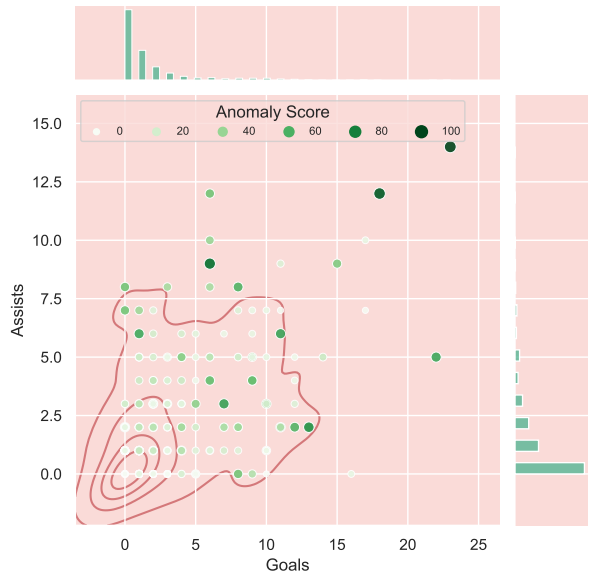


Figure 4.7: Distribution of all players in the behavioral space *Assists* × *Goals*, and their corresponding anomaly scores given by QCAD. The red curves represent the estimated densities of the joint distribution at different levels, the histograms indicate the marginal distributions. A larger and darker green dot represents a larger anomaly score as given by QCAD.

such as HBOS, LOF, IForest will treat objects in dense areas (i.e., objects residing inside the red curves) as normal objects. In contrast, our contextual anomaly detector QCAD can detect anomalies in dense areas by considering contextual information (i.e., green objects residing inside the red curves). Concretely, Figure 4.7 also presents the anomaly scores given by QCAD (with default settings). For example, player Matheus Pereira—who achieved 11 goals and 6 assists—is reported as an 'anomaly' (i.e., as having a relatively high anomaly score) by QCAD but as 'normal' by traditional anomaly detectors. Conversely, QCAD considers Patrick Bamford, a player with 17 goals and 7 assists (in a sparse area), to be normal, while traditional anomaly detectors consider him to be an anomaly.

In addition to giving an anomaly score for each data object, QCAD can also provide an explanation. For instance, for Matheus Pereira ($Position1$ = "MF", $Position2$ = "FW", Age=24, $Matches$=33, $Starts$=30, $Mins$=2577), Figure 4.8 shows the distribution of the values of each contextual feature for all players in his contextual neighborhood (i.e., similar players). It can be seen that most of these players are Midfielder (MF) and/or Forward(FW), aged from 22 to 28, playing matches more than

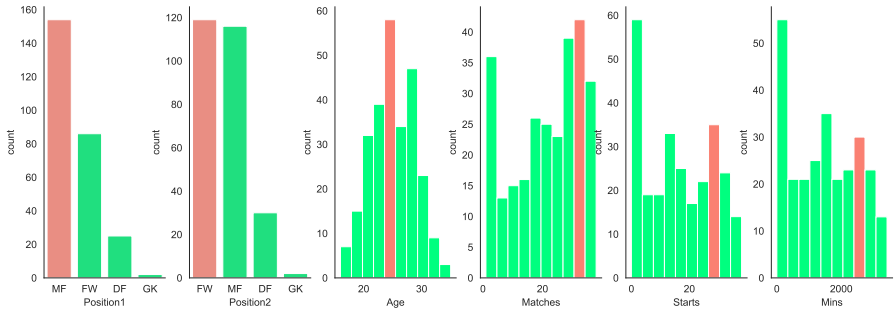


Figure 4.8: Explanations for the identified anomalous player Matheus Pereira based on his reference group. Shown are the values for Matheus Pereira (red bins) relative to the distribution of his contextual neighbors for each contextual feature.

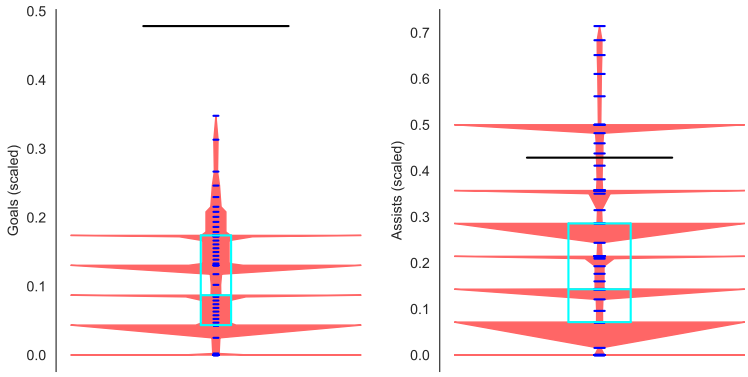


Figure 4.9: Explaining why Matheus Pereira is considered an anomaly: exceptionally many goals compared to contextually similar players, and relatively many assists. The beanplot shows the estimated conditional distribution of each behavioral feature, with a wider red area representing a higher probability of occurrence. The horizontal black lines represent the values reported for the player being investigated.

25 times, etc. The anomaly score is 65.8, which puts this player in the top-10 of most anomalous players. This score can be decomposed according to the behavioral features to give the behavioral feature(s) that contribute the most to the obtained anomaly score; *Goals*, in this case. Moreover, Figure 4.9 shows how the beanplot-like visualization can help to give insight in how the behavioral feature values deviate from those in the player's contextual neighborhood. From these plots we can see that Matheus Pereira has been exceptionally good at scoring goals and slightly better than expected with regard to giving assists.

These results can be interpreted as: Matheus Pereira performed surprisingly well in terms of goals and slightly better than expected in terms of assists when compared to other midfielders and/or forwards aged 22-28 who played more than 25 games and played more than 2000 minutes. Although we are not football experts, we expect the automated detection and interpretable score explanation of such 'anomalies' to be meaningful and possibly useful to football coaches and scouts.

4.7 Conclusions

In this paper, we for the first time explicitly establish a connection between dependency-based traditional anomaly detection methods and contextual anomaly detection methods. On this basis, we propose a novel approach to contextual anomaly detection and explanation. Specifically, we use Quantile Regression Forests to develop a accurate and interpretable anomaly detection method, QCAD, that explores dependencies between features. QCAD can handle tabular datasets with mixed contextual features and numerical behavioral features. Extensive experiment results on various synthetic and real-world datasets demonstrate that QCAD outperforms state-of-the-art anomaly detection methods in identifying contextual anomalies in terms of accuracy and interpretability.

From the case study on football player data, we conclude that QCAD can detect potentially meaningful and useful contextual anomalies that are directly interpretable by a domain expert. The beanplot-based visualizations help to explain why a certain object is (not) considered an anomaly within its context. This is important because anomaly detection is an unsupervised problem and the explanations can help analysts and domain experts to verify the results. It also opens up opportunities for human-guided anomaly detection, where feedback from the analyst can be used to guide the analysis.

In the future, given that QCAD can only handle static features, we plan to extend

QCAD to streaming settings.

Appendix A: Complexity Analysis

The computational overhead of our QCAD framework mainly comes from two parts: the calculation of Gower's Distance matrix using contextual features, and the calculation of anomaly score using all features based on Quantile Regression Forests.

The time complexity of calculating Gower's Distance matrix is $O(N^2 D_{cnt})$, where N represents the sample size and D_{cnt} denotes the number of contextual features. In addition, the time complexity of constructing a random forest is $O(n_{tree} \cdot n_{feature} \cdot klog(k))$ and using it for prediction is $O(n_{tree} \cdot n_{feature})$ [184], where n_{tree} is the number of trees used to form a random forest, $n_{feature}$ denotes the number of dimensions (i.e., D_{cnt} at most) and k indicates the number of samples used to construct the forest (i.e., the number of nearest neighbors in our case). Hence, the time complexity of constructing N quantile regression forests in D_{bhv} behavioral features is $O(n_{tree} \cdot D_{cnt} \cdot klog(k) \cdot N \cdot D_{bhv})$. Moreover, we have to estimate 100 different quantiles using each quantile regression forest, resulting in $O((n_{tree} \cdot D_{cnt} \cdot klog(k) + n_{tree} \cdot D_{cnt} \cdot 100) \cdot N \cdot D_{bhv})$. Therefore, using our default setting leads to $O((100 \cdot D_{cnt} \cdot \min(500, \frac{N}{2}) \cdot \log(\min(500, \frac{N}{2})) + 100 \cdot D_{cnt} \cdot 100) \cdot N \cdot D_{bhv})$. In the worst case, it is $O(10^5 N D_{cnt} D_{bhv})$.

Overall, the time complexity of our method is $O(10^5 N D_{cnt} D_{bhv} + N^2 D_{cnt})$ in the worst case. Particularly, the time complexity becomes $O(10^5 N D_{cnt} D_{bhv})$ when $N < 10^5$ (i.e., for small and medium datasets). Besides, the construction of quantile regression forests for each object in each behavioral feature can easily be performed in a parallel way, thus reducing the time complexity to $O(10^5 N D_{cnt} + N^2 D_{cnt})$ in the worst case.

Appendix B: Dataset Description

Abalone The dataset contains 4177 abalone physical measurement records. Specifically, we use the features Sex, Length, Diameter and Height as contextual features. Accordingly, we use the features Whole Weight, Shucked Weight, Viscera Weight, Shell Weight and Rings as behavioral features. This dataset is downloaded from UCI, [4] containing no missing values.

[4] https://archive.ics.uci.edu/ml/datasets/abalone

Airfoil Self-Noise This dataset contains 1503 records from a series of aerodynamic and acoustic tests of airfoil blade sections. Specifically, we use the features describing the frequency, the angle of attack, the chord length, the free-stream velocity and the suction side displacement thickness (e.g., f, alpha, c, U_infinity and delta) as contextual features. Meanwhile, the feature describing the scaled sound pressure level (e.g., SSPL) as behavioral feature. This dataset is downloaded from UCI, [5] containing no missing values.

Bodyfat This dataset contains the estimates of body density and the percentage of body fat (used as behavioral features), which are determined by various body circumference measurements, such as, Age, Weight, Height, Neck circumference, Chest circumference, Abdomen 2 circumference, Hip circumference, Thigh circumference, Knee circumference, Ankle circumference, Biceps (extended) circumference, Forearm circumference, and Wrist circumference (used as contextual features) for 252 men. The raw dataset includes no categorical features and 0 missing values, downloaded from the CMU statlib. [6]

Boston House Price This dataset contains 583 records of the Boston house price. We use the features describing the properties of the house and its surroundings, i.e., CRIM, ZN, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B, LSTAT as contextual features, and the median value of owner-occupied homes, i.e., MED, as behavioral feature. This dataset is released by [185] and downloaded from the CMU statlib [7]. It remains 506 records after dropping rows containing missing values.

Concrete Compressive Strength This dataset contains 1030 records about the compressive strength of concrete. We use the features describing the age and different ingredients as contextual features, including C1, C2, C3, C4, C5, C6, C7 and Age. In addition, we use the feature Strength as behavioral feature. This dataset is downloaded from UCI, [8] containing no missing values.

El Nino This dataset contains 178080 records of the oceanographic and surface meteorological readings, which are taken from buoys located throughout the equatorial Pacific. We use the temporal and spatial features, i.e., Year, Month, Day, Date,

[5] https://archive.ics.uci.edu/ml/datasets/Airfoil+Self-Noise
[6] http://lib.stat.cmu.edu/datasets/bodyfat
[7] http://lib.stat.cmu.edu/datasets
[8] https://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength

Latitude, Longitude as contextual features, and other features, i.e., Zonal_Winds, Meridional_Winds, Humidity, Air_Temp, Sea_Surface_Temp as behavioral features. This dataset is downloaded from the UCI machine learning repository.[9] It remains 93935 records after dropping rows containing missing values. However, in order for all algorithms to complete the experiment on this dataset within 12 hours, we randomly downsample the original dataset to 20,000 records.

Energy Efficiency This dataset contains 768 records about a study which assesses the energy efficiency as a function of building parameters. Accordingly, the building parameters such as X1, X2, X3, X4, X5, X6, X7 and X8 are regarded as contextual features, and the heating load and cooling load (namely Y1 and Y2) are treated as behavioral features. This dataset is downloaded from UCI, [10] containing no missing values.

Fish Weight This dataset contains 157 records about the features of common species of fish in the market. Accordingly, we use the features including Species, Length1, Length2, Length3, Height, Width as contextual features, and Weight as behavioral feature. This dataset is downloaded from Kaggle, [11] containing no missing values.

Forest Fires This dataset contains 517 records concerning meteorological and spatiotemporal information about forest fires in the northeast region of Portugal. We use the spatiotemporal features such as X, Y, month, day as contextual features, and the rest features, i.e., FFMC, DMC, DC, ISI, temp, RH, wind, rain, area as behavioral features. It is downloaded from the UCI machine learning repository,[12] containing no missing values.

Gas Turbine CO and NOx Emission The original dataset includes 36733 records of sensor measures data, which is collected from a gas turbine in Turkey from 2011 to 2015. The dataset is collected from the same power plant to study flue gas emissions and predict the hourly net energy yield. Consequently, we use the features describing the turbine parameters (e.g., AT, AP, AH, AFDP, GTEP, TIT, TAT and CDP) as

[9] archive.ics.uci.edu/ml/datasets/El+Nino
[10] https://archive.ics.uci.edu/ml/datasets/Energy+efficiency
[11] https://www.kaggle.com/aungpyaeap/fish-market
[12] https://archive.ics.uci.edu/ml/datasets/forest+fires

contextual features. The variables characterizing the turbine energy yield and emissions of gas (e.g., TEY, CO and NOX) are used as behavioral features. However, for all algorithms to complete the experiments within 12 hours, we only use the 7383 records in 2015. This dataset is downloaded from UCI, [13] containing no missing values.

Heart Failure This dataset contains heart failure clinical records of 299 patients suffering from heart failure. It contains 13 clinical features, of which we use age, sex, smoking, diabetes, high_blood_pressure, and anaemia as contextual features, and creatinine_phosphokinase, ejection_fraction, platelets, serum_creatinine, serum_sodium, time as behavioral features. Besides, the death_event feature is removed and there is no missing value in this dataset. The original dataset is released by [186] and we download it from the UCI machine learning repository. [14]

Hepatitis This dataset contains 615 records concerning the laboratory values of blood donors and Hepatitis C patients. We use the demographic features such as sex and age, and Category of donors as contextual features, and the rest of features, i.e., ALB, ALP, ALT, AST, BIL, CHE, CHOL, CREA, GGT and PROT as behavioral features. The original dataset is downloaded from the UCI machine learning Repository. [15] 26 records contain missing values and therefore are removed.

Indian Liver Patient This dataset contains the medical records of 583 Indian liver patients, 4 of which contain missing values and are therefore deleted from further analysis. We treat Age, Gender and Selector as contextual features, and Total_Bilirubin, Direct_Bilirubin, Alkaline_Phosphotase, Alamine_Aminotransferase, Aspartate_Aminotransferase, Total_Protiens, Albumin, Albumin_and_Globulin_Ratio as behavioral features. This dataset is downloaded from the UCI machine learning repository. [16]

Maintenance of Naval Propulsion Plants The data set contains 11934 experimental records, which were performed by a numerical simulator of a naval vessel featuring a gas turbine propulsion plant. We use the features describing the gas turbine measures of the physical asset as contextual features, including LeverPosition, GTT, GTn, GGn, Ts, Tp, T48, T1, T2, P48, P1, P2, Pexh, TIC, mf. Meanwhile,

[13] https://archive.ics.uci.edu/ml/datasets/Gas+Turbine+CO+and+NOx+Emission+Data+Set
[14] archive.ics.uci.edu/ml/datasets/Heart+failure+clinical+records
[15] https://archive.ics.uci.edu/ml/datasets/HCV+data
[16] https://archive.ics.uci.edu/ml/datasets/ILPD+(Indian+Liver+Patient+Dataset)

we use the features containing the ship speed and the performance decay over time of gas turbine components, e.g., ShipSpeed, CompressorDecay and TurbineDecay, as behavioral features. This dataset is downloaded from UCI, [17] containing no missing values.

Parkinsons Telemonitoring The data set contains 5875 records of a series of biomedical voice measurements from 42 early-stage Parkinson's disease patients. There people were recruited into a six-month trial of remote monitoring equipment for remote symptom progression monitoring. The features such as subject, age, sex, test_time, Jitter, Jitter_Abs, Jitter_RAP, Jitter_PPQ5, Jitter_DDP, Shimmer, Shimmer_dB, Shimmer_APQ3, Shimmer_APQ5, Shimmer_APQ11, Shimmer_DDA, NHR, HNR, RPDE, DFA, PPE descirbe the background information, and thus are used as contextual features. Meanwhile, the corresponding scores motor_UPDRS and total_UPDRS are used as behavioral features. This dataset is downloaded from UCI, [18] containing no missing values.

Power Plant This dataset contains 9568 records from a combined cycle power plant which worked with full load from 2006 to 2011. Features such as hourly average ambient variables temperature, ambient pressure, relative humidity and exhaust vacuum (e.g., T, AP, RH and EP) are considered as contextual features, while the net hourly electrical energy output (EP) is regarded as behavioral feature. This dataset is downloaded from UCI, [19] containing no missing values.

QS University Ranking This dataset contains the QS rankings of the world universities from 2018 to 2020. We consider the wolrd rankings, national rankings and location of each university from 2018 to 2020, i.e., World2018, National2018, World2019, National2019,World2020, National2020, Country as contextual features. We use six features that are considered for the ranking (i.e., Academic Reputation, Employer Reputation, Faculty to Student Ratio, Number of citations per faculty, International Faculty, International Students) as behavioral features. The original data is crawled from the QS website, [20] containing 475 records after dropping missing values.

[17] https://archive.ics.uci.edu/ml/datasets/Condition+Based+Maintenance+of+Naval+Propulsion+Plants
[18] https://archive.ics.uci.edu/ml/datasets/Parkinsons+Telemonitoring
[19] https://archive.ics.uci.edu/ml/datasets/Combined+Cycle+Power+Plant
[20] https://www.topuniversities.com/qs-world-university-rankings

QSAR Fish Toxicity This dataset contains 908 records about the fish Pimephales promelas. Specifically, the six features describing the molecular of 908 chemicals will be used as contextual features, and the corresponding acute aquatic toxicity measure will be used as behavioral feature. This dataset is downloaded from UCI, [21] containing no missing values.

Synchronous Machine This dataset contains 557 records from a real experimental set. This experiment aims to construct a model to estimate the excitation current of synchronous motors. Specifically, we use the features describing the load current, power factor, power factor error and changing of excitation current (e.g., Iy, PF, e and dIf) as contextual features. Accordingly, the feature which describes the excitation current of synchronous machine (namely If) is used as behavioral feature. This dataset is downloaded from UCI, [22] containing no missing values.

Yacht Hydrodynamics This dataset contains 308 records about the features of sailing yachts. We use the features describing the dimensions, velocity and hydrodynamic performance of yachts, namely Longitudinal_position, Prismatic_coefficient, Length_displacement_ratio, Beam_draught_ratio, Length_beam_ratio, Froude_number as contextual features, and the feature concerning the residuary resistance per unit weight of displacement, namely resistance, as behavioral feature. This dataset is downloaded from the UCI machine learning repository, [23] containing no missing values.

Appendix C: Parameter Sensitivity Analysis

The parameter k directly affects the effectiveness and efficiency of anomaly detector in anomaly detection phase. If our dataset is labeled, we can use techniques like grid-search and cross-validation to set an optimal k for a specific dataset. However, anomaly detection is usually an unsupervised learning problem, which makes it impossible to set an optimal k. Hence, we can only empirically give some rules of thumb to set k according to the properties of specific dataset (i.e., sample size, dimensionality and estimated rate of anomaly). Therefore, we perform intensive experiments on synthetic and real-world datasets with a wide range of sample sizes, dimensionalities and rates of injected anomalies to investigate the sensitivity of our algorithm on this parameter.

[21] https://archive.ics.uci.edu/ml/datasets/QSAR+fish+toxicity
[22] https://archive.ics.uci.edu/ml/datasets/Synchronous+Machine+Data+Set
[23] archive.ics.uci.edu/ml/datasets/Yacht+Hydrodynamics

As shown in Figure 4.10, increasing the number of neighbors, i.e., k, will lead to a better performance in terms of PRC AUC, ROC AUC and Precision@n when k is small (about $N/10$ for most datasets). However, the performance gain gradually slows down as k increases, and the performance finally reaches a plateau with an increase of k. After that, further increasing k yields only a negligible performance gain, at the cost of runtime. Therefore, we set k to $N/2$ for small dataset and 500 for medium dataset after taking a trade-off between the accuracy and runtime cost. Overall, different from other k-NN based anomaly detectors which are sensitive to this parameter, our method QCAD is stable on parameter k as long as its value is not too small.

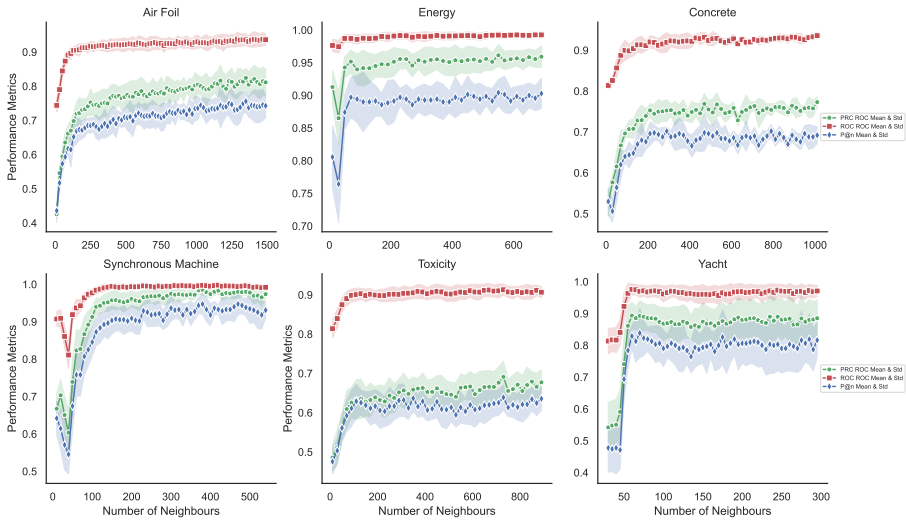


Figure 4.10: Sensitivity analysis on parameter k, where lines represent the mean values and the shaded areas indicate the corresponding standard deviations of each metric on 10 independent trials. Increasing the number of neighbors will first largely improve the performance in terms of PRC AUC (green line with asterisks), ROC AUC (red line with squares) and Precision@n (blue line with diamonds), and then yields negligible performance gains.

Appendix D: Ablation Study

4.7.1 Scaling Conditional Quantile Interval Length

In our method section, we have defined a matched conditional quantile interval length for an observation b^q when $b^q > \tau^q_{100}$ or $b^q < \tau^q_0$ as in Equation (4). We define the matched interval length for b^q based on $\max(w(\mathbf{x}|\mathbf{b}^q))$ and further scale it by

considering its distance to τ^q_{100} or τ^q_0. Alternatively, we can simply define the matched interval length as $\max(w(\mathbf{x}|\mathbf{b}^q))$ without scaling it. In other words, we could treat all observations residing outside $[\tau^q_0, \tau^q_{100}]$ equally. However, as shown in Figure 4.11, this will lead to a large performance degradation in terms of PRC AUC and Precision@n for most datasets.

Concretely, Figure 4.11 shows the difference of performance metrics, i.e., PRC ROC, ROC AUC and Precision@n between scaling $\max(w(\mathbf{x}|\mathbf{b}^q))$ with considering the distance to τ^q_{100} or τ^q_0, and not scaling $\max(w(\mathbf{x}|\mathbf{b}^q))$. For all datasets, the differences are positive. Particularly, these differences are significantly large in terms of PRC AUC and Precision@n for most datasets such as Energy, Hepatitis, Indian Liver Patient, Synthetic 5 and Synthetic 10. Therefore, it is pivotal to define the matched interval length by considering the distance from b^q to τ^q_{100} or τ^q_0 when $b^q > \tau^q_{100}$ or $b^q < \tau^q_0$, respectively.

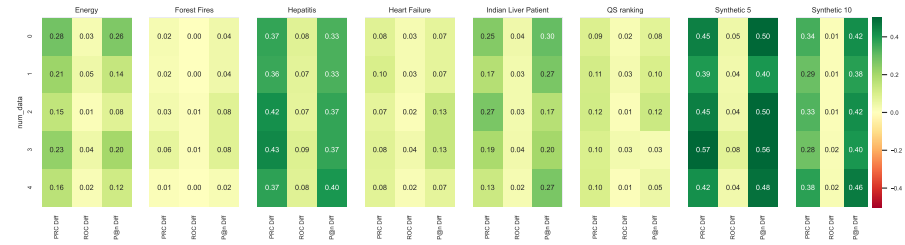


Figure 4.11: Effects of scaling matched conditional quantile interval length. For each dataset, the results are obtained by performing 5 independent trials of injecting contextual anomalies. For each trial, which is represented by the y-axis (namely *num_data*), the results are the difference (in terms of PRC AUC, ROC AUC, Precision@n, respectively) of methods with or without scaling the matched conditional quantile interval length. The large differences on most datasets imply the critical importance of scaling matched conditional quantile interval length.

4.7.2 Clipping Conditional Quantile Interval Length

To mitigate *dictator effect*, we have proposed to clip large matched conditional quantile interval lengths with an upper bound, which is set to $\frac{\eta}{100}$. More concretely, we set $\eta = 10$. To demonstrate the efficacy of this strategy on various datasets, we compute the performance metrics of QCAD by varying the hyper-parameter $\eta \in \{0.1, 0.2, 0.5, 1, 2, 3, ..., 11, 12, 15, 20, 30, 40, None\}$, where *None* means no clipping is performed.

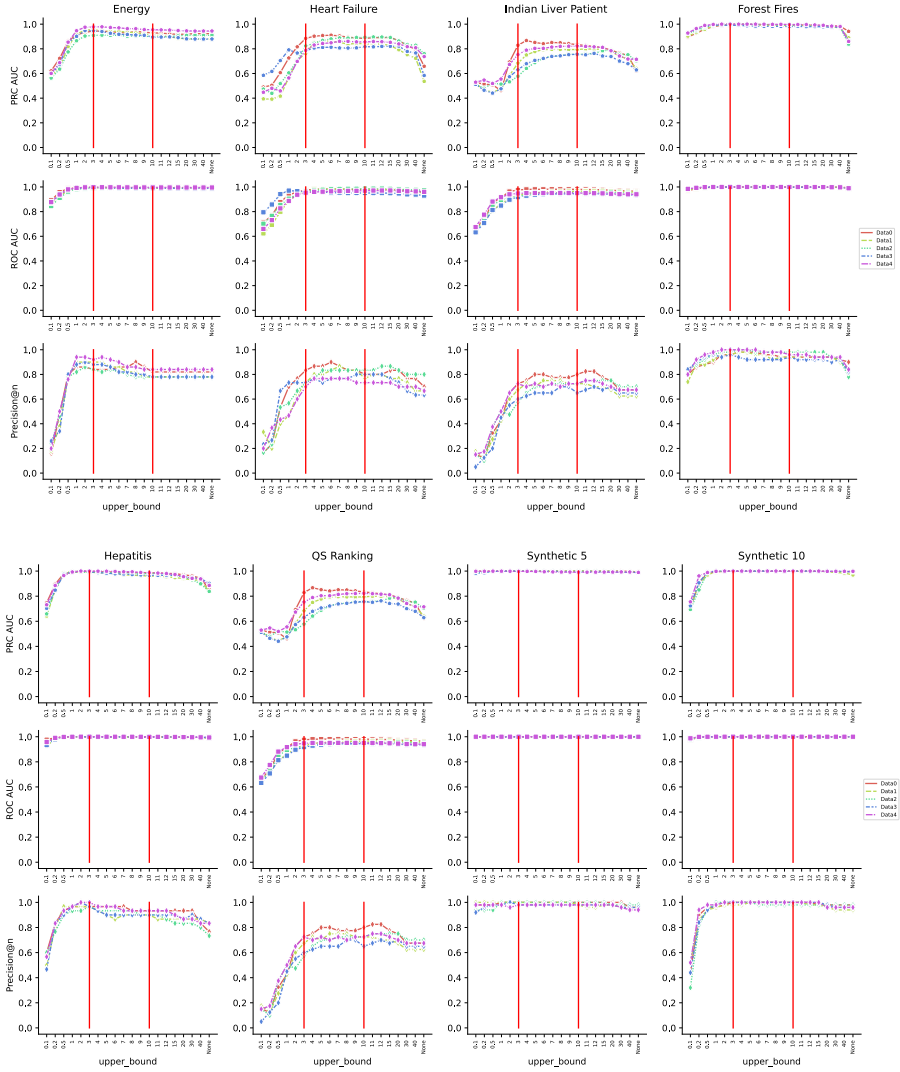


Figure 4.12: Effects of clipping matched conditional quantile interval length. For each dataset, the results are obtained by performing 5 independent trials (represented by 5 different curves) of injecting contextual anomalies. Then we compute the performance metrics (i.e., PRC AUC, ROC AUC, and Precision@n) of QCAD by varying the hyper-parameter η in $\{0.1, 0.2, 0.5, 1, 2, 3, ..., 11, 12, 15, 20, 30, 40, None\}$, where *None* means no clipping is performed. On most datasets, the best performances are achieved when $3 < \eta < 10$, as shown by the two vertical red lines.

As shown in Figure 4.12, increasing η will lead to higher PRC AUC, ROC AUC and Precision@n values when η is small (i.e., less than 1 for most datasets). Next, on the one hand, the ROC AUC stays stable as η increases, even without clipping (i.e. $\eta = None$). In other words, the clipping strategy does not affect ROC AUC metric. On the other hand, the PRC AUC and Precision@n gradually climb to a plateau as η increases. After a certain period, the PRC AUC and Precision@n start decreasing with an increase of η. Overall, on most datasets, PRC AUC and Precision@n generally achieve higher values with $3 \leq \eta \leq 10$ than those without clipping. Therefore, the *dictator effects* indeed exist and they mainly reduce the performance of QCAD in terms of PRC AUC and Precision@n. Moreover, our proposed clipping strategy can effectively overcome this problem.

Chapter 5

Explainable Graph Neural Networks Under Fire

Authors: **Zhong Li**, Simon Geisler, Yuhang Wang, Stephan Günnemann, Matthijs van Leeuwen

Submitted to IEEE Transactions on Knowledge and Data Engineering.

Abstract

Predictions made by graph neural networks (GNNs) usually lack interpretability due to their complex computational behavior and the abstract nature of graphs. In an attempt to tackle this, many GNN explanation methods have emerged. Their goal is to explain a model's predictions and thereby obtain trust when GNN models are deployed in decision critical applications. Most GNN explanation methods work in a post-hoc manner and provide explanations in the form of a small subset of important edges and/or nodes. In this paper we demonstrate that these explanations can unfortunately not be trusted, as common GNN explanation methods turn out to be highly susceptible to adversarial perturbations. That is, even small perturbations of the original graph structure that preserve the model's predictions may yield drastically different explanations. This calls into question the trustworthiness and practical utility of post-hoc explanation methods for GNNs. To be able to attack GNN explanation models, we devise a novel attack method dubbed *GXAttack*, the first *optimization-based* adversarial white-box attack method for post-hoc GNN explanations under such settings. Due to the devastating effectiveness of our attack, we call for an adversarial evaluation of future GNN explainers to demonstrate their robustness. For reproducibility, our code is available via GitHub.

5.1 Introduction

Graph Neural Networks (GNNs) [187] have achieved promising results in various learning tasks, including representation learning [188, 189], node classification [190, 191], link prediction [192, 193], and anomaly detection [55, 59]. However, intricate data representations and non-linear transformations render their interpretation and thus understanding their predictions non-trivial [194].

This has led to the introduction of GNN explanation methods, which can improve a model's transparency and thus increase trust in a GNN model, especially when it is deployed in a decision-critical application (e.g., where fairness, privacy, or safety are important) [195]. Moreover, they can help identify the scenarios where a GNN model may fail [196]. Explainability of a GNN model can be obtained by designing a self-explainable GNN model, which usually employs a simple model architecture and few parameters, leading to explainability at the cost of suboptimal prediction accuracy; or by extracting post-hoc explanations that do not influence the internal workings of a GNN model and thus maintains its high prediction accuracy. In this paper we only consider post-hoc GNN explanation methods, as many such methods have merged over the past decade, including but not limited to [195, 197, 198, 199, 196, 200, 201, 202, 203].

It has been shown that traditional deep neural networks (DNNs) are vulnerable to adversarial attacks [204, 205], where an imperceptible well-designed perturbation to input data can lead to substantially different prediction results. Recently, it was demonstrated that the explanations of predictions made by these DNNs are also fragile [206]. That is, an unnoticeable, well-designed perturbation to input data can result in completely different explanations while the model's predictions remain unchanged. However, the vulnerability of explanation methods has been primarily studied for DNN models designed for image and text data. In contrast, the vulnerability of GNN explanation methods to adversarial attacks has received very limited attention [207, 208], although a plethora of GNN explanation methods have recently been proposed [101, 209, 210].

Importantly, existing studies have only investigated the (in)stability of GNN explanations under relatively "mild" settings: they only consider *random* perturbations and/or *prediction-altering* perturbations. Specifically, [211] theoretically analyzed the stability of three GNN explainers under random perturbations. Further, [208] considered two scenarios: 1) graphs whose predictions are not changed after randomly flipping 20% edges; and 2) severe perturbations using an off-the-shelf attack algorithm

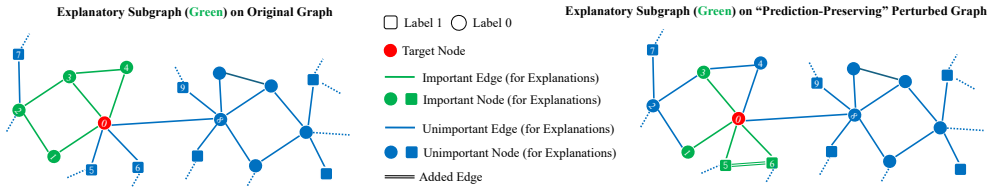


Figure 5.1: When using post-hoc GNN explainers, the explanatory subgraph on a graph with "prediction-preserving" perturbations (right) can strongly differ from that on the original graph (left).

to perturb at most 10% edges to change the GNN predictions (as well as the explanations). Besides, [212] proposed an adversarial attack to simultaneously change the GNN predictions and their explanations. By altering predictions and explanations, however, the graph can become inherently different. [207] explored evaluation metrics for GNN explanations from the perspective of adversarial robustness and resistance to out-of-distribution input, where they defined "adversarial robustness" as the difficulty of reversing a GNN prediction by perturbing the complementary of the explanatory subgraph.

Different from previous studies, we aim to perform structural perturbations that are both carefully crafted (i.e., *optimized* rather than *random*) and imperceptible (i.e., such that the GNN predictions remain unchanged but their explanations differ substantially). As shown in Figure 5.1, we empirically found that small *prediction-preserving* perturbations can result in largely different explanations generated by post-hoc GNN explainers. To our knowledge, we are the first to formally formulate and systematically investigate this problem under a white-box setting. Specifically, we devise *GXAttack*, the first *optimization-based* adversarial white-box attack on post-hoc GNN explanations. We employ a widely used GNN explainer, PGExplainer [199], as example target when designing our attack algorithm. Results on various datasets demonstrate the effectiveness of our approach. Moreover, our experiments show that other widely used GNN explanation methods, such as GradCAM [197], GNNExplainer [195], and SubgraphX [201], are also fragile under the attacks optimized for PGExplainer. Due to the devastating effectiveness of our attack, we call for an adversarial evaluation of future GNN explainers to demonstrate their robustness.

We first summarize related work in Chapter 5.2. Following this, we provide necessary background in Chapter 5.3. Then, we formalize the GNN explanation attack problem and introduce our novel attack algorithm GXAttack in Chapter 5.4. Next, we describe the experimental setups in Chapter 5.5. Experiment results and correspond-

ing analyses are given in Chapter 5.6. Finally, we conclude the paper in Chapter 5.7.

5.2 Related Work

5.2.1 GNN Explanations

[101, 209, 210] perform comprehensive surveys and propose excellent taxonomies of GNN explanation methods. Specifically, GNN explanations methods mainly include: 1) **Gradients-based methods**: Grad [213], GradCAM [197], GuidedBP [104], Integrated Gradients (IG) [214]; 2) **Decomposition-based methods**: GraphLRP [103], GNN-LRP [215]; 3) **Surrogate-based methods**: GraphLIME [203], PGM-Explainer [196]; 4) **Generation-based methods**: XGNN [198]; 5) **Perturbation-based methods**: GNNExplainer [195], PGExplainer [199], GraphMask [216], SubgraphX [201]; and 6) **CF-based methods**: CF-GNNExplainer [202], RCExplainer [200].

5.2.2 Attacks and Defenses of Traditional XAI Methods

It has been shown that traditional XAI methods (namely those designed to explain deep neural networks for image and text data) are susceptible to malicious perturbations [217, 218, 219]. Please refer to [206] for a comprehensive survey.

5.2.3 Robustness/Stability of GNN Explanations

[101, 211] point out that a reliable GNN explainer should exhibit stability, namely minor perturbations that do not impact the model predictions should not largely change the explanations. Particularly, [211] theoretically analyze the stability of three GNN explainers, including vanilla Grad, GraphMask and GraphLIME, for which they derive upper bounds on the instability of explanations. Importantly, [200, 207, 208, 212] are closely related but are different from our work and they are detailed as follows.

OAR. [207] present a novel metric for evaluating GNN explanations, termed Out-Of-Distribution-resistant Adversarial Robustness (OAR). OAR aims to solve the limitations of current removal- and generative-based evaluations. More concretely, they evaluate post-hoc explanation subgraphs by computing their robustness under attack, which consists of three steps: 1) they formulate the adversarial robustness tailored for GNN explanations problem, which is the minimum adversarial perturbation on

the structure of complementary subgraph; 2) they introduce a tractable and easy-to-implement objective of adversarial robustness for GNN explanations; after relaxation, the objective becomes a constrained graph generation problem (namely random perturbations) rather than an adversarial attack problem; 3) they present an Out-Of-Distribution (OOD) reweighting block that aims to confine the evaluation on original data distribution. This work is an initial attempt to explore evaluation metrics for GNN explanations from the perspectives of adversarial robustness and resistance to OOD. In contrast, we focus on optimization-based adversarial attack on post-hoc GNN explanations rather than providing an evaluation metric.

V-InFoR. [208] point out that "existing GNN explainers are not robust to the structurally corrupted graphs, namely graphs with noisy or adversarial edges." This study investigates the negative effect of minor structural corruptions (which do not change the predictions) and severe structural corruptions (which change the predictions) on GNN explainers. Particularly, they provide quantitative evidence that existing GNN explainers are fragile to structurally corrupted graphs. They evaluate 6 GNN explainers under two settings: 1) minor corruptions where they select the perturbed graphs whose predictions are not changed after randomly flipping 20% edges; 2) severe corruptions where they employ adversarial attack algorithm GRABNEL [220] to perturb at most 10% edges to change the predictions. In contrast, our method focus on optimization-based (rather than random) and prediction-preserving (rather than prediction-altering) adversarial attacks.

RCExplainer. [200] aim to find a small set of edges of the input graph such that the prediction result will substantially change if we remove those edges. Specifically, they propose RCExplainer to generate robust counter-factual explanations for GNNs in two steps: 1) they employ a set of decision regions to model the decision logic of a GNN, where each decision region determines the predictions on multiple graphs that are predicted to be the same class; 2) they leverage a DNN model to explore the decision logic, and thereby extract robust counter-factual explanations as a small set of edges of the input graph. This method is only applicable to GNNs that belong to Piecewise Linear Neural Networks. In this paper, we focus on attacking differentiable post-hoc GNN explanation methods that can be applied to any GNNs.

GEAttack. [212] empirically demonstrate that GNN explanation methods can be employed as tools to detect the adversarial perturbations on graphs. On this basis, they propose an attack framework dubbed *GEAttack* that can jointly attack both GNN and its explanations. Specifically, they formulate *GEAttack* as a bi-level optimization problem: 1) they mimic the GNN explanation method optimization process to obtain

graph explanations in the inner loop; 2) they compute the gradient of the attack objective w.r.t. the explanations in the outer loop. Different from GEAttack, we consider prediction-preserving attacks, which are inherently more challenging.

Concurrent Work. While the concurrent work [221] also addresses a similar topic, our study provides an alternative perspective by performing white-box attacks (namely we assume knowing the full knowledge about the GNN classifier and the GNN explainer), while they perform gray-box attacks by assuming knowing only the explanation loss and the generated explanatory edges of the GNN explainer.

5.2.4 Adversarial Robustness of GNNs

Starting with the seminal works [222, 223], that both proposed adversarial attacks on GNNs, a rich literature formed in the realm of GNN-specific adversarial attacks, defenses, and certifications [224, 225]. Most attacks primarily focus on altering predictions via perturbations of the discrete graph structure (edge insertions or deletions), either between existing nodes [222] or via the insertion of new (adversarial) nodes [226]. Optimization-based attacks either operate globally (number of edge insertions or deletions) [227, 228], alter a local neighborhood [222], or can handle a combination of both constraints [229]. While adversarial attacks on GNN explanation methods may benefit from optimization methods over the discrete graph structure, a careful discussion and derivation of appropriate attack objectives and metrics was missing.

5.3 Preliminaries

We utilize lowercase letters, bold lowercase letters, uppercase letters and calligraphic fonts to represent scalars (x), vectors ($\mathbf{x}$), matrices ($\mathbf{X}$), and sets ($\mathcal{X}$), respectively.

Definition 5.1 (Attributed Graph)**.** We denote an attributed graph as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$ where $\mathcal{V} = \{v_1, ..., v_n\}$ is the set of nodes. Let $\mathcal{E} = \{e_{ij}\}_{i,j \in \{1,...,n\}}$ be the set of edges, where $e_{ij} = 1$ if there exists an edge between v_i and v_j and $e_{ij} = 0$ otherwise. We use $\mathbf{A} \in \{0, 1\}^{n \times n}$ to denote the corresponding adjacency matrix. $\mathbf{X} \in \mathbb{R}^{n \times d}$ represents the node attribute matrix, where the i-th row vector $\mathbf{x}_i$ denotes the node attribute of v_i. Thus, a graph can also be represented as $\mathcal{G} = (\mathbf{A}, \mathbf{X})$.

5.3.1 Node Classification with GNNs

We consider the task of node classification. Given an input graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$ (or $\mathcal{G} = (\mathbf{A}, \mathbf{X})$), a prediction model $f_\theta(\cdot)$ assigns a label c from a pre-defined set $\mathcal{C}$ to

each node $v \in \mathcal{V}$. In other words, we have $f_\theta(\mathcal{G}(v)) = c$, where $\mathcal{G}(v)$ denotes the computation graph of node v. Typically, $f_\theta(\cdot)$ consists of two components: 1) a GNN model used to learn node representations, and 2) a multi-layer perceptron (MLP) classifier that assigns labels. More specifically, given a set of training nodes and their labels $\{(v_1, y_1), ..., (v_n, y_n)\}$ from graph $\mathcal{G}$, the objective function of a GNN classifier can be defined as

$$\min_\theta \mathcal{L}_{\text{GNN}}(f_\theta) := \sum_{i=1}^{n} l(f_\theta(\mathcal{G}(v_i), y_i)) \tag{5.1}$$

$$= \sum_{i=1}^{n} \sum_{c}^{C} \mathbb{I}(y_i = c) \ln(f_\theta(\mathcal{G}(v_i), y_i)^{(c)}), \tag{5.2}$$

where θ is the set of learnable parameters for $f_\theta(\cdot)$, $l(\cdot)$ is the cross-entropy loss function, and $\mathbb{I}(\cdot)$ denotes the Kronecker function. Further, $f_\theta(\mathcal{G}(v_i), y_i)^{(c)}$ indicates the c-th softmax output for the prediction of node v_i, and C is the cardinality of label set $\mathcal{C}$.

Now we define the design detail of the GNN classifier $f_\theta(\cdot)$. For the first component, for simplicity, we employ a 2-layer GCN model with parameters $(\mathbf{W}_1, \mathbf{W}_2)$, i.e.,

$$GCN(\mathcal{G}) = GCN(\mathbf{A}, \mathbf{X}) := \tilde{\mathbf{A}}\sigma(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}_1)\mathbf{W}_2, \tag{5.3}$$

where $\tilde{\mathbf{A}} = \tilde{\mathbf{D}}^{-1/2}(\mathbf{A} + \mathbf{I})\tilde{\mathbf{D}}^{-1/2}$ (with $\mathbf{I}$ the identity matrix and $\tilde{\mathbf{D}}$ the diagonal matrix of $\mathbf{A} + \mathbf{I}$). Further, $\sigma(\cdot)$ is an activation function such as ReLU. For the second component, we simply use a softmax$(\cdot)$ function rather than a MLP. Therefore, the full GNN classifier f_θ is defined as

$$f_\theta(\mathbf{A}, \mathbf{X}) := \text{softmax}(\tilde{\mathbf{A}}\sigma(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}_1)\mathbf{W}_2). \tag{5.4}$$

5.3.2 GNN Explainer

We consider post-hoc GNN explanation methods that provide instance-level explanations: given a graph $\mathcal{G}$, a target node v, and prediction $f_\theta(\mathcal{G}(v))$, the explainer $g_\lambda(v, f(\mathcal{G}(v)))$ aims to 'explain' this particular prediction. Following common explanation methods, we assume that explanations are given in the form of an *explanatory subgraph*, namely $\mathcal{G}_s(v) = g(v, f(\mathcal{G}(v)))$, consisting of a small set of important edges and their associated nodes, where the edge importance is measured by a so-called *edge importance score*.

In this paper, we take PGExplainer [199] as an example to attack for three reasons. First, [230] pointed out that GNN explanations providing edge importance are preferable to explanations providing node importance, due to the fact that edges have more fine-grained information than nodes. Second, [231] empirically showed that PGExplainer generates the least unstable explanations, exhibiting a 35.35% reduction in explanation instability relative to the average instability of other GNN explainers. Third, the objective function of PGExplainer is differentiable. For completeness, we briefly revisit the goal of PGExplainer. For brevity, we denote $\mathcal{G}(v)$ as $\mathcal{G}$ (and $\mathcal{G}_s(v)$ as $\mathcal{G}_s$ analogously) throughout the remainder of the paper.

PGExplainer. Given a graph $\mathcal{G}$ and a target node v, PGExplainer provides an *explanatory subgraph* $\mathcal{G}_s(v) = (\mathbf{A}_s, \mathbf{X}_s)$ to explain why $f_\theta(\mathcal{G}(v)) = c$, where $\mathbf{A}_s$ is the subgraph structure and $\mathbf{X}_s$ is the node features associated with nodes residing in the subgraph. Specifically, PGExplainer maximizes the mutual information between the GNN's prediction $\mathbf{y}$ and the explanatory subgraph $\mathcal{G}_s$, i.e.,

$$\max_{\mathcal{G}_s} \text{MI}(\mathbf{y}, \mathcal{G}_s) := H(\mathbf{y}) - H(\mathbf{y}|\mathcal{G}_s), \tag{5.5}$$

where $H(\mathbf{y})$ is the entropy of $f_\theta(\mathcal{G})$, and $H(\mathbf{y}|\mathcal{G}_s)$ is the conditional entropy of the GNN's prediction on the explanatory graph, namely $f_\theta(\mathcal{G}_s)$.

5.4 GXAttack: GNN Explanation Adversarial Attacks

Given an input graph $\mathcal{G}$, a GNN classifier $f_\theta(\cdot)$, and a GNN explainer $g_\lambda(\cdot)$, the original prediction for the target node v is $f_\theta(\mathcal{G}(v))$ and its corresponding original *explanatory subgraph* is given by $\mathcal{G}_s(v) = g_\lambda(f_\theta(v, \mathcal{G}(v)))$. From an attacker's perspective, a desirable manipulated graph $\mathcal{G}_{adv} := \hat{\mathcal{G}} = \phi(\mathcal{G})$ for attacking node v should have the following properties, where $\phi(\cdot)$ is an attack function that perturbs its input, i.e., it removes and/or adds a small set of edges:

1. The norm of the perturbation is so small that the change is considered imperceptible, i.e., $\mathcal{G} \approx \hat{\mathcal{G}}$. This can be formalized as $\|\mathbf{A} - \hat{\mathbf{A}}\| < B$, where B is the perturbation budget;

2. As GNNs can be highly sensitive to input perturbations, we require that the output of the GNN classifier remains approximately the same, i.e., perturbations must be *prediction-preserving*. Formally, we have $f(\mathcal{G}(v)) = f(\hat{\mathcal{G}}(v))$. We hypothesize

that perturbations that are both small and prediction-preserving are very likely to preserve the essence of the learned predictive models;

3. The explanatory subgraph for node v on original graph $\mathcal{G}_s(v) = g_\lambda(f_\theta(v, \mathcal{G}(v)))$ and that on perturbed graph $\hat{\mathcal{G}}_s(v) = g_\lambda(f_\theta(\hat{\mathcal{G}}(v)))$ are substantially different: $\mathcal{G}_s(v) \neq \hat{\mathcal{G}}_s(v)$. In particular, we aim to make $\hat{\mathbf{A}}_s$ (i.e., the adjacency matrix of explanatory subgraph on perturbed graph) strongly different from $\mathbf{A}_s$ (i.e., the adjacency matrix of explanatory subgraph on original graph).

5.4.1 Attack Objectives

Generic Objective. This paper focuses on adversarial structure attacks on post-hoc GNN explanation for node classification:

$$\max_{\hat{\mathbf{A}} \text{ s.t. } \|\mathbf{A}-\hat{\mathbf{A}}\|_0 < B} \mathcal{L}(g_\lambda(f_\theta(\hat{\mathbf{A}}, \mathbf{X}))), \tag{5.6}$$

with loss function $\mathcal{L}$, perturbation budget B, and *fixed* parameters θ and λ. The GNN predictor $f_\theta(\cdot)$ is applied to a graph $\mathcal{G} = \{\mathbf{A}, \mathbf{X}\}$ to obtain node label predictions $f_\theta(\mathbf{A}, \mathbf{X})$. Next, GNN explainer $g_\lambda(\cdot)$ is applied to GNN predictions $f_\theta(\mathbf{A}, \mathbf{X})$ to generate post-hoc explanations $g_\lambda(f_\theta(\mathbf{A}, \mathbf{X}))$. We focus on *evasion* (test time) attacks and *local* attacks on a single node with a budget B. Moreover, we study white box attacks as they represent the "worst-case noise" of a model. In other words, we assume perfect knowledge about the graph, labels, prediction model, and post-hoc explanation model.

Loss Function. We should instantiate the loss function $\mathcal{L}$ such that it pertains to the desirable properties of manipulations discussed before. More specifically, it is defined as

$$\mathcal{L} := \mathrm{MI}(f_\theta(\mathbf{A}, \mathbf{X}), f_\theta(\hat{\mathbf{A}}, \mathbf{X})) - \beta \cdot \mathrm{MI}(g_\lambda(f_\theta(\mathbf{A}, \mathbf{X})), g_\lambda(f_\theta(\hat{\mathbf{A}}, \mathbf{X}))). \tag{5.7}$$

The first term ensures that the GNN predictions before and after perturbations are approximately the same. Further, the second term enforces that the manipulated explanation is different from the original explanatory subgraph. Their relative importance is weighted by hyperparameter $\beta \in \mathbb{R}_+$.

Final Objective. We directly model the edge perturbation matrix $\mathbf{P} \in \{0, 1\}^{n \times n}$, where we flip $\mathbf{A}_{ij}$ (namely $0 \to 1$ or $1 \to 0$) if $\mathbf{P}_{ij} = 1$, and keep $\mathbf{A}_{ij}$ intact otherwise. On this basis, Eq 5.6 can be rewritten as

$$\max_{\substack{\mathbf{P}\in\{0,1\}^{n\times n}\\ \text{s.t. } \sum \mathbf{P}_{ij}<B}} \mathrm{MI}(f_\theta(\mathbf{A},\mathbf{X}), f_\theta(\mathbf{A}\oplus\mathbf{P},\mathbf{X})) - \beta\cdot \mathrm{MI}(g_\lambda(f_\theta(\mathbf{A},\mathbf{X})), g_\lambda(f_\theta(\mathbf{A}\oplus\mathbf{P},\mathbf{X}))), \tag{5.8}$$

where $\oplus$ means *element-wise exclusive or* operation, and B is the edge flipping budget.

5.4.2 Relaxations and Loss Definitions

Unfortunately, directly minimizing $\mathcal{L}$ w.r.t. $\mathbf{P}$ is intractable since there are $\mathcal{O}(2^{n\times n})$ candidates for $\mathbf{P}$. We therefore approximate this discrete optimization problem, i.e., we make some relaxations to make $\mathcal{L}$ (approximately) differentiable w.r.t. $\mathbf{P}$ so that we can perform gradient-based attacks.

Relaxations. We assume that edge flipping graph $\mathcal{M}$ corresponding to $\mathbf{P}\in\{0,1\}^{n\times n}$ (namely using $\mathbf{P}$ as the adjacency matrix) is a Gilbert random graph. In other words, edge occurrences are conditionally independent from each other. As a result, with slight abuse of notations, the probability of graph $\mathcal{M}$ can be factorized as $P(\mathcal{M}) = \prod_{1\le i,j\le n} P(m_{ij})$, dubbed $\tilde{\mathbf{P}}$. Next, we instantiate $P(m_{ij})$ using a Bernoulli distribution, namely $P(m_{ij}) \sim \text{Bernoulli}(\tilde{\mathbf{P}}_{ij})$ with $P(m_{ij}=1) = \tilde{\mathbf{P}}_{ij}$. As a result, Eq 5.8 is equivalent to

$$\max_{\mathbf{P}} \mathcal{L}(\mathbf{P}) = \max_{\mathcal{M}} \mathbb{E}_{\mathcal{M}}[\mathcal{L}(\mathcal{M})] \approx \max_{\tilde{\mathbf{P}}} \mathbb{E}_{\mathcal{M}\sim\tilde{\mathbf{P}}}[\mathcal{L}(\mathcal{M})]. \tag{5.9}$$

In this way, we relax the $\mathbf{P}\in\{0,1\}^{n\times n}$ from binary variables to continuous variables $\tilde{\mathbf{P}}\in[0,1]^{n\times n}$. Hence, we can utilize gradient-based methods to optimize the objective function. At the last step of the attack, we then discretize the result (see Chapter 5.4.3).

Loss Term 1. We approximate the first term in $\mathcal{L}$ as follows:

$$\max_{\tilde{\mathbf{P}}} \mathrm{MI}(f_\theta(\mathbf{A},\mathbf{X}), f_\theta(\mathbf{A}\oplus\tilde{\mathbf{P}},\mathbf{X})) \tag{5.10}$$

$$:= \max_{\tilde{\mathbf{P}}} H(f_\theta(\mathbf{A},\mathbf{X})) - H(f_\theta(\mathbf{A},\mathbf{X})|f_\theta(\mathbf{A}\oplus\tilde{\mathbf{P}},\mathbf{X})) \tag{5.11}$$

$$\propto \max_{\tilde{\mathbf{P}}} \; - H(f_\theta(\mathbf{A},\mathbf{X})|f_\theta(\mathbf{A}\oplus\tilde{\mathbf{P}},\mathbf{X})) \tag{5.12}$$

$$\approx \max_{\tilde{\mathbf{P}}} \sum_{c=1}^{C} P_\theta(y=c|\mathcal{G}) \ln(P_\theta(y=c|\hat{\mathcal{G}})). \tag{5.13}$$

Eq 5.11 holds by definition of mutual information; Eq 5.12 holds as $\theta, \mathbf{A}, \mathbf{X}$ are fixed; Eq 5.13 holds when we replace conditional entropy $H(f_\theta(\mathbf{A}, \mathbf{X})|f_\theta(\mathbf{A} \oplus \tilde{\mathbf{P}}, \mathbf{X}))$ with cross entropy $H(f_\theta(\mathbf{A}, \mathbf{X}), f_\theta(\mathbf{A} \oplus \tilde{\mathbf{P}}, \mathbf{X}))$ for practical optimization reason. Moreover, we define $\mathbf{A}_{ij} \oplus \tilde{\mathbf{P}}_{ij} = \mathbf{A}_{ij} + \tilde{\mathbf{P}}_{ij}$ if $\mathbf{A}_{ij} = 0$, and $\mathbf{A}_{ij} \oplus \tilde{\mathbf{P}}_{ij} = \mathbf{A}_{ij} - \tilde{\mathbf{P}}_{ij}$ otherwise.

Loss Term 2. Next, we approximate the second term in $\mathcal{L}$ as follows:

$$\min_{\tilde{\mathbf{P}}} \text{MI}(g_\lambda(f_\theta(\mathbf{A}, \mathbf{X})), g_\lambda(f_\theta(\mathbf{A} \oplus \tilde{\mathbf{P}}, \mathbf{X}))) \tag{5.14}$$

$$\propto \max_{\tilde{\mathbf{P}}} \text{Dist}(\mathbf{M}_s, \hat{\mathbf{M}}_s) \tag{5.15}$$

$$=: \max_{\tilde{\mathbf{P}}} \left(1 - \frac{\text{vec}(\mathbf{M}_{pr}) \cdot \text{vec}(\hat{\mathbf{M}}_{pr})}{\|\mathbf{M}_{pr}\|_F \|\hat{\mathbf{M}}_{pr}\|_F}\right), \tag{5.16}$$

where $\mathbf{M}_s$ and $\hat{\mathbf{M}}_s$ are the explanatory subgraphs on original graph and perturbed graph, respectively; $\mathbf{M}_{pr}$ and $\hat{\mathbf{M}}_{pr}$ are the predicted explanations (edge importance matrices) on original graph and perturbed graph, respectively. In other words, the explanatory subgraph is represented as an edge importance matrix in this study. Eq 5.15 holds by definition; Eq 5.16 holds when we define the *Dist* function as the *cosine distance metric*, where $\text{vec}(\cdot)$ flattens a matrix to a vector, and $\|\cdot\|_F$ represents the Frobenius Norm.

Final Relaxed Objective. As a result, optimizing Eq 5.8 amounts to optimizing

$$\max_{\substack{\tilde{\mathbf{P}} \in [0,1]^{n \times n} \\ \text{s.t. } \sum \tilde{\mathbf{P}}_{ij} < B}} \sum_{c=1}^{C} P_\theta(y = c|\mathcal{G}) \ln(P_\theta(y = c|\hat{\mathcal{G}}) + \beta \cdot \left(1 - \frac{\text{vec}(\mathbf{M}_{pr}) \cdot \text{vec}(\hat{\mathbf{M}}_{pr})}{\|\mathbf{M}_{pr}\|_F \|\hat{\mathbf{M}}_{pr}\|_F}\right). \tag{5.17}$$

As a result, $\mathcal{L}(\tilde{\mathbf{P}})$ (i.e., Eq 5.17) is differentiable w.r.t. $\tilde{\mathbf{P}}$.

5.4.3 Optimization

To guarantee that $\tilde{\mathbf{P}}$ parametrizes a valid distribution and that we obey the budget in expectation, we employ Projected Gradient Ascent Algorithm (based on [227]) to optimize objective function (5.17), and the details are given in Algorithm 4. To go back from the continuous $\tilde{\mathbf{P}}_{ij}$ to discrete $\mathbf{P}_{ij}$, we perform Bernoulli sampling with $\mathbf{P}_{ij} \sim \text{Bernoulli}(\tilde{\mathbf{P}}_{ij})$ after optimization (Line 10). We note that the Projected Randomized Block Coordinate Descent (PR-BCD) [228] can perform this optimization more efficiently, but do not compare both methods due to the low scalability of the

evaluated explainers.

Algorithm 4 Pseudo-code for GXAttack

Input: graph $\mathcal{G} = (\mathbf{A}, \mathbf{X})$, predictor $f_\theta(\cdot)$, explainer $g_\lambda(\cdot)$, loss $\mathcal{L}$, budget B, #epochs T, learning rate α_t
Output: perturbed adjacency matrix $\hat{\mathbf{A}}$

$\mathbf{y} \leftarrow f_\theta(\mathbf{A}, \mathbf{X})$ ▷ Get predictions on original graph
$\mathcal{G}_s \leftarrow g_\lambda(f_\theta(\mathbf{A}, \mathbf{X}))$ ▷ Get explanations for the predictions on original graph
$\tilde{\mathbf{P}} \leftarrow \mathbf{0} \in \mathbb{R}^{n \times n}$ ▷ Initialize zeros for continuous perturbation variables
for $t \in \{1, 2, ..., T\}$ **do**
  $\hat{\mathbf{y}} \leftarrow f_\theta(\mathbf{A} \oplus \tilde{\mathbf{P}}_{t-1}, \mathbf{X})$ ▷ Get predictions on perturbed graph
  $\hat{\mathcal{G}}_s \leftarrow g_\lambda(f_\theta(\mathbf{A} \oplus \tilde{\mathbf{P}}_{t-1}, \mathbf{X}))$ ▷ Get explanations for the predictions on perturbed graph
  $\tilde{\mathbf{P}}_t \leftarrow \tilde{\mathbf{P}}_{t-1} + \alpha_t \cdot \nabla_{\tilde{\mathbf{P}}_{t-1}} \mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}, \hat{\mathcal{G}}_s, \mathcal{G}_s)$ ▷ Update gradients
  $\tilde{\mathbf{P}}_t \leftarrow \prod_{\mathbb{E}[\text{Bernoulli}(\tilde{\mathbf{P}}_t) \leq B]}(\tilde{\mathbf{P}}_t)$ ▷ Project gradients, see Appendix 5.7 for more detail
end for
$\mathbf{P} \sim \text{Bernoulli}(\tilde{\mathbf{P}})$ s.t. $\sum \mathbf{P}_{ij} \leq B$ ▷ Sample binary perturbation variables
Return $\mathbf{A} \oplus \mathbf{P}$ ▷ Which is defined as $\mathbf{A}_{ij} + \mathbf{P}_{ij}(1 - 2\mathbf{A}_{ij})$

5.4.4 Complexity Analysis

We first discuss the dense case here and then make a discussion of GNNs/explainers implemented with sparse matrices. The time complexity is as follows:

1. **Initial Operations**: $O(f_\theta + g_\lambda)$, typically $O(n^2)$ for a dense GNN and explainer (sometimes even $O(n^3)$).

2. **Each Epoch**:

   - Prediction and explanation on perturbed graph: $O(2T \cdot (f_\theta + g_\lambda))$, where T is the number of epochs.
   - Gradient computation and update: Dependent on the efficiency of backpropagation through the GNN, but generally $O(T \cdot n^2)$ for gradient computation and a similar order for updates.

3. **Sampling Step**: $O(n^2)$.

Meanwhile, the space complexity is as follows:

1. **Space for Storing Matrices**: Storing $\mathbf{A}$, $\mathbf{X}$, $\tilde{\mathbf{P}}$ each requires $O(n^2)$ space.

2. **Additional Space for Operations**: Space required for intermediate gradients and outputs during predictions, typically $O(n^2)$.

Since the explainer uses dense matrices, we also choose an attack operating on all n^2 edges. Thus, the total complexity $O((T+1) \cdot (f_\theta + g_\lambda) + n^2)$ since we have T attack steps plus the initial predictions as well as explanations and, last, we sample the final perturbations. Assuming that $O(f_\theta) = O(g_\lambda) = O(n^2 d + nd^2)$ and $d \ll n$, this yields a total time complexity of $O(n^2)$ with number of attack iterations $T \ll n$, and number of nodes n. This is the same asymptotic complexity as the dense GNN f_θ and explainer g_λ.

Assuming GNN and explainer are in $O(f_\theta) = O(g_\lambda) = O(Ed+nd^2)$ with number of edges E, a dense attack would dominate the overall complexity ($O(n^2)$). An immediate remedy would yield PRBCD [228] since its complexity is $O(B+E)$ with edge flipping budget B and all additional components (e.g., loss function) could be implemented in $O(E)$ with sparse matrices. This yields an overall time and space complexity of $O(E)$, assuming $B \ll E$ and hidden dimensions $d \ll n$, which is the same asymptotic complexity as the sparse GNN f_θ and explainer g_λ.

5.5 Experimental Setup

We aim to answer the following research questions: (**RQ1**) How effective are the attacks generated by GXAttack on PGExplainer? (**RQ2**) How effective are the attacks generated by GXAttack when compared to trivial attackers that use random perturbations? and (**RQ3**) Can the attacks generated by GXAttack (optimized for PGExplainer) transfer to other GNN explanation methods?

5.5.1 Datasets

To answer these questions, we need to evaluate the GNN explanation quality before and after the attacks. However, as pointed out by [231], evaluating the quality of post-hoc GNN explanations is challenging. This is because existing benchmark graph datasets lack ground-truth explanations or their explanations are not reliable. To mitigate this, they proposed a synthetic graph data generator *ShapeGGen*. It can generate a variety of graph datasets with ground-truth explanations, where graph size, node degree distributions, the ground-truth explanation sizes, and more can be varied. As a result, *ShapeGGen* allows us to mimic graph data in various real-world applications. More importantly, *ShapeGGen* can ensure that the generated data and its ground-truth explanations do not suffer from GNN explanation evaluation pitfalls [230], namely trivial explanations, redundant explanations, and weak GNN predictors.

Table 5.1: Summary of synthetic datasets with ground-truth explanations. #Nodes and #Edges represent the number of expected nodes and edges, respectively. We do not consider larger datasets for two reasons: 1) explanation accuracy will be very low (< 0.3) for most explainers, and it is not meaningful to attack low accuracy explainers; 2) computation will be slow for the attacker (and the explainer).

Dataset	Syn1	Syn2	Syn3	Syn4	Syn5	Syn6	Syn7
Shape	House	House	House	Circle	House	House	House
#Subgraphs	10	50	50	50	50	50	100
Subgraph Size	6	6	10	10	10	10	10
P(Connection)	0.3	0.06	0.06	0.06	0.12	0.20	0.06
#Classes	2	2	2	2	2	2	2
#Nodes	59	299	521	511	489	492	1018
#Edges	164	970	1432	1314	1664	1898	3374
Average Degree	2.8	3.2	2.7	2.6	3.4	3.9	3.3
Max Budget	10	15	15	15	15	15	15

This makes *ShapeGGen* very suitable for studying the limitations of GNN explainers. We consider the synthetic datasets in Table 5.1.

Why not real-world datasets? We do not consider real-world datasets for two key reasons: 1) synthetic datasets with ground truth are *de-facto* standards for evaluating post-hoc GNN explainers (for node classification), including GNNExplainer, PGExplainer, PGM-Explainer, SubgraphX, CF-GNNExplainer, RCExplainer, etc. 2) without ground-truth information we can only compute some metrics, such as cosine similarity of explanation before and after attack (which will be defined later) to quantify attack performance. However, as shown in Table 5.2, the cosine similarity and the real attack performance (in terms of ΔGEA, which will be defined later) are not necessarily related. Therefore, considering such metrics may not be meaningful.

5.5.2 Metrics and Baselines

Graph Explanation Accuracy (GEA). We employ the *graph explanation accuracy* proposed in [231] to quantify the quality of generated explanations when the ground-truth explanations are available. For simplicity, we assume that each edge is binary coded as '0' (non-contributory) or '1' (contributory) to model predictions in both ground-truth (provided by *ShapeGGen*) and predicted explanation masks (provided by the GNN explainer). Specifically, *graph explanation accuracy* measures the correctness of generated explanations by computing the Jaccard Index between ground-truth

explanation $\mathbf{M}_{gt}$ and predicted explanation $\mathbf{M}_{pr}$ as

$$\mathrm{JAC}(\mathbf{M}_{gt}, \mathbf{M}_{pr}) = \frac{\mathrm{TP}(\mathbf{M}_{gt}, \mathbf{M}_{pr})}{\mathrm{TP}(\mathbf{M}_{gt}, \mathbf{M}_{pr}) + \mathrm{FP}(\mathbf{M}_{gt}, \mathbf{M}_{pr}) + \mathrm{FN}(\mathbf{M}_{gt}, \mathbf{M}_{pr})}, \tag{5.18}$$

where TP, FP, FN denote True Positives, False Positives, and False Negatives, respectively. Higher values indicate larger consistency between the generated explanation and the ground-truth explanation. As a result, the difference in this metric before and after perturbation can be used to measure the performance of attacks:

$$\Delta\mathrm{GEA} =: \mathrm{JAC}(\mathbf{M}_{gt}, \mathbf{M}_{pr}) - \mathrm{JAC}(\mathbf{M}_{gt}, \hat{\mathbf{M}}_{pr}), \tag{5.19}$$

where higher values indicates more successful attacks.

Cosine Similarity. To quantify the performance of attacks when ground-truth explanations are **not** available, we use cosine similarity to measure the stability of graph explanations:

$$\mathrm{Sim}_{\cos}(\mathbf{M}_{pr}, \hat{\mathbf{M}}_{pr}) = \frac{\mathrm{vec}(\mathbf{M}_{pr}) \cdot \mathrm{vec}(\hat{\mathbf{M}}_{pr})}{\|\mathbf{M}_{pr}\|_F \|\hat{\mathbf{M}}_{pr}\|_F}, \tag{5.20}$$

where $\mathrm{vec}(\cdot)$ flattens its input as a vector, and $\|\cdot\|_F$ represents the Frobenius Norm. Higher values of $\mathrm{Sim}_{\cos}$ (within $[-1, 1]$) indicate higher graph explanation stability, and thus less successful attacks.

Prediction Change. To measure the impact of edge perturbations on GNN predictions, we consider two metrics as follows: 1) **ΔLabel**: the ratio of nodes of which the predicted label has changed after attacks; and 2) **ΔProb**: the average absolute change of predicted probability (of the original predicted class) of all nodes, which is formally defines as

$$\Delta\mathrm{Prob} = \frac{\sum_{i=1}^{N} |P(y_i = y_i^*) - \hat{P}(y_i = y_i^*)|}{N},$$

where $P(y_i = y_i^*)$ is the prediction probability of node v_i being classified as y_i^* (the original predicted class with the highest probability) on the original graph. Moreover, $\hat{P}(y_i = y_i^*)$ is the prediction probability of node v_i being classified as y_i^* (the original predicted class with the highest probability on the original graph) on the perturbed graph.

Attacker Baselines. Given the novelty of our problem setting, we only consider two trivial baselines for edge perturbations: 1) **random flipping**: we randomly flip

B_1 edges; and 2) **random rewiring**: we randomly rewire B_2 edges within k-hop neighbors of the target node.

5.5.3 Attack Transferability Settings

We consider attack transferability on the following five types of GNN explainers: 1) **Gradient**-based: GradCAM [197], Integrated Gradients (IG) [214]; 2) **Perturbation**-based: GNNExplainer [195], SubgraphX [201]; 3) **Surrogate**-based: PGMExplainer [196]; 4) **Decomposition-based methods**: GNN-LRP [215] and 5) **Random** Explainer that randomly generates random important edges.

5.5.4 Training Protocols and Compute Resources

GNN Predictor Training Protocols. To avoid pitfalls caused by weak GNN predictor [230], we should train the GNN predictor close to its maximum possible performance. When training the GNN predictor, we employ an Adam optimizer where the learning rate is 1e-2, weight decay is 1e-5, the number of epochs is 300, and the hidden dimension is 32.

GNN Explainer Training Protocols. We followed the authors' recommendations for setting the hyperparameters of GNN explanation methods. We select top-k ($k = 25\%$) important edges to generate explanations for all GNN explainers. Particularly, when training the PGExplainer, the number of training epochs is 10.

GNN Explanation Attacker Training Protocols. We set the trade-off hyperparameter in the final loss as 0.1 such that the magnitudes of the two loss terms are approximately the same. Besides, the learning rate in gradient ascent is set to 0.1. Importantly, we set the maximal training epochs as 100 due to computation time reason, while further increasing this number can largely improve the attack performance (in terms of ΔGEA). Moreover, we set the perturbation budget to 15 at maximum for synthetic data (note that we attempt to employ a hard minimum budget of 5 with 2000 maximum trials of Bernoulli sampling). We perform sensitivity analysis regarding these hyperparameters, and the results and analysis will be discussed in Chapter 5.6.4.

Explanation Accuracy Computation. The $get\_acc(\cdot)$ function of GraphXAI considers all non-zero edges as predicted positives (namely with binary code '1') by default when computing JAC. Following [231], we set the top 25% (of edges with the highest edge importance scores) as 1 and the rest as zeros in our evaluations when computing both GEA and cosine similarity.

Software and Hardware. The evaluated GNN explainers are implemented mainly based on GraphXAI [1] [231]. Algorithms are implemented in Python 3.8 (using PyTorch [109] and PyTorch Geometric [110] libraries when applicable) and ran on internal clusters with AMD 128 EPYC7702 CPUs (with 512GB RAM). All experiments can be finished within 7 days if using 10 such machine instances. All code and datasets are available on GitHub[2].

5.6 Experimental Results and Analysis

Before answering the research questions, we state three presumptions and check them empirically in Chapter 5.6.1. Next, we answer RQ1 and RQ2 in Chapter 5.6.2, and RQ3 in Chapter 5.6.3. Following this, we perform sensitivity analysis in Chapter 5.6.4 and property analysis in Chapter 5.6.5.

5.6.1 Experiments to Check Presumptions

Presumption 1. *The prediction correctness of a GNN predictor and explanation accuracy of a GNN explainer are* ***not*** *related.* For each dataset, we first get the set of correctly predicted nodes and the set of wrongly predicted nodes; then we compare the distribution of their explanation accuracy. As shown in Figure 5.6 in Appendix, there is no notable difference. In other words, we do **not** observe that wrongly labelled nodes tend to have lower explanation accuracy despite [230] claim that it is unfair to use wrongly labelled nodes to evaluate GNN explanations (as they may not use the ground-truth explanations to make predictions and thus tend to have lower explanation accuracy).

Presumption 2. *The prediction confidence of a GNN predictor and explanation accuracy of a GNN explainer are related,* where prediction confidence is defined as the probability assigned to the predicted class. For each dataset, we first divide the nodes into five groups based on their prediction confidence; then we compare the distribution of their explanation accuracy. Table 5.4 in Appendix show that group 'G9' (i.e., the set of nodes with prediction confidence in $[0.9, 1.0]$) indeed tends to have lower explanation accuracy on the original graph than other groups.

Presumption 3. *The prediction confidence of a GNN predictor and attack performance of a GNN explanation attacker are related.* To check this presumption, we

[1] https://github.com/mims-harvard/GraphXAI
[2] `https://github.com/ZhongLIFR/GXAttack`

Table 5.2: Results on all synthetic datasets (average ± standard deviations across 5 trials). **O. GEA** means explanation accuracy on original graph and **P. GEA** indicates explanation accuracy on perturbed graph. Moreover, **ΔGEA** is the explanation accuracy change after perturbation, $\mathbf{Sim}_{\cos}$ is the cosine similarity between explanations (before and after perturbations), and **#Pert.** represents the number of flipped edges. **ΔLabel** is the average change of predicted labels, while **ΔProb** denotes the average absolute change of predicted probability (of the original predicted class). "GXAttack", "Rnd. Flipping", and "Rnd. Rewiring" correspond to our attack method, random flipping, and random rewiring, respectively. ' ↑' indicates larger values are preferred while ' ↓' means the opposite.

	Dataset	Syn1	Syn2	Syn3	Syn4	Syn5	Syn6	Syn7
	O. GEA	$0.678_{\pm 0.046}$	$0.641_{\pm 0.022}$	$0.494_{\pm 0.012}$	$0.469_{\pm 0.006}$	$0.439_{\pm 0.014}$	$0.306_{\pm 0.025}$	$0.429_{\pm 0.013}$
GXAttack	**P. GEA**(↓)	$0.497_{\pm 0.041}$	$0.375_{\pm 0.011}$	$0.308_{\pm 0.010}$	$0.312_{\pm 0.003}$	$0.290_{\pm 0.004}$	$0.217_{\pm 0.013}$	$0.287_{\pm 0.014}$
GXAttack	**ΔGEA** (↑)	$0.181_{\pm 0.020}$	$0.267_{\pm 0.014}$	$0.186_{\pm 0.013}$	$0.157_{\pm 0.010}$	$0.149_{\pm 0.015}$	$0.088_{\pm 0.014}$	$0.142_{\pm 0.009}$
GXAttack	$\mathbf{Sim}_{\cos}$(↓)	$0.921_{\pm 0.007}$	$0.975_{\pm 0.001}$	$0.980_{\pm 0.004}$	$0.976_{\pm 0.003}$	$0.978_{\pm 0.002}$	$0.988_{\pm 0.001}$	$0.971_{\pm 0.040}$
GXAttack	**#Pert.** (↓)	$3.300_{\pm 0.255}$	$5.620_{\pm 0.045}$	$6.680_{\pm 0.148}$	$5.900_{\pm 0.122}$	$6.560_{\pm 0.055}$	$5.940_{\pm 0.167}$	$7.020_{\pm 0.130}$
GXAttack	**ΔLabel**(↓)	$0.000_{\pm 0.000}$	$0.002_{\pm 0.003}$	$0.001_{\pm 0.001}$	$0.001_{\pm 0.001}$	$0.001_{\pm 0.001}$	$0.000_{\pm 0.001}$	$0.001_{\pm 0.001}$
GXAttack	**ΔProb**(↓)	$0.106_{\pm 0.008}$	$0.140_{\pm 0.004}$	$0.163_{\pm 0.003}$	$0.150_{\pm 0.002}$	$0.143_{\pm 0.004}$	$0.089_{\pm 0.003}$	$0.147_{\pm 0.001}$
Rnd. Flipping	**P. GEA**(↓)	$0.640_{\pm 0.056}$	$0.638_{\pm 0.023}$	$0.493_{\pm 0.012}$	$0.467_{\pm 0.009}$	$0.438_{\pm 0.013}$	$0.306_{\pm 0.024}$	$0.429_{\pm 0.013}$
Rnd. Flipping	**ΔGEA**(↑)	$0.038_{\pm 0.029}$	$0.001_{\pm 0.005}$	$0.001_{\pm 0.002}$	$0.002_{\pm 0.003}$	$0.001_{\pm 0.001}$	$0.000_{\pm 0.005}$	$0.000_{\pm 0.001}$
Rnd. Flipping	$\mathbf{Sim}_{\cos}$(↓)	$0.800_{\pm 0.006}$	$0.945_{\pm 0.004}$	$0.964_{\pm 0.006}$	$0.943_{\pm 0.005}$	$0.966_{\pm 0.001}$	$0.978_{\pm 0.001}$	$0.981_{\pm 0.001}$
Rnd. Flipping	**#Pert.**(↓)	$15_{\pm 0.000}$	$15_{\pm 0.000}$	$15_{\pm 0.000}$	$15_{\pm 0.000}$	$15_{\pm 0.000}$	$15_{\pm 0.000}$	$15_{\pm 0.000}$
Rnd. Flipping	**ΔLabel**(↓)	$0.051_{\pm 0.012}$	$0.011_{\pm 0.007}$	$0.012_{\pm 0.004}$	$0.009_{\pm 0.006}$	$0.004_{\pm 0.002}$	$0.002_{\pm 0.002}$	$0.004_{\pm 0.002}$
Rnd. Flipping	**ΔProb**(↓)	$0.048_{\pm 0.003}$	$0.009_{\pm 0.005}$	$0.006_{\pm 0.001}$	$0.008_{\pm 0.001}$	$0.006_{\pm 0.001}$	$0.003_{\pm 0.001}$	$0.003_{\pm 0.001}$
Rnd. Rewiring	**P. GEA**(↓)	$0.149_{\pm 0.012}$	$0.045_{\pm 0.005}$	$0.069_{\pm 0.004}$	$0.067_{\pm 0.004}$	$0.020_{\pm 0.002}$	$0.013_{\pm 0.001}$	$0.020_{\pm 0.002}$
Rnd. Rewiring	**ΔGEA**(↑)	$0.530_{\pm 0.043}$	$0.596_{\pm 0.022}$	$0.425_{\pm 0.010}$	$0.402_{\pm 0.003}$	$0.419_{\pm 0.014}$	$0.293_{\pm 0.025}$	$0.409_{\pm 0.012}$
Rnd. Rewiring	$\mathbf{Sim}_{\cos}$(↓)	$0.723_{\pm 0.031}$	$0.941_{\pm 0.006}$	$0.975_{\pm 0.006}$	$0.959_{\pm 0.008}$	$0.951_{\pm 0.004}$	$0.970_{\pm 0.002}$	$0.967_{\pm 0.009}$
Rnd. Rewiring	**#Pert.**(↓)	$16_{\pm 0.000}$	$16_{\pm 0.000}$	$16_{\pm 0.000}$	$16_{\pm 0.000}$	$16_{\pm 0.000}$	$16_{\pm 0.000}$	$16_{\pm 0.000}$
Rnd. Rewiring	**ΔLabel**(↓)	$0.274_{\pm 0.030}$	$0.199_{\pm 0.022}$	$0.202_{\pm 0.004}$	$0.184_{\pm 0.011}$	$0.214_{\pm 0.007}$	$0.159_{\pm 0.013}$	$0.274_{\pm 0.014}$
Rnd. Rewiring	**ΔProb**(↓)	$0.234_{\pm 0.016}$	$0.210_{\pm 0.013}$	$0.198_{\pm 0.007}$	$0.200_{\pm 0.003}$	$0.221_{\pm 0.005}$	$0.174_{\pm 0.013}$	$0.259_{\pm 0.005}$

first divide the nodes into five groups based on their prediction confidence; then we compare the distribution of their attack performance. Table 5.4 in Appendix show that group 'G9' also tends to have less successful attacks (i.e., smaller ΔGEA) when compared to other groups. Possible reasons are that: 1) the nodes in this group have lower explanation accuracy on original graph; and 2) the prediction confidence is close to 1, leaving no much room for perturbations. These two facts together make the attack more challenging.

5.6.2 Experiments to Answer RQ1 and RQ2: Attack Effectiveness

From Table 5.2, we have the following main observations:

Observation 1. *The larger the subgraph size, the lower the original explanation accuracy of PGExplainer and thus the poorer the attack performance* (i.e., the smaller the value of ΔGEA). For this, compare the results on Syn2 (Subgraph Size = 6) to those on Syn3 (Subgraph Size = 10).

Observation 2. *The type of motifs (namely ground-truth explanations) appears to have minimal impact on both explanation accuracy and attack performance.* This can be obtained by comparing the results on Syn3 (Motif ="House") with those on Syn4 (Motif ="Circle").

Observation 3. *The larger the connection probability (namely node degree), the lower the original explanation accuracy of PGExplainer and thus the poorer the attack performance* (i.e., the smaller the value of ΔGEA). This can be obtained by comparing the results on Syn3 (P(Connection) = 0.06), Syn5 (P(Connection) = 0.12), and Syn6 (P(Connection) = 0.20).

Observation 4. *The more subgraphs, the lower the original explanation accuracy of PGExplainer and thus the poorer the attack performance* (i.e., the smaller the value of ΔGEA). This can be obtained by comparing the results on Syn3 (#Subgraphs=50) with those on Syn7 (#Subgraphs=100).

Answer to RQ1. From Table 5.2, we can see that GXAttack can achieve a ΔGEA ranging from 0.088 to 0.267 while using a very small perturbation budget (less than 7.1 on all datasets) and keeping the predicted labels nearly unchanged (i.e., ΔLabel is nearly zero) after perturbation.

Answer to RQ2. In contrast, the random flipping attack employs a budget of 15 but achieves ΔGEA of almost zero on all datasets, indicating it is ineffective. Meanwhile, the random rewiring attack uses a budget of 16 and achieves a very large ΔGEA (ranging from 0.293 to 0.596). However, this comes at the cost of large ΔLabel (ranging from 0.159 to 0.274). In other words, the predicted labels of many nodes have been changed after the attacks, and this violates our objective.

5.6.3 Experiments to Answer RQ3: Attack Transferability

Answer to RQ3. From Table 5.3, we can see that some post-hoc explainers, including PGMExplainer, and IG, suffer from poor explanation accuracy on the original graphs, even giving lower accuracy than the Random Explainer. As a result, investigating the attack transferability on these explainers may not be meaningful. On the contrary, like PGExplainer, explainers such as GradCAM, GNNExplainer, SubgraphX, and GNN-LRP (highlighted in gray) can achieve good performance on most original graphs. For those, the attacks optimized for PGExplainer are also harmful, leading to a dramatic degradation of explanation accuracy. For instance, the explanation accuracy of GNNexplainer decreases from 0.711 to 0.397 on Syn2. This demonstrates the transferability of the attacks generated by GXAttack.

5.6.4 Sensitivity Analysis

Specifically, we perform analysis in terms of ΔGEA, ΔProb and used budget when varying the following hyperparameters, respectively: 1) the loss trade-off hyperparameter; 2) the maximally allowed perturbation budget (Max Budget) in Figure 5.2; and 3) the maximal training epochs of attacker in Figure 5.3.

Table 5.3: Transferability of attacks on synthetic datasets Syn1-6 with a typical run (Results on Syn7 are omitted due to excessive computation time). We report the explanation accuracy (i.e., GEA) on the original graph → explanation accuracy on the perturbed graph.

Explainer	Syn1	Syn2	Syn3	Syn4	Syn5	Syn6
PGExplainer	0.728 → 0.510	0.648 → 0.347	0.485 → 0.304	0.471 → 0.311	0.450 → 0.285	0.332 → 0.225
GradCAM	0.745 → 0.551	0.708 → 0.397	0.510 → 0.300	0.511 → 0.310	0.455 → 0.278	0.414 → 0.241
IG	0.118 → 0.104	0.134 → 0.103	0.136 → 0.101	0.109 → 0.092	0.116 → 0.085	0.078 → 0.065
GNNExplainer	0.738 → 0.550	0.711 → 0.397	0.539 → 0.341	0.514 → 0.327	0.483 → 0.283	0.431 → 0.205
SubgraphX	0.113 → 0.015	0.532 → 0.375	0.296 → 0.141	0.417 → 0.282	0.394 → 0.312	0.408 → 0.305
PGMExplainer	0.015 → 0.064	0.046 → 0.039	0.031 → 0.034	0.036 → 0.041	0.038 → 0.043	0.037 → 0.040
GNN-LRP	0.752 → 0.573	0.713 → 0.394	0.509 → 0.300	0.515 → 0.311	0.457 → 0.284	0.410 → 0.241
RandomExplainer	0.184 → 0.146	0.170 → 0.096	0.135 → 0.102	0.124 → 0.090	0.128 → 0.100	0.133 → 0.086

Figure 5.2: Sensitivity analysis w.r.t. maximally allowed perturbation budget (Max Budget) on Syn2 using GXAttack.

Trade-off Hyperparameter vs Attack Performance. Our sensitivity analysis shows that the attack performance (in term of ΔGEA), ΔProb, and the used budget are nearly constant when we vary the trade-off hyper-parameter β's value from 0.000001 to 10. We omit the figures here.

Attack Budget vs Attack Performance. From Figure 5.2, we can see that as the Max Budget increases from 1 to 10, ΔGEA and ΔProb rise sharply, showing large gains from higher allowed budgets. Beyond a (maximally allowed) budget of 10, ΔGEA and ΔProb both stabilize, indicating no further gains. The underlying reason is that the used budget stop increasing after 10 (maximally allowed) when we set the training epochs as 100. In Tables 5.2 and 5.3, we report the results with a maximum budget equal to 15.

Attack Training Epochs vs Attack Performance. From Figure 5.3, we observe a steady increase in the used budget as epochs increase. However ΔProb first increases and then decreases (due to overfitting) when increasing the training epochs. Similarly, ΔGEA first increases and then decreases (due to overfitting) with the increase of training epochs. For computation time reason, we only report the results with 100 epochs in Tables 5.2 and 5.3. However, it is important to note that the attack performance (in terms of ΔGEA) can be largely improved if we increase the number of training epochs. For instance, ΔGEA can be improved from 0.269 to 0.364 if we increase the training epochs from 100 to 300.

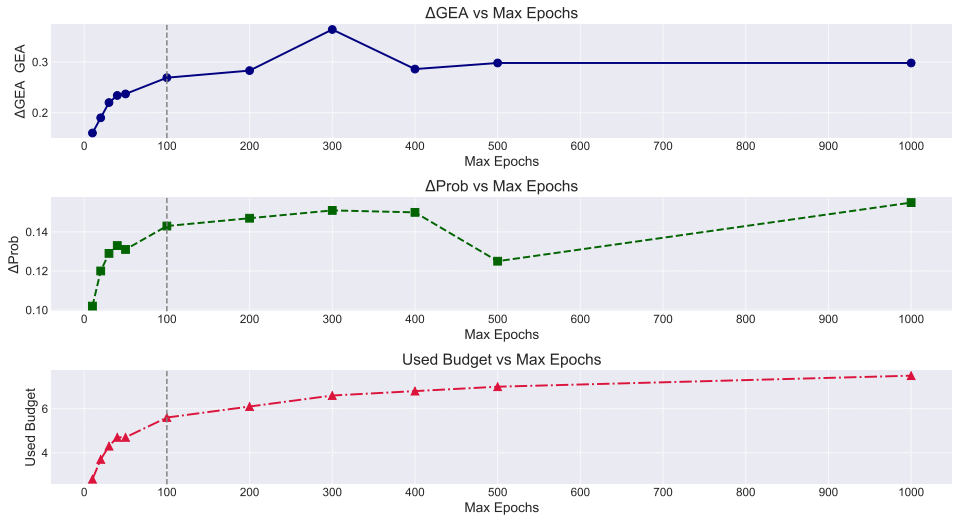
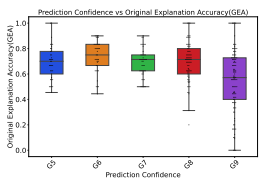
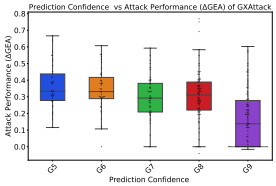


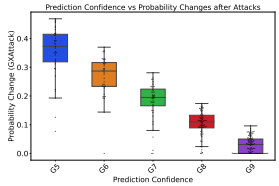
Figure 5.3: Sensitivity analysis w.r.t. maximal training epochs on Syn2 using GXAttack.



(a) Prediction confidence vs. original explanation accuracy on Syn2.



(b) Prediction confidence vs. attack performance (ΔGEA) on Syn2.



(c) Prediction confidence vs. probability change after attacks on Syn2.

Figure 5.4: Property analysis regarding prediction confidence on Syn2. Other datasets show consistent results and are omitted.

5.6.5 Property Analysis

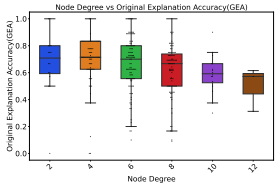
We also conducted some property analysis of GXAttack in terms of the prediction confidence, and node degree.

Property Analysis on Prediction Confidence. From Figure 5.4(a), we can clearly see that nodes in 'G9' tend to have lower original explanation accuracy. As a result, as shown in Figure 5.4(b), the attack performance of GXAttack on these nodes (in 'G9') tend to be less successful (i.e., smaller ΔGEA). This is because the nodes in 'G9' leave no much room for perturbations under the constraint that the predictions remain unchanged, namely much smaller probability changes as shown in Figure 5.4(c).

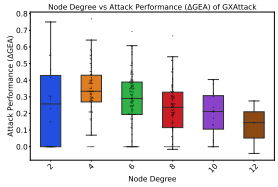
Property Analysis on Node Degree. From Figure 5.5(a), we can see that nodes with higher degrees tend to have lower original explanation accuracy. Figure 5.5(b) shows that the attack performance of GXAttack on these nodes with higher degrees tends to be less successful (i.e., smaller ΔGEA). Possible reasons are that 1) the original explanation accuracy tends to be lower; and 2) small perturbations on nodes with high degrees have limited impact on the predictions, resulting in smaller probability changes as shown in Figure 5.5(c).

5.7 Conclusions

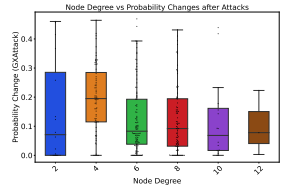
GNN explanations for enhancing transparency and trust of graph neural networks are becoming increasingly important in decision-critical domains. Our research reveals that existing GNN explanation methods are vulnerable to adversarial perturbations that strongly change the explanations without affecting the predictions. This vulner-



(a) Node degree vs. original explanation accuracy on Syn2.



(b) Node degree vs. attack performance (ΔGEA) on Syn2.



(c) Node degree vs. probability change after attacks on Syn2.

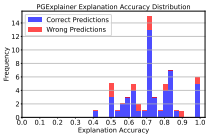
Figure 5.5: Property analysis regarding node degree on Syn2. Other datasets show consistent results and thus are omitted.

ability undermines the reliability of these methods in crucial settings. Specifically, we introduced *GXAttack*, an optimization-based adversarial method to assess the robustness of post-hoc GNN explanations. GXAttack's effectiveness underscores the need for new evaluation standards that incorporate adversarial robustness as a fundamental aspect. Future research should focus on developing explanation methods that are both interpretable and robust against adversarial attacks, as enhancing the stability of GNN explanations is crucial for ensuring their consistency and reliability in practical applications.

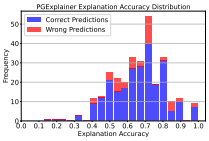
Appendix A: Projected Gradient Descent

After each gradient update, $\tilde{\mathbf{P}}_t \in \mathbb{R}^{n \times n}$, although it must lie in $[0,1]^{n \times n}$ for its interpretation as a Bernoulli random variable. Moreover, we must ensure that sampling $\mathbf{P}_t \in \{0,1\}^{n \times n} \sim \text{Bernoulli}(\tilde{\mathbf{P}}_t)$ will yield a discrete perturbation $\hat{\mathbf{A}} = \mathbf{A} \oplus \mathbf{P}_t$ obeying the admissible perturbations $\|\mathbf{A} - \hat{\mathbf{A}}\|_0 < B$ with sufficient likelihood. Following [227, 228], we ensure that $\mathbb{E}_{\mathbf{P} \sim \text{Bernoulli}(\tilde{\mathbf{P}}_t)}[\|\mathbf{P}\|_0] \leq B$ and then obtain the final perturbation form a small set of samples. To guarantee that $\tilde{\mathbf{P}}_t$ parametrizes a valid distribution and that we obey the budget in expectation, we employ the projection $\prod_{\mathbb{E}[\text{Bernoulli}(\tilde{\mathbf{P}}_t) \leq B]}(\tilde{\mathbf{P}}_t)$.

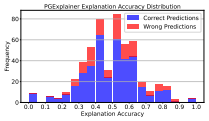
The space spanned by $\tilde{\mathbf{P}}_t \in [0,1]^{n \times n}$ describes a $n \times n$ dimensional hypercube intersected with the region below the B-simplex defining $\mathbb{E}[\text{Bernoulli}(\tilde{\mathbf{P}}_t) = B]$. The region below the B-simplex can be expressed as $\vec{1}^\top \tilde{\mathbf{P}} \vec{1} = \sum_{i,j} \tilde{\mathbf{P}}_{i,j} \leq B$. Thus, after



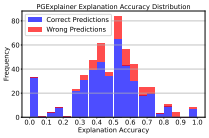
(a) Syn1



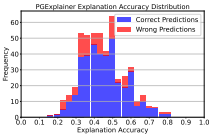
(b) Syn2



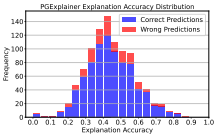
(c) Syn3



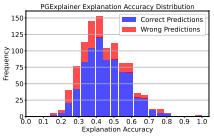
(d) Syn4



(e) Syn5



(f) Syn6



(g) Syn7

Figure 5.6: Comparative analysis of explanation accuracy with respect to prediction correctness on all synthetic datasets. The results show that explanation accuracy is not necessarily correlated with the prediction correctness. Particularly, we did **not** observe that wrongly predicted nodes tend to have lower explanation accuracy.

each gradient update, we solve

$$\prod_{\mathbb{E}[\text{Bernoulli}(\tilde{\mathbf{P}})\leq B]}(\tilde{\mathbf{P}}) = \arg\min_{\tilde{\mathbf{S}}} \|\tilde{\mathbf{P}} - \tilde{\mathbf{S}}\|_F^2 \tag{5.21}$$
$$\text{subject to } \sum_{i,j} \tilde{S}_{i,j} \leq B \text{ and } \tilde{\mathbf{S}} \in [0,1]^{n\times n}$$

for clipping to the hyperplane and projecting back towards the simplex using a bisection search. See [227] for details and the derivation.

Appendix B: Presumptions Checking Results

Table 5.4: Complete results on Syn1-Syn7. **O. GEA** means explanation accuracy on original graph and **P. GEA** indicates explanation accuracy on perturbed graph. Moreover, **ΔGEA** is the explanation accuracy change after perturbation, $\mathbf{Sim}_{\cos}$ is the cosine similarity between explanations (before and after perturbations), and **#Pert.** represents the number of flipped edges. Besides, **ΔLabel** is the average change of predicted labels, while **ΔProb** denotes the average absolute change of predicted probability (of the original predicted class). "G5" is the set of nodes of which the original prediction confidence located in [0.5,0.6). Similarly, "G6" for [0.6,0.7), "G7" for [0.7,0.8), "G8" for [0.8,0.9), "G9" for [0.9,1.0], and "All" for all nodes.

Dataset	Confidence Group	G5	G6	G7	G8	G9	All
Syn1	**O. GEA**	0.696	0.763	0.726	0.775	0.683	0.728
	P. GEA (↓)	0.403	0.463	0.485	0.584	0.679	0.560
	ΔGEA (↑)	0.293	0.299	0.241	0.190	0.004	0.168
	$\mathbf{Sim}_{\cos}$ (↓)	0.817	0.868	0.898	0.937	0.987	0.923
	#Pert. (↓)	6.5	5.0	4.5	2.7	0.2	3.0
	ΔLabel (↓)	0	0	0	0	0	0
	ΔProb (↓)	0.327	0.204	0.147	0.065	0.002	0.108
Syn2	**O. GEA**	0.694	0.739	0.690	0.694	0.546	0.648
	P. GEA (↓)	0.329	0.396	0.390	0.387	0.381	0.347
	ΔGEA (↑)	0.365	0.293	0.300	0.307	0.165	0.269
	$\mathbf{Sim}_{\cos}$ (↓)	0.958	0.960	0.971	0.979	0.991	0.977
	#Pert. (↓)	7.1	6.6	6.2	6.3	3.8	5.6
	ΔLabel (↓)	0	0	0	0	0	0
	ΔProb (↓)	0.353	0.268	0.181	0.103	0.003	0.143
Syn3	**O. GEA**	0.484	0.496	0.479	0.510	0.440	0.480
	P. GEA (↓)	0.305	0.303	0.279	0.335	0.310	0.306
	ΔGEA (↑)	0.179	0.193	0.200	0.175	0.130	0.179
	$\mathbf{Sim}_{\cos}$ (↓)	0.966	0.973	0.962	0.981	0.989	0.978
	#Pert. (↓)	8.0	7.4	6.6	5.8	4.7	6.7
	ΔLabel (↓)	0.008	0	0	0	0	0.002
	ΔProb (↓)	0.242	0.206	0.166	0.099	0.039	0.165
Syn4	**O. GEA**	0.500	0.500	0.493	0.495	0.416	0.471
	P. GEA (↓)	0.309	0.314	0.319	0.326	0.296	0.311
	ΔGEA (↑)	0.192	0.187	0.179	0.168	0.120	0.159
	$\mathbf{Sim}_{\cos}$ (↓)	0.973	0.972	0.962	0.979	0.989	0.980
	#Pert. (↓)	6.9	6.8	6.4	6.2	3.9	5.7
	ΔLabel (↓)	0	0	0	0	0	0
	ΔProb (↓)	0.333	0.267	0.177	0.111	0.030	0.149
Syn5	**O. GEA**	0.456	0.470	0.443	0.461	0.430	0.450
	P. GEA (↓)	0.252	0.264	0.282	0.305	0.336	0.293
	ΔGEA (↑)	0.204	0.207	0.161	0.157	0.094	0.157
	$\mathbf{Sim}_{\cos}$ (↓)	0.967	0.971	0.975	0.979	0.984	0.976
	#Pert. (↓)	7.5	7.6	7.0	6.6	5.2	6.6
	ΔLabel (↓)	0.013	0	0	0	0	0.002
	ΔProb (↓)	0.287	0.226	0.147	0.085	0.031	0.138
Syn6	**O. GEA**	0.415	0.412	0.407	0.382	0.291	0.332
	P. GEA (↓)	0.260	0.249	0.239	0.259	0.217	0.231
	ΔGEA (↑)	0.155	0.162	0.168	0.123	0.074	0.101
	$\mathbf{Sim}_{\cos}$ (↓)	0.975	0.975	0.979	0.986	0.984	0.989
	#Pert. (↓)	7.3	7.8	7.5	7.0	5.0	5.9
	ΔLabel (↓)	0	0	0	0	0	0
	ΔProb (↓)	0.348	0.270	0.189	0.106	0.029	0.092
Syn7	**O. GEA**	0.428	0.415	0.406	0.405	0.401	0.409
	P. GEA (↓)	0.266	0.250	0.294	0.263	0.278	0.263
	ΔGEA (↑)	0.161	0.165	0.156	0.142	0.123	0.146
	$\mathbf{Sim}_{\cos}$ (↓)	0.986	0.983	0.989	0.991	0.992	0.990
	#Pert. (↓)	8.2	7.9	7.5	7.1	6.1	7.2
	ΔLabel (↓)	0	0	0	0	0	0
	ΔProb (↓)	0.319	0.253	0.169	0.097	0.035	0.147

Chapter 6

Cross-Domain Graph Level Anomaly Detection

Authors: **Zhong Li**, Sheng Liang, Jiayang Shi, Matthijs van Leeuwen

Published in IEEE Transactions on Knowledge and Data Engineering, 36(12), 7839–7850.

Abstract

Existing graph level anomaly detection methods are predominantly unsupervised due to high costs for obtaining labels, yielding sub-optimal detection accuracy when compared to supervised methods. Moreover, they heavily rely on the assumption that the training data exclusively consists of normal graphs. Hence, even the presence of a few anomalous graphs can lead to substantial performance degradation. To alleviate these problems, we propose a *cross-domain graph level anomaly detection method*, aiming to identify anomalous graphs from a set of unlabeled graphs (*target domain*) by using easily accessible normal graphs from a different but related domain (*source domain*). Our method consists of four components: a feature extractor that preserves semantic and topological information of individual graphs while incorporating the distance between different graphs; an adversarial domain classifier to make graph level representations domain-invariant; a one-class classifier to exploit label information in the source domain; and a class aligner to align classes from both domains based on pseudolabels. Experiments on seven benchmark datasets show that the proposed method largely outperforms state-of-the-art methods.

6.1 Introduction

Graph-structured data is ubiquitous, as it can represent relations between objects using edges and semantic characteristics of objects using node attributes. As a result, graph level anomaly detection has a wide range of potential applications, such as criminal detection in financial network [82], error detection in system logs [232], identifying specific molecules in drug discovery [233], and detecting unhealthy brain structures [234].

Graph neural networks (GNNs) are capable of learning discriminative feature representations for graphs and have significantly advanced benchmark results for graph level anomaly detection [79]. Similar to other types of neural networks, the impressive performance of graph neural networks (GNNs) is often attained by using a substantial amount of labeled data. However, the process of manually annotating graph data is laborious and thus often impractical. To circumvent this challenge, recent studies have turned to unsupervised (or semi-supervised) learning instead. These methods, however, strongly rely on the assumption that the training data exclusively consists of normal graphs. Our experiments demonstrate that even a minor presence of anomalous graphs in the training data can lead to substantial performance degradation for these methods (c.f. Table 6.3).

In practice, labeled data may be accessible or relatively cheap to obtain in some domains. Hence, in situations where a certain 'target' domain of interest suffers from a dearth of labeled data (or its purity cannot be guaranteed), there is a strong motivation to construct learners that can exploit abundant labeled data from a different but related domain.

Recent work on graph level anomaly detection [78, 79, 81, 82] is mostly unsupervised (or semi-supervised), and has been limited to detecting anomalies within a single domain. That is, the potential benefits of incorporating labeled information from a related domain has not yet been researched. In this paper we investigate how to transfer 'anomaly knowledge' from a source to a target graph database.

Unsupervised domain adaptation (UDA) is an attractive approach to achieve this: it adapts models learned from a source domain with plenty labeled data to a target domain without labels, and has demonstrated remarkable performance in computer vision and natural language processing [235, 236]. Although a few studies have explored UDA for cross-domain node classification, there has been no prior research on cross-domain graph level anomaly detection. Two challenges need to be overcome. First, most existing UDA methods are developed for vector-based data, such as im-

age and text data, for which a distance in a Euclidean space can be defined, while for graph-structured data distance is typically defined in a non-Euclidean space due to graph isomorphism. This makes directly applying off-the-shelf UDA methods to graphs impractical. Second, graph level anomaly detection is inherently more challenging than node level anomaly detection, as anomalies at the graph level may involve global patterns and interactions that cannot be easily discerned by examining individual nodes.

To fill this gap, being motivated and supported by domain adaptation theory [237], we propose an unsupervised domain adaptation based graph level anomaly detection method called ARMET. It addresses the following cross-domain graph level anomaly detection problem: given a target graph database with fully unlabeled graphs and a *different but related* source graph database that contains only normal graphs, learn a one-class classifier that identifies anomalous graphs from the target graph database.

To achieve this, ARMET leverages an adversarial learning approach consisting of four main components. First, to learn graph level representations, it utilizes a two-part feature extractor: a semantic feature extractor to jointly preserve the semantic and topological information of each graph, and a structure feature extractor to extract the structure of each graph domain. Second, a domain classifier is learned to make graph level representations domain-invariant, thereby reducing the domain discrepancy. Third, a one-class classifier is trained using normal source graphs, aiming to make the learned graph level representations label-discriminative. Finally, a class aligner is trained to align normal graphs in both domains while separating anomalous graphs and normal graphs in the target domain. As a result, in an end-to-end manner, ARMET can learn both domain-invariant and label-discriminative graph level representations, and thus effectively identify anomalous graphs from the target domain.

Our contributions can be summarized as follows:

- We introduce the cross-domain graph level anomaly detection problem, and develop ARMET, an effective approach to address this problem;
- ARMET is the first attempt to combine graph neural networks and adversarial domain adaptation techniques for performing graph level anomaly detection tasks;
- Experiments demonstrate its improved performance for graph level anomaly detection when compared to state-of-the-art unsupervised graph level anomaly detectors.

The rest of this work is organized as follows. Chapter 6.2 reviews related literature. Chapter 6.3 introduces relevant notations and formulates the research problem. Chapter 6.4 presents the framework of our method ARMET. Chapter 6.5 and 6.6 describe the design details of ARMET. Chapter 6.7 provides experimental setup, and Chapter 6.8 presents results and corresponding analysis. Chapter 6.10 concludes this work.

6.2 Related Work

We review work that is most closely related; readers are referred to the following surveys for more on transfer learning [238, 239], graph representation learning [188], and graph anomaly detection [240, 241].

Graph Level Anomaly Detection Unsupervised/semi-supervised methods include OCGIN [78], GLAM [79], GLocalKD [81], OCGTL [82] and CODEtect [84]. Unlike ARMET, CODEtect can only handle unattributed graphs. Meanwhile, OCGIN, GLAM, GLocalKD, and OCGTL can handle attributed graphs, but they heavily depend on the assumption that the training data exclusively contains normal graphs. This assumption is impractical or expensive in real-world applications. As we will show later, the presence of anomalies in the training data may largely decrease the performance of these methods (c.f. Table 6.3). Moreover, the supervised method iGAD requires fully labeled training data, which is expensive or sometimes impossible to obtain. In contrast, ARMET only requires that the source domain data exclusively contains normal graphs, while imposing no additional assumptions on the target domain data. Further, it is the first graph level anomaly detection method that leverages labeled data from a different but related domain.

Traditional Unsupervised Domain Adaptation Traditional UDA techniques, such as DeepCoral [242], DANN [243], ADDA [244], and CDAN [245], are primarily designed for addressing multi-class classification problems in computer vision and natural language processing. Consequently, these methods fail to account for the unique characteristics of graph-structured data, as well as the highly imbalanced nature of anomaly detection problems. Moreover, these methods assume that the source domain contains all classes and is fully labeled. As a result, their effectiveness diminishes when the source domain contains only partial classes (e.g., only the normal class in anomaly detection), and this will be shown later in Table 6.6.

Domain Adaptation on Graphs Most existing domain adaptation methods on graphs, such as SDA-DAGL [246], DANE [247], UDA-GCN [248], DASGA [249], ACDNE [250], CDNE [251], DANE [252], COMMANDER [253], AdaGCN [254], DGL

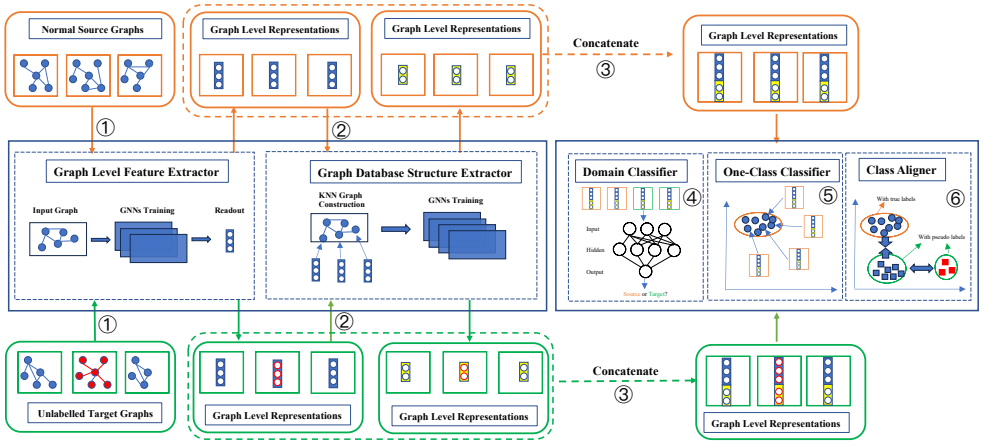


Figure 6.1: An overview of ARMET. Graphs and their learned representations are framed in oval rectangles, where the target domain is highlighted in green and the source domain in orange. Meanwhile, the semantic feature extractor, structure feature extractor, and other learners including one-class classifier, domain classifier, and class aligner are depicted in blue rectangles.

[255], AdaGIn [256] and CD-GAD [257], only consider domain adaptation from an individual graph to another graph.

Only a few works, including DGDA [258], CDA [259], and [260], consider domain adaptation from a set of graphs to another set of graphs. However, they focus on the multi-class graph classification problem, while ARMET is the first to consider cross-domain graph level anomaly detection, which is more challenging due to the extreme class imbalance. Moreover, unlike ARMET, these methods require the source domain data to contain all classes and be fully labeled.

6.3 Problem Statement

Following the notation commonly used in transfer learning [238], a *domain* $\mathcal{D}$ consists of two components, namely a feature space $\mathcal{X}$ and its marginal probability distribution $\mathbb{P}(\mathcal{X})$. Meanwhile, a *task* contains two components, that is, a label space $\mathcal{Y}$ and a predictive function $h(\cdot)$ that can be expressed as $\mathbb{P}(\mathcal{Y}|\mathcal{X})$. Moreover, we consider an attributed and undirected graph $G = (\mathcal{V}, \mathcal{E})$, where the node set $\mathcal{V}$ is associated with a node feature matrix $\mathbf{X} \in \mathcal{R}^{|\mathcal{V}|\times C}$ and the the edge set $\mathcal{E}$ is associated with the adjacency matrix $\mathbf{A} \in \mathcal{R}^{|\mathcal{V}|\times|\mathcal{V}|}$. In the following sections, we use the superscripts $(\cdot)^s$ and $(\cdot)^t$ to denote concepts in the source and target domains (also tasks), respectively.

Traditional Unsupervised Domain Adaptation on Graphs Traditional unsupervised domain adaptation (UDA) assumes that there is a set of labeled source graphs $\mathcal{G}^s = \{G_n^s, Y_n^s\}_{n=1}^{N_s}$ that are i.i.d drawn from $\mathbb{P}(\mathcal{X}^s, \mathcal{Y}^s)$, and another set of unlabeled target graphs $\mathcal{G}^t = \{G_n^s\}_{n=1}^{N_t}$ that are i.i.d drawn from $\mathbb{P}(\mathcal{X}^t)$. Moreover, UDA usually imposes the *covariate shift assumption*, i.e., $\mathbb{P}^s(\mathcal{X}^s) \neq \mathbb{P}^t(\mathcal{X}^t)$ but $\mathbb{P}^s(\mathcal{Y}^s|\mathcal{X}^s) = \mathbb{P}^t(\mathcal{Y}^t|\mathcal{X}^t)$. In other words, it assumes that the source and the target are different (in the sense that $\mathbb{P}^s(\mathcal{X}^s) \neq \mathbb{P}^t(\mathcal{X}^t)$) but related (in the sense that $\mathcal{X}^s = \mathcal{X}^t$ and $\mathbb{P}^s(\mathcal{Y}^s|\mathcal{X}^s) = \mathbb{P}^t(\mathcal{Y}^t|\mathcal{X}^t)$). On this basis, UDA aims to learn a classifier $h : \mathcal{X}^t \to \mathcal{Y}^t$ by using fully labeled source graphs and unlabeled target graphs.

For simplicity, we assume that the node feature matrices of the source (i.e., $\mathbf{X}^s$) and target graphs (i.e., $\mathbf{X}^t$) have the same dimensionality and their columns share the same semantic meanings. Otherwise, we can construct a unified node feature set $\mathcal{X} = \mathcal{X}^s \cup \mathcal{X}^t$ by following the practice in [254, 256]. This assumption is important for utilizing parameters-shared graph embedding models and fulfilling the covariate shift assumption in UDA.

Cross-Domain Graph Level Anomaly Detection The anomaly detection problem can be considered as a binary classification problem. Hence, we have $\mathcal{Y}^s = \mathcal{Y}^t = \{0, 1\}$, where 0 and 1 represent normal and abnormal graphs, respectively. To further relax the dependency on fully labeled source data, we assume that $\mathcal{G}^s$ only contains normal graphs. Specifically, given $\mathcal{G}^s = \{G_n^s, Y_n^s\}_{n=1}^{N_s} \in \mathcal{X}^s \times \mathcal{Y}^s$ with $Y_n^s = 0$ for $n \in 1, ..., N_s$, and $\mathcal{G}^t = \{G_n^t\}_{n=1}^{N_t} \in \mathcal{X}^t$, *Cross-Domain Graph Level Anomaly Detection* (CD-GLAD) aims to learn a binary classifier $g : \mathcal{X}^t \to \mathcal{Y}^t$ that accurately predicts anomaly labels for graphs in the target domain, with the assistance of both normal graphs from the source domain and unlabeled graphs from the target domain. Hence, the CD-GLAD problem is more practical but entails greater challenges than traditional UDA.

6.4 Proposed Method: ARMET

Motivation According to domain adaptation theory [237], given the source domain $\mathcal{D}^s = \{\mathcal{X}^s, \mathbb{P}(\mathcal{X}^s)\}$ and the target domain $\mathcal{D}^t = \{\mathcal{X}^t, \mathbb{P}(\mathcal{X}^t)\}$, given any binary classifier h drawn from a hypothesis class $\mathcal{H}$, for any $\delta \in (0, 1)$, with the probability at least of $1 - \delta$, we have

$$\epsilon^t(h) \leq \epsilon^s(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}^s, \mathcal{D}^t) + [\epsilon^s(h^*) + \epsilon^t(h^*)] + \omega, \tag{6.1}$$

where $\epsilon^t(h)$ and $\epsilon^s(h)$ indicate the expected error of classifier h on the target domain and source domain, respectively. Moreover, $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}^s, \mathcal{D}^t)$ represents the domain discrepancy. Importantly, $[\epsilon^s h^*) + \epsilon^s(h^*)]$ means the combined error of the ideal joint classifier h^* on both domains, where $h^* =: \arg\min_{h\in\mathcal{H}}[\epsilon^s(h) + \epsilon^t(h)]$. Besides, ω is the item associated with the model complexity and the sample sizes.

In this work, we aim to learn a classifier h that has the minimal expected error on the target domain. Therefore, the sum of terms on the right side of (6.1) should be minimized. This sheds key insights into the design of our algorithm, which considers the following overall objective:

$$\mathcal{L}(\mathcal{X}^s, \mathcal{Y}^s, \mathcal{X}^t) = \lambda_1\mathcal{L}_{SC}(\mathcal{X}^s, \mathcal{Y}^s) - \lambda_2\mathcal{L}_{DA}(\mathcal{X}^s, \mathcal{X}^t) + \lambda_3\mathcal{L}_{CA}(\mathcal{X}^s, \mathcal{Y}^s, \mathcal{X}^t), \qquad (6.2)$$

where $\mathcal{L}(\mathcal{X}^s, \mathcal{Y}^s, \mathcal{X}^t)$ corresponds to the upper bound of $\epsilon^t(h)$, $\mathcal{L}_{SC}(\mathcal{X}^s, \mathcal{Y}^s)$ is the source classifier loss corresponding to $\epsilon^s(h)$, $\mathcal{L}_{DA}(\mathcal{X}^s, \mathcal{X}^t)$ is the domain classifier loss that approximates $d(\mathcal{D}^s, \mathcal{D}^t)$ (larger loss indicates smaller discrepancy), and $\mathcal{L}_{CA}(\mathcal{X}^s, \mathcal{Y}^s, \mathcal{X}^t)$ is the class alignment loss that corresponds to $[\epsilon^s(h^*) + \epsilon^s(h^*)]$. Further, the balance parameters $\lambda_1 > 0$, $\lambda_2 > 0$ and $\lambda_3 > 0$.

Approach Figure 6.1 depicts the architecture of our proposed method, dubbed ARMET (adversarial cross domain graph level anomaly detection). Specifically, ARMET adapts an adversarial learning framework to perform cross-domain graph level anomaly detection. It involves the four following main components:

- Parameters-Shared Feature Extractor h^{FE}: takes a source graph database $\mathcal{G}^s$ consisting of exclusively normal graphs and an unlabeled target graph database $\mathcal{G}^t$ as inputs, and aims to learn a representation vector $h^{FE}(G_i)$ for each graph $G_i \in \mathcal{G}^s \cup \mathcal{G}^t$ such that similar graphs (in terms of semantic and structure properties) from both domains have similar embeddings;

- One-Class Classifier h^s: takes the representation of each graph $h^{FE}(G_i)$ from the source domain as input, and learns a hypersphere to include embeddings of normal graphs while excluding those of anomalous graphs. This classifier can be directly used to predict labels for graphs in the target domain and corresponds to $\mathcal{L}_{SC}$;

- Domain Classifier h^d: takes the representation of each graph $h^{FE}(G_i)$ as input, and attempts to discriminate whether it is drawn from the source domain or the target domain such that the distributions of embeddings of both domains are aligned. And it corresponds to $\mathcal{L}_{DA}$;

- Class Aligner: takes $\{G_n^s, Y_n^s\}_{n=1}^{N_s}$ and $\{G_n^t, \hat{Y}_n^t\}_{n=1}^{N_t}$ as inputs, further making normal graphs from both domains have close embeddings, while normal graphs and anomalous graphs in the target domain have distant embeddings. Particularly, we apply the One-Class Classifier h^s to obtain pseudo-labels $\hat{Y}_n^t$ for graphs in the target domain while using the true labels Y_n^s for graphs in the source domain; This component corresponds to $\mathcal{L}_{CA}$.

For better organization, we divide these components into two modules, as shown in Figure 6.1: the *graph feature extraction module* that contains the feature extractor (steps ①,②, ③), and the *cross-domain anomaly detection module* that includes the domain classifier (step ④), the one-class classifier (step ⑤), and the class aligner (step ⑥). Importantly, the two modules are trained jointly in an end-to-end manner, although they are introduced separately in the following.

6.5 Module 1: Graph Feature Extraction

The graph feature extraction module consists of a feature extractor dubbed $h^{FE}(\cdot)$ with trainable parameters Θ^{FE}, aiming to extract graph level representations for graphs from source domain and target domain. This component is further decomposed to three sub-components: a semantic feature extractor (step ①) and a structure feature extractor (step ②), followed by a feature concatenation operator (step ③).

Semantic Feature Extractor We denote the semantic feature extractor as $h^{SE}(\cdot)$ with trainable parameters Θ^{SE}, which learns a graph level representation for each input graph. Specifically, we adapt a K-layer GIN model [73] followed by a READOUT function to obtain graph level representations due to the superior performance of GIN compared to other competing methods [261]. Concretely, GIN updates node representations as

$$f_v^{(k+1)} = \text{MLP}\left(\left(1+\alpha^{(k)}\right) f_v^{(k)} + \sum\nolimits_{u\in\mathcal{N}(v)} f_u^{(k)}\right), \tag{6.3}$$

where $f_v^{(k)}$ denotes the representation of node v learned at the k-th layer, $\mathcal{N}(v)$ indicates the neighbor set for node v, $\alpha^{(k)}$ is a trainable parameter while MLP represents a multilayer perceptron. Next, we obtain the graph level representation by leveraging $\text{READOUT}\left(f_v^{(k)}|k=1,...,K\right)$, which can be a simple permutation-invariant function such as the maximum, sum or mean. Using this, we can preserve the semantic and topological information of each graph.

Structure Feature Extractor We denote the structure extractor as $h^{ST}(\cdot)$ with parameters Θ^{ST}. We assume that $\mathcal{G}^s = \{G_1^s, ..., G_j^s, ..., G_N^s\}$ and $\mathcal{G}^t = \{G_1^t, ..., G_j^t, ..., G_M^t\}$ exhibit some inherent data structures such as clusters in $\mathcal{X}^s$ and $\mathcal{X}^t$, respectively. Inspired by [262], for each graph database, we construct a k-nearest neighbors graph (*KNN graph*) in the latent space to model its data structure, aiming to capture the neighborhood information between different graphs. Without loss of generality, we use $\mathcal{G}^s$ to demonstrate the construction of a *KNN graph*.

The *KNN graph* of $\mathcal{G}^s$ contains N nodes, with each node representing a source graph and its node attribute indicating the corresponding graph level representation vector learned by $h^{ST}(\cdot)$. Next, to generate edges, we can construct the adjacency matrix as

$$A_{ij} = \begin{cases} 1, & \text{if } G_i \in \mathcal{N}_k(G_j) \text{ or } G_j \in \mathcal{N}_k(G_i) \\ 0, & \text{otherwise} \end{cases} \tag{6.4}$$

where $\mathcal{N}_k(G_j)$ represents the k-nearest neighbors set of graph G_j based on the Euclidean distance between graph level embeddings. After constructing the *KNN graph*, we can leverage another GIN model (without READOUT function) to learn the node representations, wherein a node represents a source graph instance.

Feature Concatenation Operator We utilize a concatenation operator that directly appends the structure feature for each graph to its semantic feature. Hence, this operator has no parameters. Overall, given a graph G_i, its final extracted feature can be expressed as $h^{FE}(G_i) = h^{SE}(G_i) \parallel h^{ST}(G_i)$. As a result, the corresponding trainable parameter $\Theta^{FE} = (\Theta^{SE}, \Theta^{ST})$. Importantly, these parameters are shared when learning representations for graphs from the source domain and the target domain, respectively.

6.6 Module 2: Cross-Domain Graph Level Anomaly Detection

This module contains three components: the one-class classifier, the domain classifier, and the class aligner. We first elucidate the rationale and design of each component, followed by introducing the theory of adversarial training.

One-Class Classifier: We use $h^s(\cdot)$ to represent the source classifier, which has no additional parameters beyond the feature extraction parameters Θ^{FE}. We train a one-class classifier on the source domain by minimizing the following One-Class Deep

SVDD objective [99]:

$$\mathcal{L}_{SC}(\mathcal{X}^s, \mathcal{Y}^s; \Theta^{FE}) =: \frac{1}{N_s} \sum_{m=1}^{N_s} \|h^{FE}(G_m) - \mathbf{o}\|_2^2, \tag{6.5}$$

where $h^{FE}(G_m)$ is the final extracted feature of graph G_m (from the source domain), and $\mathbf{o}$ is the learned center that represents the normality defined in the source domain. Minimizing this SVDD loss ensures the model learns the label information from the source domain.

Domain Classifier: We denote the domain classifier as $h^d(\cdot)$, with trainable parameters Θ^d in addition to parameters Θ^{FE}. We maximize the binary cross-entropy loss to learn domain-invariant features:

$$\mathcal{L}_{DA}(\mathcal{X}^s, \mathcal{X}^t; \Theta^d, \Theta^{FE}) =: \left[\frac{1}{N_s + N_t} \sum_{i=1}^{N_s+N_t} \left[d_i \log(\frac{1}{\hat{d}_i}) + (1 - d_i) \log(\frac{1}{1 - \hat{d}_i}) \right] \right], \tag{6.6}$$

where d_i denotes the binary ground-truth domain label for graph G_i. Specifically, d_i is 0 for graphs from the source domain and 1 for graphs from the target domain. Additionally, $\hat{d}_i$ represents the predicted probability that G_i belongs to the target domain, as determined by $h^d(\cdot)$. More precisely, the prediction $\hat{d}_i$ is given by $\hat{d}_i = h^d(h^{FE}(G_i))$, where $h^{FE}(\cdot)$ is the final feature extractor and $h^d(\cdot)$ is the domain classifier. In loss term (6.6), we consider the negative log-probability, expressed as $-\log(\hat{d}_i)$. This can be rewritten as $\log\left(\frac{1}{h^d(h^{FE}(G_i))}\right)$. The goal of this loss function is to maximize it, which encourages the feature extractor $h^{FE}(\cdot)$ to create similar representations for graphs from both the source and target domains. This helps align the domains, making the feature distributions of the source and target domains indistinguishable and reducing discrepancies between them.

Class Aligner: Structure consistency between domains (via parameters-shared feature extractor), discriminability in source domain (via source classifier), and domain-invariant features (via domain classifier) do not necessarily lead to discriminability in the target domain. That is, although we manage to align features cross domains, the learned features may be distorted in the sense that they are not representative of the underlying patterns in the target domain. As a result, features of the normal and abnormal classes in the target domain may exhibit close proximity or even overlap, leading to a high value of $\epsilon^t(h)$ in the target domain. This is known as excessive alignment in [263] and collapse of target neighborhood structure in [264].

To alleviate this problem, we should consider the third term in (6.1), namely $[\epsilon^s(h^*) + \epsilon^t(h^*)]$, that considers labels from both domains. Although the true label information in the target domain cannot be obtained, we can generate pseudolabels and minimize the class centroid alignment loss:

$$\mathcal{L}_{CA}(\mathcal{X}^s, \mathcal{Y}^s, \mathcal{X}^t, \hat{\mathcal{Y}}^t; \Theta^{FE}) =: \left[\psi(\mathcal{C}_n^s, \mathcal{C}_n^t) - \psi(\mathcal{C}_a^t, \mathcal{C}_n^t)\right], \tag{6.7}$$

where $\hat{\mathcal{Y}}^t$ are the **pseudolabels** obtained by directly applying $h^s(\cdot)$ to the target graphs, and $h^s(\cdot)$ is obtained by optimizing loss term (6.5) in previous iteration. Moreover, $\mathcal{C}_n^s$, $\mathcal{C}_n^t$, and $\mathcal{C}_a^t$ represent the centroid of normal source class, normal target class, and anomalous target class, respectively. The function $\psi(\cdot, \cdot)$ is a distance metric such as the Euclidean distance. Minimizing this loss can reduce the inter-domain distance between centroids of normal classes, while simultaneously maximizing the intra-domain distances between centroids of normal and abnormal classes within the target domain. Particularly, with an increase of training epochs, we expect that the discrepancy between source domain and target domain is reduced, the data structures are better aligned, the $h^s(\cdot)$ is further improved, and thus the pseudolabels are gradually updated to approach the ground-truth labels, progressively improving the class centroid alignment.

Adversarial Training: It can be seen that the optimization of (6.5) and (6.7) involves a minimization w.r.t. parameters Θ^{FE}, while the optimization of (6.6) concerns a maximization w.r.t. parameters Θ^d and Θ^{FE}. In other words, objective (6.6) competes against objectives (6.5) and (6.7) during training over the overall objective (6.2). To obtain a good trade-off, adversarial training techniques have been explored, with impressive results [243].

For simplicity, we rewrite $\mathcal{L}(\mathcal{X}^s, \mathcal{Y}^s, \mathcal{X}^t; \Theta^{FE}, \Theta^d)$ as $\mathcal{L}(\Theta^{FE}, \Theta^d)$, which contains two sets of parameters. Adversarial training attempts to find a saddle point $(\hat{\Theta}^{FE}, \hat{\Theta}^d)$ such that $\hat{\Theta}^{FE} = \underset{\Theta^{FE}}{\operatorname{argmin}}\, \mathcal{L}(\Theta^{FE}, \hat{\Theta}^d)$ and $\hat{\Theta}^d = \underset{\Theta^d}{\operatorname{argmax}}\, \mathcal{L}(\hat{\Theta}^{FE}, \Theta^d)$. In other words, we perform a minmax optimization over (6.2):

$$\min_{\Theta^{FE}} \max_{\Theta^d} \left[\lambda_1 \mathcal{L}_{SC}(\Theta^{FE}) - \lambda_2 \mathcal{L}_{DA}(\Theta^{FE}, \Theta^d) + \lambda_3 \mathcal{L}_{CA}(\Theta^{FE})\right]. \tag{6.8}$$

Hence, the trainable parameters can be optimized alternatively as follows:

$$\begin{aligned}\Theta^d &\leftarrow \Theta^d - \mu\frac{\partial \mathcal{L}_{DA}}{\partial \Theta^d},\\ \Theta^{FE} &\leftarrow \Theta^{FE} - \mu\left(\lambda_1\frac{\partial \mathcal{L}_{SC}}{\partial \Theta^{FE}} - \lambda_2\frac{\partial \mathcal{L}_{DA}}{\partial \Theta^{FE}} + \lambda_3\frac{\partial \mathcal{L}_{CA}}{\partial \Theta^{FE}}\right),\end{aligned} \tag{6.9}$$

where μ is the learning rate. We perform mini-batch training with gradient descent and the corresponding pseudo-code is provided in Algorithm 5.

Algorithm 5 Mini-batch Algorithm of ARMET

Input: Labeled source graphs $\mathcal{G}^s = \{G_n^s, Y_n^s\}_{n=1}^{N_s}$; Unlabeled target graphs $\mathcal{G}^t = \{G_n^s\}_{n=1}^{N_t}$; Balance parameters λ_1, λ_2 and λ_3; Batch size N_b; Maximal training epochs N_e; Maximal iteration per epoch N_i
Output: Predicted labels $\hat{\mathcal{Y}}^t$ of target graphs

Initialize parameters Θ^{FE}, Θ^d
for `epoch` $< N_e$ and not converge **do**
  for `iteration` $< N_i$ **do**
    Sample a batch $\mathcal{B}^s$ and $\mathcal{B}^t$
    Learn representations using h^{FE} for $\mathcal{B}^s$ and $\mathcal{B}^t$
    Compute source classifier loss $\mathcal{L}_{SC}$ using (6.5)
    Compute domain classifier loss $\mathcal{L}_{DC}$ using (6.6)
    Backpropagate $\mathcal{L}_{DC}$ and update Θ^d using (6.9)
    Compute class aligner loss $\mathcal{L}_{CA}$ using (6.7)
    Compute total loss $\mathcal{L}$ using (6.2)
    Backpropagate $\mathcal{L}$ and update Θ^{FE} using (6.9)
  end for
end for
$\hat{h}^{FE}(\mathcal{G}^t) \leftarrow$ Use feature extractor with optimized parameters $\hat{\Theta}^{FE}$ to extract features for $\mathcal{G}^t$
$\hat{\mathcal{Y}}^t \leftarrow$ Use optimized source classifier $\hat{h}^s(\cdot)$ to predict labels for $\hat{h}^{FE}(\mathcal{G}^t)$

6.7 Experiment Setup

We aim to answer the following research questions (RQ) via experiments:

RQ1 How does ARMET perform when compared to state-of-the-art graph level anomaly detection methods?

RQ2 How does each component of ARMET affect the performance? (Ablation study)

Table 6.1: Datasets. **#Nodes**, **#Edges**, **Degree**, **#Attr** denote the average number of nodes and edges, the average degree, and the dimensionality of node attributes, respectively.

Data	#Graphs	#Nodes	#Edges	Degree	#Attr
BG	5000(5%)	10	30	3	200
HD	5000(5%)	7	20	2.86	200
SP	5000(5%)	6	24	4	200
TB	5000(5%)	16	52	3.25	200
LL	500(10%)	4.7	4.5	0.96	2
LM	500(10%)	4.7	3.1	0.66	2
LH	500(10%)	4.7	3.3	0.70	2

RQ3 How does the performance of ARMET change with different hyperparameter values? (Sensitivity analysis)

In addition, to get a better understanding of ARMET, we utilize t-SNE [265] to visualize the learned graph representations from both domains.

6.7.1 Benchmark Datasets

As summarized in Table 6.1, we study the following datasets collected from various application fields of graph mining:

System Logs We use four benchmark datasets for log anomaly detection, as converting logs into graphs and then leveraging GNNs to detect anomalies can achieve superior performance [88, 232]. Hence, following [232], we construct four graph datasets: HDFS (HD) [64], BGL (BG), SPIRIT (SP), and THUNDERBIRD (TH) [123], where each dataset contains 5000 graphs with 5% anomalous graphs. Particularly, HD consists of Hadoop Distributed File System logs while BG, SP, and TH containing system logs collected from three different supercomputing systems. For this group of datasets, we create 12 transfer tasks.

Letter Drawings We use three benchmark graph datasets with letter drawings of varying levels of distortion: low (LL), medium (LM), and high (LH) [266]. Following the practice of downsampling classification datasets for anomaly detection [168], for each dataset the letters N, M, and W are selected as the normal class, comprising a total of 450 instances (150 instances per letter), while the letter F is chosen as the anomalous class (downsampled to 50 instances). For each graph, a node denotes the end point of a line, an edge represents a line, and a node attribute represents its two-dimensional coordinate. For these datasets, we create 6 transfer tasks.

Discussion on Dataset Selection: Some commonly used benchmark graph datasets, including BZR, DHFR and COX2 (small molecules), as well as IMDB-Binary and REDDIT (social networks), have been excluded from our analysis for the following reasons: 1) the UDA assumptions do not hold for them, as these datasets have different definitions of classification problem, namely they have different label semantics; 2) transferability cannot be guaranteed due to huge domain discrepancy between datasets. For instance, node attributes of graphs in these datasets have different dimensionalities and/or meanings, and these datasets have very different graph statistics; and 3) even supervised classification methods can only achieve very limited in-domain classification accuracy (i.e., with an accuracy lower than 0.7) on these datasets.

6.7.2 Baselines

We compare ARMET to the following baselines:

- **Unsupervised graph level anomaly detection methods:** We directly apply OCGIN [78], GLAM [79], and GLocalKD [81] on unlabeled target graphs.
- **Traditional Domain Adaptation Methods:** ADDA [244], CDAN [245], and DeepCoral [242] are state-of-the-art UDA methods for image classification. As they are not designed for graph-structured data, we modify these models for cross-domain graph level anomaly detection by following [257].

To explore the ceiling performance of GLAD methods, we additionally present the outcomes of iGAD [83], a supervised method that is not considered a direct competitor due to its reliance on labeled target data.

6.7.3 Evaluation

We employ the widely used Area Under the Curve of the Receiver Operating Characteristics curve (AUC ROC), Area Under the Curve of Precision Recall (AUC PR), and F1-Score to evaluate and compare the different methods, where a higher value (closer to 1) represents better anomaly detection accuracy. Particularly, we report the average values of AUC ROC, AUC PR, F1-Score and the corresponding standard deviations across 10 independent runs.

6.7.4 Implementation and Model Configuration

We use the publicly available implementations of OCGIN [1], GLAM [1], GLocalKD [2] and iGAD [3] with their recommended configurations. Besides, ADDA[4], CDAN [5] and DeepCoral [5] are adapted from their publicly available implementations. As they are not designed for graph-structured data, we modify these models for cross-domain graph level anomaly detection by following the practice in [257]:

- CDAN: We replace the AlexNet encoder with a GIN plus a mean readout function;
- ADDA: Similarly, we replace the CNN encoder with a GIN plus a mean readout function;
- DeepCoral: We replace the CNN encoder (CaffeNet) with a GIN plus a mean readout function and apply CORAL loss to the last classification layer.

For ARMET, by following the practice in [242], the values of hyperparameters λ_1, λ_2 and λ_3 can be set such that, after the training process (e.g., 100 epochs), the losses associated with one-class classification $\mathcal{L}_{SC}$, domain discrimination ($\mathcal{L}_{DA}$) and class-alignment ($\mathcal{L}_{CA}$) are approximately at the same magnitude (after being multiplied by their corresponding weights). The rational behind it is that we aim to learn feature representations that are both label-discriminative and domain-invariant [242]. Particularly, we found that there is usually not a single set of optimal weights for each transfer task, as pointed out in multi-task learning by [267]. Although this hyperparameter tuning method works well, it is time-consuming and labor-intensive. For simplicity, we can adapt a *de-facto* strategy for hyperparameter tuning in UDA, namely splitting the target dataset according to a ratio of 20% : 80%, where the 20% data with labels is used as the validation dataset to select these three hyperparameters and the remaining 80% data without labels is used as the test data.

For fair comparisons, all GIN models used in different domain adaptation methods are configured with the same backbone architecture, namely a backbone of two layers (64 hidden units) with each layer followed by a ReLU activation. Besides, all neural network based methods are trained using mini-batch gradient descent, with a batch-size of 512 on log anomaly detection datasets and a batchsize of 128 on other datasets

[1] https://github.com/lingxiaoshawn/glam
[2] https://github.com/RongrongMa/GLocalKD
[3] https://github.com/graph-level-anomalies/iGAD
[4] https://github.com/yuhui-zh15/pytorch-adda
[5] https://github.com/agrija9/deep-unsupervised-domain-adaptation

Table 6.2: Description of hyperparameters and their recommended values. **Range** indicates the values that we have tested in the sensitivity analysis, and boldfaced values represent the values suggested to use in the experiments.

Symbol	Meaning	Range
d	embedding dimension	{16, 32, **64**, 96, 128, 160, 192, 224, 256, 300}
k	number of neighbors in KDTree	{2, **5**, 8, 11, 14, 17, 20}
L	number of layers	$\{1, \mathbf{2}, 3, 4, 5\}$
Bs	batch size	{16, 32, 64, **128**, 256, **512**, 1024}
λ	weight decay parameter	$\{\mathbf{0.0001}, 0.001, 0.01, 0.1\}$
η	learning rate	$\{0.0001, 0.001, \mathbf{0.01}\}$
Re	readout function	**mean**, sum, max
Ep	Epochs for training	{**100**, 200, 300, 400}

respectively, initial learning rate of 0.01, weight decay rate of 0.0001, and a maximum of 200 training epochs. The settings for these hyperparameters for ARMET are summarized in Table 6.2. Other algorithm-specific hyperparamters are set in accordance with their respective references given by the original authors.

6.7.5 Training Hardware and Reproducibility

We implemented and ran all algorithms in Python 3.8, using PyTorch [109] and PyTorch Geometric [110] when applicable, on a workstation equipped with an Intel i7-11700KF CPU and Nvidia RTX3070 GPU. For reproducibility, all code and datasets are made available on GitHub[6].

6.8 Experiment Results and Analysis

We answer the three research questions as follows.

6.8.1 Detection Accuracy (RQ1)

We perform the following analysis based on the experiment results in Tables 6.3, 6.4 and 6.6. Particularly, the results in terms of AUC PR and F1-Score are consistent with those in terms of AUR ROC. Therefore, these results are either deferred to Table 6.8 or omitted.

[6]https://github.com/ZhongLIFR/ARMET/

Table 6.3: Anomaly detection accuracy (Average AUC ROC and corresponding standard deviations across 10 runs) of unsupervised methods (OCCIN, GLAM, and GLocalKD) under two scenarios: 1) when the training dataset is clean, namely containing exclusively normal graphs (shown on the left side of '→'), and 2) when the training dataset is contaminated, namely containing both normal and abnormal instances (shown on the right side of '→').

Dataset	OCGIN	GLAM	GLocalKD
BG	$0.93_{\pm 0.04} \rightarrow 0.71_{\pm 0.11}(\downarrow)$	$0.64_{\pm 0.09} \rightarrow 0.74_{\pm 0.06}(\uparrow)$	$0.93_{\pm 0.02} \rightarrow 0.91_{\pm 0.01}(\downarrow)$
SP	$0.62_{\pm 0.16} \rightarrow 0.59_{\pm 0.16}(\downarrow)$	$0.70_{\pm 0.09} \rightarrow 0.71_{\pm 0.09}(\uparrow)$	$0.81_{\pm 0.08} \rightarrow 0.66_{\pm 0.15}(\downarrow)$
HD	$0.98_{\pm 0.01} \rightarrow 0.88_{\pm 0.03}(\downarrow)$	$0.87_{\pm 0.06} \rightarrow 0.77_{\pm 0.09}(\downarrow)$	$0.45_{\pm 0.05} \rightarrow 0.45_{\pm 0.05}(-)$
TH	$0.81_{\pm 0.20} \rightarrow 0.43_{\pm 0.28}(\downarrow)$	$0.93_{\pm 0.05} \rightarrow 0.96_{\pm 0.09}(\uparrow)$	$0.93_{\pm 0.05} \rightarrow 0.76_{\pm 0.11}(\downarrow)$
LL	$0.99_{\pm 0.00} \rightarrow 0.70_{\pm 0.10}(\downarrow)$	$0.82_{\pm 0.18} \rightarrow 0.72_{\pm 0.18}(\downarrow)$	$0.56_{\pm 0.11} \rightarrow 0.53_{\pm 0.10}(\downarrow)$
LM	$0.91_{\pm 0.02} \rightarrow 0.64_{\pm 0.05}(\downarrow)$	$0.57_{\pm 0.10} \rightarrow 0.55_{\pm 0.07}(\downarrow)$	$0.56_{\pm 0.11} \rightarrow 0.53_{\pm 0.10}(\downarrow)$
LH	$0.65_{\pm 0.10} \rightarrow 0.52_{\pm 0.08}(\downarrow)$	$0.66_{\pm 0.08} \rightarrow 0.54_{\pm 0.10}(\downarrow)$	$0.56_{\pm 0.11} \rightarrow 0.53_{\pm 0.10}(\downarrow)$

Accuracy of Single-Domain GLAD

Table 6.3 demonstrates that unsupervised/semi-supervised methods OCGIN, GLAM[7], and GLocalKD generally suffer from large performance degradation when the training data is contaminated with anomalies. Moreover, these unsupervised/semi-supervised methods deliver very unstable results across different datasets even when the training data is clean. For example, GLocalKD yields high performance on BGL (ROC = 0.91), but very poor performance on HDFS (ROC = 0.45). In contrast, Table 6.4 shows that supervised method iGAD achieves perfect results, with an ROC of 1.0, on all cases. This indicates that these anomaly detection problems are solvable when sufficient labeled data is available in the target domain. However, this is unrealistic in many real-world scenarios as fully labeled data is usually expensive and often even impossible to obtain in practice.

Accuracy of Traditional UDA Methods

Table 6.4 shows that when there is only one class in the source domain, ADDA, CDAN, and DeepCoral behave like random guessing (e.g., with ROC $\approx$ 0.50), performing much worse than ARMET for most transfer tasks. The potential factors contributing to their subpar performance are as follows. First, CDAN considers the conditional distribution that captures the cross-variance between feature representations and classifier predictions, but the conditional distribution degenerates to the marginal distribution when there is only one class in the source domain. Second,

[7]With a few exceptions for GLAM, where the performance is slightly increased or stable.

Table 6.4: Anomaly detection accuracy (Average AUC ROC and corresponding standard deviations across 10 runs). The best and runner-up results are highlighted in **bold** and underlined, respectively. For unsupervised methods, we report the ROC values when the training dataset contains both normal and abnormal instances. For cross-domain methods, we report the ROC values when the source dataset contains only normal graphs. The results of supervised method are shown **only** to be able to put the performance of graph level anomaly detection methods in perspective (i.e., iGAD should **not** be considered as baseline). Moreover, the poor performance of traditional UDA methods (namely ADDA, CDAN and DeepCoral) are not due to intentionally choosing weak baselines or poor hyper-parameters. The underlying reasons for their subpar performance are given in Chapter 6.8.1.

Dataset	Supervised	Unsupervised			Traditional UDA			Ours
	iGAD	OCGIN	GLAM	GLocalKD	ADDA	CDAN	DeepCoral	ARMET
HD → BG					$0.50_{\pm 0.00}$	$0.50_{\pm 0.02}$	$0.50_{\pm 0.00}$	$\underline{0.77}_{\pm 0.06}$
SP → BG	$1.0_{\pm 0.00}$	$0.71_{\pm 0.11}$	$0.74_{\pm 0.06}$	$\mathbf{0.91}_{\pm 0.01}$	$0.50_{\pm 0.00}$	$0.49_{\pm 0.03}$	$0.50_{\pm 0.00}$	$\underline{0.85}_{\pm 0.07}$
TH → BG					$0.50_{\pm 0.00}$	$0.50_{\pm 0.00}$	$0.50_{\pm 0.00}$	$\underline{0.83}_{\pm 0.05}$
BG → SP					$0.50_{\pm 0.00}$	$0.50_{\pm 0.01}$	$0.50_{\pm 0.00}$	$\mathbf{0.89}_{\pm 0.07}$
HD → SP	$1.0_{\pm 0.00}$	$0.59_{\pm 0.16}$	$0.71_{\pm 0.09}$	$0.66_{\pm 0.15}$	$0.50_{\pm 0.00}$	$0.51_{\pm 0.04}$	$0.50_{\pm 0.00}$	$\mathbf{0.91}_{\pm 0.04}$
TH → SP					$0.50_{\pm 0.00}$	$0.50_{\pm 0.00}$	$0.50_{\pm 0.00}$	$\mathbf{0.88}_{\pm 0.06}$
BG → HD					$0.50_{\pm 0.00}$	$0.50_{\pm 0.01}$	$0.50_{\pm 0.00}$	$\underline{0.79}_{\pm 0.04}$
SP → HD	$1.0_{\pm 0.00}$	$0.88_{\pm 0.03}$	$0.77_{\pm 0.09}$	$0.45_{\pm 0.05}$	$0.50_{\pm 0.00}$	$0.51_{\pm 0.01}$	$0.50_{\pm 0.00}$	$\mathbf{0.90}_{\pm 0.03}$
TH → HD					$0.50_{\pm 0.00}$	$0.50_{\pm 0.00}$	$0.50_{\pm 0.00}$	$\underline{0.81}_{\pm 0.06}$
BG → TH					$0.50_{\pm 0.00}$	$0.60_{\pm 0.20}$	$0.50_{\pm 0.00}$	$\mathbf{1.0}_{\pm 0.00}$
SP → TH	$1.0_{\pm 0.00}$	$0.43_{\pm 0.28}$	$0.96_{\pm 0.09}$	$0.76_{\pm 0.11}$	$0.50_{\pm 0.00}$	$0.69_{\pm 0.24}$	$0.50_{\pm 0.00}$	$\mathbf{1.0}_{\pm 0.00}$
HD → TH					$0.50_{\pm 0.00}$	$0.72_{\pm 0.24}$	$0.50_{\pm 0.00}$	$\mathbf{1.0}_{\pm 0.00}$
LM → LL	$1.0_{\pm 0.00}$	$0.70_{\pm 0.10}$	$0.72_{\pm 0.18}$	$0.53_{\pm 0.10}$	$0.50_{\pm 0.00}$	$0.50_{\pm 0.00}$	$0.50_{\pm 0.00}$	$\mathbf{0.98}_{\pm 0.02}$
LH → LL					$0.50_{\pm 0.00}$	$0.50_{\pm 0.00}$	$0.50_{\pm 0.00}$	$\mathbf{0.96}_{\pm 0.00}$
LL → LM	$1.0_{\pm 0.00}$	$0.64_{\pm 0.05}$	$0.55_{\pm 0.07}$	$0.53_{\pm 0.10}$	$0.50_{\pm 0.00}$	$0.50_{\pm 0.00}$	$0.50_{\pm 0.00}$	$\mathbf{0.88}_{\pm 0.02}$
LH → LM					$0.50_{\pm 0.00}$	$0.50_{\pm 0.00}$	$0.50_{\pm 0.00}$	$\mathbf{0.78}_{\pm 0.03}$
LL → LH	$1.0_{\pm 0.00}$	$0.52_{\pm 0.08}$	$0.54_{\pm 0.10}$	$0.53_{\pm 0.10}$	$0.50_{\pm 0.00}$	$0.50_{\pm 0.00}$	$0.50_{\pm 0.00}$	$\mathbf{0.79}_{\pm 0.02}$
LM → LH					$0.50_{\pm 0.00}$	$0.50_{\pm 0.00}$	$0.50_{\pm 0.00}$	$\mathbf{0.80}_{\pm 0.05}$

DeepCoral aligns the second-order statistics of layer activations in the source encoder and the target encoder, leading to random-guessing results since the source training data contains only normal graphs while the target training data includes both normal and abnormal graphs (and thus the statistics of layer activations should be different by nature). Third, ADDA performs poorly as it utilizes different encoders for the source and target domains, implying the importance of parameters-shared models when encoding graphs from two domains. The efficacy of the source encoder degrades to random-guessing when the source domain contains only a single class, as the source classifier fails to acquire any informative knowledge. Furthermore, these traditional UDA methods are primarily designed for balanced multiple classification problems, making them ill-suited for anomaly detection, which involves an extremely imbal-

Table 6.5: Ablation studies with the following stripped-down variations, where '✓' and '✗' mean the corresponding component is included or excluded, respectively.

Components	SD	/SF	/SC	/DC	/CA	ARMET
Structure Feature Extractor	✓	✗	✓	✓	✓	✓
One-Class Classifier	✓	✓	✗	✓	✓	✓
Domain Classifier	✗	✓	✓	✗	✓	✓
Class Aligner	✗	✓	✓	✓	✗	✓

anced binary classification task. As a reference, we report the performance of ADDA, CDAN, and DeepCoral when the source domain contains both labeled anomalous and normal instances in Table 6.6. One can see that the performance of ADDA, CDAN, and DeepCoral is largely boosted with auxiliary label information in most transfer tasks. However, even with auxiliary labels, they are still outperformed by ARMET in most cases.

Accuracy of ARMET

Table 6.4 shows that ARMET achieves the best performance on 13 out of 18 transfer tasks, and the second-best performance on the remaining tasks. Further, it largely outperforms unsupervised graph level anomaly detection methods on certain target datasets, demonstrating the benefit of leveraging label information from a different but related domain. For example, transferring knowledge from **HD** to **SP** leads to 54%, 28%, and 38% performance gains compared to OCGIN, GLAM, and GLocalKD, respectively. Finally, ARMET achieves impressive results under the setting that the source domain contains only normal graphs, while other traditional unsupervised domain adaptation methods provide "random-guessing" results, making ARMET more applicable to real-world scenarios.

6.8.2 Ablation Study (RQ2)

As summarized in Table 6.5, we here compare ARMET to five its stripped-down variations: SD is trained on the source domain and then directly applied on the target domain; /SF is ARMET trained without the structure feature extractor; /SC is ARMET trained without the source one-class classifier; /DC is ARMET trained without the domain classifier; and /CA is ARMET trained without the class aligner. We have the following important observations from Table 6.7:

- **SD vs ARMET.** ARMET is always superior to the case where we train the

Table 6.6: Anomaly detection accuracy (Average AUC ROC and corresponding standard deviations across 10 runs) of traditional UDA methods (including ADDA, CDAN, and DeepCoral) when the source domain contains both labeled anomalous and normal instances. ('↑' indicates the performance is boosted compared to the case where the source domain contains only normal instances, while '↓' means it is degraded.)

Dataset	ADDA*	CDAN*	DeepCoral*	ARMET
HD → BG	$0.50_{\pm 0.03}(-)$	$0.64_{\pm 0.11}(\uparrow)$	$0.51_{\pm 0.11}(\uparrow)$	$\mathbf{0.77}_{\pm 0.02}$
SP → BG	$0.47_{\pm 0.00}(\downarrow)$	$0.48_{\pm 0.06}(\downarrow)$	$0.49_{\pm 0.01}(\downarrow)$	$\mathbf{0.85}_{\pm 0.07}$
TH → BG	$0.49_{\pm 0.02}(\downarrow)$	$0.50_{\pm 0.00}(-)$	$0.50_{\pm 0.00}(-)$	$\mathbf{0.83}_{\pm 0.05}$
BG → SP	$0.56_{\pm 0.17}(\uparrow)$	$0.50_{\pm 0.08}(-)$	$0.49_{\pm 0.19}(\downarrow)$	$\mathbf{0.89}_{\pm 0.07}$
HD → SP	$0.63_{\pm 0.24}(\uparrow)$	$0.50_{\pm 0.03}(\downarrow)$	$0.60_{\pm 0.16}(\uparrow)$	$\mathbf{0.91}_{\pm 0.04}$
TH → SP	$0.54_{\pm 0.13}(\uparrow)$	$0.50_{\pm 0.00}(-)$	$0.50_{\pm 0.01}(-)$	$\mathbf{0.88}_{\pm 0.06}$
BG → HD	$0.62_{\pm 0.13}(\uparrow)$	$0.50_{\pm 0.08}(-)$	$0.48_{\pm 0.11}(\downarrow)$	$\mathbf{0.79}_{\pm 0.04}$
SP → HD	$0.57_{\pm 0.14}(\uparrow)$	$0.51_{\pm 0.03}(-)$	$0.51_{\pm 0.01}(\uparrow)$	$\mathbf{0.90}_{\pm 0.03}$
TH → HD	$0.57_{\pm 0.06}(\uparrow)$	$0.51_{\pm 0.02}(\uparrow)$	$0.50_{\pm 0.00}(-)$	$\mathbf{0.81}_{\pm 0.06}$
BG → TH	$0.48_{\pm 0.01}(\downarrow)$	$0.48_{\pm 0.20}(\downarrow)$	$0.54_{\pm 0.24}(\uparrow)$	$\mathbf{1.00}_{\pm 0.00}$
SP → TH	$0.53_{\pm 0.17}(\uparrow)$	$0.77_{\pm 0.24}(\uparrow)$	$0.73_{\pm 0.27}(\uparrow)$	$\mathbf{1.00}_{\pm 0.00}$
HD → TH	$0.58_{\pm 0.22}(\uparrow)$	$0.50_{\pm 0.17}(\downarrow)$	$0.84_{\pm 0.21}(\uparrow)$	$\mathbf{1.00}_{\pm 0.00}$
LM → LL	$0.93_{\pm 0.15}(\uparrow)$	$1.0_{\pm 0.00}(\uparrow)$	$1.0_{\pm 0.00}(\uparrow)$	$0.98_{\pm 0.02}$
LH → LL	$0.78_{\pm 0.23}(\uparrow)$	$0.99_{\pm 0.00}(\uparrow)$	$0.95_{\pm 0.05}(\uparrow)$	$0.96_{\pm 0.00}$
LL → LM	$0.53_{\pm 0.04}(\uparrow)$	$0.74_{\pm 0.00}(\uparrow)$	$0.63_{\pm 0.01}(\uparrow)$	$\mathbf{0.88}_{\pm 0.02}$
LH → LM	$0.65_{\pm 0.05}(\uparrow)$	$0.89_{\pm 0.00}(\uparrow)$	$0.76_{\pm 0.03}(\uparrow)$	$0.78_{\pm 0.03}$
LL → LH	$0.50_{\pm 0.01}(-)$	$0.78_{\pm 0.00}(\uparrow)$	$0.59_{\pm 0.02}(\uparrow)$	$\mathbf{0.79}_{\pm 0.02}$
LM → LH	$0.66_{\pm 0.08}(\uparrow)$	$0.87_{\pm 0.00}(\uparrow)$	$0.80_{\pm 0.01}(\uparrow)$	$0.80_{\pm 0.05}$

model on the source domain and then directly apply it to the target domain. This exemplifies the benefits and necessity of performing transfer learning.

- **/SF vs ARMET.** ARMET consistently outperforms its counterpart without the structural feature extractor. This underlines the importance of considering the neighborhood information in a set of graphs for CD-GLAD.
- **/SC vs ARMET.** It shows that explicitly incorporating the source label information via a source classifier is typically beneficial.
- **/DC vs ARMET.** In most cases, explicitly aligning the embeddings of both domains via a domain classifier is favorable. In certain cases, the presence of a domain classifier may have a small adverse impact on performance. One possible reason is that the domain classifier overly distorts the embedding space by aligning the embedding space in a brute-force manner. Another possible reason is that the embeddings are also implicitly aligned via the parameters-shared feature extractor and the class aligner.

Table 6.7: Ablation study (Average AUC ROC with 10 runs). SD: trained on source domain and then directly applied on target domain; /SF: ARMET without structure feature extractor; /SC: ARMET without source one-class classifier; /DC: ARMET without domain classifier; /CA: ARMET without class aligner.

Dataset	ARMET	SD	/SF	/SC	/DC	/CA
HD → BG	0.77	0.55	0.74	0.48	0.55	0.57
SP → BG	0.85	0.66	0.81	0.55	0.60	0.58
TH → BG	0.83	0.60	0.74	0.64	0.64	0.77
BG → SP	0.89	0.71	0.55	0.67	0.62	0.59
HD → SP	0.91	0.74	0.62	0.73	0.72	0.64
TH → SP	0.88	0.61	0.52	0.73	0.77	0.58
BG → HD	0.79	0.60	0.58	0.59	0.59	0.60
SP → HD	0.90	0.61	0.74	0.67	0.53	0.67
TH → HD	0.81	0.68	0.66	0.70	0.64	0.68
BG → TH	1.0	0.77	0.61	1.0	0.90	0.73
SP → TH	1.0	0.56	0.06	1.0	0.77	0.68
HD → TH	1.0	0.88	0.79	1.0	0.96	0.88
LM → LL	0.98	0.98	0.71	0.70	0.99	0.98
LH → LL	0.96	0.92	0.65	0.73	0.87	0.90
LL → LM	0.88	0.74	0.75	0.62	0.77	0.85
LH → LM	0.78	0.76	0.55	0.42	0.71	0.73
LL → LH	0.79	0.74	0.60	0.59	0.72	0.71
LM → LH	0.80	0.71	0.56	0.63	0.67	0.70

- **/CA vs ARMET.** The removal of the class aligner always leads to a performance degradation. This corroborates the importance of considering labels (or pseudo-labels) in the target domain when performing CD-GLAD.

6.8.3 Sensitivity Analysis (RQ3)

We examine the effects of the following hyperparameters on the detection performance, and Figure 6.2 depicts the selected results on four representative transfer tasks:

- The number of embedding dimensions d in parameters-shared feature extractor: most transfer tasks can achieve the best performance with an embedding dimension of 64. Small fluctuations can be observed with varying number of dimensions. Moreover, an overly small value of d may lead to suboptimal performance, while a large value introduces a considerable computational burden.

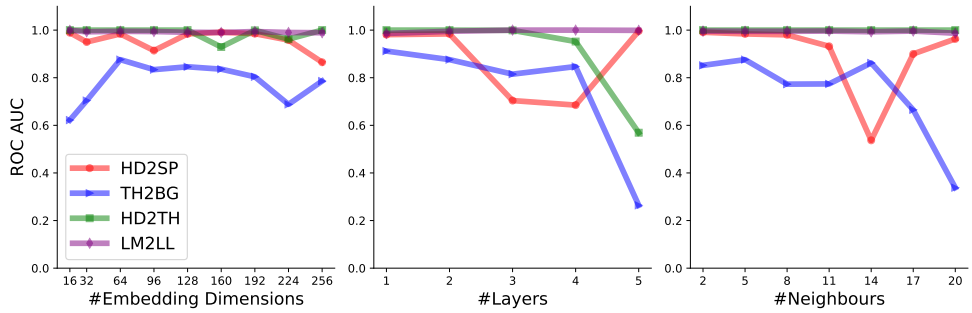


Figure 6.2: Sensitivity analysis of hyperparameters on four representative transfer tasks (HD → SP, TH → BG, HD → TH, LM → LL). Average results over five runs.

- The number of hidden layers L in the GIN model in the parameters-shared feature extractor: optimal performances are obtained when $L = 1$ or $L = 2$ on most transfer tasks. A further increase of its value usually results in performance degradation, which is widely known as over-smoothing [268].
- The number of neighbors k in the KNN graph: most transfer tasks can obtain the best performance on a wide range of k's values (namely $1 \leq k \leq 10$). However, further increasing the value of k may cause large performance fluctuations.

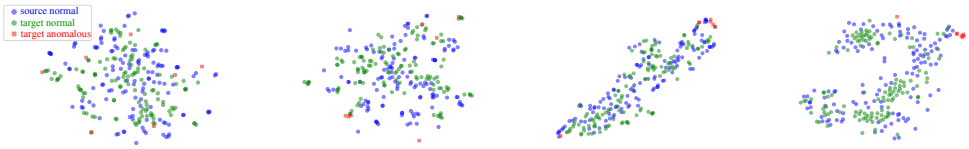


Figure 6.3: T-SNE visualization of the graph representation space during domain adaptation on task **LM** → **LL**. From left to right: the number of epochs correspond to zero, two, five and one hundred, respectively.

6.8.4 Visualization with T-SNE

Figure 6.3 visualizes the domain alignment and anomaly separation process on the transfer task **LM** → **LL**. The target normal graphs (green dots) are progressively adapted to the source normal graphs (blue dots), while the target anomalous graphs (red crosses) become increasingly separable in the embedding space.

6.9 Discussion on Transferability

As shown in Table 6.7, when comparing ARMET with its counterpart without explicit transfer learning (namely the column 'SD' that represents the scenarios where we train the model on the source domain and then directly apply it to the target domain), the performance gains from transfer learning are consistently non-negative. Moreover, when comparing ARMET with the best performing single-domain graph level anomaly detectors that are trained directly on the target domain, the performance gains are often positive (in 13 out 18 cases, see Table 6.4). This further demonstrates the benefits of transfer learning.

The underlying reason why transfer learning is beneficial in this context is that the extent of "relatedness" among System Logs datasets (i.e., BG, HD, SP and TH) is large enough, and so is the extent of "relatedness" among Letter Drawings datasets (i.e., LL, LM, and LH). In cross domain graph-level anomaly detection, it is crucial to ensure that the semantic meanings of normal patterns in the source and target domains are approximately the same. For instance, in System Logs data, these are normal patterns of system operations, while in Letter Drawings data, they represent the same set of letters.

As in other typical unsupervised domain adaptation settings, we assume that the source and target domains are different yet related. However, the extent of "relatedness" in this study is ensured based on our domain knowledge rather than a quantifiable transferability metric. To our knowledge, how to quantify the extent of this "relatedness" (namely *transferability* cross datasets) remains a long-standing problem [269]. Notably, when this "relatedness" is low, it may introduce negative transfer [270]. Moreover, it is critical to note that the transferability is usually *asymmetrical*, e.g., the performance gain for transfer task SP $\rightarrow$ TH is 0.44 (namely 1.0 minus 0.56 in terms of ROC AUC), which is different from the performance gain for TH $\rightarrow$ SP (namely 0.23 in terms of ROC AUC). Therefore, we should exercise caution (especially in safety-critical fields such as healthcare) when the transferability from the source domain to the target domain cannot be guaranteed, whether from a quantifiable metric or domain knowledge perspective.

6.10 Conclusions

This paper studies the problem of cross-domain graph level anomaly detection, wherein a set of unlabeled graphs from the target domain and a set of normal graphs from a

different but related domain are given. We propose ARMET, a theoretically motivated, novel method to solve this widely encountered but largely understudied problem. Specifically, ARMET consists of four components: a feature extractor, an adversarial domain classifier, a one-class classifier, and a class aligner. It is the first attempt to combine graph neural networks and adversarial domain adaptation techniques for performing graph level anomaly detection tasks. Extensive experiments demonstrate the efficacy of ARMET and its superiority to single-domain graph level anomaly detection methods and traditional unsupervised domain adaptation methods. Moreover, extensive ablation studies validate the benefits of incorporating each component into ARMET. Understanding and quantifying the transferability across different graph domains should be addressed in future research.

Table 6.8: Anomaly detection accuracy: average **AUC PR** and corresponding standard deviations across 10 runs (Top), and average **F1-Score**(Binary) and corresponding standard deviations across 10 runs (Bottom)

Dataset	Supervised	Unsupervised			Traditional UDA			Ours
	iGAD	OCGIN	GLAM	GLocalKD	ADDA	CDAN	DeepCoral	ARMET
HD → BG					$0.05_{\pm 0.00}$	$0.05_{\pm 0.01}$	$0.05_{\pm 0.00}$	$0.14_{\pm 0.09}$
SP → BG	$1.0_{\pm 0.00}$	$0.19_{\pm 0.09}$	$0.14_{\pm 0.03}$	$\mathbf{0.46}_{\pm 0.10}$	$0.05_{\pm 0.00}$	$0.05_{\pm 0.00}$	$0.05_{\pm 0.00}$	$\underline{0.25}_{\pm 0.13}$
TH → BG					$0.05_{\pm 0.00}$	$0.05_{\pm 0.00}$	$0.05_{\pm 0.00}$	$\underline{0.29}_{\pm 0.11}$
BG → SP					$0.05_{\pm 0.00}$	$0.05_{\pm 0.00}$	$0.05_{\pm 0.00}$	$\mathbf{0.76}_{\pm 0.12}$
HD → SP	$1.0_{\pm 0.00}$	$0.08_{\pm 0.03}$	$0.09_{\pm 0.02}$	$0.06_{\pm 0.01}$	$0.05_{\pm 0.00}$	$0.09_{\pm 0.08}$	$0.05_{\pm 0.00}$	$\mathbf{0.83}_{\pm 0.15}$
TH → SP					$0.05_{\pm 0.00}$	$0.05_{\pm 0.00}$	$0.05_{\pm 0.00}$	$\mathbf{0.90}_{\pm 0.05}$
BG → HD					$0.05_{\pm 0.00}$	$0.05_{\pm 0.00}$	$0.05_{\pm 0.00}$	$\underline{0.48}_{\pm 0.15}$
SP → HD	$1.0_{\pm 0.00}$	$0.43_{\pm 0.06}$	$\mathbf{0.62}_{\pm 0.07}$	$0.40_{\pm 0.04}$	$0.05_{\pm 0.00}$	$0.06_{\pm 0.02}$	$0.05_{\pm 0.00}$	$0.29_{\pm 0.10}$
TH → HD					$0.05_{\pm 0.00}$	$0.05_{\pm 0.00}$	$0.05_{\pm 0.00}$	$\underline{0.49}_{\pm 0.10}$
BG → TH					$0.06_{\pm 0.00}$	$0.06_{\pm 0.00}$	$0.06_{\pm 0.00}$	$\mathbf{0.93}_{\pm 0.13}$
SP → TH	$1.0_{\pm 0.00}$	$0.12_{\pm 0.15}$	$0.51_{\pm 0.14}$	$0.08_{\pm 0.02}$	$0.06_{\pm 0.00}$	$0.19_{\pm 0.29}$	$0.06_{\pm 0.00}$	$\mathbf{1.0}_{\pm 0.00}$
HD → TH					$0.06_{\pm 0.00}$	$0.41_{\pm 0.48}$	$0.06_{\pm 0.00}$	$\mathbf{0.98}_{\pm 0.01}$
LM → LL	$1.0_{\pm 0.00}$	$0.19_{\pm 0.06}$	$0.42_{\pm 0.20}$	$0.13_{\pm 0.03}$	$0.03_{\pm 0.00}$	$0.03_{\pm 0.00}$	$0.03_{\pm 0.00}$	$\mathbf{0.92}_{\pm 0.06}$
LH → LL					$0.03_{\pm 0.00}$	$0.03_{\pm 0.00}$	$0.03_{\pm 0.00}$	$\mathbf{0.51}_{\pm 0.15}$
LL → LM	$1.0_{\pm 0.00}$	$0.36_{\pm 0.04}$	$0.30_{\pm 0.07}$	$0.13_{\pm 0.03}$	$0.10_{\pm 0.00}$	$0.10_{\pm 0.00}$	$0.10_{\pm 0.00}$	$\mathbf{0.62}_{\pm 0.11}$
LH → LM					$0.10_{\pm 0.00}$	$0.10_{\pm 0.00}$	$0.10_{\pm 0.00}$	$\mathbf{0.37}_{\pm 0.05}$
LL → LH	$1.0_{\pm 0.00}$	$0.25_{\pm 0.04}$	$0.39_{\pm 0.09}$	$0.13_{\pm 0.03}$	$0.10_{\pm 0.00}$	$0.10_{\pm 0.00}$	$0.10_{\pm 0.00}$	$\mathbf{0.43}_{\pm 0.08}$
LM → LH					$0.10_{\pm 0.00}$	$0.10_{\pm 0.00}$	$0.10_{\pm 0.00}$	$\mathbf{0.47}_{\pm 0.12}$

HD → BG					$0.00_{\pm 0.00}$	$0.03_{\pm 0.02}$	$0.00_{\pm 0.00}$	$0.08_{\pm 0.15}$
SP → BG	$1.0_{\pm 0.00}$	$0.15_{\pm 0.08}$	$0.18_{\pm 0.09}$	$\mathbf{0.34}_{\pm 0.06}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.15_{\pm 0.18}$
TH → BG					$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$\underline{0.29}_{\pm 0.11}$
BG → SP					$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$\mathbf{0.76}_{\pm 0.18}$
HD → SP	$1.0_{\pm 0.00}$	$0.05_{\pm 0.08}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.09_{\pm 0.18}$	$0.00_{\pm 0.00}$	$\mathbf{0.79}_{\pm 0.39}$
TH → SP					$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$\mathbf{0.85}_{\pm 0.06}$
BG → HD					$0.00_{\pm 0.00}$	$0.02_{\pm 0.04}$	$0.00_{\pm 0.00}$	$\underline{0.50}_{\pm 0.13}$
SP → HD	$1.0_{\pm 0.00}$	$0.47_{\pm 0.07}$	$\mathbf{0.56}_{\pm 0.10}$	$0.26_{\pm 0.02}$	$0.00_{\pm 0.00}$	$0.04_{\pm 0.06}$	$0.00_{\pm 0.00}$	$0.09_{\pm 0.09}$
TH → HD					$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$\underline{0.50}_{\pm 0.10}$
BG → TH					$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$\mathbf{0.97}_{\pm 0.05}$
SP → TH	$1.0_{\pm 0.00}$	$0.07_{\pm 0.22}$	$0.55_{\pm 0.32}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.17_{\pm 0.37}$	$0.00_{\pm 0.00}$	$\mathbf{1.0}_{\pm 0.00}$
HD → TH					$0.00_{\pm 0.00}$	$0.39_{\pm 0.53}$	$0.00_{\pm 0.00}$	$\mathbf{0.99}_{\pm 0.00}$
LM → LL	$0.99_{\pm 0.01}$	$0.14_{\pm 0.07}$	$0.39_{\pm 0.21}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$\mathbf{0.91}_{\pm 0.01}$
LH → LL					$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$\mathbf{0.57}_{\pm 0.10}$
LL → LM	$0.99_{\pm 0.01}$	$0.40_{\pm 0.03}$	$0.29_{\pm 0.09}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$\mathbf{0.62}_{\pm 0.11}$
LH → LM					$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$\mathbf{0.41}_{\pm 0.07}$
LL → LH	$0.99_{\pm 0.01}$	$0.25_{\pm 0.08}$	$0.37_{\pm 0.09}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$\mathbf{0.49}_{\pm 0.06}$
LM → LH					$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$\mathbf{0.48}_{\pm 0.12}$

Chapter 7

Towards Automated Self-Supervised Learning for Truly Unsupervised Graph Anomaly Detection

Authors: **Zhong Li**, Yuhang Wang, Matthijs van Leeuwen
Manuscript submitted to Data Mining and Knowledge Discovery.

Abstract

Self-supervised learning (SSL) is an emerging paradigm that exploits supervisory signals generated from the data itself, and many recent studies have leveraged SSL to conduct graph anomaly detection. However, we empirically found that three important factors can substantially impact detection performance across datasets: 1) the specific SSL strategy employed; 2) the tuning of the strategy's hyperparameters; and 3) the allocation of combination weights when using multiple strategies. Most SSL-based graph anomaly detection methods circumvent these issues by arbitrarily or selectively (i.e., guided by label information) choosing SSL strategies, hyperparameter settings, and combination weights. While an arbitrary choice may lead to subpar performance, using label information in an unsupervised setting is label information leakage and leads to severe overestimation of a method's performance. Leakage has been criticized as "one of the top ten data mining mistakes", yet many recent studies on SSL-based graph anomaly detection have been using label information to select hyperparameters. To mitigate this issue, we propose to use an internal evaluation strategy (with theoretical analysis) to select hyperparameters in SSL for unsupervised anomaly detection. We perform extensive experiments using 10 recent SSL-based graph anomaly detection algorithms on various benchmark datasets, demonstrating both the prior issues with hyperparameter selection and the effectiveness of our proposed strategy.

7.1 Introduction

Graph anomaly detection (GAD) refers to the tasks of identifying anomalous graph objects—such as nodes, edges or sub-graphs—in an individual graph, or identifying anomalous graphs from a set of graphs [241]. GAD has numerous successful applications, e.g., in finance fraud detection [271], fake news detection [272], system fault diagnosis [232], and network intrusion detection [273]. In this paper, we focus on unsupervised node anomaly detection on static attributed graphs, namely identifying which nodes in a static attributed graph are anomalous. Recently, Graph Neural Networks (GNNs) have become prevalent in detecting node anomalies in graphs and have shown promising performance [98]. Specifically, GNNs can learn an embedding for each node by considering both the node attributes and the graph topological information, enabling them to capture and exploit complex patterns for anomaly detection.

Like with other neural networks, the high performance of GNNs is typically achieved at the cost of a substantial volume of labeled data. However, the process of labeling graphs is often a laborious and time-consuming effort, necessitating domain-specific expertise. For these reasons, GAD is preferably tackled in an unsupervised manner, without relying on any ground-truth labels. Self-supervised learning (SSL) has emerged as a promising unsupervised learning technique on graphs [274], and recent studies have shown its usefulness for node anomaly detection [275, 276, 277, 278, 279, 280, 281, 282].

Graph SSL can be roughly divided into *generative*, *contrastive*, and *predictive* methods [283]. First, *generative* methods such as DOMINANT [284], GUIDE [279], and AnomalyDAE [275] aim to detect graph anomalies by reconstructing ('generating') the adjacency matrix and/or the node attribute matrix. Next, *contrastive* methods such as CoLA [278], ANEMONE [277], GRADATE [285], and Sub-CR [286] train a graph encoder to pull positive pairs closer while pushing negative pairs away in the embedding space. The nodes with relatively large contrastive loss values are deemed anomalies. Finally, *predictive* methods such as SL-GAD [276] try to predict node properties using its local context (e.g., a subgraph), and nodes with large prediction errors are considered anomalies.

Contrastive learning is arguably the most successful SSL strategy for graphs [287]. Most contrastive graph learning methods consist of two main modules: 1) a *data augmentation module* that generates augmented data by operations such as edge dropping, node attribute masking, node addition, subgraph sampling, and/or graph diffusion. The augmented view of an instance is generally regarded as a positive pair with the

original instance; and 2) a *contrastive learning module* that contrasts positive pairs (and often involves negative pairs) at different levels, such as node-node contrast, node-subgraph contrast, and subgraph-subgraph contrast.

Although SSL-based graph anomaly detection has been successful, using it in practice is often not straightforward. The most important reason for this is that most methods require a large number of choices to be made, leading to three challenges:

C1. How should we select appropriate data augmentation functions?

C2. How should we choose appropriate values for hyperparameters (HPs) of a given augmentation function? (e.g., subgraph size in a subgraph sampling function, or the proportion of edges to drop in an edge dropping function)

C3. How to combine the contrast losses at different levels? (i.e., how to set their combination weights?)

Further, a recent study [276] shows that combining multiple SSL strategies for GAD can achieve better performance than using a single SSL strategy. This leads to the fourth challenge:

C4. How should we combine different SSL strategies?(i.e., how to set the combination weights of different SSL loss functions?)

Previous work [288, 289, 290] showed that the choice of SSL strategie(s) and hyperparameter values can strongly impact performance. In a *supervised* setting, these choices can be systematically and rigorously made by using separate labeled data for validation. In an *unsupervised setting* such as anomaly detection, however, one should assume that no labels are available even for hyperparameter tuning. In our extensive literature study, we found that existing SSL-based GAD methods typically either 1) arbitrarily choose settings or 2) do use labeled data, corroborating the findings in [290].

In the former case, practitioners typically heuristically select an augmentation function (C1) and fix its associated HPs (C2) across all datasets, and set the combination weights all equal to 1 or other fixed values (for C3 and Q4). Although this approach is not flawed, it is likely to result in suboptimal detection performance: graphs from different domains usually have different properties [291], implying that the optimal SSL strategy is in general data-dependent [288, 289]. Therefore, utilizing a unified and pre-fixed combination weights and/or HPs in SSL strategies for all graphs can result in sub-optimal performance.

In the latter case, practitioners pick the optimal combination weights and other hyperparameter values following a 'hyperparameters sensitivity analysis' using labeled data. By using ground-truth labels on test data to check model performance with different hyperparameter values and using that to select the best model, however, *label leakage* occurs. That is, information about the target of a data mining problem is used for learning/selecting model, while this information should not be legitimately accessible for learning purposes [292, 293]. Specifically, label information should never be used (whether implicitly or explicitly) in an unsupervised learning scenario. As shown in Figure 7.1, label leakage leads to huge overestimation of the model's performance, which is also corroborated in [294] by comparing the max and average performance with different hyperparameter configurations (cf. Appendix 7.8 for more details).

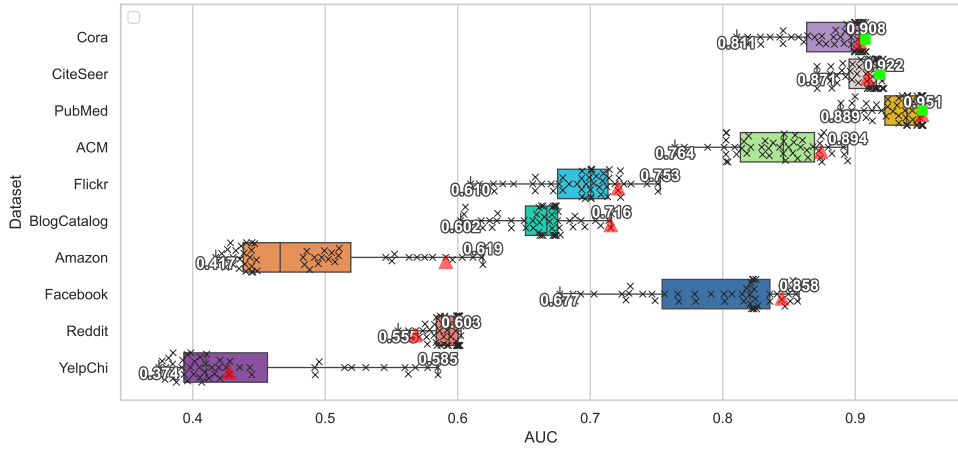


Figure 7.1: Large performance variations (here measured by AUC) over different hyperparameter configurations for ANEMONE [277] on various benchmark datasets. Using labeled data and only reporting the best possible performance leads to severe overestimation of model performance. For instance, the green squares on Cora, CiteSeer, and PubMed are reported by [277] (the other datasets were not used). Similar results are observed for other algorithms (see Appendix 7.8 for details). The red triangles represent the results obtained by our internal evaluation strategy, showing its potential for automating truly unsupervised anomaly detection.

The reason that hyperparameter values are often chosen either arbitrarily or using label information is probably that it is challenging to construct an internal evaluation strategy for anomaly detection without using any labels. There have been some research efforts aimed at automating graph SSL though. For instance, JOAO [295] aims to automatically combine several predefined graph augmentations via learning a

sampling distribution, where the augmentations themselves are not learnable. Meanwhile, AD-GCL [296] uses learnable edge dropping augmentation and AutoGCL [297] proposes a learnable graph view generator that learns a probability distribution over node-level augmentations, which can well preserve the semantic labels of graphs for graph-level tasks. However, all these automated graph augmentation methods are agnostic to the downstream tasks, making the learned graph embeddings sub-optimal for a specific downstream task, namely anomaly detection in our case. Additionally, these methods are specifically designed for certain SSL frameworks, and it is non-trivial (if at all possible) to extend them to the general SSL framework. Moreover, these automated SSL strategies are computationally expensive, rendering them impractical in real-world applications.

As an initial step towards mitigating this long-standing but neglected issue, we propose a lightweight and plug-and-play approach dubbed *AutoGAD*, to automate SSL for truly unsupervised graph anomaly detection. Specifically, *AutoGAD* leverages a so-called internal evaluation strategy [298], without relying on any ground-truth labels (whether explicitly or implicitly), to select optimal combination weights and/or SSL-specific hyperparameter values. Moreover, we theoretically analyze the internal evaluation strategy to prove why it is effective and empirically demonstrate this.

Overall, our main contributions can be summarized as follows:

- We raise renewed awareness to the *label information leakage* issue, which is critical but often overlooked in the unsupervised GAD field;
- Although there exists a plethora of graph SSL methods and GAD approaches, we are the first to investigate automated SSL specifically for unsupervised GAD;
- We propose a lightweight, plug-and-play approach to automate SSL for truly unsupervised GAD and provide a theoretical analysis;
- Extensive experiments are conducted using 10 state-of-the-art SSL-based GAD algorithms on 10 benchmark datasets, demonstrating the effectiveness of our approach.

7.2 Related Work

Our work is related to node anomaly detection on static attributed graphs, self-supervised learning for graph anomaly detection, automated self-supervised learning, and automated anomaly detection.

7.2.1 Anomaly Detection on Attributed Graphs

Early methods for node anomaly detection in static attributed graphs, such as AMEN [299], Radar [300], and Anomalous [301], are not based on deep learning. These methods work well on low-dimensional attributed graphs, but their performance is limited on complex graphs with high-dimensional node attributes.

Recently, deep learning-based methods, including DOMINANT [284], AnomalyDAE [275], and GUIDE [279], have been proposed for GAD. These methods usually employ graph autoencoders to encode nodes followed by decoders to reconstruct the adjacency matrix and/or node attributes. As a result, nodes with large reconstruction errors are considered anomalies. Despite their superior performance to non-deep learning methods, these reconstruction-based methods still suffer from sub-optimal performance, as reconstruction is a generic unsupervised learning objective. Besides, these methods require the full attribute and adjacency matrices as model input, making them unsuitable or even impossible for large graphs.

7.2.2 Self-Supervised Learning for Graph Anomaly Detection

Graph SSL aims to learn a model by using supervision signals generated from the graph itself, without relying on human-annotated labels [274]. It has achieved promising performance on typical graph mining tasks such as representation learning [302] and graph classification [303]. [278] first applied SSL to the GAD problem. Their proposed method CoLA performs single scale comparison (node-subgraph) for anomaly detection. However, ANEMONE [277] argues that modeling the relationships in a single contrastive perspective leads to limited capability of capturing complex anomalous patterns. Hence, they propose additional node-node contrast. Additionally, GRADATE [285] and M-MAG [304] combines various multi-contrast objectives, namely node-node, node-subgraph, and subgraph-subgraph contrasts for node anomaly detection. To achieve better performance, SL-GAD [276] combines multi-view contrastive learning and generative attribute regression, while Sub-CR [286] combines multi-view contrastive learning and graph autoencoder. Finally, CONAD [280] considers both contrastive learning and generative reconstruction for better node anomaly detection.

7.2.3 Automated Self-Supervised Learning

Seminal work on *automated data augmentation for images* [305, 306] was followed by work improving [306] via faster searching mechanisms [307, 308, 309] or advanced

optimization methods [310, 311, 312].

In the context of *automated data augmentation for graphs*, related work exists on graph representation learning [313, 296, 314, 287, 297, 295], node classification [315, 316], and graph-level classification [317, 318, 297]. For example, JOAO [295] learns the sampling distribution of a set of predefined graph augmentations. AD-GCL [296] designs a learnable edge dropping augmentation and employs adversarial training strategy, and AutoGCL [297] proposes a learnable graph view generator that learns a probability distribution over the node-level augmentations. Further, [317] augment graph data samples, while [318] perturb the representation vector. However, these methods focus on other typical graph learning tasks and it is unclear how to use them for unsupervised GAD.

7.2.4 Automated Anomaly Detection

Recent studies [319, 320, 321, 322] pointed out that unsupervised anomaly detection methods tend to be highly sensitive to the values of their hyperparameters (HPs). For example, [319] show that a 10x performance difference is observed for LOF [173] by changing the number of nearest neighbors. Even more, [321] indicate that deep anomaly detection methods suffer more from such HP sensitivity issues. Concretely, [322] demonstrate that RAE [323] exhibits a 37x performance difference with different HPs configurations.

To tackle this issue, automated HP tuning and model selection for unsupervised anomaly detection has received increasing but insufficient attention; [320] present an overview. Inspired by [320, 322], we subdivide existing approaches into two main categories:

- *Supervised evaluation* methods which require ground-truth labels although anomaly detection algorithms are unsupervised. Methods include PyODDS [324], TODS [325], AutoOD [326], and AutoAD [327];
- *Unsupervised evaluation* methods which do not require ground-truth labels. They include
  - randomly selecting an HP configuration;
  - selecting an HP configuration via an internal evaluation strategy [328, 329, 330, 331];
  - averaging the outputs of a set of randomly selected HP configurations [332];

– meta-learning based methods [333, 334, 322].

However, existing automated anomaly detection methods are primarily designed for non-graph data.

7.3 Problem Statement

We utilize lowercase letters, bold lowercase letters, uppercase letters, and calligraphic fonts to represent scalars (x), vectors ($\mathbf{x}$), matrices ($\mathbf{X}$), and sets ($\mathcal{X}$), respectively.

Definition 7.1 (Attributed Graph)**.** We denote an attributed graph as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$, where $\mathcal{V} = \{v_1, ..., v_n\}$ is the set of nodes. Besides, $\mathcal{E} = \{e_{ij}\}_{i,j\in\{1,...,n\}}$ is the set of edges, where $e_{ij} = 1$ if there exists an edge between v_i and v_j and $e_{ij} = 0$ otherwise. Moreover, $\mathbf{X} \in \mathbb{R}^{n\times d}$ represents the node attribute matrix, where the i-th row vector $\mathbf{x}_i$ means the node attribute of v_i.

Formally, we consider unsupervised node anomaly detection on attributed graphs (dubbed GAD hereafter), which is defined as follows:

Problem 1 (Node Anomaly Detection on Attributed Graph)**.** *Given an attributed graph as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$, we aim to learn an anomaly scoring function $f(\cdot)$ that assigns an anomaly score $s = f(v_i)$ to each node v_i, with a higher score representing a higher degree of being anomalous. Next, the anomaly scores are used to rank the nodes such that the top-k nodes can be considered as anomalies.*

In this paper, we consider the *transductive unsupervised anomaly detection* setting: the graph containing both normal and abnormal nodes are given at the training stage. Node labels are not accessible during the training stage and they are only used for performance evaluation. Importantly, the labels of nodes are **not** (and should not be) used for HP tuning under this unsupervised setting.

Formally, we consider the hyperparameter optimization problem for unsupervised graph anomaly detection (dubbed HPO for GAD):

Problem 2 (HPO for GAD)**.** *Given a graph $\mathcal{G}$ without labels and a graph anomaly detection algorithm $f(\cdot)$ with hyperparameter space $\mathbf{\Lambda}$, we aim to identify a hyperparameter configuration $\boldsymbol{\lambda} \in \mathbf{\Lambda}$ such that the resulting model $f(\boldsymbol{\lambda})$ can achieve the best performance on $\mathcal{G}$. I.e., suppose $\boldsymbol{\lambda}$ consists of K different hyperparameters $\{\boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_k, ..., \boldsymbol{\lambda}_K\}$, where $\boldsymbol{\lambda}_k \in \mathbf{\Lambda}_k$ can be discrete or continuous, we aim to find*

$$\underset{\boldsymbol{\lambda}_1\in\mathbf{\Lambda}_1,...,\lambda_k\in\mathbf{\Lambda}_k,...,\lambda_K\in\mathbf{\Lambda}_K}{\arg\max} \; \mathit{Metric}\left[f(\boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_k, ..., \boldsymbol{\lambda}_K; \mathcal{G})\right], \tag{7.1}$$

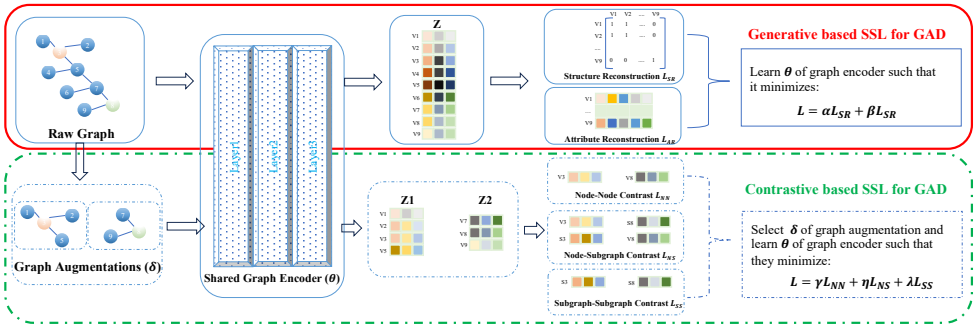


Figure 7.2: Self-supervised learning based graph anomaly detection methods can be subdivided into *generative based* methods and *contrastive based* methods. A *generative based* method generally involves *graph structure reconstruction* and *node attributes reconstruction*. A *contrastive based* method usually consists of a *graph augmentation module* and a *contrastive learning module*.

where $Metric[\cdot]$ *is a given performance metric.*

7.4 SSL for Unsupervised GAD

In this section, we first revisit existing self-supervised learning methods for "unsupervised" graph anomaly detection, followed by an analysis and experiments to showcase pitfalls in existing studies.

7.4.1 Existing SSL for "Unsupervised" GAD

Figure 7.2 shows how existing SSL based GAD methods can be divided into *generative* methods and *contrastive* methods.

That is, a *generative* method usually consists of two individual SSL tasks, namely 1.1) *structure reconstruction* that aims to reconstruct the adjacency matrix, and 1.2) *attribute reconstruction* that aims to reconstruct the node attribute matrix. On this basis, the attribute reconstruction error and the structure reconstruction error are combined to obtain an anomaly score, where higher reconstruction error indicates a higher degree of anomalousness.

Meanwhile, a *contrastive* method often consists of two modules: 2.1) *data augmentation* module, and 2.2) *contrastive learning* module. First, for each target node, the *data augmentation* module utilizes one augmentation function $f(\boldsymbol{\delta})$ to produce augmented samples, which usually include positive samples and negative samples. The

scenario of using multiple augmentation functions can be obtained in a similar way. Second, three contrastive perspectives can be applied to contrast positive pairs and negative pairs: 2.2.1) *node-node contrast* that contrasts node embedding with node embedding, and 2.2.2) *node-subgraph contrast* that contrasts node embedding with subgraph embedding, and 2.2.3) *subgraph-subgraph contrast* that contrasts subgraph embedding with subgraph embedding.

7.4.2 Pitfalls in Existing Methods

In this subsection, we revisit existing SSL-based unsupervised GAD methods by checking the following three aspects for each method:

- Which SSL framework does the method employ: *generative*, *contrastive*, or both?
- How many SSL-specific hyperparameters are involved? (E.g., combination weights and others.)
- How are values for key SSL hyperparameters chosen? (E.g., the ratio of node attribute masking or dropping edges, and the combination weights of multiple loss functions?)

By doing so, we point out that these studies have noticeable pitfalls. More importantly, we perform experiments to show that the high performance that these methods claim to achieve is often strongly overestimated due to label leakage issues (cf. Table 7.1).

Due to space constraints and to enhance readability, we revisit three representative SSL-based GAD algorithms in the main paper, including a contrastive method: ANEMONE [277], a generative method: AnomalyDAE [275], and a combined contrastive and generative method: SL-GAD [276].

Revisiting ANEMONE

ANEMONE [277] is a *contrastive* method for unsupervised GAD.

Graph Augmentation Module. A single graph augmentation operation is used, namely Random Ego-Nets generation with a fixed size K. Specifically, taking the target node as the center, they employ RWR [335] to generate two different subgraphs as ego-nets with a fixed size K. This results in one critical HP, namely K.

Contrast Learning Module. Two contrast perspectives are considered: 1) node-node contrast between the embedding of a masked target node within the ego-net and

the embedding of the original node, leading to loss term $\mathcal{L}_{NN}$, and 2) node-subgraph contrast within each view, leading to loss term $\mathcal{L}_{NS}$. These loss terms are combined as $\mathcal{L} = (1-\alpha)\mathcal{L}_{NN} + \alpha\mathcal{L}_{NS}$, where $\alpha \in [0, 1]$ is the trade-off HP, giving one more critical HP, namely α.

HPs Sensitivity & Tuning. By using ground-truth label information, they heuristically set α to $0.8, 0.6, 0.8$ on Cora, CiterSeer, and PubMed respectively, and report the corresponding results. The setting of K is not studied, and is set to 4 for all datasets.

Revisiting AnomalyDAE

AnomalyDAE [275] is a *generative* method using autoencoders (based on GNNs) for unsupervised GAD.

Generative Framework. AnomalyDAE consists of two components: 1) an attribute autoencoder to reconstruct the node attributes, where the encoder consists of two non-linear feature transform layers and the decoder is simply a dot product operation. This leads to the loss term $\mathcal{L}_A$, and $\mathcal{L}_A$ is associated with a penalty HP η; and 2) a structure autoencoder to reconstruct the structure, where the encoder is based on GAT [336] and the decoder is a dot product operation followed by a *sigmoid* function. This leads to the loss term $\mathcal{L}_S$, and $\mathcal{L}_S$ is associated with a penalty HP θ.

Their overall optimization objective is then defined as $\mathcal{L} = \alpha\mathcal{L}_S + (1-\alpha)\mathcal{L}_A$, where $\alpha \in (0, 1)$ balances the two objectives.

HPs Sensitivity & Tuning. The paper finds that the AUC usually increases first and then decreases with the increase of α. However, the specific value of α on each dataset is selected using label information. The HPs (α, η, θ) are heuristically set as $(0.7, 5, 40)$, $(0.9, 8, 90)$, $(0.7, 8, 10)$ on BlogCatalog, Flickr, and ACM respectively.

Revisiting SL-GAD

SL-GAD [276] is an unsupervised GAD method that combines both contrastive and generative objectives.

Contrastive Framework—Data Augmentation Module. The method uses a single graph augmentation operation, namely Random Ego-Nets generation with a fixed size K. Specifically, taking the target node as the center, RWR [335] is used to generate two different subgraphs as ego-nets with a fixed size K, where K controls the radius of the surrounding contexts. This gives one critical HP for graph augmentation, namely K.

Contrastive Framework—Contrast Learning Module. The Multi-View Contrastive Learning module compares the similarity between a node embedding and the embedding of sampled sub-graphs in augmented views (namely node-subgraph contrast), leading to loss terms $\mathcal{L}_{con,1}$ and $\mathcal{L}_{con,2}$. Combining those leads to contrastive objective $\mathcal{L}_{con} = \frac{1}{2}(\mathcal{L}_{con,1} + \mathcal{L}_{con,2})$.

Generative Framework. The Generative Attribute Regression module reconstructs node attributes, with the aim to achieve node-level discrimination. Specifically, they minimize the Mean Square Error between the target node's original and reconstructed attributes in augmented views, leading to loss terms $\mathcal{L}_{gen,1}$ and $\mathcal{L}_{gen,2}$. Combining those with equal weights leads to generative objective $\mathcal{L}_{gen} = \frac{1}{2}(\mathcal{L}_{gen,1}+\mathcal{L}_{gen,2})$.

The overall optimization objective is then defined as $\mathcal{L} = \alpha\mathcal{L}_{con} + \beta\mathcal{L}_{gen}$, where $\alpha, \beta \in (0, 1]$ are trade-off HPs to balance the importance of the two SSL objectives.

HPs Sensitivity & Tuning. The authors conducted a sensitive analysis and found that: 1) the performance first increases and then decreases with the increase of K. For efficiency considerations, they heuristically set the sampled subgraph size $K = 4$ for all datasets; 2) they heuristically fix $\alpha = 1$ for all datasets as they found that this achieves good performance on most datasets (with the help of label information); and 3) the selection of β is highly dependent on the specific dataset. Hence, they "fine-tune" the value of β for each dataset via selecting β from $\{0.2, 0.4, 0.6, 0.8, 1.0\}$ using labels.

Other SSL-based GAD methods

Due to space constraints, the analyses of other SSL-based GAD methods, including CoLA [278] , GRADATE [285], Sub-CR [286], CONAD [280], DOMINANT [284], GUIDE [279], and GAAN [337], are given in Appendix 7.8. These methods are all representatives of recent advancements in using SSL to conduct unsupervised graph anomaly detection, and have yielded outstanding detection performance. Likewise, however, these methods also exhibit pitfalls with regard to hyperparameter tuning, similar to those of ANEMONE [277], AnomalyDAE [275], and SL-GAD [276].

7.4.3 Sensitivity Analysis

After revisiting recent SSL-based unsupervised GAD methods, we now empirically investigate their sensitivity to SSL-related HPs in a systematic way. More concretely, we report their performance variations in terms of RUC-AUC values under different hyperparameter configurations (see Chapter 7.6 for experiment settings).

Table 7.1: Performance variation, quantified as $\frac{\max(AUC)-\min(AUC)}{\max(AUC)}$, under different hyperparameter settings on six benchmark datasets. Results are the mean of five independent runs, each with a unique random seed. OOM means out of memory, while OOR indicates that runtime exceeded a 7-day limit for a single trial. Cells marked as 'UNF' denote persistent underfitting of algorithms, even after reaching the maximum allowed training epochs (e.g., loss values change by less than 10^{-2} after 400 epochs). 'NAN' indicates execution errors caused by excessive NaN values; these cases are excluded from further analysis. Refer to Chapter 7.6 for details on the experimental setup.

	Cora	CiteSeer	PubMed	ACM	Flickr	BlogCatalog	Amazon	Facebook	Reddit	YelpChi	**Average**
CoLA	1.1%	1.7%	1.6%	4.2%	3.3%	4.7%	18.0%	3.4%	2.9%	31.9%	7.3%
ANEMONE	8.9%	6.6%	6.3%	11.3%	16.9%	16.8%	32.7%	23.9%	8.9%	37.7%	17.0%
GRADATE	6.9%	14.1%	OOM	OOM	OOR	OOR	5.9%	22.9%	OOR	OOR	12.5%
SL-GAD	17.4%	16.2%	19.5%	17.7%	16.3%	23.4%	25.4%	47.8%	21.8%	OOR	22.8%
Sub-CR	15.1%	8.3%	OOM	OOM	9.8%	6.3%	28.6%	20.3%	OOM	OOM	14.7%
CONAD	5.8%	7.0%	2.3%	UNF	OOM	OOM	17.3%	27.3%	9.7%	40.7%	15.7%
DOMINANT	5.1%	6.0%	1.9%	UNF	UNF	UNF	12.4%	19.1%	8.4%	34.5%	12.5%
A-DAE	19.1%	25.3%	23.8%	20.8%	23.6%	14.3%	48.1%	64.9%	NAN	NAN	30.0%
GUIDE	4.8%	4.8%	1.8%	UNF	8.5%	UNF	11.5%	18.6%	8.0%	34.4%	11.6%
GAAN	28.1%	30.0%	30.3%	25.6%	10.1%	7.2%	13.1%	72.6%	11.9%	11.5%	24.0%

As shown in Figure 7.1, for a typical run with different hyperparameter configurations, the performance of ANEMONE [277] can vary strongly on each of the ten datasets. Other SSL-based GAD algorithms exhibit similar behavior; extensive results and analysis are deferred to Appendix 7.8 for space reasons.

For an in-depth yet compact analysis, Table 7.1 presents average results over five independent runs when varying SSL-related hyperparameter values. Specifically, CoLA [278], GUIDE [279], DOMINANT [284], GRADATE [285], and Sub-CR [286] demonstrate moderate performance variations (namely between 7.3% and 14.7% on average). Meanwhile, CONAD [280], ANEMONE [277], SL-GAD [276], GAAN [337], and AnomalyDAE [275] suffer from large performance variations (namely ranging from 15.7% to 30.0% on average). From Chapter 7.4.2 and Appendix 7.8, we see that the results reported in existing papers are often obtained by manually tuned HPs (in a post-hoc way with label information), thereby leading to strongly overestimated performance for real-world applications where labels are not accessible. To mitigate this severe issue, we propose *AutoGAD*, a method for automating hyperparameter selection in SSL for GAD and achieving truly unsupervised graph anomaly detection. Importantly, *AutoGAD* does not need any ground-truth labels.

7.5 AutoGAD: Using Internal Evaluation to Automate SSL for GAD

Our proposed approach, called AutoGAD, consists of two parts: 1) an unsupervised performance metric, and 2) an effective search method. Importantly, and as mentioned before, the chosen performance metric—denoted $Metric[\cdot]$ in Equation 7.1—should not use any ground-truth label information, simply because this is not available in a truly unsupervised setting. We therefore propose to utilize an internal evaluation strategy, which will be elucidated later. Next, given the impracticality of evaluating an infinite number of configurations for continuous hyperparameter domains, another challenge is the efficient exploration of the search space. Chapter 7.5.2 describes a straightforward approach using discretization and grid search that works well in practice, as shown in the next section.

7.5.1 Internal Evaluation Strategy

The intuition behind the internal evaluation strategy that we use is to measure the similarity of anomaly scores within the same predicted anomaly class and the dissimilarity between anomaly scores across different predicted classes (i.e., 'anomaly' or 'no anomaly'). As we will prove later, optimizing the resulting measure is equivalent to simultaneously minimizing the false positive rate and the false negative rate. In this way, we aim to evaluate and optimize the performance of the anomaly detector under different SSL configurations *without having to rely on any ground-truth labels.*

Contrast Score Margin

The metric that we use is Contrast Score Margin [338], which was introduced before but not for graph anomaly detection, and is defined as

$$T(f) = \frac{\hat{\mu}_{\mathbf{O}} - \hat{\mu}_{\mathbf{I}}}{\sqrt{\frac{1}{k}(\hat{\delta}^2_{\mathbf{O}} + \hat{\delta}^2_{\mathbf{I}})}}, \tag{7.2}$$

where $\hat{\mu}_{\mathbf{O}}$ and $\hat{\delta}^2_{\mathbf{O}}$ denote the average and variance of the anomaly scores of the k predicted anomalous objects ($\hat{\mathbf{O}}$), respectively. Moreover, $\hat{\mu}_{\mathbf{I}}$ and $\hat{\delta}^2_{\mathbf{I}}$ represent the average and variance of the anomaly scores of the k predicted normal objects ($\hat{\mathbf{I}}$) with the highest scores, respectively. Intuitively, the metric focuses on the k predicted normal objects that are most similar to the k predicted anomalous objects, and aims

to measure the margin of the anomaly scores between them. It only takes linear time with respect to n to compute.

Analysis

We now analyze why the internal evaluation metric Contrast Score Margin should work for our purposes.

Theorem 1 (Minimizing False Positives and Negatives)**.** *For an anomaly detector $f(\cdot)$ on dataset $\mathbf{X}$, assume the anomaly scores of the top k true anomalies ($\mathbf{O}$) have the expected value $\mu_{\mathbf{O}}$ and variance $\delta^2_{\mathbf{O}}$, and the anomaly scores of the top k true normal objects with the highest anomaly scores ($\mathbf{I}$) have the expected value $\mu_{\mathbf{I}}$ and variance $\delta^2_{\mathbf{I}}$, then maximizing T is equal to simultaneously minimizing the false positive rate and the false negative rate .*

Proof. According to *Cantelli's inequality*, which makes no assumptions on specific probability distributions, on the one hand, for $\mathbf{x} \in \mathbf{O}$ we have $P(f(\mathbf{x}) \leq \mu_{\mathbf{O}} - \alpha) \leq \frac{\delta^2_{\mathbf{O}}}{\delta^2_{\mathbf{O}}+\alpha^2}$, where $\alpha \geq 0$ is a small constant chosen based on a desired bound on the false negative. By replacing $\alpha = a\delta_{\mathbf{O}}$, we have $P(f(\mathbf{x}) \leq \mu_{\mathbf{O}} - a\delta_{\mathbf{O}}) \leq \frac{1}{1+a^2}$, which is the False Negative Bound. In other words, $f(x)$ has a maximum probability of $\frac{1}{1+a^2}$ to be less than $\mu_{\mathbf{O}} - a\delta_{\mathbf{O}}$.

On the other hand, for $\mathbf{y} \in \mathbf{I}$ we have $P(f(\mathbf{y}) \geq \mu_{\mathbf{I}} + \beta) \leq \frac{\delta^2_{\mathbf{I}}}{\delta^2_{\mathbf{I}}+\beta^2}$, where $\beta \geq 0$ is a small constant chosen based on a desired bound on the false positive. By replacing $\beta = b\delta_{\mathbf{I}}$, we have $P(f(\mathbf{y}) \geq \mu_{\mathbf{I}} + b\delta_{\mathbf{I}}) \leq \frac{1}{1+b^2}$, which is the False Positive Bound. In other words, $f(y)$ has a maximum probability of $\frac{1}{1+b^2}$ to be larger than $\mu_{\mathbf{I}} + b\delta_{\mathbf{I}}$.

Furthermore, $(\mu_{\mathbf{O}} - a\delta_{\mathbf{O}}) - (\mu_{\mathbf{I}} + b\delta_{\mathbf{I}}) = (\mu_{\mathbf{O}} - \mu_{\mathbf{I}}) - (b\delta_{\mathbf{I}} + a\delta_{\mathbf{O}})$. Hence, to ensure a small false positive rate and a small false negative rate, we want $\mu_{\mathbf{O}} - \mu_{\mathbf{I}}$ to be as large as possible while $b\delta_{\mathbf{O}} + a\delta_{\mathbf{I}}$ as small as possible. In fact, this is equivalent to optimize the Contrast Score Margin, i.e.,

$$T(f) = \frac{\mu_{\mathbf{O}} - \mu_{\mathbf{I}}}{\sqrt{\frac{1}{k}(\delta^2_{\mathbf{O}} + \delta^2_{\mathbf{I}})}}$$

Note that if an anomaly detector $f(\cdot)$ produces a perfect anomaly detection result, i.e., for any $\mathbf{x} \in \mathbf{O}$ and any $\mathbf{y} \in \mathbf{X} \setminus \mathbf{O}$, we have $f(\mathbf{x}) > f(\mathbf{y})$, then we will obtain $\mu_{\mathbf{O}} - \mu_{\mathbf{I}} > 0$. In another extreme, if $f(\cdot)$ produces a poor anomaly detection result, i.e., for all $\mathbf{x} \in \mathbf{O}$ and any $\mathbf{y} \in \mathbf{X} \setminus \mathbf{O}$, we have $f(\mathbf{x}) < f(\mathbf{y})$, then we will obtain $\mu_{\mathbf{O}} - \mu_{\mathbf{I}} < 0$. Meanwhile, if an anomaly detector $f(\cdot)$ produces a random result, i.e.,

for some $\mathbf{x} \in \mathbf{O}$ and any $\mathbf{y} \in \mathbf{X} \backslash \mathbf{O}$, we have $f(\mathbf{x}) < f(\mathbf{y})$, then we may obtain $\mu_{\mathbf{O}} - \mu_{\mathbf{I}} < 0$ or $\mu_{\mathbf{O}} - \mu_{\mathbf{I}} \approx 0$. □

Improvements and Remarks

In practice we observed that Equation 7.2 is not always stable. Possible reasons are that 1) the proportion of anomalies is usually very small (namely less than 5% in most datasets); and 2) the exact number of anomalies is generally not known (even for a dataset with injected anomalies, there may exist some natural samples that exhibit similar behaviors as anomalies). Therefore, we propose to modify Equation 7.2 as follows:

$$T(f) = \frac{\hat{\mu}_{\mathbf{O}} - \tilde{\mu}_{\mathbf{I}}}{\sqrt{\hat{\delta}^2_{\mathbf{O}} + \tilde{\delta}^2_{\mathbf{I}}}}, \tag{7.3}$$

where $\hat{\mu}_{\mathbf{O}}$ and $\hat{\delta}^2_{\mathbf{O}}$ denote the average and variance of the anomaly scores of the k predicted anomalous objects, respectively. Importantly, $\tilde{\mu}_{\mathbf{I}}$ and $\tilde{\delta}^2_{\mathbf{I}}$ represent the average and variance of the anomaly scores of the remaining $n - k$ objects, respectively. This change should lead to more stable performance compared to using anomaly scores of the top-k predicted normal objects in Equation 7.2. This is because the true labels are not accessible, and thus we utilize the pseudo-labels to identify the top-k anomalous and the top-k normal objects. However, the pseudo-labels of the top-k "pseudo-normal" objects may not be reliable due to the two facts stated above.

Moreover, to ensure the effectiveness of this internal evaluation strategy, we have to make sure that: 1) we use the same algorithm with different hyperparameter configurations; and 2) the scales of the loss values are approximately the same when combining multiple loss functions in the same algorithm. In other words, we should not directly use the strategy to select among different heterogeneous anomaly detection algorithms (please refer to Appendix 7.8 for empirical evidence of this).

7.5.2 Discretization and Grid Search

To find the optimal hyperparameter configuration, we first perform discretization of the continuous search space and then conduct grid search. The corresponding pseudo-code is provided in Algorithm 6, with a detailed explanation presented below.

Discretization of Continuous Search Space (Lines 1–2). To make the overall search process feasible, we discretize the hyperparameter space. Assume we are given a GAD algorithm $f(\cdot)$ with its set of hyperparameters $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$. Without loss of generality, we assume there are L different hyperparameters and let $\boldsymbol{\lambda} = \{\lambda^{(1)}, \lambda^{(2)}, \ldots, \lambda^{(L)}\}$,

Algorithm 6 Grid Search for Anomaly Detector Hyperparameter Optimization

Input: Graph anomaly detection algorithm $f(\cdot)$, graph $\mathcal{G}$, hyperparameter domains $\mathbf{\Lambda} = \{\mathbf{\Lambda}^{(1)}, \ldots, \mathbf{\Lambda}^{(L)}\}$, internal evaluation function $T(\cdot)$
Output: Best hyperparameter configuration $\boldsymbol{\lambda}_{\text{best}}$
1: Discretize each continuous domain $\mathbf{\Lambda}^{(l)}$ into a finite set if necessary
2: Generate hyperparameter search set $\boldsymbol{\lambda}_{\text{search}} = \{\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_M\}$ where $M = \prod_{l=1}^{L} |\mathbf{\Lambda}^{(l)}|$
3: Initialize best score $t_{\text{best}} \leftarrow -\infty$ and best configuration $\boldsymbol{\lambda}_{\text{best}} \leftarrow \emptyset$
4: **for** each $\boldsymbol{\lambda}_m \in \boldsymbol{\lambda}_{\text{search}}$ **do**
5: Compute anomaly scores $\boldsymbol{s}_m(\mathcal{G}) = f(\boldsymbol{\lambda}_m; \mathcal{G})$
6: Compute evaluation score $t_m(\mathcal{G}) = T(\boldsymbol{s}_m(\mathcal{G}))$
7: **if** $t_m(\mathcal{G}) > t_{\text{best}}$ **then**
8: Update $t_{\text{best}} \leftarrow t_m(\mathcal{G})$
9: Update $\boldsymbol{\lambda}_{\text{best}} \leftarrow \boldsymbol{\lambda}_m$
10: **end if**
11: **end for**
12: **return** $\boldsymbol{\lambda}_{\text{best}}$

where each $\lambda^{(l)} \in \mathbf{\Lambda^{(l)}}$ for $l = 1, 2, \ldots, L$. If a hyperparameter domain $\mathbf{\Lambda^{(l)}}$ is continuous, we discretize it into a finite set of values (with cardinality $|\mathbf{\Lambda^{(l)}}|$). This results in M possible hyperparameter configurations, represented by the set $\boldsymbol{\lambda}_{\text{search}} = \{\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_m, \ldots, \boldsymbol{\lambda}_M\}$, where $\boldsymbol{\lambda}_m = \{\lambda_m^{(1)}, \lambda_m^{(2)}, \cdots, \lambda_m^{(L)}\}$ and $M = \prod_{l=1}^{L} |\mathbf{\Lambda^{(l)}}|$.

Grid Search (Lines 3–11). Once the hyperparameter search space is discretized, we apply grid search to evaluate each configuration. For each hyperparameter configuration $\boldsymbol{\lambda}_m \in \boldsymbol{\lambda}_{\text{search}}$, we run the GAD algorithm $f(\boldsymbol{\lambda}_m)$ on the given graph $\mathcal{G}$ to produce a vector of anomaly scores $\boldsymbol{s}_m(\mathcal{G}) = f(\boldsymbol{\lambda}_m; \mathcal{G})$. These scores are evaluated using an internal unsupervised performance metric $T(\cdot)$ (with Equation 7.3) to yield a final score $t_m(\mathcal{G}) = T(\boldsymbol{s}_m(\mathcal{G}))$. The configuration that maximizes $T(\cdot)$ is selected as the optimal values of hyperparameters.

Note that more advanced strategies than grid search, such as SMBO-based optimization [339], could be employed (see Appendix 7.8 for an example). However, these methods often introduce additional hyperparameters (whose tuning may be non-trivial), which contradicts our goal of automated anomaly detection.

7.6 Experiments

We aim to answer the following research questions (RQ):

RQ1 How sensitive are existing SSL-based GAD methods to the values of their hy-

Table 7.2: Summary of datasets: anomalies in Cora, CiteSeer, PubWeb, ACM, BlogCatalog, and Flickr are synthetically injected following established methods [277, 278], while Amazon, Facebook, Reddit, and YelpChi contain real-world anomalies.

Dataset	#Nodes	#Edges	#Attributes	#Anomalies
Cora [340]	2708	11060	1433	138(5.1%)
CiteSeer [340]	3327	4732	3703	150(4.5%)
PubMed [340]	19717	44338	500	150(2.5%)
ACM [341]	16484	71980	8337	600(3.6%)
BlogCataLog [342]	5196	171743	8189	300(5.8%)
Flickr [342]	7575	239738	12407	450(5.9%)
Amazon [343]	10244	175608	25	693(6.7%)
Facebook [344]	1081	55104	576	27(2.5%)
Reddit [345]	10984	168016	64	366(3.3%)
YelpChi [346]	24741	49315	32	1217(4.9%)

perparameters?

RQ2 How effective is *AutoGAD* in tuning SSL-related hyperparameter values for these methods?

We describe the experiment settings, including the datasets, baselines, evaluation metrics, and software and hardware used, which is followed by the experiment results and their interpretation.

7.6.1 Datasets

We use three popular citation networks, namely Cora, Citeseer, and Pubmed [340] with injected anomalies, one social network Flickr (less homophily) with injected anomalies, ACM as well as BlogCataLog with injected anomalies. Particularly, we follow the methods used by ANEMONE [277] and CoLA [278] to inject structure and contextual anomalies. Note that [294] have slightly modified this injection procedure. Following [347], we also consider four commonly-used graph datasets with real anomalies: Amazon [343], Facebook [344], Reddit [345], and YelpChi [346]. The resulting datasets are summarized in Table 7.2.

7.6.2 Baselines

We study the performance of the following SSL-based graph anomaly detection methods:

- Generative methods: DOMINANT [284], AnomalyDAE [275], GUIDE [279], GAAN [337];
- Contrastive methods (and some also generative): CoLA [278], ANEMONE [277], GRADATE [285], SL-GAD [276], Sub-CR [286], CONAD [280].

Particularly, the SSL-related HPs for each GAD algorithm and their discretized search spaces are given in Table 7.6 in the Appendix. These GAD methods are further summarized in Table 7.7 in the Appendix.

7.6.3 Evaluation Metrics

To evaluate the effectiveness of various GAD algorithms, we utilize the ROC-AUC metric [348] (*AUC* for short hereinafter), where a value approaching 1 denotes the best possible performance.

Moreover, to quantify the performance variation of an individual GAD method under different SSL-related HP configurations, we define the following *performance variation* metric:

$$\frac{\max(AUC) - \min(AUC)}{\max(AUC)}, \tag{7.4}$$

where $\max(AUC)$ and $\min(AUC)$ represent the maximum and minimum of achieved AUC values for the evaluated GAD algorithm with different configurations, respectively. Hence, the smaller this value is, the less sensitive the algorithm is to SSL-related HPs.

Further, we define the *performance gain over minimal AUC* as

$$\frac{\mathrm{CSM}(AUC) - \min(AUC)}{\min(AUC)}, \tag{7.5}$$

where $\mathrm{CSM}(AUC)$ indicates the AUC value obtained for the evaluated GAD algorithm when configured with the HPs selected using the Contrast Score Margin. This metric can quantify the effectiveness of our strategy relative to the worst case hyperparameter setting. Next, we define *performance gain over median AUC* as

$$\frac{\mathrm{CSM}(AUC) - \mathrm{median}(AUC)}{\mathrm{median}(AUC)}, \tag{7.6}$$

where $\mathrm{median}(AUC)$ represents the median of the obtained AUC values for the GAD algorithm with different configurations. Thus, if the value of this metric is positive,

the GAD algorithm configured with our selected HPs can at least outperform its counterparts configured with 50% of the other sampled hyperparameter values.

Furthermore, we define *performance gain over maximal AUC* as

$$\frac{\text{CSM}(AUC) - \max(AUC)}{\max(AUC)}, \tag{7.7}$$

where $\max(AUC)$ represents the maximum of the obtained AUC values for the GAD algorithm with different configurations. Thus, if the value of this metric is close to zero, the GAD algorithm configured with our selected HPs can approximately achieve the best possible performance.

7.6.4 Software and Hardware

All algorithms are implemented in Python 3.8 (using PyTorch [109] and PyTorch Geometric [110] libraries when applicable) and ran on workstations equipped with AMD EPYC7453 CPUs (with 64GB RAM) and/or Nvidia RTX4090 GPUs (with 24.0 GB video memory). All code and datasets are available on GitHub[1].

7.6.5 Results and Analysis

Table 7.3: Performance gain over minimal AUC defined as $\frac{\text{CSM}(AUC)-\min(AUC)}{\min(AUC)}$. Results are averaged on five independent runs. CSM is contrast score margin defined in Equation 7.3, while OOM, OOR, UNF, and NAN convey the same meanings as in Table 7.1.

	Cora	CiteSeer	PubMed	ACM	Flickr	BlogCatalog	Amazon	Facebook	Reddit	YelpChi	**Average**
CoLA	0.5%	1.2%	1.6%	2.8%	2.2%	4.7%	22%	1.8%	1.5%	2.5%	4.1%
ANEMONE	8.6%	5.9%	6.6%	6.8%	15.8%	19.7%	44.7%	28.1%	4.0%	11.9%	15.2%
GRADATE	4.0%	14.3%	OOM	OOM	OOR	OOR	4.3%	29.7%	OOR	OOR	13.1%
SL-GAD	21.2%	19.1%	23.7%	21.3%	18.2%	30.4%	13.0%	16.3%	15.8%	OOR	19.9%
Sub-CR	16.2%	4.3%	OOM	OOM	4.3%	2.4%	19.2%	25.3%	OOM	OOM	12.0%
CONAD	5.4%	2.3%	2.1%	UNF	OOM	OOM	6.5%	18.3%	2.4%	24.3%	8.8%
DOMINANT	5.2%	1.3%	1.8%	UNF	UNF	UNF	13.7%	14.9%	0.8%	28.0%	9.4%
A-DAE	11.3%	4.6%	11.9%	5.6%	30.8%	32.2%	67.3%	114.6%	NAN	NAN	34.8%
GUIDE	5.0%	1.2%	1.8%	UNF	0.1%	UNF	9.4%	14.3%	3.8%	28.8%	8.1%
GAAN	7.7%	34.1%	43.6%	5.6%	1.3%	6.6%	12.4%	77.9%	0.4%	14.5%	20.4%

We answer the research questions as follows:

RQ1: Sensitivity of SSL-based GAD methods to HPs

The results are summarized in Table 7.1 for five independent runs. Typical runs are depicted in Figure 7.1 and in Figures 7.5-7.13 in Appendix 7.8. We briefly analyzed the

[1] `https://github.com/ZhongLIFR/AutoGAD2024`

Table 7.4: Performance gain over median AUC defined as $\frac{\text{CSM}(AUC)-\text{median}(AUC)}{\text{median}(AUC)}$. Results are averaged on five independent runs. CSM is contrast score margin defined in Equation 7.3, while OOM, OOR, UNF, and NAN convey the same meanings as in Table 7.1. For enhanced readability, cells are color-coded based on their values, as specified in the legend.

Dark Orange	Light Orange	Light Green	Dark Green	Grey
$(-\infty, -5.0\%]$	$(-5.0\%, 0.0\%)$	$[0.0\%, 5.0\%)$	$[5.0\%, +\infty)$	Excluded

	Cora	CiteSeer	PubMed	ACM	Flickr	BlogCatalog	Amazon	Facebook	Reddit	YelpChi	**Average**
CoLA	-0.1%	0.1%	0.2%	-0.7%	0.6%	2.0%	15.4%	0.1%	-0.2%	-3.3%	1.4%
ANEMONE	0.3%	1.0%	1.5%	-0.8%	2.0%	7.2%	26.4%	3.5%	-2.4%	1.6%	4.0%
GRADATE	-0.6%	4.0%	OOM	OOM	OOR	OOR	0.7%	18.3%	OOR	OOR	5.6%
SL-GAD	3.3%	3.7%	4.8%	4.3%	2.8%	5.0%	-1.4%	-31.6%	-2.0%	OOR	-1.2%
Sub-CR	-1.6%	-0.4%	OOM	OOM	-3.2%	-2.2%	2.6%	9.8%	OOM	OOM	0.8%
CONAD	4.0%	1.2%	1.5%	UNF	OOM	OOM	-3.7%	2.5%	-3.1%	1.5%	0.6%
DOMINANT	3.7%	0.5%	1.3%	UNF	UNF	UNF	4.4%	-1.8%	-3.0%	4.8%	1.4%
A-DAE	2.9%	-3.0%	-2.9%	-4.1%	0.9%	-3.4%	2.6%	0.7%	NAN	NAN	-0.8%
GUIDE	3.8%	0.6%	1.5%	UNF	-0.3%	UNF	2.1%	-2.3%	0.2%	5.3%	1.4%
GAAN	2.8%	27.5%	35.6%	3.5%	0.7%	4.7%	-2.4%	-45.6%	-1.3%	-0.5%	2.5%

Table 7.5: Performance gain over maximal AUC defined as $\frac{\text{CSM}(AUC)-\max(AUC)}{\max(AUC)}$. Results are averaged on five independent runs. CSM is contrast score margin defined in Equation 7.3, while OOM, OOR, and NAN convey the same meanings as in Table 7.1.

	Cora	CiteSeer	PubMed	ACM	Flickr	BlogCatalog	Amazon	Facebook	Reddit	YelpChi	**Average**
CoLA	-0.6%	-0.5%	-0.1%	-1.5%	-1.3%	-0.3%	0%	-1.7%	-1.5%	-30.2%	-3.8%
ANEMONE	-1.1%	-1.1%	-0.2%	-5.4%	-3.9%	-0.4%	-2.6%	-2.6%	-5.3%	-30.3%	-5.3%
GRADATE	-3.2%	-1.9%	OOM	OOM	OOR	OOR	-1.8%	0.0%	OOR	OOR	-1.7%
SL-GAD	-0.4%	-0.3%	-0.4%	-0.4%	-1.2%	-0.2%	-15.8%	-39.4%	-9.4%	OOR	-7.5%
Sub-CR	-3.8%	-4.5%	OOM	OOM	-5.9%	-4.1%	-15.3%	-0.2%	OOM	OOM	-5.6%
CONAD	-0.8%	-4.9%	-0.2%	UNF	OOM	OOM	-11.9%	-14.0%	-7.6%	-26.3%	-9.3%
DOMINANT	-0.2%	-4.8%	-0.1%	UNF	UNF	UNF	-0.6%	-7.1%	-7.8%	-16.2%	-5.3%
A-DAE	-10.0%	-21.8%	-14.6%	-16.4%	-0.1%	-4.8%	-18.5%	-26.3%	NAN	NAN	-14.1%
GUIDE	0%	-3.6%	0%	UNF	-8.4%	UNF	-3.1%	-7.1 %	-4.6%	-15.4%	-5.3%
GAAN	-22.6%	-6.7%	0%	-21.6%	-8.9%	-1.2%	-2.6%	-53.7%	-11.6%	-0.9%	-13.0%

results in Chapter 7.4.3; more detailed analyses are given in Appendix 7.8. To recall, five out of ten algorithms show moderate performance variations, while the remaining five algorithms demonstrate large performance variations when the values of SSL-related HPs are varied. In other words, SSL-based GAD methods are (sometimes highly) sensitive to hyperparameter values.

RQ2: Effectiveness of AutoGAD in tuning SSL-related HPs

The results are summarized in Tables 7.3, 7.4 and 7.5 for five independent runs, while Figure 7.1 and Figures 7.5-7.13 depict typical runs. We have the following main observations:

1) From Table 7.3, one can see that *AutoGAD* can result in moderate *performance gain over minimal AUC* (namely between 4.1% and 13.1% on average) for CoLA,

GUIDE, CONAD, DOMINANT, Sub-CR, and GRADATE. Recall that five of these algorithms (including CoLA, GUIDE, DOMINANT, GRADATE, and Sub-CR) exhibit moderate performance variations, ranging from 7.3% to 14.7% on average. Moreover, *AutoGAD* leads to large *performance gain over minimal AUC* (namely between 6.8% and 24.1% on average) for the remaining four algorithms, which suffer from large performance variations (namely between 15.1% and 34.8% on average). Overall, *AutoGAD* is substantially better than the worst case, i.e., when one happens to select the HP values that give the smallest *AUC* value.

2) From Table 7.4, one can see that *AutoGAD* can result in positive *performance gain over median AUC* in 8 out 10 algorithms (ranging from 0.6% to 5.6% on average), implying that the HP values selected by *AutoGAD* are better than at least 50% of randomly selected HP values. Particularly, the *performance gains over median AUC* for GRADATE [285], ANEMONE [277], and GAAN [337] are 5.6%, 4.0%, and 2.5% respectively, which shows that *AutoGAD* is highly effective for these methods.

3) From Table 7.5, one can see that *AutoGAD* can result in *performance gain over max AUC* larger than -10% in 8 out 10 algorithms, implying that the HP values selected by *AutoGAD* can achieve performances that are comparable to optimal performances. For instance, the *performance gains over max AUC* for GRADATE and SL-GAD are -1.7% and -7.5% respectively, which shows that *AutoGAD* is highly effective for these methods while they show moderate or large performance variations (12.5% and 22.8% respectively).

4) Following the above observations, we check the details in Figure 7.13 for SL-GAD, Figure 7.11 for GRADATE, and Figure 7.1 for ANEMONE. For SL-GAD and GRADATE, *AutoGAD* often selects HP values better than 90% of randomly selected HPs values on most datasets. For ANEMONE, the HP values selected by *AutoGAD* often outperform 75% of randomly selected HP values.

Sensitivity Analysis

Sensitivity to k. The selection of the value of k in our experiments acknowledges the varying anomaly ratios across different datasets, implying that k should ideally differ to reflect the unique characteristics of each dataset. We operated under the assumption that the anomaly ratio within a dataset is approximately known, a premise

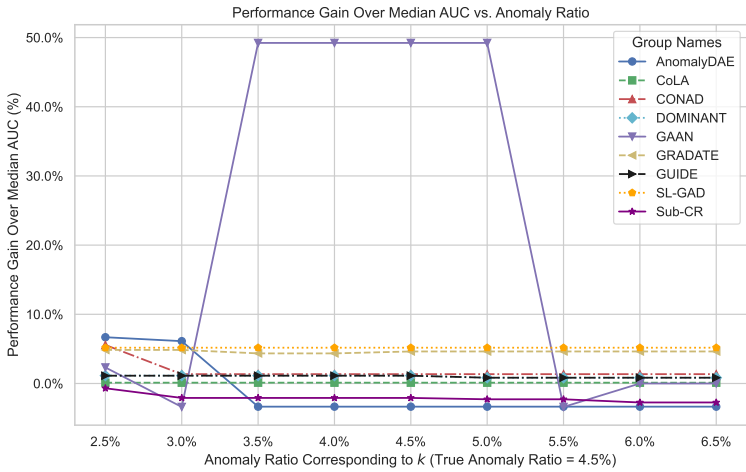


Figure 7.3: Sensitivity analysis of k (for our proposed AutoGAD) on dataset CiteSeer with all investigated SSL-based GAD algorithms. It can be seen that AutoGAD remains stable as long as k is not drastically distant from the actual anomaly ratio (namely 4.5%) all for SSL-based GAD algorithms.

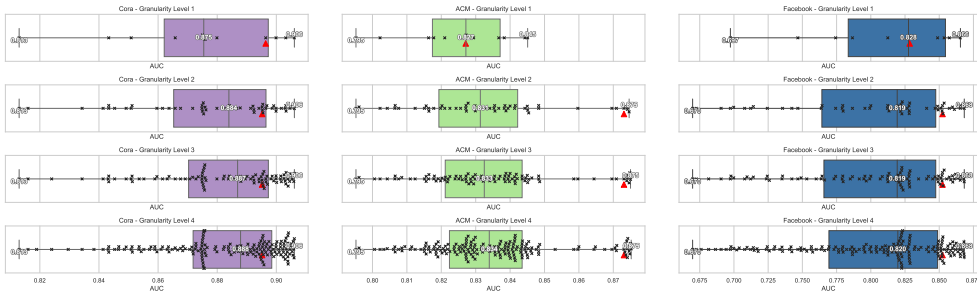


Figure 7.4: Performance of AutoGAD across different granularity levels of search grids using ANEMONE on the Cora, ACM, and Facebook datasets. Similar trends were observed for other anomaly detectors and datasets, which are omitted for brevity.

that aligns with real-world anomaly detection tasks where some prior knowledge about the frequency of anomalies is often available.

As shown in Figure 7.3, we conducted a sensitivity analysis on k to assess the stability of AutoGAD against deviations from the true anomaly ratio. The findings from this analysis indicate that the effectiveness of AutoGAD remains stable as long as k is not drastically distant from the actual anomaly ratio, reinforcing the practical applicability of our approach even when exact anomaly proportions are not precisely determined.

Sensitivity to the Granularity of the Search Grid. Acknowledging the significance of search space granularity in the performance of AutoGAD, we conduct a sensitivity analysis by varying the granularity levels of the search grids in grid search. Figure 7.4 presents representative results using ANEMONE [277] on the Cora, ACM, and Facebook datasets with four levels of search granularity, as follows:

- Granularity Level 1: $\alpha \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$, $K \in \{2, 4\}$;
- Granularity Level 2: $\alpha \in \{0, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 1\}$, $K = \{2, 3, 4, 5\}$;
- Granularity Level 3: $\alpha \in \{0, 0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 0.99, 1\}, K \in \{2, 3, 4, 5\}$;
- Granularity Level 4: $\alpha \in \{0, 0.01, 0.025, 0.05, 0.075, 0.1, 0.125, 0.15, 0.175, 0.2, 0.225, 0.25, 0.275, 0.3, 0.325, 0.35, 0.375, 0.4, 0.425, 0.45, 0.475, 0.5, 0.525, 0.55, 0.575, 0.6, 0.625, 0.65, 0.675, 0.7, 0.725, 0.75, 0.775, 0.8, 0.825, 0.85, 0.875, 0.9, 0.925, 0.95, 0.975, 0.99, 1\}, K \in \{2, 3, 4, 5, 6, 7\}$.

The results indicate that finer search grids tend to improve the performance of AutoGAD. This is expected, as the optimal value achievable in a finer search grid cannot be worse than that in a coarser grid. Similar observations were made for other anomaly detection methods and datasets, which are omitted here for brevity.

7.7 Alternative Strategies and Discussion

Internal evaluation strategies aim to assess the quality of a model based solely on internal information, without relying on external information such as ground-truth labels. Internal information can typically be derived from two sources: 1) the input samples, such as feature values of instances in tabular data or node attributes in graph

data; or 2) the anomaly scores generated by an anomaly detection model. Beyond the Contrast Score Margin [338] discussed in this paper, additional internal evaluation strategies exist for unsupervised model selection in anomaly detection. According to [298], these strategies can be categorized as *stand-alone* or *consensus-based* internal evaluation strategies; we will next discuss each category.

7.7.1 Stand-alone Internal Evaluation Strategies

Stand-alone strategies rely solely on input samples or individual anomaly detection methods (or models with specific HP configurations in our setting) and their output anomaly scores. Key methods include:

- **IROES** [349, 330] quantifies the separability of each input sample, assuming that a good anomaly detection model assigns high anomaly scores to highly separable samples. However, separability scores are defined only for tabular data, making extension to graph data non-trivial. Additionally, computing separability scores is computationally expensive, posing challenges for large datasets.
- **Mass-Volume and Excess-Mass** [328] use statistical tools to measure the quality of an anomaly scoring function. These methods operate on the raw input samples rather than anomaly scores and assume that anomalies occur in the distribution's tail. However, they are restricted to tabular data and are not applicable to graph data.
- **Clustering Validation Metrics** [350] assume that an anomaly detector divides input samples into two clusters: abnormal and normal. Clustering validation metrics, such as the Xie-Beni index [351], are then used to evaluate performance. While clustering coefficients on graphs could be analogous [352], these metrics are computationally expensive, particularly for large datasets.

7.7.2 Consensus-based Internal Evaluation Strategies

Consensus-based strategies assess the agreement among multiple anomaly detection models (or the same model with varying HP configurations in our setting). Key methods include:

- **UDR** [353] assumes that good HP configurations yield consistent results under different random initializations, while poor configurations do not. [298] repur-

posed UDR to select among heterogeneous anomaly detectors, assuming that good detectors produce consistent results across HP configurations.

- **Model Centrality** [354] hypothesizes that good models are close to the optimal model and thus to each other.
- **Model Centrality by HITS** [355] follows a similar hypothesis but employs a different computation approach.
- **Unsupervised Anomaly Detection Ensembling** [298] infers pseudo anomaly labels by aggregating outputs from a predefined subset of good models. However, this method is less feasible in our setting as there is no such pre-defined good models.

Two challenges remain when utilizing these strategies in our setting: 1) validating the underlying assumptions, which often lack theoretical justification, and 2) addressing their computational expenses, as consensus-based methods require pairwise comparisons. In contrast, Contrast Score Margin is computationally efficient, as it operates on anomaly scores rather than on raw data points and it avoids pairwise comparisons.

7.7.3 Discussion and Future Work

Although [298] demonstrated that many internal evaluation strategies perform suboptimally for selecting heterogeneous anomaly detectors, we hypothesize that some can be valuable for hyperparameter tuning within a single anomaly detection model. However, this is beyond the scope of this paper and is left for future work. The primary objectives of this paper are twofold:

- We highlight flaws in existing studies on using SSL for unsupervised graph anomaly detection. Specifically, we:
    1. Review these studies, showing that most tune HPs arbitrarily or selectively.
    2. Demonstrate empirically, through extensive experiments, that these methods are highly sensitive to HP settings. Consequently, we argue that these methods may suffer from label information leakage under unsupervised learning settings, leading to overstated performance in practical scenarios where label-based tuning is inaccessible.

- We propose an initial solution to these issues by utilizing and improving the Contrast Score Margin. This internal evaluation metric was selected for two reasons:

    1. It operates on anomaly scores rather than on raw data points and avoids pairwise computations, making it computationally efficient and suitable for large datasets.
    2. Theoretical guarantees for its properties are provided by Theorem 1, which may not hold for other internal evaluation strategies.

This paper does not aim to provide a perfect solution to the issues mentioned above. Instead, our goal is to spark interest in the research community to address these challenges. Unlike [298], we do not aim to conduct a comprehensive review and evaluation of internal evaluation strategies for SSL-based graph anomaly detection, as this requires significant computational resources and in-depth analysis. Nevertheless, we aim to explore this direction in future work by considering and potentially repurposing the internal evaluation strategies reviewed in [298]. We have described a more advanced search strategy than grid search, namely SMBO-based optimization [339], in Appendix 7.8, without experimental evaluation. This is because this method introduces additional hyperparameters and their tuning is non-trivial, contradicting our goal of automated anomaly detection. Other advanced hyper-parameter tuning methods [356, 357, 358] to speed up the search are possible, and we leave their explorations for future work.

7.8 Conclusions

SSL has received much attention in recent years, and many recent studies have explored SSL to perform unsupervised GAD. However, we found that most existing studies tune hyperparameters arbitrarily or selectively (i.e., guided by labels), and our empirical findings reveal that most methods are highly sensitive to hyperparameter settings. Using label information to tune hyperparameters in an unsupervised setting, however, is label information leakage and leads to severe overestimation of model performance. To mitigate this issue, we introduce AutoGAD, the first automated hyperparameter selection method for SSL-based unsupervised GAD. Extensive experiments demonstrate the effectiveness of our proposed strategy. Overall, we aim to raise awareness to the label information leakage issue in the unsupervised GAD field, and AutoGAD provides a first step towards achieving truly unsupervised SSL-based GAD.

Appendix A: Pitfalls in Existing Methods (Full Analysis)

A.1 CoLA

Particularly, CoLA [278] is the first *contrastive-based* framework for unsupervised GAD. The design of its *data augmentation* module and *contrast learning* module is as follows.

Data Augmentation Module They consider one type of data augmentation, subgraph sampling, to obtain local augmented view for each node. Particularly, they employ RWR [335] to generate a sub-graph with a fixed size K in subgraph sampling, resulting in one critical HP in graph augmentation, namely K.

Contrast Learning Module They consider a single contrast aspect, namely node-subgraph contrast between the embedding of the target node and the aggregated embedding of its local sug-graph, without resulting in any HPs.

HPs Sensitivity & Tuning They conducted sensitive analysis and found that the selection of subgraph size K is dependent on the specific dataset. The AUC performance usually increases first and then decreases with the increasing of K. However, for efficiency and robustness consideration, they heuristically set the sampled subgraph size $K = 4$ for all datasets.

A.2 ANEMONE

ANEMONE [277] is a *contrastive-based* framework for unsupervised GAD. They argue that modeling the relationships in a single contrastive perspective leads to limited capability of capturing complex anomalous patterns, and thus propose additional contrast perspectives as follows.

Graph Augmentation Module They consider a single graph augmentation operation, namely Random Ego-Nets generation with a fixed size K. Specifically, taking the target node as the center, they employ RWR [335] to generate two different subgraphs as ego-nets with a fixed size K. Overall, they result in one critical HP in graph augmentation, namely K.

Contrast Learning Module They consider two contrast perspectives: 1) node-node contrast between the embedding of masked target node within ego-net and the embedding of the original node, leading to loss term $\mathcal{L}_{NN}$, and 2) node-subgraph contrast within each view, leading to loss term $\mathcal{L}_{NS}$. On this basis, they combine

these loss terms as

$$\mathcal{L} = (1-\alpha)\mathcal{L}_{NN} + \alpha\mathcal{L}_{NS}$$

where $\alpha \in [0,1]$ is the trade-off HP. Hence, they result in one critical HP in graph contrast, namely α.

HPs Sensitivity & Tuning In their ablation studies: 1) by using ground-truth label information, they heuristically set α as $0.8, 0.6, 0.8$ on Cora, CiterSeer and PubMed respectively, and report the corresponding results; and 2) the setting of K was not studied, and it is set to 4 for all datasets.

A.3 GRADATE

GRADATE [285] is also a *contrastive-based* framework. They argue that subgraph-subgraph contrast is also critical in detecting graph anomalies, and design it as follows.

Data Augmentation Module They consider a single graph augmentation operation, namely Edge Modification that removes edges in the adjacency matrix as well as add the same number of edges. Concretely, they fix a proportion P, and then uniformly and randomly sample $\frac{P \cdot M}{2}$ edges from a total of M edges to remove. Meanwhile, $\frac{P \cdot M}{2}$ edges are added into the adjacency matrix. Overall, they result in one critical HP in graph augmentation, namely P.

Contrast Learning Module They consider three contrast aspects: 1) node-node contrast within each view, leading to loss term $\mathcal{L}_{NN}$), 2) node-subgraph contrast within each view, leading to loss term $\mathcal{L}_{NS}$, and 3) subgraph-subgraph contrast between original view and augmented view, leading to loss term $\mathcal{L}_{SS}$. On this basis, they combine these loss terms as

$$\mathcal{L} = (1-\beta)\mathcal{L}_{NN} + \beta\mathcal{L}_{NS} + \gamma\mathcal{L}_{SS},$$

where $\beta, \gamma \in (0,1)$ are trade-off HPs. More, $\mathcal{L}_{NN} = \alpha\mathcal{L}_{NN,1} + (1-\alpha)\mathcal{L}_{NN,2}$, and $\mathcal{L}_{NS} = \alpha\mathcal{L}_{NS,1} + (1-\alpha)\mathcal{L}_{NS,2}$, with $\mathcal{L}_{NN,1}$ and $\mathcal{L}_{NN,2}$ being the loss term in the first and second views respectively. Overall, they result in three critical HPs in graph contrast, namely the combination weights α, β, γ.

HPs Sensitivity & Tuning In their ablation studies, 1) they compared four different graph augmentation strategies, including Gaussian Noise Feature, Feature Masking, Graph Diffusion, and Edge Modification, and they found that Edge Modification performs the best across different datasets (with ground-truth labels on test data to measure the performance); 2) with the help of ground-truth label in-

formation on test data, they heuristically set (α, β) as $(0.9, 0.3), (0.1, 0.7), (0.7, 0.1)$, $(0.9, 0.3), (0.7, 0.5), (0.5, 0.5)$ on EAT, WebKB, UAT, Cora, UAI2010, and Citation respectively; 3) similarly, they set $\gamma = 1$ for all datasets; and 4) they also heuristically set $P = 0.2$ for all datasets.

A.4 SL-GAD

Different from CoLA, ANEMONE and GRADATE, SL-GAD [276] combines the *contrastive-based* framework and the *generative-based* framework for unsupervised GAD.

First, the design of the *contrastive-based* framework is as follows.

Contrastive Framework—Data Augmentation Module They consider a single graph augmentation operation, namely Random Ego-Nets generation with a fixed size K. Specifically, taking the target node as the center, they employ RWR [335] to generate two different subgraphs as ego-nets with a fixed size K, where K controls the radius of the surrounding contexts. Overall, they result in one critical HP in graph augmentation, namely K. Particularly, they indicate that other augmentation strategies such as attribute masking and edge modification may introduce extra anomalies, while random ego-nets and graph diffusion can augment data without changing the underlying graph semantic information.

Contrastive Framework—Contrast Learning Module They introduce a Multi-View Contrastive Learning module that compare the similarity between node embedding and embedding of sampled sub-graphs in augmented views (namely node-subgraph contrast), leading to two loss terms $\mathcal{L}_{con,1}$ and $\mathcal{L}_{con,2}$ corresponding to two augmented views, respectively. On this basis, they obtain the contrastive objective $\mathcal{L}_{con} = \frac{1}{2}(\mathcal{L}_{con,1} + \mathcal{L}_{con,2})$, which combines the two loss terms with equal weights.

Second, the *generative-based* framework is designed as follows.

Generative Framework They introduce a Generative Attribute Regression module that reconstructs node attributes, with the aim to achieve node-level discrimination, where the encoder is a GCN and the decoder is another GCN. Specifically, they minimize the Mean Square Error between the target node's original and reconstructed attributes in augmented views, leading to two loss terms $\mathcal{L}_{gen,1}$ and $\mathcal{L}_{gen,2}$ corresponding to two augmented views, respectively. Then they combine them with equal weights, leading to the generative objective $\mathcal{L}_{gen} = \frac{1}{2}(\mathcal{L}_{gen,1} + \mathcal{L}_{gen,2})$.

At last, their final optimization objective is defined as follows:

$$\mathcal{L} = \alpha\mathcal{L}_{con} + \beta\mathcal{L}_{gen},$$

where $\alpha, \beta \in (0, 1]$ are trade-off HPs to balance the importance of two SSL objectives.

HPs Sensitivity & Tuning They conducted sensitive analysis and found that: 1) the performance first increases and then decreases with the increasing of K. For efficiency consideration, they heuristically set the sampled subgraph size $K = 4$ for all datasets; 2) they heuristically fix $\alpha = 1$ for all datasets as they found that this achieves good performance on most datasets (with the help of label information); and 3) the selection of β is high dependent on the specific dataset. Hence, they "fine-tune" the value of β for each dataset via selecting β from $\{0.2, 0.4, 0.6, 0.8, 1.0\}$ with labels.

A.5 Sub-CR

Similar to SL-GAD, Sub-CR [286] also combines the *contrastive-based* framework and the *generative-based* framework for unsupervised GAD.

First, the design of the *contrastive-based* framework is as follows.

Contrastive Framework—Contrast Learning Module They consider two types of data augmentation: 1) subgraph sampling to obtain local augmented views for each node (so-called local view subgraph), 2) graph diffusion plus subgraph sampling (in a sequential order) to obtain global augmented views for each node (so-called global view subgraph). Particularly, they employ RWR [335] to generate a sub-graph with a fixed size K in subgraph sampling. Besides, they apply Persnonalized PageRank to power the graph diffusion [359], wherein the teleport probability α needs to be determined. Overall, they result in two critical HPs in graph augmentation, namely K and α.

Contrastive Framework—Contrast Learning Module This module consists of: 1) intra-view contrastive learning that maximizes the agreement between the node and its sub-graph level representations in the local view (with loss term $\mathbf{L}_{intra,1}$), and the agreement between the node and its sub-graph level representations in the global view (with loss term $\mathbf{L}_{intra,2}$), where they combine the local view and global view loss terms with equal weights to obtain the intra-view loss term $\mathbf{L}_{intra} = \mathbf{L}_{intra,1} + \mathbf{L}_{intra,2}$; and 2) inter-view contrastive learning that makes closer the discriminative scores of node-subgraph pairs in local view and global view, leading to the loss term $\mathbf{L}_{inter}$. On this basis, they combine the intra-view loss term and inter-view loss term with equal weights to obtain the multi-view contrastive learning loss term $\mathbf{L}_{con} = \mathbf{L}_{intra} + \mathbf{L}_{inter}$.

Second, the *generative-based* framework is designed as follows.

Generative Framework They introduce a masked Autoencoder-based Reconstruction module, where the encoder is a GCN and the decoder is a multilayer per-

ceptron with *PReLU* activation function, aiming to reconstruct the attributes of the target node based on the attributes of neighboring nodes in the local view (with loss term $\mathbf{L}_{res,1}$), and in the global view (with loss term $\mathbf{L}_{res,2}$). Next, they combine the local view and global view loss terms with equal weights to obtain the overall reconstruction loss term $\mathbf{L}_{res} = \mathbf{L}_{res,1} + \mathbf{L}_{res,2}$ for each node.

At last, their final optimization objective is defined as follows:

$$\mathcal{L} = \mathcal{L}_{con} + \gamma \mathcal{L}_{res},$$

where $\gamma \in (0, 1]$ is the trade-off HP to balance the importance of two different SSL objectives.

HPs Sensitivity & Tuning They conducted sensitive analysis and found that: 1) the selection of K is dependent on the specific dataset. However, for efficiency and performance consideration, they heuristically set the sampled subgraph size $K = 4$ for all datasets; 2) they did not discuss the setting of teleport probability α; and 3) they claim that most datasets are not sensitive to the value of γ when $\gamma > 0.4$. Hence, they heuristically set $\gamma = 0.6$ for Cora, Citeseer, Flickr, and BlogCatalog while $\gamma = 0.4$ for PubMed with the help of label information.

A.6 CONAD

Similar to SL-GAD and Sub-CR, CONAD [280] also combines the *contrastive-based* framework and the *generative-based* framework for unsupervised GAD.

First, the design of the *contrastive-based* framework is as follows.

Contrastive Framework—Data Augmentation Module They consider four different types of data augmentations, with each type of data augmentation operation corresponding to a specific type of node anomaly. They include 1) edge adding augmentation that connects a node with many other non-connected nodes (structure - high degree), 2) edge removing augmentation that removes most edges of a node (structure - outlying); 3) attribute replacement augmentation that replaces the target node's attributes with another dissimilar node's attributes (attribute - deviated), and 4) attribute scaling augmentation that scales the target node's attributes to much larger or smaller values (attribute - disproportionate); This leads to four HPs p_1, p_2, p_3, p_4, which represent the sampling probability of each augmentation strategy. Moreover, the rate r of augmented anomalies (namely modified nodes) is also a HP.

Contrastive Framework—Contrast Learning Module They consider two different contrast strategies: 1) Siamese contrast $\mathcal{L}_{SC} = \sum_{i \in \mathbf{NM}} d(\mathbf{z}_i, \hat{\mathbf{z}}_i) + \sum_{j \in \mathbf{MM}} max\{0, m-$

$d(\mathbf{z}_j, \hat{\mathbf{z}}_j)\}$ where $d(\mathbf{z}_i, \hat{\mathbf{z}}_i)$ is the distance between embeddings of node i in the original view and in the augmented view. **MM** and **NM** mean the node is modified or non-modified, respectively; 2) Triplet contrast $\mathcal{L}_{TC} = \sum max\{0, m - [d(\mathbf{z}_i, \hat{\mathbf{z}}_j) - d(\mathbf{z}_i, \mathbf{z}_j)]\}$ where $d(\mathbf{z}_i, \mathbf{z}_j)$ is the distance between embeddings of node i and its neighbor j in the original view, and $d(\mathbf{z}_i, \hat{\mathbf{z}}_j)$ is the distance between embeddings of node i in the original view and its neighbor j in the augmented view. Particularly, the contrastive loss term $\mathcal{L}_{Contr} = \mathcal{L}_{SC}$ or $\mathcal{L}_{Contr} = \mathcal{L}_{TC}$. This module contains a HP, namely the margin m.

Second, the *generative-based* framework is designed as follows.

Generative Framework This framework consists of two components: 1) an attribute autoencoder to reconstruct the node attributes, where the encoder is a GAT [336] and the decoder is another GAT. This leads to the loss term L_A; and 2) a structure autoencoder to reconstruct the structure, where the encoder is a GAT and the decoder is a dot product operation followed by a *sigmoid* function (namely $sigmoid(\mathbf{z}^t\mathbf{z})$). This leads to the loss term L_S. Combining these two loss terms leads to a loss term $L_{Recon} = \lambda L_A + (1-\lambda)L_S$, where $\lambda \in (0, 1)$ is a trade-off HP to balance the two reconstruction errors. Unlike SL-GAD and Sub-CR, CONAD requires the whole adjacency matrix and node attribute matrix as input, and thus it can reconstruct the graph structure, making it unsuitable to large graphs. In contrast, SL-GAD and Sub-CR only require subgraphs as inputs, and thus are unable to perform structure reconstruction while being scalable.

At last, the final optimization objective is defined as follows:

$$\mathcal{L} = \eta\mathcal{L}_{Contr} + (1 - \eta)\mathcal{L}_{Recon},$$

where $\eta \in (0, 1)$ is the trade-off HP to balance the importance of two SSL objectives.

HPs Sensitivity & Tuning They did not perform sensitivity analysis over the HPs. Instead, 1) They heuristically set the ration of augmented anomalies $r = 0.1$ and $r = 0.2$ for small and large datasets, respectively; 2) The sampling probability of each augmentation strategy is set to $p_i = 0.25$ for $i \in \{1, 2, 3, 4\}$; 3) They heuristically set the margin $m = 0.5$ for all datasets; and 4) They heuristically set the trade-off hyper-parameters $\lambda = 0.9$ and $\eta = 0.7$ for all datasets

A.7 DOMINANT

DOMINANT [284] is arguably the first work that utilizes *generative-based* framework and GNNs to perform unsupervised anomaly detection on attribute graphs.

Generative Framework They first employ GCN [360] to obtain node embeddings. Next, they construct two decoders: 1) an attribute decoder, which consists of another GCN, to reconstruct the node attributes, leading to the loss term L_A, and 2) a structure decoder, which is a dot product operation followed by a *sigmoid* function (namely $sigmoid(\mathbf{z}^t\mathbf{z})$), to reconstruct topological structures, leading to the loss term L_S.

At last, their final optimization objective is defined as follows:

$$\mathcal{L} = \alpha\mathcal{L}_A + (1-\alpha)\mathcal{L}_S,$$

where $\alpha \in (0,1)$ is the trade-off HP to balance the importance of two objectives.

HPs Sensitivity & Tuning Specifically, they found that the AUC performance usually increases first and then decreases with the increasing of α. However, the specific value of α on each dataset is heuristically selected with the help of labels. The HP α is selected from $[0.4, 0.7], [0.4, 0.7], [0.5, 0.8]$ on BlogCatalog, Flickr, and ACM respectively.

A.8 AnomalyDAE

Similar to DOMINANT, AnomalyDAE [275] leverages *generative-based* framework and autoencoders (based on GNNs) to perform unsupervised GAD.

Generative Framework AnomalyDAE consists of two components: 1) an attribute autoencoder to reconstruct the node attributes, where the encoder consists of two non-linear feature transform layers and the decoder is simply a dot product operation. This leads to the loss term L_A, and L_A is associated with a penalty HP $\eta > 1$); and 2) a structure autoencoder to reconstruct the structures, where the encoder is based GAT [336] and the decoder is a dot product operation followed by a *sigmoid* function (namely $sigmoid(\mathbf{z}^t\mathbf{z})$). This leads to the loss term L_S, and L_S is associated with a penalty HP $\theta > 1$.

At last, their final optimization objective is defined as follows:

$$\mathcal{L} = \alpha\mathcal{L}_S + (1-\alpha)\mathcal{L}_A,$$

where $\alpha \in (0,1)$ is the trade-off HP to balance the importance of two objectives.

HPs Sensitivity & Tuning Specifically, they found that the AUC performance usually increases first and then decreases with the increasing of α. However, the specific value of α on each dataset is selected using label information. The HPs (α, η, θ) are

heuristically set as $(0.7, 5, 40)$, $(0.9, 8, 90)$, $(0.7, 8, 10)$ on BlogCatalog, Flickr, and ACM respectively.

A.9 GUIDE

Similar to AnomalyDAE, GUIDE [279] leverages *generative-based* framework and autoencoders (based on GNNs) to perform unsupervised GAD. Particularly, they consider reconstructing the high-order structures.

Generative Framework GUIDE consists of two components: 1) an attribute autoencoder to reconstruct the node attributes, where the encoder is a GCN and the decoder is another GCN. This leads to the loss term L_A; and 2) a structure autoencoder to reconstruct the high-order structures, where the encoder is a graph node attention network based on [361] and the decoder is another graph node attention layer. This leads to the loss term L_S. Moreover, structure matrix is composed of node motif degrees, which leads to a HP, namely the degree of motifs D.

At last, their final optimization objective is defined as follows:

$$\mathcal{L} = \alpha\mathcal{L}_A + (1-\alpha)\mathcal{L}_S,$$

where $\alpha \in (0, 1)$ is the trade-off HP to balance the importance of two SSL objectives.

HPs Sensitivity & Tuning They mention that the HPs are optimized via a parameter sensitivity analysis experiment for each dataset. Specifically, they found that: 1) the AUC performance usually increases first and then decreases with the increasing of α, and most datasets can achieve a good performance when $0.1 < \alpha < 0.3$. However, the specific value of α on each dataset is selected using labels; and 2) they heuristically set the degree of motifs as $D = 4$.

A.10 GAAN

GAAN [337] combines the *generative-based* framework and GAN [362] for unsupervised GAD. Particularly, GAN can be considered as a special case of *contrastive-based* framework.

Contrastive Framework—Data Augmentation Module GAAN employs GAN, which consists of a generator and a discriminator, to generate adversarial samples as augmented views, without involving any HPs.

Contrastive Framework—Contrastive Learning Module For each target node, GAAN computes the sum of cross-entropy losses of its 1-hop neighboring nodes

(where the edge is considered as from real distribution by the discriminator) as anomaly score, leading to a loss term $\mathcal{L}_D$. In particular, this discriminator loss can be regarded as contrastive loss, and it considers both node attributes and graph structures.

Generative Framework GAAN utilizes the generator to reconstruct the node attribute, and employs the reconstruction error to compute anomaly score, leading to a loss term $\mathcal{L}_G$.

At last, their final optimization objective is defined as

$$\mathcal{L} = \alpha\mathcal{L}_G + (1-\alpha)\mathcal{L}_D,$$

where $\alpha \in [0, 1]$ is the trade-off HP to balance the importance of two objectives

HPs Sensitivity & Tuning Specifically, they found that the AUC performance usually increases first and then decreases with the increasing of α. However, the specific value of α on each dataset is selected using label information. The HP α is heuristically set as $0.2, 0.3, 0.1$ on BlogCatalog, Flickr, and ACM respectively.

Appendix B: Performance Variations under Different HP Settings

In this section, we present a comprehensive analysis of the performance exhibited by various semi-supervised learning (SSL) based graph anomaly detection techniques. This evaluation encompasses an extensive array of hyperparameter (HP) configurations and is conducted across multiple benchmark datasets.

Specifically, the results for GAAN [337] is provided in Figure 7.5, from which we can see huge performance variations under different HP settings. For example, the AUC value can vary from 0.474 to 0.747 if one utilizes different HP configurations on dataset CiteSeer (namely by changing the HP α from 0.5 to 0). Moreover, the results for CoLA [278] is provided in Figure 7.6. Compared to GAAN, CoLA is less sensitive to the setting of HPs, while we can still see moderate performance variations on some datasets (e.g., from 0.693 to 0.733 on Flickr, and from 0.767 to 0.795 on ACM). Besides, Figure 7.7 shows that DOMINANT is also sensitive to HPs except for the cases where the algorithm is largely underfitted (i.e., on ACM, Flickr and BlogCatalog the loss values change only by 10^{-2} after 400 epochs of training).

Particularly, AnomalyDAE and SL-GAD are very sensitive to HPs as shown in Figures 7.8 and 7.13. For example, the performance of AnomalyDAE ranges from 0.702 to 0.941 on CiteSeer, and the performance of SL-GAD vary from 0.787 to 0.920.

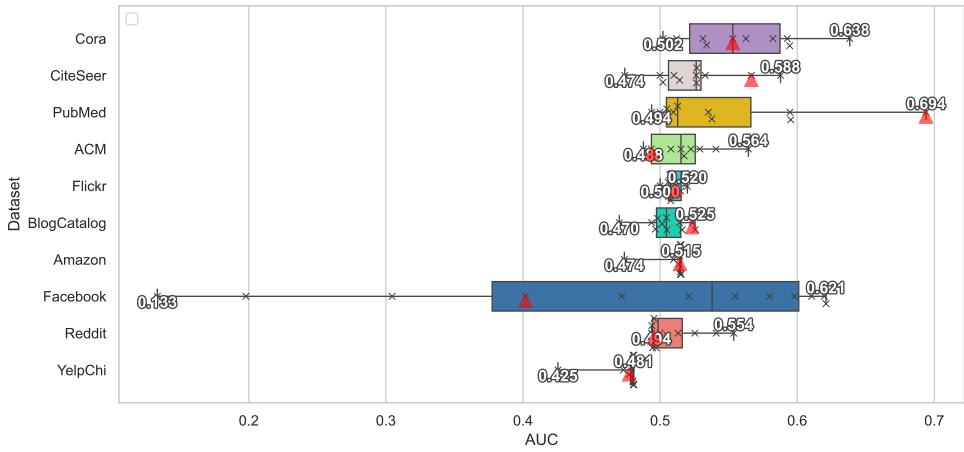


Figure 7.5: Performance variations over different HP configurations for GAAN [337] on different benchmark datasets.

As shown in Figure 7.9, CONAD shows similar behaviors except for the cases where CONAD is largely underfitted (namely on ACM) or suffers from OOM errors (namely on Flickr and BlogCatalog). The analysis for GUIDE in Figure 7.10, GRADATE in Figure 7.11, and Sub-CR in Figure 7.12 is similar and conveys the same issues.

Appendix C: Similar Observations in Other Papers

[294] conduct a comprehensive benchmark for unsupervised graph anomaly detection. From their results (note that their experiment setting is slightly different from ours), we can have similar observations as follows by comparing the average AUC vs max AUC:

- **Radar** [300] is not sensitive to hyper-parameters (0.65 VS 0.66 on Cora, 0.99 VS 0.99 on Weibo, 0.55 VS 0.57 on Reddit, 0.52 VS 0.52 on Disney, 0.53 VS 0.53 on Books), but it will suffer from OOM errors for large graphs;
- **ANOMALOUS** [301] is very sensitive to hyper-parameters on some datasets (0.55 VS 0.68 on Cora, 0.99 VS 0.99 on Weibo, 0.55 VS 0.60 on Reddit, 0.52 VS 0.52 on Disney, 0.53 VS 0.53 on Books), and it will suffer from OOM errors for large graphs;
- **DOMINANT** [284] is very sensitive to hyper-parameters on some datasets (0.83 VS 0.84 on Cora, 0.76 VS 0.85 on Flickr, 0.85 VS 0.93 on Weibo, 0.50 VS 0.58

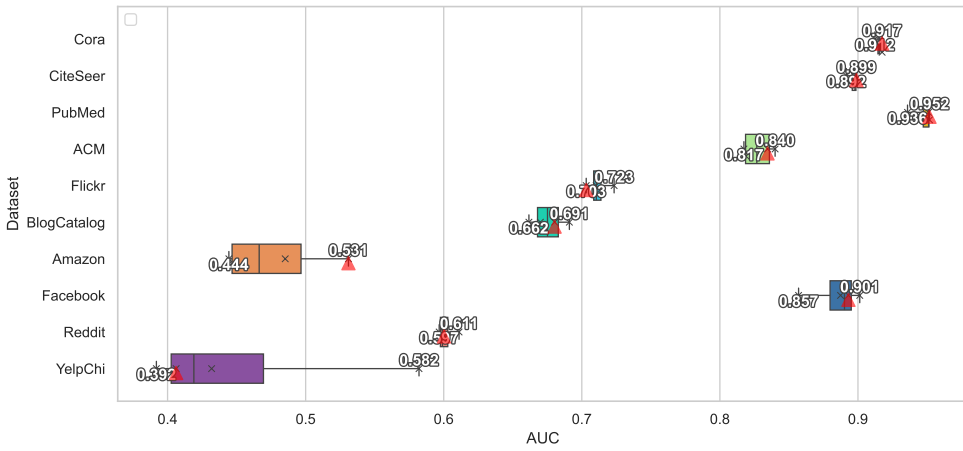


Figure 7.6: Performance variations over different HP configurations for CoLA [278] on different benchmark datasets.

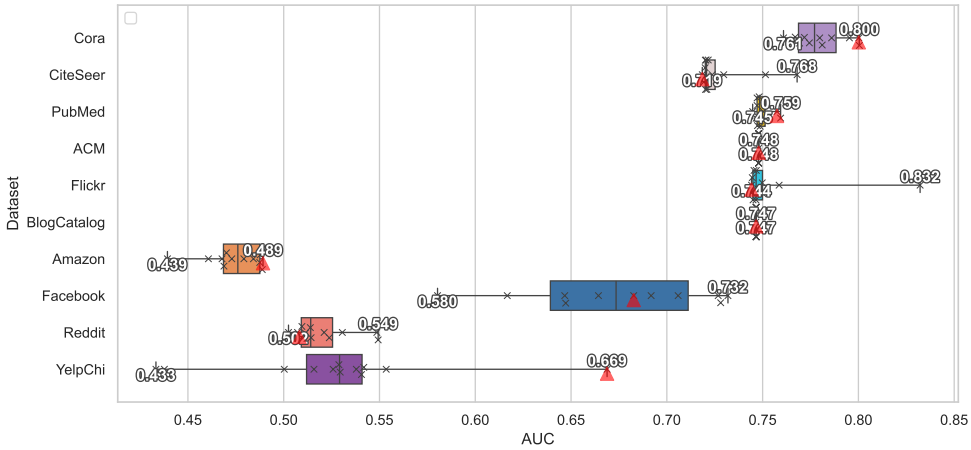


Figure 7.7: Performance variations over different HP configurations for DOMINANT [284] on different benchmark datasets.

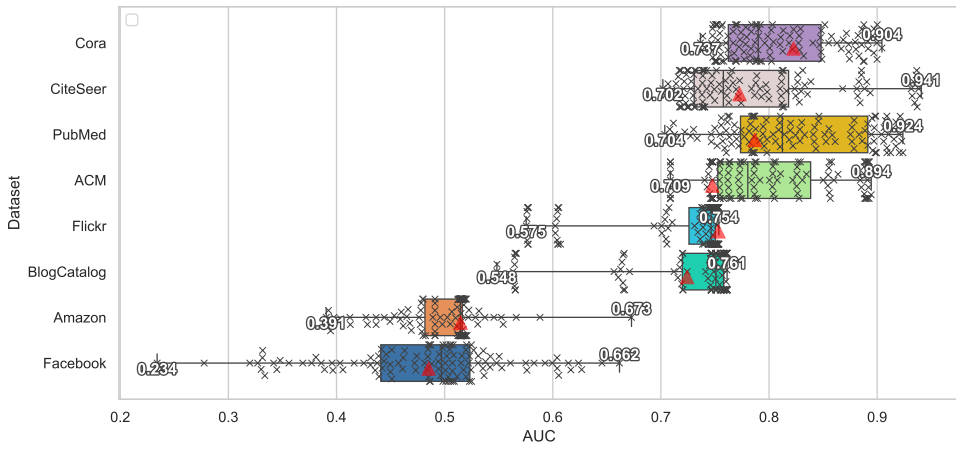


Figure 7.8: Performance variations over different HP configurations for AnomalyDAE [275] on different benchmark datasets.

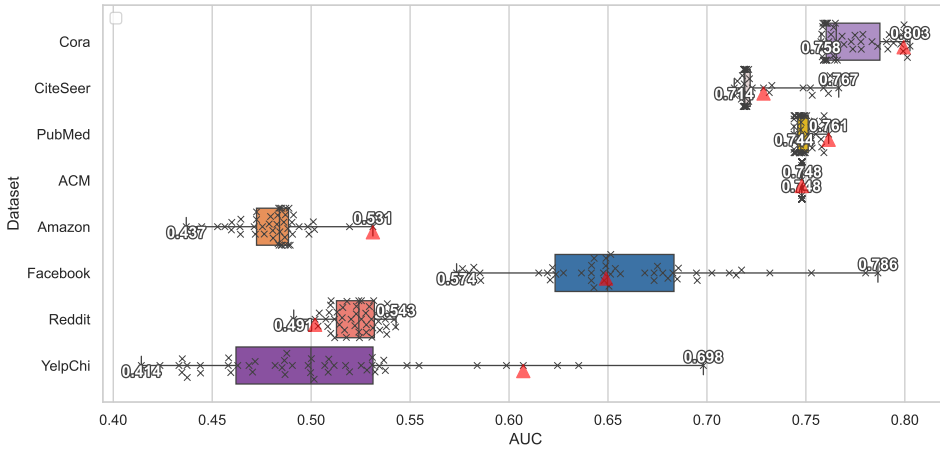


Figure 7.9: Performance variations over different HP configurations for CONAD [280] on different benchmark datasets.

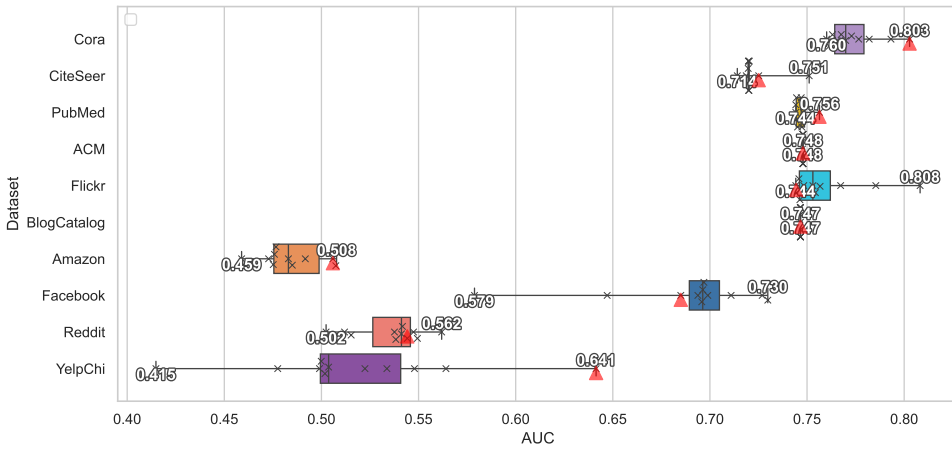


Figure 7.10: Performance variations over different HP configurations for GUIDE [279] on different benchmark datasets.

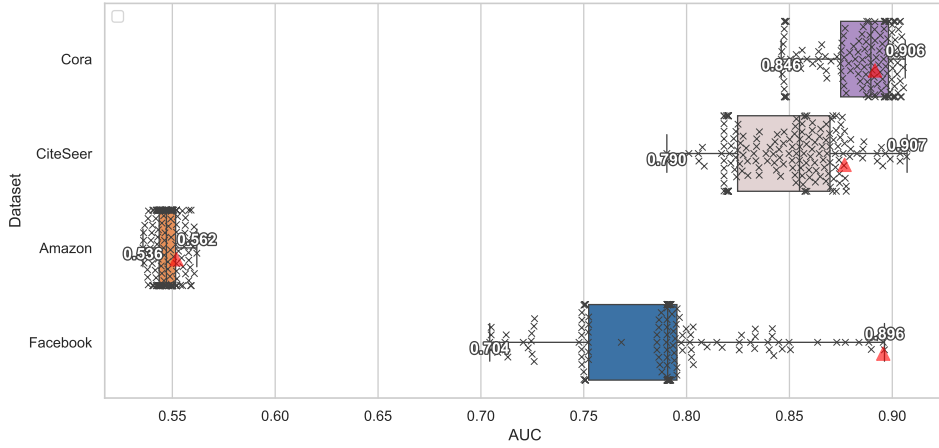


Figure 7.11: Performance variations over different HP configurations for GRADATE [285] on different benchmark datasets.

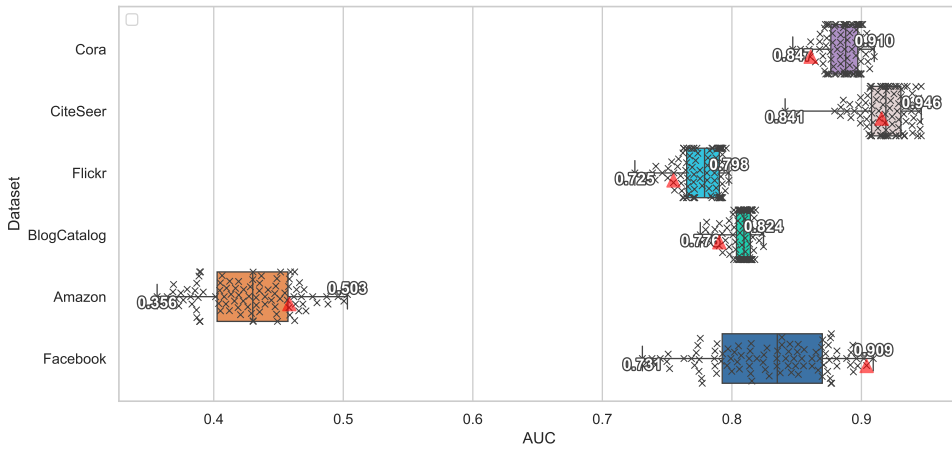


Figure 7.12: Performance variations over different HP configurations for Sub-CR [286] on different benchmark datasets.

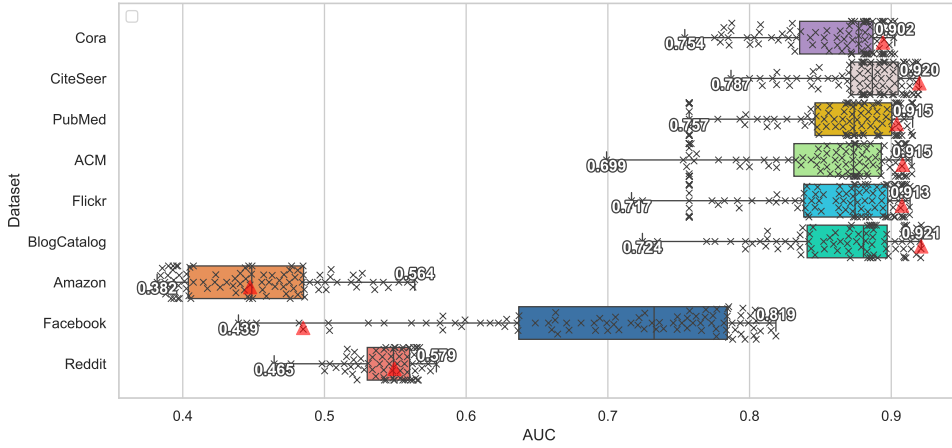


Figure 7.13: Performance variations over different HP configurations for SL-GAD [276] on different benchmark datasets.

on Books, 0.56 VS 0.56 on Reddit, 0.47 VS 0.55 on Disney)

- **AnomalyDAE** [275] is very sensitive to hyper-parameters on some datasets (0.83 VS 0.85 on Cora, 0.86 VS 0.91 on Amazon, 0.66 VS 0.70 on Flickr, 0.91 VS 0.93 on Weibo, 0.56 VS 0.56 on Reddit, 0.49 VS 0.55 on Disney, 0.54 VS 0.69 on Books);
- **GUIDE** [279] is very sensitive to hyper-parameters on some datasets (0.39 VS 0.53 on Disney, 0.52 VS 0.63 on Books, 0.75 VS 0.78 on Cora), and it will suffer from OOM errors on large graph (including Amazon, Flickr, Weibo, Reddit). It needs much time and memory for training as it employs a graph motif counting algorithm to extract structural information;
- **CONAD** [280] is very sensitive to hyper-parameters on some datasets (0.79 VS 0.84 on Cora, 0.81 VS 0.82 on Amazon, 0.65 VS 0.67 on Flickr, 0.85 VS 0.93 on Weibo, 0.56 VS 0.56 on Reddit, 0.48 VS 0.53 on Disney, 0.52 VS 0.63 on Books).

Appendix D: Summary of existing SSL-based graph anomaly detection methods

Existing SSL-based graph anomaly detection methods are summarized in Table 7.7, which includes the datasets used to test, the core principles of SSL techniques, the involved hyper-parameters (only SSL related ones), and their public implementations.

Appendix E: Search Space Approximation based on SMBO

Performance Surrogate Functions

Although discretization of continuous domains can largely reduce the search space, it is still computationally prohibitive to search the full discretized HP space when the number of HPs is large. Therefore, we learn a regressor $g(\cdot)$ which aims to to learn the mapping from HP settings onto the performance metric (namely the domain of $T(\cdot)$). Note that $g(\cdot)$ should be different for different combinations of graph and graph anomaly detector $[\mathcal{G}, f(\cdot)]$, and we call these functions *performance surrogate functions*. Gaussian Process (GP) [363] is one popular choice for $g(\cdot)$. Based on these *performance*

Table 7.6: SSL-related HPs for different algorithms, where "**Range**" indicates the tested values in grid search.

Algo	HPs	Range
CoLA [278]	K	{2, 3, 4, 5}
ANEMONE [277]	K	{2, 3, 4, 5}
	α	{0, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 1}
GRADATE [285]	P	{0.20}
	α	{0.9}
	β	{0, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 1}
	γ	{0, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 1}
SL-GAD [276]	K	{2, 3, 4, 5, 6, 7, 8, 9}
	α	{0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 1}
	β	{0.6}
Sub-CR [286]	K	{2, 3, 4, 5, 6, 7, 8, 9}
	α	{0.01}
	γ	{0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 1}
CONAD [286]	r	{0.10}
	$p1$	{0.25}
	$p2$	{0.25}
	$p3$	{0.25}
	$p4$	{0.25}
	m	{0.5}
	λ	{0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 1}
	η	{0.01, 0.5, 0.99, 1}
DOMINANT [284]	α	{0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 1}
AnomalyDAE [275]	α	{0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 1}
	η	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
	θ	{10}
GUIDE [279]	D	{4}
	α	{0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99}
GAAN [337]	α	{0, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 1}

Table 7.7: Summary of existing SSL-based graph anomaly detection methods.

Method	Venue	Datasets	SSL methods	Hyperparameters	Code
CoLA [278]	TNNLS'21	Cora, Citeseer, Pubmed, BlogCatalog, Flickr, ACM, ogbn-arxiv	Node-Sub CL	Random walk length (K)	Github, PyGOD
ANEMONE [277]	CIKM'21	Cora, Citeseer, PubMed	Node-Node CL, Node-Sub CL	Ego-Net size (K), Combination weights	Github
GRADATE [285]	AAAI'23	EAT, WebKB, UAT, Cora, UAI2010, Citation	Node-Node CL, Node-Sub CL, Sub-Sub CL	Proportion of modified edges (P), Combination weights	Github
SL-GAD [276]	TKDE'21	Cora, Citeseer, PubMed, ACM, Flickr, BlogCataLog	Node-Sub CL, Attribute Recon	Random walk length (K), Combination weights	Github
Sub-CR [286]	IJCAI'22	Cora, Citeseer, PubMed, Flickr, BlogCataLog	Node-Sub CL, Attribute Recon	Random walk length (K), Teleport probability α, Combination weights	Github
CONAD [280]	PAKDD'22	Amazon, Flickr, Enron, Facebook, Twitter	Node-Sub CL, Attribute Recon, Structure Recon	Augmentation sampling probabilities, combination weights	PyGOD, Github
DOMINANT [284]	ICDM'19	ACM, Flickr, BlogCataLog	Attribute Recon, Structure Recon	Combination weight	PyGOD
AnomalyDAE [275]	ICASSP'20	ACM, Flickr, BlogCataLog	Attribute Recon, Structure Recon	Penalty HPs, Combination weights	PyGOD
GUIDE [279]	BigData'21	Cora, Citation, Pubmed, ACM, DBLP	Attribute Recon, Structure Recon	Combination weight	PyGOD
GAAN [337]	CIKM'20	ACM, Flickr, BlogCataLog	Attribute Recon, Discr Loss	Combination weight	PyGOD

surrogate functions, we can identify promising HPs without running experiments on all possible HPs, which will be illustrated in next subsection.

SMBO-based Optimization

Particularly, we leverage Sequential Model-based Optimization (SMBO) [339] to iteratively and efficiently identify promising HP configurations to evaluate, and finally output the optimal one as follows. Similar idea is also explored in [364].

Initialization Specifically, we first randomly sample a small number of HPs $\boldsymbol{\lambda}_{eval} = \{\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, ..., \boldsymbol{\lambda}_J\}$ with $J \ll M$. Second, for each HP, we compute its unsupervised performance metric score $t(\mathcal{G})$, leading to pairs $\{(\boldsymbol{\lambda}_1, t_1(\mathcal{G})), (\boldsymbol{\lambda}_2, t_2(\mathcal{G})),..., (\boldsymbol{\lambda}_J, t_J(\mathcal{G}))\}$. Third, we employ these pairs to train a specific *performance surrogate function* $g(\cdot)$.

Iteration For each iteration, we leverage $g(\cdot)$ to predict the performance for a sampled HP $\boldsymbol{\lambda}_j$, denoted as $\eta_j = g(\boldsymbol{\lambda}_j)$. Moreover, we also utilize $g(\cdot)$ to predict the uncertainty around the prediction of $\boldsymbol{\lambda}_j$, denoted as $\sigma_j = \sigma[g(\boldsymbol{\lambda}_l | \boldsymbol{\lambda}_l \in \boldsymbol{\lambda}_{sample})]$. Note that $\boldsymbol{\lambda}_{sample}$ is different from $\boldsymbol{\lambda}_{eval}$, and it is a finite number of HPs that is randomly sampled from the full HP space before discretization. Next, we utilize a so-called *acquisition function* $h(\cdot)$, which can make a trade-off between predicted performance

and uncertainty, to select the most promising HP to evaluate. Particularly, we leverage Expected Improvement (EI) [339] as the *acquisition function* since it has shown prominent performances in many studies [322]. Under the mild Gaussian assumption, the EI value of HP setting $\boldsymbol{\lambda}_j$ has the following closed-form expression:

$$EI(g(\boldsymbol{\lambda}_j)) = [\phi(\hat{\eta}_j) + \hat{\eta}_j \cdot \Phi(\hat{\eta}_j)]\,\sigma_j, \tag{7.8}$$

where $\hat{\eta}_j = \frac{\eta_j - \eta^*_{eval}}{\sigma_j}$ if $\sigma_j > 0$ and $\hat{\eta}_j = 0$ otherwise. Moreover, $\phi(\cdot)$ and $\Phi(\cdot)$ denote the probability density function and the cumulative distribution function of standard Gaussian distribution, respectively. In addition, η^*_{eval} is the highest prediction performance on $\boldsymbol{\lambda}_{eval}$ so far. For each iteration, the most promising HP can be obtained as follows:

$$\boldsymbol{\lambda}^* = \underset{\boldsymbol{\lambda}_j \in \boldsymbol{\lambda}_{sample}}{\arg\max}\ h(g(\boldsymbol{\lambda}_j)), \tag{7.9}$$

where $g(\cdot) = g^{(current)}(\cdot)$ is the surrogate function in the current iteration, which can output the most promising HP $\boldsymbol{\lambda}^*$ to evaluate. On this basis, we apply $f(\boldsymbol{\lambda}^*)$ on graph $\mathcal{G}$ to obtain a vector of anomaly scores $\boldsymbol{s}^*$, followed by inputting $\boldsymbol{s}^*$ into Equation 7.3 to obtain the performance metric score t^*. At last, we update the evaluation HP set as $\boldsymbol{\lambda}_{eval} = \boldsymbol{\lambda}_{eval} \cup \boldsymbol{\lambda}^*$, and retrain $g(\cdot)$ with the updated pairs $\{(\boldsymbol{\lambda}_1, t_1(\mathcal{G})), (\boldsymbol{\lambda}_2, t_2(\mathcal{G})),..., (\boldsymbol{\lambda}_J, t_J(\mathcal{G}))\}..., (\boldsymbol{\lambda}^*, t^*)\}$. Additionally, we update η^*_{eval} using the updated $\boldsymbol{\lambda}_{eval}$.

Appendix F: AutoGAD for Selecting Heterogeneous Anomaly Detectors

To evaluate the effectiveness of AutoGAD in selecting heterogeneous anomaly detectors, we compute the Pearson Correlation Coefficient between the highest improved CSM scores (based on Eq. 7.3) and the corresponding AUC scores for all anomaly detectors on each individual dataset.

As shown in Figure 7.14, the results reveal that AutoGAD's CSM score does not effectively predict the true performance (AUC) of heterogeneous anomaly detectors. Specifically, on the Cora dataset, the Pearson correlation is very weak (0.070), indicating almost no relationship between the CSM score and AUC. On the Amazon dataset, the correlation is negative (-0.488), suggesting that higher CSM scores are, in fact, associated with lower AUC values in many cases. This weak or inverse correlation demonstrates that AutoGAD's scoring mechanism may not be suitable for selecting the

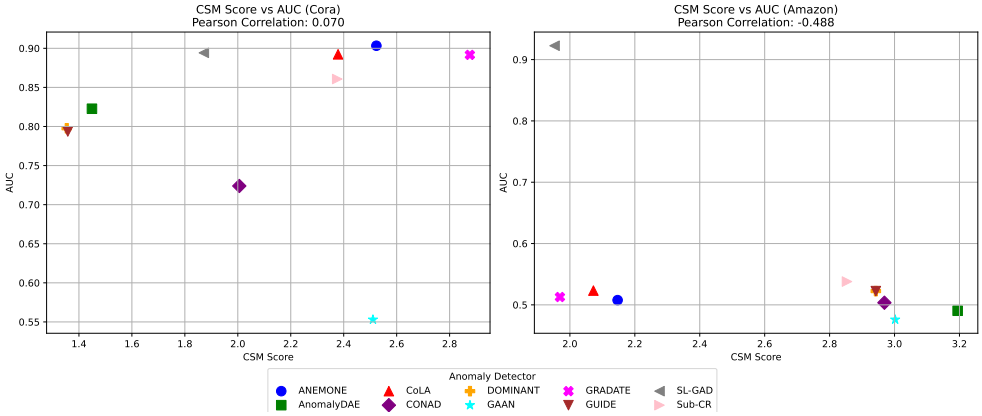


Figure 7.14: Performance of AutoGAD in selecting heterogeneous anomaly detectors on selected datasets (results on other datasets are similar and thus omitted).

best-performing anomaly detectors, as it fails to consistently align with true detector performance. Notably, detectors such as SL-GAD, which achieve high AUC, do not consistently receive high CSM scores, further underscoring the discrepancy. In summary, these findings suggest that AutoGAD's current approach to ranking anomaly detectors is unreliable and may require significant revisions to improve its predictive accuracy.

Chapter 8

Conclusions and Future Directions

In this dissertation, we developed advanced anomaly detection approaches with a primary focus on enhancing the manufacturing processes of high-tech systems. However, it is important to emphasize that many of the techniques and findings discussed here are applicable to broader scenarios beyond manufacturing, and in some cases, were even designed for more general-purpose use. Given the complexity of achieving trustworthy anomaly detection in smart manufacturing, we approached this challenge from two complementary perspectives—data-centric AI and model-centric AI—leading to two key research questions, each with several sub-questions.

The first research question focuses on improving anomaly detection from a **data-centric AI perspective**. In this approach, the emphasis is placed on systematically processing and engineering the data that serves as input for the anomaly detection systems. For instance, we explored ways to handle *complex data*—such as system logs, time series, and graphs—by introducing novel methods that effectively manage and extract meaningful patterns from such data. Additionally, we investigated the challenges posed by *high-dimensional data*, which arises even after transforming complex data into simpler forms. High dimensionality often undermines the effectiveness of traditional anomaly detection approaches.

The second research question centers on the **model-centric AI perspective**, which focuses on optimizing the anomaly detection models themselves. One critical issue we explored in this context is the *explainability* of AI models. While neural

network-based models can excel in detecting anomalies, their lack of transparency often leaves end-users in the dark about how decisions are made. Furthermore, we examined *the robustness of post-hoc explanations*, particularly in the face of adversarial attacks. In addition to explainability, we explored *generalizability*, which is essential for deploying anomaly detection algorithms in new, unseen environments. This is particularly important when systems undergo updates or modifications, such as changes in robots or software. Finally, we investigated approaches to improving the *automatability* in anomaly detection, particularly in unsupervised settings where labeled data is scarce.

In summary, this dissertation provides a comprehensive framework for developing robust, explainable, and generalizable anomaly detection systems, offering solutions to both data-related and model-related challenges in smart manufacturing. Our approaches contribute to improving the reliability and efficiency of manufacturing processes, while also providing insights applicable to broader fields of anomaly detection.

In the remainder of this chapter, we will begin by summarizing the key findings from each chapter to address the research questions outlined in Chapter 1. This will provide a cohesive overview of how the research outcomes contribute to answering these questions. Afterward, we will explore the limitations and challenges encountered across multiple chapters, highlighting the areas that require further attention. This discussion aims to offer valuable insights into unresolved issues and guide potential future research directions.

8.1 Conclusions

In Chapter 1, we outlined six key research questions. In this subsection, we provide detailed answers to each of these questions, drawing upon the findings presented in Chapters 2 through 7. For each research question, we summarize the main conclusions, highlighting how the results contribute to addressing the research objectives.

8.1.1 Develop and/or Improve Anomaly Detection for Smart Manufacturing from A Data-Centric AI Perspective

Q1.1 **How to deal with complex data in system logs to effectively detect and explain anomalies?**

In response to this problem, we introduced *Logs2Graphs*, a new approach for unsupervised log anomaly detection. It first converts log files to attributed,

directed, and edge-weighted graphs, translating the problem to an instance of graph-level anomaly detection. Next, this problem is solved by OCDiGCN, a novel method based on graph neural networks that performs graph representation learning and graph-level anomaly detection in an end-to-end manner. Important properties of OCDiGCN include that it can deal with directed graphs and do unsupervised learning. Moreover, we furnish a concise set of nodes pivotal in OCDiGCN's prediction as explanations for each detected anomaly, offering valuable insights for subsequent root cause analysis. Extensive results on five benchmark datasets reveal that *Logs2Graphs* is at least comparable to and often outperforms state-of-the-art log anomaly detection methods such as DeepLog [65] and LogAnomaly [66].

The above findings are all described in Chapter 2. The main conclusion to this research question is that: by leveraging the rich and expressive power of attributed, directed, and edge-weighted graphs to represent logs, followed by using graph neural networks to effectively detect graph-level anomalies, taking into account both semantic information of log events and structure information among log events, we can better detect anomalies in system logs. Moreover, by decomposing the anomaly score of a graph into individual nodes and visualizing these nodes based on their contributions, we provide understandable explanations for identified anomalies.

Q1.2 **How to handle high-dimensionality in system logs to effectively detect anomalies?**

By regarding each log event as a feature, we developed a feature selection method that aims to select relevant features for log-based fault detection and prediction. In brief, our method consists of three main modules, namely *Log Event Vectorization*, *Selection of Relevant Features* and *Removal of Redundant Features*. Specifically, the *Log Event Vectorization* module aims at converting unstructured log events into time series data; the *Selection of Relevant Features* module attempts to select relevant features for fault detection and prediction by using the variables measured by sensors as target; and the *Removal of Redundant Features* module focuses on eliminating redundant features to further reduce the number of selected features. Extensive experiments on real-world datasets show that our proposed feature selection method can help improve log-based anomaly detection performance. Specifically, based on the selected log features, KNN [124] was able to accurately detect or predict faults in 24 out of 25 machines.

In contrast, KNN can only accurately detect or predict faults in 17 out of 25 datasets based on all log features. One possible reason is that logs from all subsystems are entangled and the inclusion of many irrelevant log events renders the detection/prediction of gradual faults difficult.

The above findings are described in Chapter 3. The main conclusion is that: by considering the associations between sensor data and system logs, we can select relevant and non-redundant log events as features. As a result, the accuracy of log-based fault detection and prediction can be significantly improved, as irrelevant log events often obscure gradual fault patterns.

8.1.2 Develop and/or Improve Anomaly Detection for Smart Manufacturing from A Model-Centric AI Perspective

Q2.1 **How to achieve intrinsic explainability of anomaly detection systems?**

To explore this problem, we for the first time explicitly establish a connection between dependency-based traditional anomaly detection methods and contextual anomaly detection methods. On this basis, we propose a novel approach to contextual anomaly detection and explanation. Specifically, we use Quantile Regression Forests to develop an accurate and interpretable anomaly detection method, QCAD, that explores dependencies between features. QCAD can handle tabular datasets with mixed contextual features and numerical behavioral features. Extensive experiment results on various synthetic and real-world datasets demonstrate that QCAD outperforms state-of-the-art anomaly detection methods in identifying contextual anomalies in terms of accuracy and interpretability. Particularly, the beanplot-based visualizations help to explain why a certain object is (not) considered an anomaly within its context.

These findings are described in Chapter 4. The main conclusion is that: by establishing a connection between dependency-based traditional anomaly detection methods and contextual anomaly detection methods, we developed a self-interpretable anomaly detection method QCAD, which can effectively identify contextual anomalies by exploring dependencies between features and provide beanplot-based visualizations to further explain why an object is (or is not) considered an anomaly.

Q2.2 **Are post-doc explanations for Graph Neural Networks robust to adversarial attacks?**

GNN explanations for enhancing transparency and trust of graph neural networks are becoming increasingly important in decision-critical domains. We showed that existing GNN explanation methods are vulnerable to adversarial perturbations, namely small *prediction-preserving* perturbations can result in largely different explanations generated by post-hoc GNN explainers. To achieve this, we performed perturbations that are both carefully crafted (i.e., *optimized* rather than *random*) and imperceptible (i.e., such that the GNN predictions remain unchanged but their explanations differ substantially). More concretely, we devise *GXAttack*, the first *optimization-based* adversarial attack on post-hoc GNN explanations under this setting. We employ a widely used GNN explainer, PGExplainer [199], as example target when designing our attack algorithm. Results on various datasets demonstrate the effectiveness of our approach. Moreover, our experiments show that other widely used GNN explanation methods, such as GradCAM [197], GNNExplainer [195], and SubgraphX [201], are also fragile under the attacks optimized for PGExplainer.

These findings are described in Chapter 5. The main conclusion is that: existing GNN explanation methods are vulnerable to adversarial perturbations, leading to drastically different explanations while maintaining the predictions unchanged. This is achieved by GXAttack, the first optimization-based adversarial white-box attack on post-hoc GNN explanations. Additionally, other widely used GNN explanation methods, such as GradCAM, GNNExplainer, and SubgraphX, are also fragile under attacks optimized for PGExplainer.

Q2.3 **How to detect graph-level anomalies in an unseen target domain with the help of labeled normal graphs from a different but related source domain?**

Being motivated and supported by domain adaptation theory [237], we propose an unsupervised domain adaptation based graph level anomaly detection method called ARMET. It leverages an adversarial learning approach consisting of four main components. First, to learn graph level representations, it utilizes a two-part feature extractor: a semantic feature extractor to jointly preserve the semantic and topological information of each graph, and a structure feature extractor to extract the structure of each graph domain. Second, a domain classifier is learned to make graph level representations domain-invariant, thereby reducing the domain discrepancy. Third, a one-class classifier is trained using normal source graphs, aiming to make the learned graph level representations

label-discriminative. Finally, a class aligner is trained to align normal graphs in both domains while separating anomalous graphs and normal graphs in the target domain. As a result, in an end-to-end manner, ARMET can learn both domain-invariant and label-discriminative graph level representations, and thus effectively identify anomalous graphs from the target domain. Experiments on seven benchmark datasets show that the proposed method largely outperforms state-of-the-art methods.

These findings are described in Chapter 6. The main conclusion is that: by learning domain-invariant and label-discriminative graph-level representations through adversarial learning, ARMET can detect graph-level anomalies in an unseen target domain by leveraging labeled normal graphs from a related source domain, significantly outperforming state-of-the-art methods across multiple benchmark datasets.

Q2.4 **How to automatically tune hyperparamaters in anomaly detection systems without relying on labels?**

Self-Supervised Learning (SSL) has received much attention in recent years, and many recent studies have explored SSL to perform unsupervised graph anomaly detection. However, we found that most existing studies tune hyperparameters arbitrarily or selectively (i.e., guided by labels), and our empirical findings reveal that most methods are highly sensitive to hyperparameter settings. Using label information to tune hyperparameters in an unsupervised setting, however, is label information leakage and leads to severe overestimation of model performance. To mitigate this issue, we introduce AutoGAD, the first automated hyperparameter selection method for SSL-based unsupervised graph anomaly detection. AutoGAD, consists of two parts: 1) an unsupervised performance metric which is based on an internal evaluation strategy, and 2) an effective search method which leverages discretization and grid search that works well in practice. Extensive experiments demonstrate the effectiveness of our proposed strategy. Overall, we aim to raise awareness to the label information leakage issue in the unsupervised graph anomaly detection field, and AutoGAD provides a first step towards achieving truly unsupervised SSL-based graph anomaly detection.

These findings are described in Chapter 7. The main conclusion is that: AutoGAD addresses the critical issue of label information leakage by using an unsupervised performance metric combined with an straightforward grid search

strategy, ensuring hyperparameter selection without relying on labels. This approach mitigates overestimation of model performance and paves the way for truly unsupervised graph anomaly detection.

8.2 Limitations and Future Directions

Throughout the research presented in this dissertation, we identified a number of research challenges that may offer opportunities for future research, which we will summarize next.

8.2.1 Limitations of Proposed Methods

First, we discuss the limitations of the proposed methods in each chapter and suggest potential future research directions to mitigate these limitations.

8.2.1.1 Limitations of Logs2Graphs (Chapter 2)

We identify several factors that could potentially impact the validity of our findings related to the Logs2Graphs method introduced in Chapter 2:

- **Limited Datasets.** Our experimental protocol entails utilizing five publicly available log datasets, which have been commonly employed in prior research on log-based anomaly detection. However, it is important to acknowledge that these datasets may not fully encapsulate the entirety of log data characteristics. To address this limitation, our future work will conduct experiments on additional datasets, particularly those derived from industrial settings, in order to encompass a broader range of real-world scenarios.
- **Limited Competitors.** This study focuses solely on the experimental evaluation of eight competing models, which are considered representative and possess publicly accessible source code. However, it is worth noting that certain models such as GLAD-PAW [67] did not disclose their source code and it requires non-trivial efforts to re-implement these models. Moreover, certain other models, such as CODEtect [84], require several months to conduct the experiments on our limited computing resources. For these reasons, we excluded them from our present evaluation. In subsequent endeavors, we intend to re-implement certain models and attain more computing resources to test more models.

- **Purity of Training Data.** The purity of training data is usually hard to guarantee in practical scenarios. Although Logs2Graphs is shown to be robust to very small contamination of the training data, it is critical to improve model robustness by using techniques such as adversarial training [112] in the future.
- **Graph Construction.** The graph construction process, especially regarding the establishment of edges and assigning edge weights, adheres to a rule based on connecting consecutive log events. However, this rule may be considered overly simplistic in certain scenarios. Therefore, application-specific techniques will be explored to construct graphs in the future.

8.2.1.2 Limitations of FS4FDP (Chapter 3)

Several factors have been identified that may influence the validity of our findings concerning the FS4FDP method introduced in Chapter 3:

- **Limited Datasets.** Our experiments were conducted on 25 real-world datasets provided by industrial partners, which may not capture the full range of data diversity encountered in broader applications. In future work, we plan to incorporate additional datasets to further assess the generalizability of FS4FDP across varied contexts and investigate the effects of hyperparameter tuning on performance.
- **Limited Anomaly Detectors.** Currently, we only evaluated FS4FDP using a restricted selection of anomaly detectors, denoted as $\phi(\cdot)$ and $\varphi(\cdot)$. Future efforts will expand this selection to include a broader array of anomaly detection methods, enabling a more comprehensive analysis of FS4FDP's adaptability and effectiveness when paired with different detection techniques.
- **More Studies on Causal Relationships**. Particularly, the Equation (3.4) used in Granger causality requires several explicit and implicit assumptions to effectively identify Granger causal effects. Some of these assumptions may not be fulfilled by our use-case though. Therefore, we will further explore and improve Granger causality test [125] or similar techniques to find log events that can be used to predict sensor time series anomalies. Furthermore, we have not fully explored the causal relationships between different log events. In the future, by constructing a causality graph using techniques such as the PC-algorithm [126] on log events, we can investigate the causal relationships between log events. As a result, it might be possible to pinpoint the root causes of anomalies.

8.2.1.3 Limitations of QCAD (Chapter 4)

The QCAD method presented in Chapter 4 has the following limitations, which we aim to address in future work:

- **Static Features**. Currently, QCAD is limited to static features in both contextual and behavioral spaces. We plan to expand QCAD's capabilities to handle streaming data in future versions.
- **Constraints on Behavioral Space**. At present, QCAD is restricted to numerical features within the behavioral space. Future efforts will focus on extending QCAD to accommodate categorical features and mixed feature types in the behavioral space.

8.2.1.4 Limitations of GXAttack (Chapter 5)

Several factors may influence the validity of our findings concerning the GXAttack method presented in Chapter 5, and we plan to mitigate them in the future:

- **Low Scalability.** GXAttack suffers from low scalability (large requirement of memory), which however can be mitigated by using the Projected Randomized Block Coordinate Descent (PR-BCD) [228].
- **Limited Datasets.** We only employed synthetic datasets. While real-world datasets can provide quantitative evaluation results using metrics like cosine similarity (to measure the consistency of explanations before and after attacks), these metrics may not accurately reflect the true effectiveness of the attacks (i.e., in terms of ΔGEA). This discrepancy suggests that quantitative results can be challenging to interpret, and relying solely on such metrics might not yield meaningful insights. On the other hand, qualitative evaluations are also possible, but as machine learning researchers, our expertise lies not in the realm of such interpretations.
- **Differentiable Target.** GXAttack requires that the target GNN explainer to be differentiable, which is not always the case. However, the attack transferability shows its potential effectiveness on other GNN explainers.
- **Prediction-preserving Assumption.** We (implicitly) impose an assumption that a small L_0 perturbation that is prediction-preserving also preserves the causal reasoning for the GNN prediction. Unlike for images or texts, this is not

easy to verify for graph data. However, we can leverage visualizations to help understand the reasoning process.

8.2.1.5 Limitations of ARMET (Chapter 6)

The ARMET method introduced in Chapter 6 has certain limitations that we intend to explore further in future research:

- **Limited Datasets.** Our experimental protocol included only four publicly available log datasets and three image-based datasets, which do not fully capture the diversity of data characteristics encountered in real-world applications. To overcome this limitation, future work will focus on experiments using additional datasets, especially those derived from molecular, finance, and social networks, to better represent a wide range of practical scenarios.
- **Transferability Across Graph Domains.** Currently, there is limited understanding of how well ARMET generalizes across various graph domains. Future research should focus on evaluating and quantifying the transferability of ARMET across different types of graph-structured data, enabling broader applicability and enhancing its robustness in diverse domains.

8.2.1.6 Limitations of AutoGAD (Chapter 7)

The AutoGAD method presented in Chapter 7 has the following limitations, which we plan to address in future work:

- **Limited Hyperparameters Search Space.** In our experimental setup, we explored a restricted search space for hyperparameters within each SSL-based GAD method, constrained by available computational resources. In future work, we aim to expand this search space to allow for a more comprehensive exploration of hyperparameters, which may uncover more optimal configurations and improve overall performance.
- **Basic Hyperparameters Search Strategy.** Currently, our approach relies on grid search for hyperparameter tuning. Future research will investigate more sophisticated search techniques, such as Bayesian optimization, and genetic algorithms, to increase efficiency and precision in identifying optimal hyperparameter settings.

8.2.2 Other Future Directions

In addition to challenges directly associated with our proposed methods, we identify the following research directions that should receive more attention in the future.

8.2.2.1 Definition of Anomaly and Trustworthy Anomaly Detection

A long-standing problem in anomaly analysis is the lack of a uniform definition of an anomaly, leading to a wide range of anomaly detection methods [46]. The diversity of anomaly definitions and anomaly detection methods leads to the need for a large variety of trustworthy anomaly detection (TAD) techniques. Although is not necessarily problematic on itself (and may be unavoidable), the lack of uniform definitions for anomaly detection and TAD hampers communication of researchers between different (sub)fields, such as computer vision, natural language processing, data mining, and social science. This makes it hard to find related work and leads to the re-invention of methods, causing unnecessary delays in scientific progress. More importantly, the evaluation and comparison of TAD methods becomes difficult and subjective, due to the lack of a uniform, objective, and precise definition of TAD.

8.2.2.2 Enhancing Anomaly Detection for Smart Manufacturing through the Integration of Model-Centric and Data-Centric AI Approaches

Most existing anomaly detection methods (not limited to the context of smart manufacturing) are predominantly developed or enhanced from a model-centric AI perspective [27, 28, 29, 30, 31, 32, 33, 34, 35, 36]. This means that researchers focus on improving anomaly detection systems by refining algorithms to perform more effectively on existing, often fixed, datasets. These efforts involve designing better model architectures, tuning hyperparameters, developing more efficient optimization techniques, and proposing new models types or learning paradigms.

Although there has been a growing shift from model-centric AI to data-centric AI within the broader data mining and machine learning communities [365], research on data-centric anomaly detection remains limited and insufficient [366, 367, 368, 369, 370, 371, 372, 373, 374]. We therefore call for increased research efforts to address the anomaly detection problem from a data-centric AI perspective. However, we also argue that model-centric and data-centric AI approaches are inherently complementary. While data-centric AI deserves more attention, it should not overshadow the importance of model-centric AI. Effectively tackling challenges in smart manufacturing requires both the methods of action ("how-to") and deep insights hidden from

the data ("what-is") [48]. Thus, advanced research should approach the anomaly detection problem by integrating both model-centric and data-centric AI perspectives, recognizing them as dual forces driving progress in smart manufacturing.

8.2.2.3 Anomaly Detection in the Era of Large Language Models

Large Language Models (LLMs) have recently shown their superior performance not only in traditional natural language processing tasks like language comprehension and summarization but also in a wider array of applications thanks to their advanced comprehension and generative abilities [375, 376]. To unlock the full potential of LLMs beyond textual data, researchers are expanding these models into multi-modal tasks, such as vision-language understanding and generation, which are collectively referred to as Multimodal LLMs (MLLMs) [377]. Leveraging the zero- and few-shot reasoning capabilities of LLMs and MLLMs, researchers are increasingly applying these models to anomaly detection, yielding promising results [378, 379].

The integration of LLMs and MLLMs into anomaly detection has dramatically shifted the traditional learning paradigm, which previously relied mostly on statistical models and traditional machine learning techniques. Xu and Ding [378] categorize these approaches into three groups based on the primary roles played by LLMs in anomaly detection:

- **LLMs for augmentation**: In this approach, LLMs are not directly responsible for detecting anomalies but instead enhance the detection process through their advanced semantic understanding and vast knowledge. LLMs act as data augmenters, generating meaningful information to improve anomaly detection. This includes producing effective text embeddings (using LLMs as feature extractors), generating high-quality synthetic datasets with pseudo-labels, and creating detailed textual descriptions of both normal and abnormal instances.

- **LLMs for detection**: In these methods, LLMs are employed directly as anomaly detectors. This can involve prompting-based approaches in which LLMs generate detection results in response to specific prompts, or contrastive learning approaches that utilize MLLMs pretrained with contrastive objectives to detect anomalies.

- **LLMs for explanation**: Here, LLMs offer detailed explanations and insights regarding the results of anomaly detection. These explanations help address

real-world challenges by providing deeper understanding and interpretability of the detected anomalies.

While the application of LLMs and MLLMs in anomaly detection has garnered increasing attention, it remains a relatively underexplored area. Specifically, there are several key directions for future research. First, efforts should prioritize improving the trustworthiness of LLMs and MLLMs in anomaly detection, focusing on aspects such as effectiveness, explainability, and robustness. This will be crucial to ensuring their broader adoption, particularly in critical fields like smart manufacturing. Second, given the diverse range of data modalities involved in smart manufacturing—such as images, videos, time series, and system logs—the potential of MLLMs to handle multimodal data is significant. Harnessing this capability can unlock new opportunities for more advanced and accurate anomaly detection systems within the industry.

Bibliography

[1] Thorsten Wuest, Daniel Weimer, Christopher Irgens, and Klaus-Dieter Thoben. "Machine learning in manufacturing: advantages, challenges, and applications". In: *Production & Manufacturing Research* 4.1 (2016), pp. 23–45.

[2] Baicun Wang, Fei Tao, Xudong Fang, Chao Liu, Yufei Liu, and Theodor Freiheit. "Smart manufacturing and intelligent manufacturing: A comparative review". In: *Engineering* 7.6 (2021), pp. 738–757.

[3] Iñaki Elía and Miguel Pagola. "Anomaly Detection in Smart-Manufacturing Era: A Review". In: *Available at SSRN 4815859* ().

[4] Zohaib Jan, Farhad Ahamed, Wolfgang Mayer, Niki Patel, Georg Grossmann, Markus Stumptner, and Ana Kuusk. "Artificial intelligence for industry 4.0: Systematic review of applications, challenges, and opportunities". In: *Expert Systems with Applications* 216 (2023), p. 119456.

[5] Morteza Ghobakhloo. "Industry 4.0, digitization, and opportunities for sustainability". In: *Journal of cleaner production* 252 (2020), p. 119869.

[6] Andrew Kusiak. "Smart manufacturing". In: *International journal of production Research* 56.1-2 (2018), pp. 508–517.

[7] Pai Zheng, Honghui Wang, Zhiqian Sang, Ray Y Zhong, Yongkui Liu, Chao Liu, Khamdi Mubarok, Shiqiang Yu, and Xun Xu. "Smart manufacturing systems for Industry 4.0: Conceptual framework, scenarios, and future perspectives". In: *Frontiers of Mechanical Engineering* 13 (2018), pp. 137–150.

[8] Ray Y Zhong, Xun Xu, Eberhard Klotz, and Stephen T Newman. "Intelligent manufacturing in the context of industry 4.0: a review". In: *Engineering* 3.5 (2017), pp. 616–630.

[9] Klaus-Dieter Thoben, Stefan Wiesner, and Thorsten Wuest. " "Industrie 4.0" and smart manufacturing-a review of research issues and application examples". In: *International journal of automation technology* 11.1 (2017), pp. 4–16.

[10] Fei Tao, Qinglin Qi, Lihui Wang, and AYC Nee. "Digital twins and cyber–physical systems toward smart manufacturing and industry 4.0: Correlation and comparison". In: *Engineering* 5.4 (2019), pp. 653–661.

[11] Yang Liu, Yu Peng, Bailing Wang, Sirui Yao, and Zihe Liu. "Review on cyber-physical systems". In: *IEEE/CAA Journal of Automatica Sinica* 4.1 (2017), pp. 27–40.

[12] Liang Hu, Nannan Xie, Zhejun Kuang, and Kuo Zhao. "Review of cyber-physical system architecture". In: *2012 IEEE 15th international symposium on object/component/service-oriented real-time distributed computing workshops*. IEEE. 2012, pp. 25–30.

[13] Michael Grieves and John Vickers. "Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems". In: *Transdisciplinary perspectives on complex systems: New findings and approaches* (2017), pp. 85–113.

[14] Hyun Jung La and Soo Dong Kim. "A service-based approach to designing cyber physical systems". In: *2010 IEEE/ACIS 9th International Conference on Computer and Information Science*. IEEE. 2010, pp. 895–900.

[15] Rikard Söderberg, Kristina Wärmefjord, Johan S Carlson, and Lars Lindkvist. "Toward a Digital Twin for real-time geometry assurance in individualized production". In: *CIRP annals* 66.1 (2017), pp. 137–140.

[16] Ragunathan Rajkumar, Insup Lee, Lui Sha, and John Stankovic. "Cyber-physical systems: the next computing revolution". In: *Proceedings of the 47th design automation conference*. 2010, pp. 731–736.

[17] P Nunes, J Santos, and E Rocha. "Challenges in predictive maintenance–A review". In: *CIRP Journal of Manufacturing Science and Technology* 40 (2023), pp. 53–67.

[18] Pooja Kamat and Rekha Sugandhi. "Anomaly detection for predictive maintenance in industry 4.0-A survey". In: *E3S web of conferences*. Vol. 170. EDP Sciences. 2020, p. 02007.

[19] Sule Selcuk. "Predictive maintenance, its implementation and latest trends". In: *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 231.9 (2017), pp. 1670–1679.

[20] Tiago Zonta, Cristiano André Da Costa, Rodrigo da Rosa Righi, Miromar Jose de Lima, Eduardo Silveira da Trindade, and Guann Pyng Li. "Predictive maintenance in the Industry 4.0: A systematic literature review". In: *Computers & Industrial Engineering* 150 (2020), p. 106889.

[21] K Wang. "Intelligent predictive maintenance (IPdM) system–Industry 4.0 scenario". In: *WIT transactions on engineering sciences* 113 (2016), pp. 259–268.

[22] Ana Cachada, Jose Barbosa, Paulo Leitño, Carla AS Gcraldcs, Leonel Deusdado, Jacinta Costa, Carlos Teixeira, João Teixeira, António HJ Moreira, Pedro Miguel Moreira, et al. "Maintenance 4.0: Intelligent and predictive maintenance system architecture". In: *2018 IEEE 23rd international conference on emerging technologies and factory automation (ETFA)*. Vol. 1. IEEE. 2018, pp. 139–146.

[23] Zhe Li, Kesheng Wang, and Yafei He. "Industry 4.0-potentials for predictive maintenance". In: *6th international workshop of advanced manufacturing and automation*. Atlantis Press. 2016, pp. 42–46.

[24] Tianwen Zhu, Yongyi Ran, Xin Zhou, and Yonggang Wen. "A Survey of Predictive Maintenance: Systems, Purposes and Approaches". In: *arXiv preprint arXiv:1912.07383v2* (Mar. 2024). Corresponding author: Xin Zhou. arXiv: `1912.07383 [eess.SP]`.

[25] Lukas Ruff, Jacob R Kauffmann, Robert A Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G Dietterich, and Klaus-Robert Müller. "A unifying review of deep and shallow anomaly detection". In: *Proceedings of the IEEE* (2021).

[26] Edwin M Knorr and Raymond T Ng. "Algorithms for mining distance-based outliers in large datasets". In: *VLDB*. Vol. 98. Citeseer. 1998, pp. 392–403.

[27] Markos Markou and Sameer Singh. "Novelty detection: a review—part 1: statistical approaches". In: *Signal processing* 83.12 (2003), pp. 2481–2497.

[28] Markos Markou and Sameer Singh. "Novelty detection: a review—part 2:: neural network based approaches". In: *Signal processing* 83.12 (2003), pp. 2499–2521.

[29] Malik Agyemang, Ken Barker, and Rada Alhajj. "A comprehensive survey of numeric and symbolic outlier mining techniques". In: *Intelligent Data Analysis* 10.6 (2006), pp. 521–538.

[30] Animesh Patcha and Jung-Min Park. "An overview of anomaly detection techniques: Existing solutions and latest technological trends". In: *Computer networks* 51.12 (2007), pp. 3448–3470.

[31] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey". In: *ACM computing surveys (CSUR)* 41.3 (2009), pp. 1–58.

[32] Arthur Zimek, Erich Schubert, and Hans-Peter Kriegel. "A survey on unsupervised outlier detection in high-dimensional numerical data". In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 5.5 (2012), pp. 363–387.

[33] Charu C Aggarwal. "Outlier analysis". In: *Data mining*. Springer. 2015, pp. 237–263.

[34] Raghavendra Chalapathy and Sanjay Chawla. "Deep learning for anomaly detection: A survey". In: *CoRR* abs/1901.03407 (2019). arXiv: `1901.03407`.

[35] Azzedine Boukerche, Lining Zheng, and Omar Alfandi. "Outlier detection: Methods, models, and classification". In: *ACM Computing Surveys (CSUR)* 53.3 (2020), pp. 1–37.

[36] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. "Deep learning for anomaly detection: A review". In: *ACM computing surveys (CSUR)* 54.2 (2021), pp. 1–38.

[37] Vladimir Naumovich Vapnik, Vlamimir Vapnik, et al. "Statistical learning theory". In: (1998).

[38] Davinder Kaur, Suleyman Uslu, Kaley J Rittichier, and Arjan Durresi. "Trustworthy artificial intelligence: a review". In: *ACM computing surveys (CSUR)* 55.2 (2022), pp. 1–38.

[39] Bo Li, Peng Qi, Bo Liu, Shuai Di, Jingen Liu, Jiquan Pei, Jinfeng Yi, and Bowen Zhou. "Trustworthy AI: From principles to practices". In: *ACM Computing Surveys* 55.9 (2023), pp. 1–46.

[40] Huan Xu and Shie Mannor. "Robustness and Generalization". In: *Machine Learning* 86.3 (2012), pp. 391–423.

[41] Arijit Ukil, Soma Bandyoapdhyay, Chetanya Puri, and Arpan Pal. "IoT healthcare analytics: The importance of anomaly detection". In: *2016 IEEE 30th international conference on advanced information networking and applications (AINA)*. IEEE. 2016, pp. 994–997.

[42] Mohiuddin Ahmed, Abdun Naser Mahmood, and Md Rafiqul Islam. "A survey of anomaly detection techniques in financial domain". In: *Future Generation Computer Systems* 55 (2016), pp. 278–288.

[43] Daniel Bogdoll, Maximilian Nitsche, and J Marius Zöllner. "Anomaly Detection in Autonomous Driving: A Survey". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 4488–4499.

[44] Sylvain Fuertes, Gilles Picart, Jean-Yves Tourneret, Lotfi Chaari, André Ferrari, and Cédric Richard. "Improving spacecraft health monitoring with automatic anomaly detection techniques". In: *14th international conference on space operations*. 2016, p. 2430.

[45] Shuhan Yuan and Xintao Wu. "Trustworthy anomaly detection: A survey". In: *arXiv preprint arXiv:2202.07787* (2022).

[46] Zhong Li, Yuxuan Zhu, and Matthijs Van Leeuwen. "A survey on explainable anomaly detection". In: *ACM Transactions on Knowledge Discovery from Data* 18.1 (2023), pp. 1–54.

[47] Johannes Jakubik, Michael Vössing, Niklas Kühl, Jannis Walk, and Gerhard Satzger. "Data-centric artificial intelligence". In: *Business & Information Systems Engineering* (2024), pp. 1–9.

[48] Oussama H Hamid. "Data-centric and model-centric AI: Twin drivers of compact and robust industry 4.0 solutions". In: *Applied Sciences* 13.5 (2023), p. 2753.

[49] Daochen Zha, Zaid Pervaiz Bhat, Kwei-Herng Lai, Fan Yang, and Xia Hu. "Data-centric ai: Perspectives and challenges". In: *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM)*. SIAM. 2023, pp. 945–948.

[50] Daochen Zha, Zaid Pervaiz Bhat, Kwei-Herng Lai, Fan Yang, Zhimeng Jiang, Shaochen Zhong, and Xia Hu. "Data-centric artificial intelligence: A survey". In: *arXiv preprint arXiv:2303.10158* (2023).

[51] Mohammad Hossein Jarrahi, Ali Memariani, and Shion Guha. "The Principles of Data-Centric AI". In: *Commun. ACM* 66.8 (July 2023), pp. 84–92.

[52] Prerna Singh. "Systematic review of data-centric approaches in artificial intelligence and machine learning". In: *Data Science and Management* 6.3 (2023), pp. 144–157.

[53] Neoklis Polyzotis and Matei Zaharia. "What can data-centric AI learn from data and ML engineering?" In: *arXiv preprint arXiv:2112.06439* (2021).

[54] Gil Press. "Andrew ng launches a campaign for data-centric ai". In: *Forbes, Jun* 16 (2021), p. 2021.

[55] Zhong Li, Jiayang Shi, and Matthijs van Leeuwen. "Graph Neural Network based Log Anomaly Detection and Explanation". In: *2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. IEEE. 2024.

[56] Zhong Li and Matthijs van Leeuwen. "Feature selection for fault detection and prediction based on event log analysis". In: *ACM SIGKDD Explorations Newsletter* 24.2 (2022), pp. 96–104.

[57] Zhong Li and Matthijs van Leeuwen. "Explainable contextual anomaly detection using quantile regression forests". In: *Data Mining and Knowledge Discovery* 37.6 (2023), pp. 2517–2563.

[58] Zhong Li, Simon Geisler, Yuhang Wang, Stephan Günnemann, and Matthijs van Leeuwen. "Explainable Graph Neural Networks Under Fire". In: *arXiv preprint arXiv:2406.06417* (2024).

[59] Zhong Li, Sheng Liang, Jiayang Shi, and Matthijs van Leeuwen. " Cross-Domain Graph Level Anomaly Detection ". In: *IEEE Transactions on Knowledge & Data Engineering* 36.12 (Dec. 2024), pp. 7839–7850.

[60] Zhong Li, Matteo Quartagno, Stefan Böhringer, and Nan van Geloven. "Choosing and changing the analysis scale in non-inferiority trials with a binary outcome". In: *Clinical Trials* 19.1 (2022), pp. 14–21.

[61] Yuhang Wang, Zhong Li, Shujian Yu, and Matthijs van Leeuwen. *Labels Are Not All You Need: Evaluating Node Embedding Quality without Relying on Labels*. 2025.

[62] Zhong Li, Qi Huang, Lincen Yang, Jiayang Shi, Zhao Yang, Niki van Stein, Thomas Bäck, and Matthijs van Leeuwen. "Diffusion Models for Tabular Data: Challenges, Current Progress, and Future Directions". In: *arXiv preprint arXiv:2502.17119* (2025).

[63] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. "Estimating the support of a high-dimensional distribution". In: *Neural computation* 13.7 (2001), pp. 1443–1471.

[64] Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael I Jordan. "Detecting large-scale system problems by mining console logs". In: *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*. 2009, pp. 117–132.

[65] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. "Deeplog: Anomaly detection and diagnosis from system logs through deep learning". In: 2017, pp. 1285–1298.

[66] Weibin Meng, Ying Liu, Yichen Zhu, Shenglin Zhang, Dan Pei, Yuqing Liu, Yihao Chen, Ruizhi Zhang, Shimin Tao, Pei Sun, et al. "LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs." In: vol. 19. 7. 2019, pp. 4739–4745.

[67] Yi Wan, Yilin Liu, Dong Wang, and Yujin Wen. "Glad-paw: Graph-based log anomaly detection by position aware weighted graph attention network". In: Springer. 2021, pp. 66–77.

[68] Yufei Li, Yanchi Liu, Haoyu Wang, Zhengzhang Chen, Wei Cheng, Yuncong Chen, Wenchao Yu, Haifeng Chen, and Cong Liu. "GLAD: Content-aware Dynamic Graphs For Log Anomaly Detection". In: *arXiv preprint arXiv:2309.05953* (2023).

[69] Tong Jia, Pengfei Chen, Lin Yang, Ying Li, Fanjing Meng, and Jingmin Xu. "An approach for anomaly diagnosis based on hybrid graph model with logs for distributed services". In: *2017 IEEE international conference on web services (ICWS)*. IEEE. 2017, pp. 25–32.

[70] Zhong Li, Yuxuan Zhu, and Matthijs van Leeuwen. "A Survey on Explainable Anomaly Detection". In: *ACM Trans. Knowl. Discov. Data* (July 2023).

[71] Lin Yang, Junjie Chen, Zan Wang, Weijing Wang, Jiajun Jiang, Xuyuan Dong, and Wenbin Zhang. "Semi-supervised log-based anomaly detection via probabilistic label estimation". In: *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE. 2021, pp. 1448–1460.

[72] Zekun Tong, Yuxuan Liang, Changsheng Sun, Xinke Li, David Rosenblum, and Andrew Lim. "Digraph inception convolutional networks". In: *Advances in neural information processing systems* 33 (2020), pp. 17907–17918.

[73] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. "How Powerful are Graph Neural Networks?" In: *International Conference on Learning Representations* (2018).

[74] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. "graph2vec: Learning distributed representations of graphs". In: *International workshop on mining and learning with graphs (mlg)* (2017).

[75] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. "Weisfeiler-lehman graph kernels." In: *Journal of Machine Learning Research* 12.9 (2011).

[76] Marion Neumann, Roman Garnett, Christian Bauckhage, and Kristian Kersting. "Propagation kernels: efficient graph kernels from propagated information". In: *Machine Learning* 102.2 (2016), pp. 209–245.

[77] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation forest". In: IEEE. 2008, pp. 413–422.

[78] Lingxiao Zhao and Leman Akoglu. "On using classification datasets to evaluate graph outlier detection: Peculiar observations and new insights". In: *Big Data* (2021).

[79] Lingxiao Zhao, Saurabh Sawlani, Arvind Srinivasan, and Leman Akoglu. "Graph Anomaly Detection with Unsupervised GNNs". In: *IEEE International Conference on Data Mining (ICDM) Short* (2022).

[80] David MJ Tax and Robert PW Duin. "Support vector data description". In: *Machine learning* 54.1 (2004), pp. 45–66.

[81] Rongrong Ma, Guansong Pang, Ling Chen, and Anton van den Hengel. "Deep graph-level anomaly detection by glocal knowledge distillation". In: *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 2022, pp. 704–714.

[82] Chen Qiu, M. Kloft, Stephan Mandt, and Maja R. Rudolph. "Raising the Bar in Graph-level Anomaly Detection". In: *International Joint Conference on Artificial Intelligence* (2022).

[83] Ge Zhang, Zhenyu Yang, Jia Wu, Jian Yang, Shan Xue, Hao Peng, Jianlin Su, Chuan Zhou, Quan Z Sheng, Leman Akoglu, et al. "Dual-discriminative Graph Neural Network for Imbalanced Graph-level Anomaly Detection". In: *Advances in Neural Information Processing Systems*. 2022.

[84] Hung T. Nguyen, Pierre J. Liang, and Leman Akoglu. "Detecting Anomalous Graphs in Labeled Multi-Graph Databases". In: *ACM Trans. Knowl. Discov. Data* 17.2 (Feb. 2023).

[85] Markus Goldstein and Andreas Dengel. "Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm". In: *KI-2012: poster and demo track* 1 (2012), pp. 59–63.

[86] Amir Farzad and T Aaron Gulliver. "Unsupervised log message anomaly detection". In: *ICT Express* 6.3 (2020), pp. 229–237.

[87] Chenxi Zhang, Xin Peng, Chaofeng Sha, Ke Zhang, Zhenqing Fu, Xiya Wu, Qingwei Lin, and Dongmei Zhang. "DeepTraLog: Trace-log combined microservice anomaly detection through graph-based deep learning". In: *Proceedings of the 44th International Conference on Software Engineering*. 2022, pp. 623–634.

[88] Yongzheng Xie, Hongyu Zhang, and Muhammad Ali Babar. "LogGD: Detecting Anomalies from System Logs with Graph Neural Networks". In: *2022 IEEE 22nd International Conference on Software Quality, Reliability and Security (QRS)* (2022), pp. 299–310.

[89] Zhong Li and Matthijs van Leeuwen. "Explainable contextual anomaly detection using quantile regression forests". In: *Data Mining and Knowledge Discovery* (2023).

[90] Lin Yang, Junjie Chen, Zan Wang, Weijing Wang, Jiajun Jiang, Xuyuan Dong, and Wenbin Zhang. "Plelog: Semi-supervised log-based anomaly detection via probabilistic label estimation". In: *2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. IEEE. 2021, pp. 230–231.

[91] Bahman Bahmani, Abdur Chowdhury, and Ashish Goel. "Fast incremental and personalized PageRank". In: *Proceedings of the VLDB Endowment* 4.3 (2010), pp. 173–184.

[92] Pinjia He, Jieming Zhu, Zibin Zheng, and Michael R Lyu. "Drain: An online log parsing approach with fixed depth tree". In: *2017 IEEE international conference on web services (ICWS)*. IEEE. 2017, pp. 33–40.

[93] Jieming Zhu, Shilin He, Jinyang Liu, Pinjia He, Qi Xie, Zibin Zheng, and Michael R Lyu. "Tools and benchmarks for automated log parsing". In: *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE. 2019, pp. 121–130.

[94] Jeffrey Pennington, Richard Socher, and Christopher D Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[95] Juan Ramos et al. “Using tf-idf to determine word relevance in document queries”. In: *Proceedings of the first instructional conference on machine learning*. Vol. 242. 1. Citeseer. 2003, pp. 29–48.

[96] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. “Hierarchical graph representation learning with differentiable pooling”. In: *Advances in neural information processing systems* 31 (2018).

[97] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. “Neural message passing for quantum chemistry”. In: *International conference on machine learning*. PMLR. 2017, pp. 1263–1272.

[98] Hwan Kim, Byung Suk Lee, Won-Yong Shin, and Sungsu Lim. “Graph Anomaly Detection with Graph Neural Networks: Current Status and Challenges”. In: *IEEE Access* (2022).

[99] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. “Deep one-class classification”. In: *International conference on machine learning*. PMLR. 2018, pp. 4393–4402.

[100] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[101] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. “Explainability in graph neural networks: A taxonomic survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[102] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. “On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation”. In: *PloS one* 10.7 (2015), e0130140.

[103] Robert Schwarzenberg, Marc Hübner, David Harbecke, Christoph Alt, and Leonhard Hennig. “Layerwise Relevance Visualization in Convolutional Text Graph Classifiers”. In: *Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13)* (2019), pp. 58–62.

[104] Federico Baldassarre and Hossein Azizpour. “Explainability Techniques for Graph Convolutional Networks”. In: *ICML 2019 Workshop” Learning and Reasoning with Graph-Structured Representations”* (2019).

[105] Qingwei Lin, Hongyu Zhang, Jian-Guang Lou, Yu Zhang, and Xuewei Chen. "Log clustering based problem identification for online service systems". In: IEEE. 2016, pp. 102–111.

[106] Charu C Aggarwal and Charu C Aggarwal. *An introduction to outlier analysis*. Springer, 2017.

[107] Yue Zhao, Zain Nasrullah, and Zheng Li. "PyOD: A Python Toolbox for Scalable Outlier Detection". In: *Journal of Machine Learning Research* 20 (2019), pp. 1–7.

[108] Zhuangbin Chen, Jinyang Liu, Wenwei Gu, Yuxin Su, and Michael R Lyu. "Experience report: Deep learning-based system log analysis for anomaly detection". In: *arXiv preprint arXiv:2107.05908* (2021).

[109] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32 (2019).

[110] Matthias Fey and Jan Eric Lenssen. "Fast graph representation learning with PyTorch Geometric". In: *ICLR 2019 (RLGM Workshop)* (2019).

[111] Xihao Zhang. "An introduction to lithography machine". In: *2021 6th International Conference on Modern Management and Education Technology (MMET 2021)*. Atlantis Press. 2021, pp. 49–53.

[112] Tao Bai, Jinqi Luo, Jun Zhao, Bihan Wen, and Qian Wang. "Recent Advances in Adversarial Training for Adversarial Robustness". In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21* (Aug. 2021). Ed. by Zhi-Hua Zhou. Survey Track, pp. 4312–4321.

[113] Rakesh Bahadur Yadav, P Santosh Kumar, and Sunita Vikrant Dhavale. "A survey on log anomaly detection using deep learning". In: *2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*. IEEE. 2020, pp. 1215–1220.

[114] Shilin He, Jieming Zhu, Pinjia He, and Michael R Lyu. "Experience report: System log analysis for anomaly detection". In: IEEE. 2016, pp. 207–218.

[115] Haixuan Guo, Shuhan Yuan, and Xintao Wu. "Logbert: Log anomaly detection via bert". In: IEEE. 2021, pp. 1–8.

[116] Hossein Hamooni, Biplob Debnath, Jianwu Xu, Hui Zhang, Guofei Jiang, and Abdullah Mueen. "Logmine: Fast pattern recognition for log analytics". In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 2016, pp. 1573–1582.

[117] Hetong Dai, Heng Li, Che Shao Chen, Weiyi Shang, and Tse-Hsun Chen. "Logram: Efficient log parsing using n-gram dictionaries". In: *IEEE Transactions on Software Engineering* (2020).

[118] Van-Hoang Le and Hongyu Zhang. "Log-based anomaly detection with deep learning: how far are we?" In: *Proceedings of the 44th International Conference on Software Engineering*. 2022, pp. 1356–1367.

[119] Xu Zhang, Yong Xu, Qingwei Lin, Bo Qiao, Hongyu Zhang, Yingnong Dang, Chunyu Xie, Xinsheng Yang, Qian Cheng, Ze Li, et al. "Robust log-based anomaly detection on unstable log data". In: *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2019, pp. 807–817.

[120] Siyang Lu, Xiang Wei, Yandong Li, and Liqiang Wang. "Detecting anomaly in big data system logs using convolutional neural network". In: *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSc* IEEE. 2018, pp. 151–158.

[121] Maurice G Kendall. "A new measure of rank correlation". In: *Biometrika* 30.1/2 (1938), pp. 81–93.

[122] Ali Shojaie and Emily B Fox. "Granger causality: A review and recent advances". In: *Annual Review of Statistics and Its Application* 9 (2022), pp. 289–319.

[123] Adam Oliner and Jon Stearley. "What supercomputers say: A study of five system logs". In: *37th annual IEEE/IFIP international conference on dependable systems and networks (DSN'07)*. IEEE. 2007, pp. 575–584.

[124] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. "Efficient algorithms for mining outliers from large data sets". In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. 2000, pp. 427–438.

[125] Andrew Arnold, Yan Liu, and Naoki Abe. "Temporal causal modeling with graphical granger methods". In: *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2007, pp. 66–75.

[126] Peter Spirtes, Clark N Glymour, Richard Scheines, and David Heckerman. *Causation, prediction, and search*. MIT press, 2000.

[127] Douglas M Hawkins. *Identification of outliers*. Vol. 11. Springer, 1980.

[128] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. "A survey of network anomaly detection techniques". In: *Journal of Network and Computer Applications* 60 (2016), pp. 19–31.

[129] Eduardo J Spinosa and AC Carvalho. "Support vector machines for novel class detection in Bioinformatics". In: *Genet Mol Res* 4.3 (2005), pp. 608–15.

[130] Mohiuddin Ahmed, Abdun Naser Mahmood, and Md Rafiqul Islam. "A survey of anomaly detection techniques in financial domain". In: *Future Generation Computer Systems* 55 (2016), pp. 278–288.

[131] Inseok Hwang, Sungwan Kim, Youdan Kim, and Chze Eng Seah. "A survey of fault detection, isolation, and reconfiguration methods". In: *IEEE transactions on control systems technology* 18.3 (2009), pp. 636–653.

[132] Hongzhi Wang, Mohamed Jaward Bah, and Mohamed Hammad. "Progress in outlier detection techniques: A survey". In: *Ieee Access* 7 (2019), pp. 107964–108000.

[133] Sha Lu, Lin Liu, Jiuyong Li, Thuc Duy Le, and Jixue Liu. "Dependency-based Anomaly Detection: Framework, Methods and Benchmark". In: *arXiv preprint arXiv:2011.06716* (2020).

[134] Nicolai Meinshausen. "Quantile regression forests." In: *Journal of Machine Learning Research* 7.6 (2006), pp. 983–999.

[135] Scott M Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *Advances in neural information processing systems* 30 (2017).

[136] Hidde Fokkema, Rianne de Heide, and Tim van Erven. "Attribution-based explanations that provide recourse cannot be robust". In: *arXiv preprint arXiv:2205.15834* (2022).

[137] Qiao Cai, Haibo He, and Hong Man. "Spatial outlier detection based on iterative self-organizing learning model". In: *Neurocomputing* 117 (2013), pp. 161–172.

[138] Stan Salvador, Philip Chan, and John Brodie. "Learning States and Rules for Time Series Anomaly Detection." In: *FLAIRS conference*. 2004, pp. 306–311.

[139] Koen Smets, Brigitte Verdonk, and Elsa M Jordaan. "Discovering novelty in spatio/temporal data using one-class support vector machines". In: *2009 International Joint Conference on Neural Networks*. IEEE. 2009, pp. 2956–2963.

[140] Xiuyao Song, Mingxi Wu, Christopher Jermaine, and Sanjay Ranka. "Conditional anomaly detection". In: *IEEE Transactions on knowledge and Data Engineering* 19.5 (2007), pp. 631–645.

[141] Jiongqian Liang and Srinivasan Parthasarathy. "Robust contextual outlier detection: Where context meets sparsity". In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 2016, pp. 2167–2172.

[142] Egawati Panjei, Le Gruenwald, Eleazar Leal, Christopher Nguyen, and Shejuti Silvia. "A survey on outlier explanations". In: *The VLDB Journal* 31.5 (2022), pp. 977–1008.

[143] Hongzuo Xu, Yijie Wang, Songlei Jian, Zhenyu Huang, Yongjun Wang, Ning Liu, and Fei Li. "Beyond outlier detection: Outlier interpretation by attention-guided triplet deviation network". In: *Proceedings of the Web Conference 2021*. 2021, pp. 1328–1339.

[144] Ninghao Liu, Donghwa Shin, and Xia Hu. "Contextual outlier interpretation". In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 2018, pp. 2461–2467.

[145] Michal Valko, Branislav Kveton, Hamed Valizadegan, Gregory F Cooper, and Milos Hauskrecht. "Conditional anomaly detection with soft harmonic functions". In: *2011 IEEE 11th International Conference on Data Mining*. IEEE. 2011, pp. 735–743.

[146] Michael A Hayes and Miriam AM Capretz. "Contextual anomaly detection in big sensor data". In: *2014 IEEE International Congress on Big Data*. IEEE. 2014, pp. 64–71.

[147] Guanting Tang, Jian Pei, James Bailey, and Guozhu Dong. "Mining multidimensional contextual outliers from categorical relational data". In: *Intelligent Data Analysis* 19.5 (2015), pp. 1171–1192.

[148] Charmgil Hong and Milos Hauskrecht. "Multivariate conditional anomaly detection and its clinical application". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 29. 1. 2015.

[149] Guanjie Zheng, Susan L Brantley, Thomas Lauvaux, and Zhenhui Li. "Contextual spatial outlier detection with metric learning". In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2017, pp. 2161–2170.

[150] MY Meghanath, Deepak Pai, and Leman Akoglu. "ConOut: Con textual Outlier Detection with Multiple Contexts: Application to Ad Fraud". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2018, pp. 139–156.

[151] Choh-Man Teng. "Correcting Noisy Data." In: *ICML*. Citeseer. 1999, pp. 239–248.

[152] Yi-an Huang, Wei Fan, Wenke Lee, and Philip S Yu. "Cross-feature analysis for detecting ad-hoc routing anomalies". In: *23rd International Conference on Distributed Computing Systems, 2003. Proceedings.* IEEE. 2003, pp. 478–487.

[153] Weng-Keen Wong, Andrew W Moore, Gregory F Cooper, and Michael M Wagner. "Bayesian network anomaly pattern detection for disease outbreaks". In: *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*. 2003, pp. 808–815.

[154] Keith Noto, Carly Brodley, and Donna Slonim. "Anomaly detection using an ensemble of feature models". In: *2010 ieee international conference on data mining*. IEEE. 2010, pp. 953–958.

[155] Sakshi Babbar and Sanjay Chawla. "Mining causal outliers using Gaussian Bayesian networks". In: *2012 IEEE 24th International Conference on Tools with Artificial Intelligence*. Vol. 1. IEEE. 2012, pp. 97–104.

[156] Sha Lu, Lin Liu, Jiuyong Li, Thuc Duy Le, and Jixue Liu. "LoPAD: A Local Prediction Approach to Anomaly Detection". In: *Advances in Knowledge Discovery and Data Mining* 12085 (2020), p. 660.

[157] Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. "Outlier detection in axis-parallel subspaces of high dimensional data". In: *Pacific-asia conference on knowledge discovery and data mining*. Springer. 2009, pp. 831–838.

[158] Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. "Outlier detection in arbitrarily oriented subspaces". In: *2012 IEEE 12th international conference on data mining*. IEEE. 2012, pp. 379–388.

[159] Hoang Vu Nguyen, Emmanuel Müller, Jilles Vreeken, Fabian Keller, and Klemens Böhm. "CMI: An information-theoretic contrast measure for enhancing subspace cluster and outlier detection". In: *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM. 2013, pp. 198–206.

[160] Ismael Cabero, Irene Epifanio, Ana Piérola, and Alfredo Ballester. "Archetype analysis: A new subspace outlier detection approach". In: *Knowledge-Based Systems* 217 (2021), p. 106830.

[161] Roger Koenker and Kevin F Hallock. "Quantile regression". In: *Journal of economic perspectives* 15.4 (2001), pp. 143–156.

[162] Mark Segal and Yuanyuan Xiao. "Multivariate random forests". In: *Wiley interdisciplinary reviews: Data mining and knowledge discovery* 1.1 (2011), pp. 80–87.

[163] Leo Breiman. "Random forests". In: *Machine learning* 45 (2001), pp. 5–32.

[164] Jing Lei, Max G' Sell, Alessandro Rinaldo, Ryan J Tibshirani, and Larry Wasserman. "Distribution-free predictive inference for regression". In: *Journal of the American Statistical Association* 113.523 (2018), pp. 1094–1111.

[165] John C Gower. "A general coefficient of similarity and some of its properties". In: *Biometrics* (1971), pp. 857–871.

[166] Peter Kampstra. "Beanplot: A boxplot alternative for visual comparison of distributions". In: *Journal of statistical software* 28.1 (2008), pp. 1–9.

[167] Ines Färber, Stephan Günnemann, Hans-Peter Kriegel, Peer Kröger, Emmanuel Müller, Erich Schubert, Thomas Seidl, and Arthur Zimek. "On using class-labels in evaluation of clusterings". In: *MultiClust: 1st international workshop on discovering, summarizing and using multiple clusterings held in conjunction with KDD*. 2010, p. 1.

[168] Guilherme O Campos, Arthur Zimek, Jörg Sander, Ricardo JGB Campello, Barbora Micenková, Erich Schubert, Ira Assent, and Michael E Houle. "On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study". In: *Data mining and knowledge discovery* 30.4 (2016), pp. 891–927.

[169] Cedric Seger. *An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing*. 2018.

[170] Sevvandi Kandanaarachchi, Mario A Muñoz, Rob J Hyndman, and Kate Smith-Miles. "On normalization and algorithm selection for unsupervised outlier detection". In: *Data Mining and Knowledge Discovery* 34.2 (2020), pp. 309–354.

[171] Ece Calikus, Slawomir Nowaczyk, Mohamed-Rafik Bouguelia, and Onur Dikmen. "Wisdom of the Contexts: Active Ensemble Learning for Contextual Anomaly Detection". In: *arXiv preprint arXiv:2101.11560* (2021).

[172] Yu-Hsuan Kuo, Zhenhui Li, and Daniel Kifer. "Detecting outliers in data with correlated measures". In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 2018, pp. 287–296.

[173] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. "LOF: identifying density-based local outliers". In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. 2000, pp. 93–104.

[174] Fabrizio Angiulli and Clara Pizzuti. "Fast outlier detection in high dimensional spaces". In: *European conference on principles of data mining and knowledge discovery*. Springer. 2002, pp. 15–27.

[175] Marco Scutari, Maintainer Marco Scutari, and Hiton-PC MMPC. "Package 'bnlearn' ". In: *Bayesian network structure learning, parameter learning and inference, R package version* 4.1 (2019).

[176] Sandeep Yaramakala and Dimitris Margaritis. "Speculative Markov blanket discovery for optimal feature selection". In: *Fifth IEEE International Conference on Data Mining (ICDM'05)*. IEEE. 2005, 4–pp.

[177] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. *Classification and regression trees*. Routledge, 2017.

[178] Barbora Micenková, Brian McWilliams, and Ira Assent. "Learning outlier ensembles: The best of both worlds–supervised and unsupervised". In: *Proceedings of the ACM SIGKDD 2014 Workshop on Outlier Detection and Description under Data Diversity (ODD2). New York, NY, USA*. Citeseer. 2014, pp. 51–54.

[179] Barbora Micenková, Brian McWilliams, and Ira Assent. "Learning representations for outlier detection on a budget". In: *arXiv preprint arXiv:1507.08104* (2015).

[180] José Ramón Pasillas-Díaz and Sylvie Ratté. "An unsupervised approach for combining scores of outlier detection techniques, based on similarity measures". In: *Electronic notes in theoretical computer science* 329 (2016), pp. 61–77.

[181] Charu C Aggarwal and Saket Sathe. *Outlier ensembles: An introduction*. Springer, 2017.

[182] Milton Friedman. "The use of ranks to avoid the assumption of normality implicit in the analysis of variance". In: *Journal of the american statistical association* 32.200 (1937), pp. 675–701.

[183] Peter Bjorn Nemenyi. *Distribution-free multiple comparisons*. Princeton University, 1963.

[184] Anna L Buczak and Erhan Guven. "A survey of data mining and machine learning methods for cyber security intrusion detection". In: *IEEE Communications surveys & tutorials* 18.2 (2015), pp. 1153–1176.

[185] David Harrison Jr and Daniel L Rubinfeld. "Hedonic housing prices and the demand for clean air". In: *Journal of environmental economics and management* 5.1 (1978), pp. 81–102.

[186] Tanvir Ahmad, Assia Munir, Sajjad Haider Bhatti, Muhammad Aftab, and Muhammad Ali Raza. "Survival analysis of heart failure patients: A case study". In: *PloS one* 12.7 (2017), e0181001.

[187] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. "Graph neural networks: A review of methods and applications". In: *AI open* 1 (2020), pp. 57–81.

[188] Fenxiao Chen, Yun-Cheng Wang, Bin Wang, and C-C Jay Kuo. "Graph representation learning: a survey". In: *APSIPA Transactions on Signal and Information Processing* 9 (2020), e15.

[189] Shima Khoshraftar and Aijun An. "A survey on graph representation learning methods". In: *ACM Transactions on Intelligent Systems and Technology* 15.1 (2024), pp. 1–55.

[190] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. "Dropedge: Towards deep graph convolutional networks on node classification". In: *arXiv preprint arXiv:1907.10903* (2019).

[191] Shunxin Xiao, Shiping Wang, Yuanfei Dai, and Wenzhong Guo. "Graph neural networks in node classification: survey and evaluation". In: *Machine Vision and Applications* 33.1 (2022), p. 4.

[192] Ajay Kumar, Shashank Sheshar Singh, Kuldeep Singh, and Bhaskar Biswas. "Link prediction techniques, applications, and performance: A survey". In: *Physica A: Statistical Mechanics and its Applications* 553 (2020), p. 124289.

[193] Muhan Zhang and Yixin Chen. "Link prediction based on graph neural networks". In: *Advances in neural information processing systems* 31 (2018).

[194] Kenza Amara, Rex Ying, Zitao Zhang, Zhihao Han, Yinan Shan, Ulrik Brandes, Sebastian Schemm, and Ce Zhang. "Graphframex: Towards systematic evaluation of explainability methods for graph neural networks". In: *arXiv preprint arXiv:2206.09677* (2022).

[195] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. "Gnnexplainer: Generating explanations for graph neural networks". In: *Advances in neural information processing systems* 32 (2019).

[196] Minh Vu and My T Thai. "Pgm-explainer: Probabilistic graphical model explanations for graph neural networks". In: *Advances in neural information processing systems* 33 (2020), pp. 12225–12235.

[197] Phillip E Pope, Soheil Kolouri, Mohammad Rostami, Charles E Martin, and Heiko Hoffmann. "Explainability methods for graph convolutional neural networks". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 10772–10781.

[198] Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. "Xgnn: Towards model-level explanations of graph neural networks". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 430–438.

[199] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. "Parameterized explainer for graph neural network". In: *Advances in neural information processing systems* 33 (2020), pp. 19620–19631.

[200] Mohit Bajaj, Lingyang Chu, Zi Yu Xue, Jian Pei, Lanjun Wang, Peter Cho-Ho Lam, and Yong Zhang. "Robust counterfactual explanations on graph neural networks". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 5644–5655.

[201] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. "On explainability of graph neural networks via subgraph explorations". In: *International conference on machine learning*. PMLR. 2021, pp. 12241–12252.

[202] Ana Lucic, Maartje A Ter Hoeve, Gabriele Tolomei, Maarten De Rijke, and Fabrizio Silvestri. "Cf-gnnexplainer: Counterfactual explanations for graph neural networks". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, pp. 4499–4511.

[203] Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, and Yi Chang. "Graphlime: Local interpretable model explanations for graph neural networks". In: *IEEE Transactions on Knowledge and Data Engineering* (2022).

[204] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. "Intriguing properties of neural networks". In: *arXiv preprint arXiv:1312.6199* (2013).

[205] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. "Adversarial attacks and defences: A survey". In: *arXiv preprint arXiv:1810.00069* (2018).

[206] Hubert Baniecki and Przemyslaw Biecek. "Adversarial attacks and defenses in explainable artificial intelligence: A survey". In: *Information Fusion* (2024), p. 102303.

[207] Junfeng Fang, Wei Liu, Yuan Gao, Zemin Liu, An Zhang, Xiang Wang, and Xiangnan He. "Evaluating post-hoc explanations for graph neural networks via robustness analysis". In: *Advances in Neural Information Processing Systems* 36 (2024).

[208] Senzhang Wang, Jun Yin, Chaozhuo Li, Xing Xie, and Jianxin Wang. "V-InFoR: A Robust Graph Neural Networks Explainer for Structurally Corrupted Graphs". In: *Advances in Neural Information Processing Systems* 36 (2024).

[209] Yiqiao Li, Jianlong Zhou, Sunny Verma, and Fang Chen. "A survey of explainable graph neural networks: Taxonomy and evaluation metrics". In: *arXiv preprint arXiv:2207.12599* (2022).

[210] Jaykumar Kakkad, Jaspal Jannu, Kartik Sharma, Charu Aggarwal, and Sourav Medya. "A Survey on Explainability of Graph Neural Networks". In: *arXiv preprint arXiv:2306.01958* (2023).

[211] Chirag Agarwal, Marinka Zitnik, and Himabindu Lakkaraju. "Probing gnn explainers: A rigorous theoretical and empirical analysis of gnn explanation methods". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, pp. 8969–8996.

[212] Wenqi Fan, Han Xu, Wei Jin, Xiaorui Liu, Xianfeng Tang, Suhang Wang, Qing Li, Jiliang Tang, Jianping Wang, and Charu Aggarwal. "Jointly attacking graph neural network and its explanations". In: *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE. 2023, pp. 654–667.

[213] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Deep inside convolutional networks: Visualising image classification models and saliency maps". In: *arXiv preprint arXiv:1312.6034* (2013).

[214] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic attribution for deep networks". In: *International conference on machine learning*. PMLR. 2017, pp. 3319–3328.

[215] Thomas Schnake, Oliver Eberle, Jonas Lederer, Shinichi Nakajima, Kristof T Schütt, Klaus-Robert Müller, and Grégoire Montavon. "Higher-order explanations of graph neural networks via relevant walks". In: *IEEE transactions on pattern analysis and machine intelligence* 44.11 (2021), pp. 7581–7596.

[216] Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. "Interpreting graph neural networks for NLP with differentiable edge masking". In: *arXiv preprint arXiv:2010.00577* (2020).

[217] Juyeon Heo, Sunghwan Joo, and Taesup Moon. "Fooling neural network interpretations via adversarial model manipulation". In: *Advances in neural information processing systems* 32 (2019).

[218] Xinyang Zhang, Ningfei Wang, Hua Shen, Shouling Ji, Xiapu Luo, and Ting Wang. "Interpretable deep learning under fire". In: *29th {USENIX} security symposium ({USENIX} security 20)*. 2020.

[219] Amirata Ghorbani, Abubakar Abid, and James Zou. "Interpretation of neural networks is fragile". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 3681–3688.

[220] Xingchen Wan, Henry Kenlay, Robin Ru, Arno Blaas, Michael A Osborne, and Xiaowen Dong. "Adversarial attacks on graph classifiers via bayesian optimisation". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 6983–6996.

[221] Jiate Li, Meng Pang, Yun Dong, Jinyuan Jia, and Binghui Wang. "Graph Neural Network Explanations are Fragile". In: *arXiv preprint arXiv:2406.03193* (2024).

[222] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. "Adversarial attacks on neural networks for graph data". In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2018, pp. 2847–2856.

[223] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. "Adversarial attack on graph structured data". In: *International conference on machine learning*. PMLR. 2018, pp. 1115–1124.

[224] Wei Jin, Yaxing Li, Han Xu, Yiqi Wang, Shuiwang Ji, Charu Aggarwal, and Jiliang Tang. "Adversarial attacks and defenses on graphs". In: *ACM SIGKDD Explorations Newsletter* 22.2 (2021), pp. 19–34.

[225] Stephan Günnemann. "Graph neural networks: Adversarial robustness". In: *Graph Neural Networks: Foundations, Frontiers, and Applications* (2022), pp. 149–176.

[226] Xu Zou, Qinkai Zheng, Yuxiao Dong, Xinyu Guan, Evgeny Kharlamov, Jialiang Lu, and Jie Tang. "TDGIA:Effective Injection Attacks on Graph Neural Networks". In: *arXiv:2106.06663 [cs]* (June 2021). arXiv: 2106.06663.

[227] Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. "Topology attack and defense for graph neural networks: An optimization perspective". In: *arXiv preprint arXiv:1906.04214* (2019).

[228] Simon Geisler, Tobias Schmidt, Hakan Şirin, Daniel Zügner, Aleksandar Bojchevski, and Stephan Günnemann. "Robustness of graph neural networks at scale". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 7637–7649.

[229] Lukas Gosch, Simon Geisler, Daniel Sturm, Bertrand Charpentier, Daniel Zügner, and Stephan Günnemann. "Adversarial Training for Graph Neural Networks: Pitfalls, Solutions, and New Directions". In: *Neural Information Processing Systems, NeurIPS*. 2023.

[230] Lukas Faber, Amin K. Moghaddam, and Roger Wattenhofer. "When comparing to ground truth is wrong: On evaluating gnn explanation methods". In: *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 2021, pp. 332–341.

[231] Chirag Agarwal, Owen Queen, Himabindu Lakkaraju, and Marinka Zitnik. "Evaluating explainability for graph neural networks". In: *Scientific Data* 10.1 (2023), p. 144.

[232] Zhong Li, Jiayang Shi, and Matthijs van Leeuwen. "Graph Neural Network based Log Anomaly Detection and Explanation". In: *arXiv preprint arXiv:2307.00527* (2023).

[233] H Gerhard Vogel, Wolfgang H Vogel, H Gerhard Vogel, Günter Müller, Jürgen Sandow, and Bernward A Schölkens. *Drug discovery and evaluation: pharmacological assays*. Vol. 2. Springer, 1997.

[234] Tommaso Lanciano, Francesco Bonchi, and Aristides Gionis. "Explainable classification of brain networks via contrast subgraphs". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 3308–3318.

[235] Garrett Wilson and Diane J Cook. "A survey of unsupervised deep domain adaptation". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 11.5 (2020), pp. 1–46.

[236] Ercong Nie, Sheng Liang, Helmut Schmid, and Hinrich Schütze. "Cross-Lingual Retrieval Augmented Prompt for Low-Resource Languages". In: *Findings of the Association for Computational Linguistics: ACL 2023*. Association for Computational Linguistics, 2023, pp. 8320–8340.

[237] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. "A theory of learning from different domains". In: *Machine learning* 79 (2010), pp. 151–175.

[238] Sinno Jialin Pan and Qiang Yang. "A survey on transfer learning". In: *IEEE Transactions on knowledge and data engineering* 22.10 (2010), pp. 1345–1359.

[239] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. "A survey on deep transfer learning". In: *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III 27*. Springer. 2018, pp. 270–279.

[240] Leman Akoglu, Hanghang Tong, and Danai Koutra. "Graph based anomaly detection and description: a survey". In: *Data mining and knowledge discovery* 29 (2015), pp. 626–688.

[241] Xiaoxiao Ma, Jia Wu, Shan Xue, Jian Yang, Chuan Zhou, Quan Z Sheng, Hui Xiong, and Leman Akoglu. "A comprehensive survey on graph anomaly detection with deep learning". In: *IEEE Transactions on Knowledge and Data Engineering* (2021).

[242] Baochen Sun and Kate Saenko. "Deep coral: Correlation alignment for deep domain adaptation". In: *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14*. Springer. 2016, pp. 443–450.

[243] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. "Domain-adversarial training of neural networks". In: *The journal of machine learning research* 17.1 (2016), pp. 2096–2030.

[244] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. "Adversarial discriminative domain adaptation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7167–7176.

[245] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. "Conditional adversarial domain adaptation". In: *Advances in neural information processing systems* 31 (2018).

[246] Elif Vural. "Domain adaptation on graphs by learning graph topologies: theoretical analysis and an algorithm". In: *Turkish Journal of Electrical Engineering and Computer Sciences* 27.3 (2019), pp. 1619–1635.

[247] Yizhou Zhang, Guojie Song, Lun Du, Shuwen Yang, and Yilun Jin. "Dane: Domain adaptive network embedding". In: *arXiv preprint arXiv:1906.00684* (2019).

[248] Man Wu, Shirui Pan, Chuan Zhou, Xiaojun Chang, and Xingquan Zhu. "Unsupervised domain adaptive graph convolutional networks". In: *Proceedings of The Web Conference 2020*. 2020, pp. 1457–1467.

[249] Mehmet Pilancı and Elif Vural. "Domain adaptation on graphs by learning aligned graph bases". In: *IEEE Transactions on Knowledge and Data Engineering* 34.2 (2020), pp. 587–600.

[250] Xiao Shen, Quanyu Dai, Fu-lai Chung, Wei Lu, and Kup-Sze Choi. "Adversarial deep network embedding for cross-network node classification". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 03. 2020, pp. 2991–2999.

[251] Xiao Shen, Quanyu Dai, Sitong Mao, Fu-lai Chung, and Kup-Sze Choi. "Network together: Node classification via cross-network deep network embedding". In: *IEEE Transactions on Neural Networks and Learning Systems* 32.5 (2020), pp. 1935–1948.

[252] Guojie Song, Yizhou Zhang, Lingjun Xu, and Haibing Lu. "Domain adaptive network embedding". In: *IEEE Transactions on Big Data* 8.5 (2020), pp. 1220–1232.

[253] Kaize Ding, Kai Shu, Xuan Shan, Jundong Li, and Huan Liu. "Cross-domain graph anomaly detection". In: *IEEE Transactions on Neural Networks and Learning Systems* 33.6 (2021), pp. 2406–2415.

[254] Quanyu Dai, Xiao-Ming Wu, Jiaren Xiao, Xiao Shen, and Dan Wang. "Graph transfer learning via adversarial domain adaptation with graph convolution". In: *IEEE Transactions on Knowledge and Data Engineering* (2022).

[255] Jinfeng Li, Weifeng Liu, Yicong Zhou, Jun Yu, Dapeng Tao, and Changsheng Xu. "Domain-invariant graph for adaptive semi-supervised domain adaptation". In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 18.3 (2022), pp. 1–18.

[256] Jiaren Xiao, Quanyu Dai, Xiaochen Xie, Qi Dou, Ka-Wai Kwok, and James Lam. "Domain Adaptive Graph Infomax via Conditional Adversarial Networks". In: *IEEE Transactions on Network Science and Engineering* 10.1 (2022), pp. 35–52.

[257] Qizhou Wang, Guansong Pang, Mahsa Salehi, Wray Buntine, and Christopher Leckie. "Cross-Domain Graph Anomaly Detection via Anomaly-aware Contrastive Alignment". In: *arXiv preprint arXiv:2212.01096* (2022).

[258] Ruichu Cai, Fengzhu Wu, Zijian Li, Pengfei Wei, Lingling Yi, and Kun Zhang. "Graph domain adaptation: A generative view". In: *arXiv preprint arXiv:2106.07482* (2021).

[259] Mengxi Wu and Mohammad Rostami. "Unsupervised Domain Adaptation for Graph-Structured Data Using Class-Conditional Distribution Alignment". In: *arXiv preprint arXiv:2301.12361* (2023).

[260] Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. "Graph Domain Adaptation via Theory-Grounded Spectral Regularization". In: *The Eleventh International Conference on Learning Representations*. 2023.

[261] Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. "A fair comparison of graph neural networks for graph classification". In: *arXiv preprint arXiv:1912.09893* (2019).

[262] Xinhong Ma, Tianzhu Zhang, and Changsheng Xu. "Gcan: Graph convolutional adversarial network for unsupervised domain adaptation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8266–8276.

[263] Ni Xiao and Lei Zhang. "Dynamic weighted learning for unsupervised domain adaptation". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 15242–15251.

[264] Kuniaki Saito, Donghyun Kim, Piotr Teterwak, Stan Sclaroff, Trevor Darrell, and Kate Saenko. "Tune it the right way: Unsupervised validation of domain adaptation via soft neighborhood density". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9184–9193.

[265] Laurens Van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.11 (2008).

[266] Kaspar Riesen, Horst Bunke, et al. "IAM Graph Database Repository for Graph Based Pattern Recognition and Machine Learning." In: *SSPR/SPR*. Vol. 5342. 2008, pp. 287–297.

[267] Alex Kendall, Yarin Gal, and Roberto Cipolla. "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7482–7491.

[268] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. "Measuring and relieving the over-smoothing problem for graph neural networks from the topological view". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 04. 2020, pp. 3438–3445.

[269] Shibal Ibrahim, Natalia Ponomareva, and Rahul Mazumder. "Newer is not always better: Rethinking transferability metrics, their peculiarities, stability and performance". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2022, pp. 693–709.

[270] Wen Zhang, Lingfei Deng, Lei Zhang, and Dongrui Wu. "A survey on negative transfer". In: *IEEE/CAA Journal of Automatica Sinica* 10.2 (2022), pp. 305–329.

[271] Soroor Motie and Bijan Raahemi. "Financial fraud detection using graph neural networks: A systematic review". In: *Expert Systems with Applications* (2023), p. 122156.

[272] Weizhi Xu, Junfei Wu, Qiang Liu, Shu Wu, and Liang Wang. "Evidence-aware fake news detection with graph neural networks". In: *Proceedings of the ACM Web Conference 2022*. 2022, pp. 2501–2510.

[273] Pedro Garcia-Teodoro, Jesus Diaz-Verdejo, Gabriel Maciá-Fernández, and Enrique Vázquez. "Anomaly-based network intrusion detection: Techniques, systems and challenges". In: *computers & security* 28.1-2 (2009), pp. 18–28.

[274] Yixin Liu, Ming Jin, Shirui Pan, Chuan Zhou, Yu Zheng, Feng Xia, and S Yu Philip. "Graph self-supervised learning: A survey". In: *IEEE Transactions on Knowledge and Data Engineering* 35.6 (2022), pp. 5879–5900.

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

[276] Yu Zheng, Ming Jin, Yixin Liu, Lianhua Chi, Khoa T Phan, and Yi-Ping Phoebe Chen. "Generative and contrastive self-supervised learning for graph anomaly detection". In: *IEEE Transactions on Knowledge and Data Engineering* (2021).

[277] Ming Jin, Yixin Liu, Yu Zheng, Lianhua Chi, Yuan-Fang Li, and Shirui Pan. "Anemone: Graph anomaly detection with multi-scale contrastive learning". In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2021, pp. 3122–3126.

[278] Yixin Liu, Zhao Li, Shirui Pan, Chen Gong, Chuan Zhou, and George Karypis. "Anomaly detection on attributed networks via contrastive self-supervised learning". In: *IEEE transactions on neural networks and learning systems* 33.6 (2021), pp. 2378–2392.

[279] Xu Yuan, Na Zhou, Shuo Yu, Huafei Huang, Zhikui Chen, and Feng Xia. "Higher-order structure based anomaly detection on attributed networks". In: *2021 IEEE International Conference on Big Data (Big Data)*. IEEE. 2021, pp. 2691–2700.

[280] Zhiming Xu, Xiao Huang, Yue Zhao, Yushun Dong, and Jundong Li. "Contrastive attributed network anomaly detection with data augmentation". In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2022, pp. 444–457.

[281] Fanzhen Liu, Xiaoxiao Ma, Jia Wu, Jian Yang, Shan Xue, Amin Beheshti, Chuan Zhou, Hao Peng, Quan Z Sheng, and Charu C Aggarwal. "Dagad: Data augmentation for graph anomaly detection". In: *2022 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2022, pp. 259–268.

[282] Bo Chen, Jing Zhang, Xiaokang Zhang, Yuxiao Dong, Jian Song, Peng Zhang, Kaibo Xu, Evgeny Kharlamov, and Jie Tang. "Gccad: Graph contrastive learning for anomaly detection". In: *IEEE Transactions on Knowledge and Data Engineering* (2022).

[283] Lirong Wu, Haitao Lin, Cheng Tan, Zhangyang Gao, and Stan Z Li. "Self-supervised learning on graphs: Contrastive, generative, or predictive". In: *IEEE Transactions on Knowledge and Data Engineering* (2021).

[284] Kaize Ding, Jundong Li, Rohit Bhanushali, and Huan Liu. "Deep anomaly detection on attributed networks". In: *Proceedings of the 2019 SIAM International Conference on Data Mining*. SIAM. 2019, pp. 594–602.

[285] Jingcan Duan, Siwei Wang, Pei Zhang, En Zhu, Jingtao Hu, Hu Jin, Yue Liu, and Zhibin Dong. "Graph anomaly detection via multi-scale contrastive learning networks with augmented view". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37. 6. 2023, pp. 7459–7467.

[286] Jiaqiang Zhang, Senzhang Wang, and Songcan Chen. "Reconstruction enhanced multi-view contrastive learning for anomaly detection on attributed networks". In: *arXiv preprint arXiv:2205.04816* (2022).

[287] Yaochen Xie, Zhao Xu, Jingtun Zhang, Zhengyang Wang, and Shuiwang Ji. "Self-supervised learning of graph neural networks: A unified review". In: *IEEE transactions on pattern analysis and machine intelligence* 45.2 (2022), pp. 2412–2429.

[288] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. "A simple framework for contrastive learning of visual representations". In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607.

[289] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. "Graph contrastive learning with augmentations". In: *Advances in neural information processing systems* 33 (2020), pp. 5812–5823.

[290] Jaemin Yoo, Yue Zhao, Lingxiao Zhao, and Leman Akoglu. "DSV: An Alignment Validation Loss for Self-supervised Outlier Model Selection". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2023, pp. 254–269.

[291] Tong Zhao, Xianfeng Tang, Danqing Zhang, Haoming Jiang, Nikhil Rao, Yiwei Song, Pallav Agrawal, Karthik Subbian, Bing Yin, and Meng Jiang. "Autogda: Automated graph data augmentation for node classification". In: *Learning on Graphs Conference*. PMLR. 2022, pp. 32–1.

[292] Robert Nisbet, John Elder, and Gary D Miner. *Handbook of statistical analysis and data mining applications*. Academic press, 2009.

[293] Shachar Kaufman, Saharon Rosset, Claudia Perlich, and Ori Stitelman. "Leakage in data mining: Formulation, detection, and avoidance". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 6.4 (2012), pp. 1–21.

[294] Kay Liu, Yingtong Dou, Yue Zhao, Xueying Ding, Xiyang Hu, Ruitong Zhang, Kaize Ding, Canyu Chen, Hao Peng, Kai Shu, et al. "Bond: Benchmarking unsupervised outlier node detection on static attributed graphs". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 27021–27035.

[295] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. "Graph contrastive learning automated". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 12121–12132.

[296] Susheel Suresh, Pan Li, Cong Hao, and Jennifer Neville. "Adversarial graph augmentation to improve graph contrastive learning". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 15920–15933.

[297] Yihang Yin, Qingzhong Wang, Siyu Huang, Haoyi Xiong, and Xiang Zhang. "Autogcl: Automated graph contrastive learning via learnable view generators". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 36. 8. 2022, pp. 8892–8900.

[298] Martin Q Ma, Yue Zhao, Xiaorong Zhang, and Leman Akoglu. "The need for unsupervised outlier model selection: A review and evaluation of internal evaluation strategies". In: *ACM SIGKDD Explorations Newsletter* 25.1 (2023).

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

[300] Jundong Li, Harsh Dani, Xia Hu, and Huan Liu. "Radar: Residual analysis for anomaly detection in attributed networks." In: *IJCAI*. Vol. 17. 2017, pp. 2152–2158.

[301] Zhen Peng, Minnan Luo, Jundong Li, Huan Liu, Qinghua Zheng, et al. "ANOMALOUS: A Joint Modeling Approach for Anomaly Detection on Attributed Networks." In: *IJCAI*. 2018, pp. 3513–3519.

[302] Yizhu Jiao, Yun Xiong, Jiawei Zhang, Yao Zhang, Tianqi Zhang, and Yangyong Zhu. "Sub-graph contrast for scalable self-supervised graph representation learning". In: *2020 IEEE international conference on data mining (ICDM)*. IEEE. 2020, pp. 222–231.

[303] Jiaqi Zeng and Pengtao Xie. "Contrastive self-supervised learning for graph classification". In: *Proceedings of the AAAI conference on Artificial Intelligence*. Vol. 35. 12. 2021, pp. 10824–10832.

[304] Zhiyuan Liu, Chunjie Cao, Fangjian Tao, and Jingzhang Sun. "Revisiting Graph Contrastive Learning for Anomaly Detection". In: *arXiv preprint arXiv:2305.02496* (2023).

[305] Alexander J Ratner, Henry Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher Ré. "Learning to compose domain-specific transformations for data augmentation". In: *Advances in neural information processing systems* 30 (2017).

[306] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. "Autoaugment: Learning augmentation policies from data". In: *arXiv preprint arXiv:1805.09501* (2018).

[307] Daniel Ho, Eric Liang, Xi Chen, Ion Stoica, and Pieter Abbeel. "Population based augmentation: Efficient learning of augmentation policy schedules". In: *International conference on machine learning*. PMLR. 2019, pp. 2731–2741.

[308] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. "Fast autoaugment". In: *Advances in Neural Information Processing Systems* 32 (2019).

[309] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. "Randaugment: Practical automated data augmentation with a reduced search space". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 2020, pp. 702–703.

[310] Ryuichiro Hataya, Jan Zdenek, Kazuki Yoshizoe, and Hideki Nakayama. "Faster autoaugment: Learning augmentation strategies using backpropagation". In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXV 16*. Springer. 2020, pp. 1–16.

[311] Yonggang Li, Guosheng Hu, Yongtao Wang, Timothy Hospedales, Neil M Robertson, and Yongxin Yang. "Differentiable automatic data augmentation". In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16*. Springer. 2020, pp. 580–595.

[312] Xinyu Zhang, Qiang Wang, Jian Zhang, and Zhao Zhong. "Adversarial autoaugment". In: *arXiv preprint arXiv:1912.11188* (2019).

[313] Kaveh Hassani and Amir Hosein Khasahmadi. "Learning graph augmentations to learn graph representations". In: *arXiv preprint arXiv:2201.09830* (2022).

[314] Wei Jin, Xiaorui Liu, Xiangyu Zhao, Yao Ma, Neil Shah, and Jiliang Tang. "Automated self-supervised learning for graphs". In: *arXiv preprint arXiv:2106.05470* (2021).

[315] Tong Zhao, Yozen Liu, Leonardo Neves, Oliver Woodford, Meng Jiang, and Neil Shah. "Data augmentation for graph neural networks". In: *Proceedings of the aaai conference on artificial intelligence*. Vol. 35. 12. 2021, pp. 11015–11023.

[316] Junwei Sun, Bai Wang, and Bin Wu. "Automated graph representation learning for node classification". In: *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2021, pp. 1–7.

[317] Youzhi Luo, Michael McThrow, Wing Yee Au, Tao Komikado, Kanji Uchino, Koji Maruhashi, and Shuiwang Ji. "Automated data augmentations for graph classification". In: *arXiv preprint arXiv:2202.13248* (2022).

[318] Han Yue, Chunhui Zhang, Chuxu Zhang, and Hongfu Liu. "Label-invariant augmentation for semi-supervised graph classification". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 29350–29361.

[319] Yue Zhao, Ryan Rossi, and Leman Akoglu. "Automatic unsupervised outlier model selection". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 4489–4502.

[320] Maroua Bahri, Flavia Salutari, Andrian Putina, and Mauro Sozio. "AutoML: state of the art with a focus on anomaly detection, challenges, and research directions". In: *International Journal of Data Science and Analytics* 14.2 (2022), pp. 113–126.

[321] Xueying Ding, Lingxiao Zhao, and Leman Akoglu. "Hyperparameter sensitivity in deep outlier detection: Analysis and a scalable hyper-ensemble solution". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 9603–9616.

[322] Yue Zhao and Leman Akoglu. "Towards unsupervised hpo for outlier detection". In: *arXiv preprint arXiv:2208.11727* (2022).

[323] Chong Zhou and Randy C Paffenroth. "Anomaly detection with robust deep autoencoders". In: *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 2017, pp. 665–674.

[324] Yuening Li, Daochen Zha, Praveen Venugopal, Na Zou, and Xia Hu. "Pyodds: An end-to-end outlier detection system with automated machine learning". In: *Companion Proceedings of the Web Conference 2020*. 2020, pp. 153–157.

[325] Kwei-Herng Lai, Daochen Zha, Guanchu Wang, Junjie Xu, Yue Zhao, Devesh Kumar, Yile Chen, Purav Zumkhawaka, Minyang Wan, Diego Martinez, et al. "Tods: An automated time series outlier detection system". In: *Proceedings of the aaai conference on artificial intelligence*. Vol. 35. 18. 2021, pp. 16060–16062.

[326] Yuening Li, Zhengzhang Chen, Daochen Zha, Kaixiong Zhou, Haifeng Jin, Haifeng Chen, and Xia Hu. "Autood: Neural architecture search for outlier detection". In: *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE. 2021, pp. 2117–2122.

[327] Yuening Li, Zhengzhang Chen, Daochen Zha, Kaixiong Zhou, Haifeng Jin, Haifeng Chen, and Xia Hu. "Automated anomaly detection via curiosity-guided search and self-imitation learning". In: *IEEE Transactions on Neural Networks and Learning Systems* 33.6 (2021), pp. 2365–2377.

[328] Nicolas Goix. "How to evaluate the quality of unsupervised anomaly detection algorithms?" In: *arXiv preprint arXiv:1607.01152* (2016).

[329] Yue Zhao, Zain Nasrullah, Maciej K Hryniewicki, and Zheng Li. "LSCP: Locally selective combination in parallel outlier ensembles". In: *Proceedings of the 2019 SIAM International Conference on Data Mining*. SIAM. 2019, pp. 585–593.

[330] Henrique O Marques, Ricardo JGB Campello, Jörg Sander, and Arthur Zimek. "Internal evaluation of unsupervised outlier detection". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 14.4 (2020), pp. 1–42.

[331] Andrian Putina, Maroua Bahri, Flavia Salutari, and Mauro Sozio. "AutoAD: an Automated Framework for Unsupervised Anomaly Detectio." In: *2022 IEEE 9th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE. 2022, pp. 1–10.

[332] Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. "Hyperparameter ensembles for robustness and uncertainty quantification". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6514–6527.

[333] Yue Zhao, Ryan A Rossi, and Leman Akoglu. "Automating outlier detection via meta-learning". In: *arXiv preprint arXiv:2009.10606* (2020).

[334] Daochen Zha, Kwei-Herng Lai, Mingyang Wan, and Xia Hu. "Meta-AAD: Active anomaly detection with deep reinforcement learning". In: *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2020, pp. 771–780.

[335] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. "Fast random walk with restart and its applications". In: *Sixth international conference on data mining (ICDM'06)*. IEEE. 2006, pp. 613–622.

[336] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. "Graph attention networks". In: *arXiv preprint arXiv:1710.10903* (2017).

[337] Zhenxing Chen, Bo Liu, Meiqing Wang, Peng Dai, Jun Lv, and Liefeng Bo. "Generative adversarial attributed network anomaly detection". In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2020, pp. 1989–1992.

[338] Zekun Xu, Deovrat Kakde, and Arin Chaudhuri. "Automatic hyperparameter tuning method for local outlier factor, with applications to anomaly detection". In: *2019 IEEE International Conference on Big Data (Big Data)*. IEEE. 2019, pp. 4201–4207.

[339] Donald R Jones, Matthias Schonlau, and William J Welch. "Efficient global optimization of expensive black-box functions". In: *Journal of Global optimization* 13 (1998), pp. 455–492.

[340] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. "Collective classification in network data". In: *AI magazine* 29.3 (2008), pp. 93–93.

[341] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. "Arnetminer: extraction and mining of academic social networks". In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2008, pp. 990–998.

[342] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. "Graphsaint: Graph sampling based inductive learning method". In: *arXiv preprint arXiv:1907.04931* (2019).

[343] Patricia Iglesias Sánchez, Emmanuel Müller, Fabian Laforet, Fabian Keller, and Klemens Böhm. "Statistical selection of congruent subspaces for mining attributed graphs". In: *2013 IEEE 13th international conference on data mining*. IEEE. 2013, pp. 647–656.

[344] Jure Leskovec and Julian Mcauley. "Learning to discover social circles in ego networks". In: *Advances in neural information processing systems* 25 (2012).

[345] Srijan Kumar, Xikun Zhang, and Jure Leskovec. "Predicting dynamic embedding trajectory in temporal interaction networks". In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019, pp. 1269–1278.

[346] Shebuti Rayana and Leman Akoglu. "Collective opinion spam detection: Bridging review networks and metadata". In: *Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining*. 2015, pp. 985–994.

[347] Hezhe Qiao and Guansong Pang. "Truncated affinity maximization: One-class homophily modeling for graph anomaly detection". In: *Advances in Neural Information Processing Systems* 36 (2024).

[348] James A Hanley and Barbara J McNeil. "The meaning and use of the area under a receiver operating characteristic (ROC) curve." In: *Radiology* 143.1 (1982), pp. 29–36.

[349] Henrique O Marques, Ricardo JGB Campello, Arthur Zimek, and Jörg Sander. "On the internal evaluation of unsupervised outlier detection". In: *Proceedings of the 27th international conference on scientific and statistical database management*. 2015, pp. 1–12.

[350] Thanh Trung Nguyen, Uy Quang Nguyen, et al. "An evaluation method for unsupervised anomaly detection algorithms". In: *Journal of Computer Science and Cybernetics* 32.3 (2016), pp. 259–272.

[351] Xuanli Lisa Xie and Gerardo Beni. "A validity measure for fuzzy clustering". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 13.08 (1991), pp. 841–847.

[352] Yusheng Li, Yilun Shang, and Yiting Yang. "Clustering coefficients of large networks". In: *Information Sciences* 382 (2017), pp. 350–358.

[353] Sunny Duan, Loic Matthey, Andre Saraiva, Nicholas Watters, Christopher P Burgess, Alexander Lerchner, and Irina Higgins. "Unsupervised model selection for variational disentangled representation learning". In: *arXiv preprint arXiv:1905.12614* (2019).

[354] Zinan Lin, Kiran Thekumparampil, Giulia Fanti, and Sewoong Oh. "Infogan-cr and modelcentrality: Self-supervised model training and selection for disentangling gans". In: *international conference on machine learning*. PMLR. 2020, pp. 6127–6139.

[355] Jon M Kleinberg. "Authoritative sources in a hyperlinked environment". In: *Journal of the ACM (JACM)* 46.5 (1999), pp. 604–632.

[356] Li Yang and Abdallah Shami. "On hyperparameter optimization of machine learning algorithms: Theory and practice". In: *Neurocomputing* 415 (2020), pp. 295–316.

[357] Bernd Bischl, Martin Binder, Michel Lang, Tobias Pielok, Jakob Richter, Stefan Coors, Janek Thomas, Theresa Ullmann, Marc Becker, Anne-Laure Boulesteix, et al. "Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 13.2 (2023), e1484.

[358] Yue Zhao and Leman Akoglu. "HPOD: Hyperparameter Optimization for Unsupervised Outlier Detection". In: *AutoML 2024 Methods Track*. 2024.

[359] Mengchun Zhang, Maryam Qamar, Taegoo Kang, Yuna Jung, Chenshuang Zhang, Sung-Ho Bae, and Chaoning Zhang. "A survey on graph diffusion models: Generative ai in science for molecule, protein and material". In: *arXiv preprint arXiv:2304.01565* (2023).

[360] Thomas N Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907* (2016).

[361] Kaize Ding, Jundong Li, Nitin Agarwal, and Huan Liu. "Inductive anomaly detection on attributed networks". In: *Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence*. 2021, pp. 1288–1294.

[362] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets". In: *Advances in neural information processing systems* 27 (2014).

[363] Christopher Williams and Carl Rasmussen. "Gaussian processes for regression". In: *Advances in neural information processing systems* 8 (1995).

[364] Yue Zhao, Sean Zhang, and Leman Akoglu. "Toward unsupervised outlier model selection". In: *2022 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2022, pp. 773–782.

[365] Michael Nieberl, Alexander Zeiser, and Holger Timinger. "A Review of Data-Centric Artificial Intelligence (DCAI) and its Impact on manufacturing Industry: Challenges, Limitations, and Future Directions". In: *2024 IEEE Conference on Artificial Intelligence (CAI)*. IEEE. 2024, pp. 44–51.

[366] Chetana Hegde. "Anomaly detection in time series data using data-centric AI". In: *2022 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*. IEEE. 2022, pp. 1–6.

[367] Alexander Zeiser, Bekir Özcan, Christoph Kracke, Bas van Stein, and Thomas Bäck. "A data-centric approach to anomaly detection in layer-based additive manufacturing". In: *at-Automatisierungstechnik* 71.1 (2023), pp. 81–89.

[368] Alexander Zeiser, Bekir Özcan, Bas van Stein, and Thomas Bäck. "Evaluation of deep unsupervised anomaly detection methods with a data-centric approach for on-line inspection". In: *Computers in Industry* 146 (2023), p. 103852.

[369] Justin M Johnson and Taghi M Khoshgoftaar. "Data-centric ai for healthcare fraud detection". In: *SN Computer Science* 4.4 (2023), p. 389.

[370] Xiaoxiao Ma, Ruikun Li, Fanzhen Liu, Kaize Ding, Jian Yang, and Jia Wu. "Graph Anomaly Detection with Few Labels: A Data-Centric Approach". In: *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2024, pp. 2153–2164.

[371] Gimin Gwon, Yunyoung Kyeong, and Seki Park. "P-262: A Data-Centric Approach to Anomaly Detection for Multivariate Time-Series Data in Robot Diagnosis System". In: *SID Symposium Digest of Technical Papers*. Vol. 55. 1. Wiley Online Library. 2024, pp. 1827–1829.

[372] Sheldon Liu, Gordon Wang, Lei Liu, and Xuefeng Liu. "Data-centric anomaly detection with diffusion models". In: (2024).

[373] Rashmiranjan Nayak, Umesh Chandra Pati, and Santos Kumar Das. "A comprehensive review of datasets for detection and localization of video anomalies: a step towards data-centric artificial intelligence-based video anomaly detection". In: *Multimedia Tools and Applications* 83.21 (2024), pp. 59617–59674.

[374] Dylan Perdigão, Tiago Cruz, Paulo Simões, and Pedro Henriques Abreu. "Data-Centric Federated Learning for Anomaly Detection in Smart Grids and Other Industrial Control Systems". In: *NOMS 2024-2024 IEEE Network Operations and Management Symposium*. IEEE. 2024, pp. 1–5.

[375] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. "A survey of large language models". In: *arXiv preprint arXiv:2303.18223* (2023).

[376] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. "Large language models: A survey". In: *arXiv preprint arXiv:2402.06196* (2024).

[377] Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. "A survey on multimodal large language models". In: *arXiv preprint arXiv:2306.13549* (2023).

[378] Ruiyao Xu and Kaize Ding. "Large Language Models for Anomaly and Out-of-Distribution Detection: A Survey". In: *arXiv preprint arXiv:2409.01980* (2024).

[379] Jing Su, Chufeng Jiang, Xin Jin, Yuxin Qiao, Tingsong Xiao, Hongda Ma, Rong Wei, Zhi Jing, Jiajun Xu, and Junhong Lin. “Large language models for forecasting and anomaly detection: A systematic literature review”. In: *arXiv preprint arXiv:2402.10350* (2024).

Acknowledgements

First, I would like to express my deepest gratitude to my promotor and daily supervisor, Dr. Matthijs van Leeuwen. From the very first PhD interview, I felt that you are not only approachable but also have a good research taste. Coupled with my strong interest in the PhD program, I chose to embark on this journey with you without any hesitation. Four years have flown by, and they have only proven that my decision was the right one. Under your guidance, I evolved from a novice with limited (almost no) research experience into a more seasoned researcher. You taught me how to read and analyze literature, how to select meaningful research topics, and how to write papers —especially how to first expand them and then refine them. You also showed me how to engage with reviewers, how to present and promote my work, how to mentor undergraduate and master's students in their own research, and how to manage my time effectively. Additionally, you helped me navigate the process of career planning. After going through these transformative experiences, I believe I have grown into an independent researcher. Although I am unsure if I will stay in academia, the knowledge and habits that I have cultivated here will stay with me throughout my life, guiding me as I explore new horizons.

I would also like to thank my promotor Prof. Dr. Thomas Bäck. Thank you for being my promoter and for all the support.

Next, I would like to extend my heartfelt thanks to Dr. Nan van Geloven from LUMC. Thank you for hosting me as an intern during my time as a Master's student at ENSAI. I feel fortunate to have begun my research journey with you and to have written my first research paper under your guidance. Additionally, I deeply appreciate your support in writing a reference letter for me when I was applying for PhD positions.

Besides, I would like to express my gratitude to Dr. Fabien Navarro from ENSAI (now at Paris 1 Panthéon-Sorbonne University). Thank you for recommending potential PhD positions and supervisors and for writing a reference letter for me. Your

support was invaluable and greatly helped me along the way.

I would also like to express my sincere gratitude to Prof. Dr. Stephan Günnemann from the Technical University of Munich for hosting me as a visiting researcher in 2024. During my visit, I had the opportunity to learn from many brilliant researchers in the Data Analytics and Machine Learning group, for which I am very thankful.

In addition, I would like to extend my thanks to my co-authors: Yuxuan Zhu (Leiden University, now at Rensselaer Polytechnic Institute), Jiayang Shi (Leiden University), Sheng Liang (LMU Munich), Yuhang Wang (Leiden University), Simon Geisler (Technical University of Munich), and Nerea Perez Odriozola (University of the Basque Country). It was a pleasure working with all of you, and I truly enjoyed the collaborative process and the interesting work we created together.

Moreover, I am grateful to my colleagues from the Digital Twin program (Project 4), including Prof. Dr. Nathan van de Wouw (Eindhoven University of Technology), Dr. Jeroen van de Wijdeven (ASML), Dr. Peyman Mohajerin Esfahani (Delft University of Technology), Dr. Carlos Murguia (Eindhoven University of Technology), Dr. Fernanda Takahashi (VDL-ETG), Dr. Tom Callewaert-Doré (Canon), Fatima Abidine (Canon), Pim Hacking (Canon), Youri De Loore (Canon), Dr. Gabriel de Albuquerque Gleizer (Delft University of Technology), Farhad Ghanipoor (Eindhoven University of Technology), and Amin Sheikhi (Delft University of Technology). Your collaboration and support have been invaluable.

I would also like to extend my appreciation to my colleagues at LIACS, particularly those in the EDA group, for their camaraderie and support throughout my journey.

Finally, I want to thank my friends and family for being a constant source of encouragement and for being part of my life. Your support has meant the world to me.

Summary

This dissertation focuses on the development of trustworthy anomaly detection methods aimed at improving manufacturing processes, particularly in high-tech systems. While the primary focus is on smart manufacturing, many of the techniques and findings have broad applicability beyond this domain, with several approaches designed for general-purpose use. Given the complexity of achieving trustworthy anomaly detection, this work approaches the challenges from two complementary perspectives: data-centric AI and model-centric AI, each leading to a research question that consists of multiple sub-questions.

From the *data-centric AI perspective*, the dissertation approaches the challenges related to complex and high-dimensional data, both of which are prevalent in smart manufacturing environments. The first research question (Q1.1) deals with the problem of detecting and explaining anomalies in system logs. To solve this, we developed *Logs2Graphs*, a novel method that first transforms event logs into informative graphs. These graphs are then analyzed using a specialized graph neural network model, *OCDiGCN*, to identify anomalies. The method *Logs2Graphs* not only improves detection accuracy but also provides clear explanations by providing the most relevant nodes in the anomalous graph, with the potential to facilitate root cause analysis.

The second research question (Q1.2) focuses on managing high-dimensional data, which can undermine traditional log anomaly detection methods. We developed a novel approach to reduce the complexity of log data by identifying relevant log events that are highly associated with system faults. This method improves fault detection and prediction accuracy, as demonstrated in experiments with real-world datasets. By selecting relevant system log events, the model becomes more effective at identifying gradual fault patterns that might otherwise be obscured by irrelevant data.

From the *model-centric AI perspective*, this dissertation alleviates challenges related to explainability, robustness, generalizability, and automatability of anomaly detec-

tion models. The first model-centric research question (Q2.1) investigates how to achieve intrinsic explainability in anomaly detection systems. We proposed *QCAD*, a contextual anomaly detection method based on Quantile Regression Forests, which explores dependencies between features to identify and explain anomalies. This method enhances both detection accuracy and interpretability, allowing domain experts to understand why specific objects are considered anomalous.

A related research question (Q2.2) explores the robustness of post-hoc explanations for graph neural networks (GNNs) under adversarial attacks. We found that current post-hoc GNN explanation methods are highly vulnerable to small perturbations in graph structures, which can drastically alter explanations without changing model predictions. To achieve this, we proposed *GXAttack*, one of the first optimization-based adversarial attack methods targeting GNN explanations. This research exposes the fragility of widely used GNN explainers, highlighting the need for more robust interpretability methods in high-stakes applications.

The third model-centric research question (Q2.3) deals with generalizing anomaly detection models to new, unseen environments. We developed *ARMET*, an unsupervised domain adaptation method that uses labeled normal graphs from a source domain to detect anomalies in an unlabeled target domain. By learning appropriate graph representations through adversarial learning, ARMET achieves superior performance in cross-domain anomaly detection, making it particularly useful for evolving systems, such as those undergoing software updates or hardware modifications.

The final research question (Q2.4) focuses on automating the tuning of hyperparameters in unsupervised anomaly detection systems, where labeled data is scarce. Many self-supervised learning (SSL)-based approaches to anomaly detection are sensitive to hyperparameter settings, leading to overestimated performance when labels are improperly used for tuning. To address this, we introduced *AutoGAD*, the first automated hyperparameter selection method for SSL-based graph anomaly detection. AutoGAD uses an internal evaluation metric to select hyperparameters without relying on labels, thereby mitigating label leakage and improving model reliability.

In summary, this dissertation presents a comprehensive framework for improving anomaly detection in smart manufacturing by integrating data-centric and model-centric AI approaches. We demonstrate that anomaly detection in smart manufacturing can be enhanced by using graph neural networks to handle complex log data and employing feature selection to manage high-dimensionality. Moreover, explainable models like *QCAD* help make anomaly detection more interpretable, while *GXAttack* investigates the robustness of post-hoc GNN explanations and *ARMET* enables

adaptability across domains. Lastly, automated hyperparameter tuning via *AutoGAD* supports the development of reliable anomaly detection systems without relying on labels. These contributions not only improve the reliability and efficiency of manufacturing processes but also provide insights applicable to a wide range of fields that rely on anomaly detection.

Samenvatting

Deze dissertatie richt zich op de ontwikkeling van betrouwbare anomaliedetectiemethoden ter verbetering van productieprocessen, met name in high-tech systemen. Hoewel de primaire focus ligt op slimme productie, hebben veel van de technieken en bevindingen een brede toepasbaarheid buiten dit domein, waarbij verschillende methoden zijn ontworpen voor algemeen gebruik. Gezien de complexiteit van betrouwbare anomaliedetectie, benadert dit werk de uitdagingen vanuit twee complementaire perspectieven: data-centrische AI en model-centrische AI, elk leidend tot een onderzoeksvraag die uit meerdere subvragen bestaat.

Vanuit het *data-centrische AI perspectief* behandelt deze dissertatie de uitdagingen met betrekking tot complexe en hoog-dimensionale gegevens, die beide veel voorkomen in slimme productiesystemen. De eerste onderzoeksvraag (Q1.1) betreft het detecteren en verklaren van anomalieën in systeemlogboeken. Om dit op te lossen, hebben we *Logs2Graphs* ontwikkeld, een nieuwe methode die gebeurtenislogboeken eerst omzet in informatieve grafen. Deze grafen worden vervolgens geanalyseerd met een gespecialiseerd graaf neuraal netwerkmodel, *OCDiGCN*, om anomalieën te identificeren. *Logs2Graphs* verbetert niet alleen de detectie nauwkeurigheid, maar biedt ook duidelijke verklaringen door de meest relevante knooppunten in de anomalie-graaf te identificeren, wat de analyse van grondoorzaken mogelijk zou kunnen maken.

De tweede onderzoeksvraag (Q1.2) richt zich op het omgaan met hoog-dimensionale gegevens, die traditionele log-anomaliedetectiemethoden kunnen ondermijnen. We hebben een nieuwe aanpak ontwikkeld om de complexiteit van loggegevens te verminderen door relevante loggebeurtenissen die sterk samenhangen met systeemfouten te identificeren. Deze methode verbetert de foutdetectie en voorspellingsnauwkeurigheid, zoals aangetoond in experimenten met echte datasets. Door relevante systeemgebeurtenissen te selecteren, wordt het model effectiever in het identificeren van geleidelijke foutpatronen die anders verborgen zouden blijven.

Vanuit het *model-centrische AI perspectief* richt deze dissertatie zich op het verbeteren van uitlegbaarheid, robuustheid, generaliseerbaarheid en automatisering van anomaliedetectiemodellen. De eerste modelgerichte onderzoeksvraag (Q2.1) onderzoekt hoe intrinsieke uitlegbaarheid van anomaliedetectiesystemen kan worden bereikt. We hebben *QCAD* geïntroduceerd, een contextuele anomaliedetectiemethode gebaseerd op *Quantile Regression Forests*, die afhankelijkheden tussen kenmerken verkent om anomalieën te identificeren en te verklaren. Deze methode verhoogt zowel de detectie nauwkeurigheid als de interpretatie, waardoor domeinexperts beter kunnen begrijpen waarom bepaalde objecten als afwijkend worden beschouwd.

Een gerelateerde onderzoeksvraag (Q2.2) onderzoekt de robuustheid van post-hoc verklaringen voor graaf neurale netwerken (GNNs) onder adversariële aanvallen. We ontdekten dat huidige post-hoc GNN-verklaringstechnieken zeer kwetsbaar zijn voor kleine verstoringen in graafstructuren, die de verklaringen drastisch kunnen veranderen zonder de modelvoorspellingen aan te passen. Om dit aan te pakken, hebben we *GXAttack* ontwikkeld, een van de eerste optimalisatie-gebaseerde aanvallen die zich specifiek richt op GNN-verklaringen. Dit onderzoek onthult de kwetsbaarheid van veelgebruikte GNN-explainers en benadrukt de noodzaak voor robuustere interpretatiemethoden in kritieke toepassingen.

De derde modelgerichte onderzoeksvraag (Q2.3) betreft de generaliseerbaarheid van anomaliedetectiemodellen naar nieuwe, onbekende omgevingen. We hebben *ARMET* ontwikkeld, een ongesuperviseerde domeinaanpassingsmethode die gelabelde normale grafieken uit een brondomein gebruikt om anomalieën te detecteren in een niet-gelabeld doeldomein. Door middel van adversarial learning leert ARMET geschikte graafrepresentaties en behaalt het superieure prestaties in cross-domein anomaliedetectie, waardoor het bijzonder nuttig is voor evoluerende systemen zoals software-updates of hardwarewijzigingen.

De laatste onderzoeksvraag (Q2.4) richt zich op het automatiseren van de hyperparameterafstemming in ongesuperviseerde anomaliedetectiesystemen, waar gelabelde gegevens schaars zijn. Veel aanpakken gebaseerd op zelf-gesuperviseerd leren (SSL) voor anomaliedetectie zijn gevoelig voor hyperparameterinstellingen, wat leidt tot overschatte prestaties wanneer labels onjuist worden gebruikt voor afstemming. Om dit te verhelpen, hebben we *AutoGAD* geïntroduceerd, de eerste geautomatiseerde hyperparameterselectiemethode voor SSL-gebaseerde anomaliedetectie in grafen. AutoGAD maakt gebruik van een interne evaluatiemetriek om hyperparameters te selecteren zonder afhankelijk te zijn van labels, waardoor labellekkage wordt verminderd en de modelbetrouwbaarheid wordt verbeterd.

Samengevat presenteert deze dissertatie een uitgebreid raamwerk voor het verbeteren van anomaliedetectie in slimme productie door data-centrische en model-centrische AI-benaderingen te integreren. We laten zien dat anomaliedetectie in slimme productie kan worden verbeterd door graaf neurale netwerken te gebruiken om complexe loggegevens te verwerken en selectie van variabelen toe te passen om hoge dimensionaliteit aan te kunnen. Bovendien helpen uitlegbare modellen zoals *QCAD* om anomaliedetectie beter interpreteerbaar te maken, terwijl *GXAttack* de robuustheid van post-hoc GNN-verklaringen onderzoekt en *ARMET* de aanpasbaarheid over verschillende domeinen mogelijk maakt. Tot slot ondersteunt geautomatiseerde hyperparameterafstemming via *AutoGAD* de ontwikkeling van betrouwbare anomaliedetectiesystemen zonder afhankelijk te zijn van labels. Deze bijdragen verbeteren niet alleen de betrouwbaarheid en efficiëntie van productieprocessen, maar bieden ook inzichten die toepasbaar zijn in een breed scala aan domeinen die afhankelijk zijn van anomaliedetectie.

SIKS Dissertation Series

2016
01 Syed Saiden Abbas (RUN), Recognition of Shapes by Humans and Machines
02 Michiel Christiaan Meulendijk (UU), Optimizing medication reviews through decision support: prescribing a better pill to swallow
03 Maya Sappelli (RUN), Knowledge Work in Context: User Centered Knowledge Worker Support
04 Laurens Rietveld (VUA), Publishing and Consuming Linked Data
05 Evgeny Sherkhonov (UvA), Expanded Acyclic Queries: Containment and an Application in Explaining Missing Answers
06 Michel Wilson (TUD), Robust scheduling in an uncertain environment
07 Jeroen de Man (VUA), Measuring and modeling negative emotions for virtual training
08 Matje van de Camp (TiU), A Link to the Past: Constructing Historical Social Networks from Unstructured Data
09 Archana Nottamkandath (VUA), Trusting Crowdsourced Information on Cultural Artefacts
10 George Karafotias (VUA), Parameter Control for Evolutionary Algorithms
11 Anne Schuth (UvA), Search Engines that Learn from Their Users
12 Max Knobbout (UU), Logics for Modelling and Verifying Normative Multi-Agent Systems
13 Nana Baah Gyan (VUA), The Web, Speech Technologies and Rural Development in West Africa - An ICT4D Approach
14 Ravi Khadka (UU), Revisiting Legacy Software System Modernization
15 Steffen Michels (RUN), Hybrid Probabilistic Logics - Theoretical Aspects, Algorithms and Experiments
16 Guangliang Li (UvA), Socially Intelligent Autonomous Agents that Learn from Human Reward
17 Berend Weel (VUA), Towards Embodied Evolution of Robot Organisms
18 Albert Meroño Peñuela (VUA), Refining Statistical Data on the Web
19 Julia Efremova (TU/e), Mining Social Structures from Genealogical Data
20 Daan Odijk (UvA), Context & Semantics in News & Web Search
21 Alejandro Moreno Célleri (UT), From Traditional to Interactive Playspaces: Automatic Analysis of Player Behavior in the Interactive Tag Playground
22 Grace Lewis (VUA), Software Architecture Strategies for Cyber-Foraging Systems
23 Fei Cai (UvA), Query Auto Completion in Information Retrieval
24 Brend Wanders (UT), Repurposing and Probabilistic Integration of Data; An Iterative and data model independent approach

25 Julia Kiseleva (TU/e), Using Contextual Information to Understand Searching and Browsing Behavior
26 Dilhan Thilakarathne (VUA), In or Out of Control: Exploring Computational Models to Study the Role of Human Awareness and Control in Behavioural Choices, with Applications in Aviation and Energy Management Domains
27 Wen Li (TUD), Understanding Geo-spatial Information on Social Media
28 Mingxin Zhang (TUD), Large-scale Agent-based Social Simulation - A study on epidemic prediction and control
29 Nicolas Höning (TUD), Peak reduction in decentralised electricity systems - Markets and prices for flexible planning
30 Ruud Mattheij (TiU), The Eyes Have It
31 Mohammad Khelghati (UT), Deep web content monitoring
32 Eelco Vriezekolk (UT), Assessing Telecommunication Service Availability Risks for Crisis Organisations
33 Peter Bloem (UvA), Single Sample Statistics, exercises in learning from just one example
34 Dennis Schunselaar (TU/e), Configurable Process Trees: Elicitation, Analysis, and Enactment
35 Zhaochun Ren (UvA), Monitoring Social Media: Summarization, Classification and Recommendation
36 Daphne Karreman (UT), Beyond R2D2: The design of nonverbal interaction behavior optimized for robot-specific morphologies
37 Giovanni Sileno (UvA), Aligning Law and Action - a conceptual and computational inquiry
38 Andrea Minuto (UT), Materials that Matter - Smart Materials meet Art & Interaction Design
39 Merijn Bruijnes (UT), Believable Suspect Agents; Response and Interpersonal Style Selection for an Artificial Suspect
40 Christian Detweiler (TUD), Accounting for Values in Design
41 Thomas King (TUD), Governing Governance: A Formal Framework for Analysing Institutional Design and Enactment Governance
42 Spyros Martzoukos (UvA), Combinatorial and Compositional Aspects of Bilingual Aligned Corpora
43 Saskia Koldijk (RUN), Context-Aware Support for Stress Self-Management: From Theory to Practice
44 Thibault Sellam (UvA), Automatic Assistants for Database Exploration
45 Bram van de Laar (UT), Experiencing Brain-Computer Interface Control
46 Jorge Gallego Perez (UT), Robots to Make you Happy
47 Christina Weber (UL), Real-time foresight - Preparedness for dynamic innovation networks
48 Tanja Buttler (TUD), Collecting Lessons Learned
49 Gleb Polevoy (TUD), Participation and Interaction in Projects. A Game-Theoretic Analysis
50 Yan Wang (TiU), The Bridge of Dreams: Towards a Method for Operational Performance Alignment in IT-enabled Service Supply Chains

2017 01 Jan-Jaap Oerlemans (UL), Investigating Cybercrime
02 Sjoerd Timmer (UU), Designing and Understanding Forensic Bayesian Networks using Argumentation

03 Daniël Harold Telgen (UU), Grid Manufacturing; A Cyber-Physical Approach with Autonomous Products and Reconfigurable Manufacturing Machines
04 Mrunal Gawade (CWI), Multi-core Parallelism in a Column-store
05 Mahdieh Shadi (UvA), Collaboration Behavior
06 Damir Vandic (EUR), Intelligent Information Systems for Web Product Search
07 Roel Bertens (UU), Insight in Information: from Abstract to Anomaly
08 Rob Konijn (VUA), Detecting Interesting Differences:Data Mining in Health Insurance Data using Outlier Detection and Subgroup Discovery
09 Dong Nguyen (UT), Text as Social and Cultural Data: A Computational Perspective on Variation in Text
10 Robby van Delden (UT), (Steering) Interactive Play Behavior
11 Florian Kunneman (RUN), Modelling patterns of time and emotion in Twitter #anticipointment
12 Sander Leemans (TU/e), Robust Process Mining with Guarantees
13 Gijs Huisman (UT), Social Touch Technology - Extending the reach of social touch through haptic technology
14 Shoshannah Tekofsky (TiU), You Are Who You Play You Are: Modelling Player Traits from Video Game Behavior
15 Peter Berck (RUN), Memory-Based Text Correction
16 Aleksandr Chuklin (UvA), Understanding and Modeling Users of Modern Search Engines
17 Daniel Dimov (UL), Crowdsourced Online Dispute Resolution
18 Ridho Reinanda (UvA), Entity Associations for Search
19 Jeroen Vuurens (UT), Proximity of Terms, Texts and Semantic Vectors in Information Retrieval
20 Mohammadbashir Sedighi (TUD), Fostering Engagement in Knowledge Sharing: The Role of Perceived Benefits, Costs and Visibility
21 Jeroen Linssen (UT), Meta Matters in Interactive Storytelling and Serious Gaming (A Play on Worlds)
22 Sara Magliacane (VUA), Logics for causal inference under uncertainty
23 David Graus (UvA), Entities of Interest — Discovery in Digital Traces
24 Chang Wang (TUD), Use of Affordances for Efficient Robot Learning
25 Veruska Zamborlini (VUA), Knowledge Representation for Clinical Guidelines, with applications to Multimorbidity Analysis and Literature Search
26 Merel Jung (UT), Socially intelligent robots that understand and respond to human touch
27 Michiel Joosse (UT), Investigating Positioning and Gaze Behaviors of Social Robots: People's Preferences, Perceptions and Behaviors
28 John Klein (VUA), Architecture Practices for Complex Contexts
29 Adel Alhuraibi (TiU), From IT-BusinessStrategic Alignment to Performance: A Moderated Mediation Model of Social Innovation, and Enterprise Governance of IT"
30 Wilma Latuny (TiU), The Power of Facial Expressions
31 Ben Ruijl (UL), Advances in computational methods for QFT calculations
32 Thaer Samar (RUN), Access to and Retrievability of Content in Web Archives
33 Brigit van Loggem (OU), Towards a Design Rationale for Software Documentation: A Model of Computer-Mediated Activity
34 Maren Scheffel (OU), The Evaluation Framework for Learning Analytics

35 Martine de Vos (VUA), Interpreting natural science spreadsheets
36 Yuanhao Guo (UL), Shape Analysis for Phenotype Characterisation from High-throughput Imaging
37 Alejandro Montes Garcia (TU/e), WiBAF: A Within Browser Adaptation Framework that Enables Control over Privacy
38 Alex Kayal (TUD), Normative Social Applications
39 Sara Ahmadi (RUN), Exploiting properties of the human auditory system and compressive sensing methods to increase noise robustness in ASR
40 Altaf Hussain Abro (VUA), Steer your Mind: Computational Exploration of Human Control in Relation to Emotions, Desires and Social Support For applications in human-aware support systems
41 Adnan Manzoor (VUA), Minding a Healthy Lifestyle: An Exploration of Mental Processes and a Smart Environment to Provide Support for a Healthy Lifestyle
42 Elena Sokolova (RUN), Causal discovery from mixed and missing data with applications on ADHD datasets
43 Maaike de Boer (RUN), Semantic Mapping in Video Retrieval
44 Garm Lucassen (UU), Understanding User Stories - Computational Linguistics in Agile Requirements Engineering
45 Bas Testerink (UU), Decentralized Runtime Norm Enforcement
46 Jan Schneider (OU), Sensor-based Learning Support
47 Jie Yang (TUD), Crowd Knowledge Creation Acceleration
48 Angel Suarez (OU), Collaborative inquiry-based learning

2018 01 Han van der Aa (VUA), Comparing and Aligning Process Representations
02 Felix Mannhardt (TU/e), Multi-perspective Process Mining
03 Steven Bosems (UT), Causal Models For Well-Being: Knowledge Modeling, Model-Driven Development of Context-Aware Applications, and Behavior Prediction
04 Jordan Janeiro (TUD), Flexible Coordination Support for Diagnosis Teams in Data-Centric Engineering Tasks
05 Hugo Huurdeman (UvA), Supporting the Complex Dynamics of the Information Seeking Process
06 Dan Ionita (UT), Model-Driven Information Security Risk Assessment of Socio-Technical Systems
07 Jieting Luo (UU), A formal account of opportunism in multi-agent systems
08 Rick Smetsers (RUN), Advances in Model Learning for Software Systems
09 Xu Xie (TUD), Data Assimilation in Discrete Event Simulations
10 Julienka Mollee (VUA), Moving forward: supporting physical activity behavior change through intelligent technology
11 Mahdi Sargolzaei (UvA), Enabling Framework for Service-oriented Collaborative Networks
12 Xixi Lu (TU/e), Using behavioral context in process mining
13 Seyed Amin Tabatabaei (VUA), Computing a Sustainable Future
14 Bart Joosten (TiU), Detecting Social Signals with Spatiotemporal Gabor Filters
15 Naser Davarzani (UM), Biomarker discovery in heart failure
16 Jaebok Kim (UT), Automatic recognition of engagement and emotion in a group of children
17 Jianpeng Zhang (TU/e), On Graph Sample Clustering
18 Henriette Nakad (UL), De Notaris en Private Rechtspraak

19 Minh Duc Pham (VUA), Emergent relational schemas for RDF
20 Manxia Liu (RUN), Time and Bayesian Networks
21 Aad Slootmaker (OU), EMERGO: a generic platform for authoring and playing scenario-based serious games
22 Eric Fernandes de Mello Araújo (VUA), Contagious: Modeling the Spread of Behaviours, Perceptions and Emotions in Social Networks
23 Kim Schouten (EUR), Semantics-driven Aspect-Based Sentiment Analysis
24 Jered Vroon (UT), Responsive Social Positioning Behaviour for Semi-Autonomous Telepresence Robots
25 Riste Gligorov (VUA), Serious Games in Audio-Visual Collections
26 Roelof Anne Jelle de Vries (UT),Theory-Based and Tailor-Made: Motivational Messages for Behavior Change Technology
27 Maikel Leemans (TU/e), Hierarchical Process Mining for Scalable Software Analysis
28 Christian Willemse (UT), Social Touch Technologies: How they feel and how they make you feel
29 Yu Gu (TiU), Emotion Recognition from Mandarin Speech
30 Wouter Beek (VUA), The "Kïn "semantic web"stands for "knowledge": scaling semantics to the web

---

2019 01 Rob van Eijk (UL),Web privacy measurement in real-time bidding systems. A graph-based approach to RTB system classification
02 Emmanuelle Beauxis Aussalet (CWI, UU), Statistics and Visualizations for Assessing Class Size Uncertainty
03 Eduardo Gonzalez Lopez de Murillas (TU/e), Process Mining on Databases: Extracting Event Data from Real Life Data Sources
04 Ridho Rahmadi (RUN), Finding stable causal structures from clinical data
05 Sebastiaan van Zelst (TU/e), Process Mining with Streaming Data
06 Chris Dijkshoorn (VUA), Nichesourcing for Improving Access to Linked Cultural Heritage Datasets
07 Soude Fazeli (TUD), Recommender Systems in Social Learning Platforms
08 Frits de Nijs (TUD), Resource-constrained Multi-agent Markov Decision Processes
09 Fahimeh Alizadeh Moghaddam (UvA), Self-adaptation for energy efficiency in software systems
10 Qing Chuan Ye (EUR), Multi-objective Optimization Methods for Allocation and Prediction
11 Yue Zhao (TUD), Learning Analytics Technology to Understand Learner Behavioral Engagement in MOOCs
12 Jacqueline Heinerman (VUA), Better Together
13 Guanliang Chen (TUD), MOOC Analytics: Learner Modeling and Content Generation
14 Daniel Davis (TUD), Large-Scale Learning Analytics: Modeling Learner Behavior & Improving Learning Outcomes in Massive Open Online Courses
15 Erwin Walraven (TUD), Planning under Uncertainty in Constrained and Partially Observable Environments
16 Guangming Li (TU/e), Process Mining based on Object-Centric Behavioral Constraint (OCBC) Models
17 Ali Hurriyetoglu (RUN),Extracting actionable information from microtexts

18 Gerard Wagenaar (UU), Artefacts in Agile Team Communication
19 Vincent Koeman (TUD), Tools for Developing Cognitive Agents
20 Chide Groenouwe (UU), Fostering technically augmented human collective intelligence
21 Cong Liu (TU/e), Software Data Analytics: Architectural Model Discovery and Design Pattern Detection
22 Martin van den Berg (VUA),Improving IT Decisions with Enterprise Architecture
23 Qin Liu (TUD), Intelligent Control Systems: Learning, Interpreting, Verification
24 Anca Dumitrache (VUA), Truth in Disagreement - Crowdsourcing Labeled Data for Natural Language Processing
25 Emiel van Miltenburg (VUA), Pragmatic factors in (automatic) image description
26 Prince Singh (UT), An Integration Platform for Synchromodal Transport
27 Alessandra Antonaci (OU), The Gamification Design Process applied to (Massive) Open Online Courses
28 Esther Kuindersma (UL), Cleared for take-off: Game-based learning to prepare airline pilots for critical situations
29 Daniel Formolo (VUA), Using virtual agents for simulation and training of social skills in safety-critical circumstances
30 Vahid Yazdanpanah (UT), Multiagent Industrial Symbiosis Systems
31 Milan Jelisavcic (VUA), Alive and Kicking: Baby Steps in Robotics
32 Chiara Sironi (UM), Monte-Carlo Tree Search for Artificial General Intelligence in Games
33 Anil Yaman (TU/e), Evolution of Biologically Inspired Learning in Artificial Neural Networks
34 Negar Ahmadi (TU/e), EEG Microstate and Functional Brain Network Features for Classification of Epilepsy and PNES
35 Lisa Facey-Shaw (OU), Gamification with digital badges in learning programming
36 Kevin Ackermans (OU), Designing Video-Enhanced Rubrics to Master Complex Skills
37 Jian Fang (TUD), Database Acceleration on FPGAs
38 Akos Kadar (OU), Learning visually grounded and multilingual representations

2020 01 Armon Toubman (UL), Calculated Moves: Generating Air Combat Behaviour
02 Marcos de Paula Bueno (UL), Unraveling Temporal Processes using Probabilistic Graphical Models
03 Mostafa Deghani (UvA), Learning with Imperfect Supervision for Language Understanding
04 Maarten van Gompel (RUN), Context as Linguistic Bridges
05 Yulong Pei (TU/e), On local and global structure mining
06 Preethu Rose Anish (UT), Stimulation Architectural Thinking during Requirements Elicitation - An Approach and Tool Support
07 Wim van der Vegt (OU), Towards a software architecture for reusable game components
08 Ali Mirsoleimani (UL),Structured Parallel Programming for Monte Carlo Tree Search
09 Myriam Traub (UU), Measuring Tool Bias and Improving Data Quality for Digital Humanities Research
10 Alifah Syamsiyah (TU/e), In-database Preprocessing for Process Mining
11 Sepideh Mesbah (TUD), Semantic-Enhanced Training Data AugmentationMethods for Long-Tail Entity Recognition Models
12 Ward van Breda (VUA), Predictive Modeling in E-Mental Health: Exploring Applicability in Personalised Depression Treatment

13 Marco Virgolin (CWI), Design and Application of Gene-pool Optimal Mixing Evolutionary Algorithms for Genetic Programming
14 Mark Raasveldt (CWI/UL), Integrating Analytics with Relational Databases
15 Konstantinos Georgiadis (OU), Smart CAT: Machine Learning for Configurable Assessments in Serious Games
16 Ilona Wilmont (RUN), Cognitive Aspects of Conceptual Modelling
17 Daniele Di Mitri (OU), The Multimodal Tutor: Adaptive Feedback from Multimodal Experiences
18 Georgios Methenitis (TUD), Agent Interactions & Mechanisms in Markets with Uncertainties: Electricity Markets in Renewable Energy Systems
19 Guido van Capelleveen (UT), Industrial Symbiosis Recommender Systems
20 Albert Hankel (VUA), Embedding Green ICT Maturity in Organisations
21 Karine da Silva Miras de Araujo (VUA), Where is the robot?: Life as it could be
22 Maryam Masoud Khamis (RUN), Understanding complex systems implementation through a modeling approach: the case of e-government in Zanzibar
23 Rianne Conijn (UT), The Keys to Writing: A writing analytics approach to studying writing processes using keystroke logging
24 Lenin da Nóbrega Medeiros (VUA/RUN), How are you feeling, human? Towards emotionally supportive chatbots
25 Xin Du (TU/e), The Uncertainty in Exceptional Model Mining
26 Krzysztof Leszek Sadowski (UU), GAMBIT: Genetic Algorithm for Model-Based mixed-Integer opTimization
27 Ekaterina Muravyeva (TUD), Personal data and informed consent in an educational context
28 Bibeg Limbu (TUD), Multimodal interaction for deliberate practice: Training complex skills with augmented reality
29 Ioan Gabriel Bucur (RUN), Being Bayesian about Causal Inference
30 Bob Zadok Blok (UL), Creatief, Creatiever, Creatiefst
31 Gongjin Lan (VUA), Learning better – From Baby to Better
32 Jason Rhuggenaath (TU/e), Revenue management in online markets: pricing and online advertising
33 Rick Gilsing (TU/e), Supporting service-dominant business model evaluation in the context of business model innovation
34 Anna Bon (UM), Intervention or Collaboration? Redesigning Information and Communication Technologies for Development
35 Siamak Farshidi (UU), Multi-Criteria Decision-Making in Software Production

2021 01 Francisco Xavier Dos Santos Fonseca (TUD),Location-based Games for Social Interaction in Public Space
02 Rijk Mercuur (TUD), Simulating Human Routines: Integrating Social Practice Theory in Agent-Based Models
03 Seyyed Hadi Hashemi (UvA), Modeling Users Interacting with Smart Devices
04 Ioana Jivet (OU), The Dashboard That Loved Me: Designing adaptive learning analytics for self-regulated learning
05 Davide Dell'Anna (UU), Data-Driven Supervision of Autonomous Systems
06 Daniel Davison (UT), "Hey robot, what do you think?"How children learn with a social robot

07 Armel Lefebvre (UU), Research data management for open science
08 Nardie Fanchamps (OU), The Influence of Sense-Reason-Act Programming on Computational Thinking
09 Cristina Zaga (UT), The Design of Robothings. Non-Anthropomorphic and Non-Verbal Robots to Promote Children's Collaboration Through Play
10 Quinten Meertens (UvA), Misclassification Bias in Statistical Learning
11 Anne van Rossum (UL), Nonparametric Bayesian Methods in Robotic Vision
12 Lei Pi (UL), External Knowledge Absorption in Chinese SMEs
13 Bob R. Schadenberg (UT), Robots for Autistic Children: Understanding and Facilitating Predictability for Engagement in Learning
14 Negin Samaeemofrad (UL), Business Incubators: The Impact of Their Support
15 Onat Ege Adali (TU/e), Transformation of Value Propositions into Resource Re-Configurations through the Business Services Paradigm
16 Esam A. H. Ghaleb (UM), Bimodal emotion recognition from audio-visual cues
17 Dario Dotti (UM), Human Behavior Understanding from motion and bodily cues using deep neural networks
18 Remi Wieten (UU), Bridging the Gap Between Informal Sense-Making Tools and Formal Systems - Facilitating the Construction of Bayesian Networks and Argumentation Frameworks
19 Roberto Verdecchia (VUA), Architectural Technical Debt: Identification and Management
20 Masoud Mansoury (TU/e), Understanding and Mitigating Multi-Sided Exposure Bias in Recommender Systems
21 Pedro Thiago Timbó Holanda (CWI), Progressive Indexes
22 Sihang Qiu (TUD), Conversational Crowdsourcing
23 Hugo Manuel Proença (UL), Robust rules for prediction and description
24 Kaijie Zhu (TU/e), On Efficient Temporal Subgraph Query Processing
25 Eoin Martino Grua (VUA), The Future of E-Health is Mobile: Combining AI and Self-Adaptation to Create Adaptive E-Health Mobile Applications
26 Benno Kruit (CWI/VUA), Reading the Grid: Extending Knowledge Bases from Human-readable Tables
27 Jelte van Waterschoot (UT), Personalized and Personal Conversations: Designing Agents Who Want to Connect With You
28 Christoph Selig (UL), Understanding the Heterogeneity of Corporate Entrepreneurship Programs

2022 01 Judith van Stegeren (UT), Flavor text generation for role-playing video games
02 Paulo da Costa (TU/e), Data-driven Prognostics and Logistics Optimisation: A Deep Learning Journey
03 Ali el Hassouni (VUA), A Model A Day Keeps The Doctor Away: Reinforcement Learning For Personalized Healthcare
04 Ünal Aksu (UU), A Cross-Organizational Process Mining Framework
05 Shiwei Liu (TU/e), Sparse Neural Network Training with In-Time Over-Parameterization
06 Reza Refaei Afshar (TU/e), Machine Learning for Ad Publishers in Real Time Bidding
07 Sambit Praharaj (OU), Measuring the Unmeasurable? Towards Automatic Co-located Collaboration Analytics
08 Maikel L. van Eck (TU/e), Process Mining for Smart Product Design

09 Oana Andreea Inel (VUA), Understanding Events: A Diversity-driven Human-Machine Approach
10 Felipe Moraes Gomes (TUD), Examining the Effectiveness of Collaborative Search Engines
11 Mirjam de Haas (UT), Staying engaged in child-robot interaction, a quantitative approach to studying preschoolers' engagement with robots and tasks during second-language tutoring
12 Guanyi Chen (UU), Computational Generation of Chinese Noun Phrases
13 Xander Wilcke (VUA), Machine Learning on Multimodal Knowledge Graphs: Opportunities, Challenges, and Methods for Learning on Real-World Heterogeneous and Spatially-Oriented Knowledge
14 Michiel Overeem (UU), Evolution of Low-Code Platforms
15 Jelmer Jan Koorn (UU), Work in Process: Unearthing Meaning using Process Mining
16 Pieter Gijsbers (TU/e), Systems for AutoML Research
17 Laura van der Lubbe (VUA), Empowering vulnerable people with serious games and gamification
18 Paris Mavromoustakos Blom (TiU), Player Affect Modelling and Video Game Personalisation
19 Bilge Yigit Ozkan (UU), Cybersecurity Maturity Assessment and Standardisation
20 Fakhra Jabeen (VUA), Dark Side of the Digital Media - Computational Analysis of Negative Human Behaviors on Social Media
21 Seethu Mariyam Christopher (UM), Intelligent Toys for Physical and Cognitive Assessments
22 Alexandra Sierra Rativa (TiU), Virtual Character Design and its potential to foster Empathy, Immersion, and Collaboration Skills in Video Games and Virtual Reality Simulations
23 Ilir Kola (TUD), Enabling Social Situation Awareness in Support Agents
24 Samaneh Heidari (UU), Agents with Social Norms and Values - A framework for agent based social simulations with social norms and personal values
25 Anna L.D. Latour (UL), Optimal decision-making under constraints and uncertainty
26 Anne Dirkson (UL), Knowledge Discovery from Patient Forums: Gaining novel medical insights from patient experiences
27 Christos Athanasiadis (UM), Emotion-aware cross-modal domain adaptation in video sequences
28 Onuralp Ulusoy (UU), Privacy in Collaborative Systems
29 Jan Kolkmeier (UT), From Head Transform to Mind Transplant: Social Interactions in Mixed Reality
30 Dean De Leo (CWI), Analysis of Dynamic Graphs on Sparse Arrays
31 Konstantinos Traganos (TU/e), Tackling Complexity in Smart Manufacturing with Advanced Manufacturing Process Management
32 Cezara Pastrav (UU), Social simulation for socio-ecological systems
33 Brinn Hekkelman (CWI/TUD), Fair Mechanisms for Smart Grid Congestion Management
34 Nimat Ullah (VUA), Mind Your Behaviour: Computational Modelling of Emotion & Desire Regulation for Behaviour Change
35 Mike E.U. Ligthart (VUA), Shaping the Child-Robot Relationship: Interaction Design Patterns for a Sustainable Interaction

2023 01 Bojan Simoski (VUA), Untangling the Puzzle of Digital Health Interventions

02 Mariana Rachel Dias da Silva (TiU), Grounded or in flight? What our bodies can tell us about the whereabouts of our thoughts
03 Shabnam Najafian (TUD), User Modeling for Privacy-preserving Explanations in Group Recommendations
04 Gineke Wiggers (UL), The Relevance of Impact: bibliometric-enhanced legal information retrieval
05 Anton Bouter (CWI), Optimal Mixing Evolutionary Algorithms for Large-Scale Real-Valued Optimization, Including Real-World Medical Applications
06 António Pereira Barata (UL), Reliable and Fair Machine Learning for Risk Assessment
07 Tianjin Huang (TU/e), The Roles of Adversarial Examples on Trustworthiness of Deep Learning
08 Lu Yin (TU/e), Knowledge Elicitation using Psychometric Learning
09 Xu Wang (VUA), Scientific Dataset Recommendation with Semantic Techniques
10 Dennis J.N.J. Soemers (UM), Learning State-Action Features for General Game Playing
11 Fawad Taj (VUA), Towards Motivating Machines: Computational Modeling of the Mechanism of Actions for Effective Digital Health Behavior Change Applications
12 Tessel Bogaard (VUA), Using Metadata to Understand Search Behavior in Digital Libraries
13 Injy Sarhan (UU), Open Information Extraction for Knowledge Representation
14 Selma Čaušević (TUD), Energy resilience through self-organization
15 Alvaro Henrique Chaim Correia (TU/e), Insights on Learning Tractable Probabilistic Graphical Models
16 Peter Blomsma (TiU), Building Embodied Conversational Agents: Observations on human nonverbal behaviour as a resource for the development of artificial characters
17 Meike Nauta (UT), Explainable AI and Interpretable Computer Vision –From Oversight to Insight
18 Gustavo Penha (TUD), Designing and Diagnosing Models for Conversational Search and Recommendation
19 George Aalbers (TiU), Digital Traces of the Mind: Using Smartphones to Capture Signals of Well-Being in Individuals
20 Arkadiy Dushatskiy (TUD), Expensive Optimization with Model-Based Evolutionary Algorithms applied to Medical Image Segmentation using Deep Learning
21 Gerrit Jan de Bruin (UL), Network Analysis Methods for Smart Inspection in the Transport Domain
22 Alireza Shojaifar (UU), Volitional Cybersecurity
23 Theo Theunissen (UU), Documentation in Continuous Software Development
24 Agathe Balayn (TUD), Practices Towards Hazardous Failure Diagnosis in Machine Learning
25 Jurian Baas (UU), Entity Resolution on Historical Knowledge Graphs
26 Loek Tonnaer (TU/e), Linearly Symmetry-Based Disentangled Representations and their Out-of-Distribution Behaviour
27 Ghada Sokar (TU/e), Learning Continually Under Changing Data Distributions
28 Floris den Hengst (VUA), Learning to Behave: Reinforcement Learning in Human Contexts
29 Tim Draws (TUD), Understanding Viewpoint Biases in Web Search Results

2024 01 Daphne Miedema (TU/e), On Learning SQL: Disentangling concepts in data systems education
02 Emile van Krieken (VUA), Optimisation in Neurosymbolic Learning Systems

03 Feri Wijayanto (RUN), Automated Model Selection for Rasch and Mediation Analysis
04 Mike Huisman (UL), Understanding Deep Meta-Learning
05 Yiyong Gou (UM), Aerial Robotic Operations: Multi-environment Cooperative Inspection & Construction Crack Autonomous Repair
06 Azqa Nadeem (TUD), Understanding Adversary Behavior via XAI: Leveraging Sequence Clustering to Extract Threat Intelligence
07 Parisa Shayan (TiU), Modeling User Behavior in Learning Management Systems
08 Xin Zhou (UvA), From Empowering to Motivating: Enhancing Policy Enforcement through Process Design and Incentive Implementation
09 Giso Dal (UT), Probabilistic Inference Using Partitioned Bayesian Networks
10 Cristina-Iulia Bucur (VUA), Linkflows: Towards Genuine Semantic Publishing in Science
11 withdrawn
12 Peide Zhu (TUD), Towards Robust Automatic Question Generation For Learning
13 Enrico Liscio (TUD), Context-Specific Value Inference via Hybrid Intelligence
14 Larissa Capobianco Shimomura (TU/e), On Graph Generating Dependencies and their Applications in Data Profiling
15 Ting Liu (VUA), A Gut Feeling: Biomedical Knowledge Graphs for Interrelating the Gut Microbiome and Mental Health
16 Arthur Barbosa Câmara (TUD), Designing Search-as-Learning Systems
17 Razieh Alidoosti (VUA), Ethics-aware Software Architecture Design
18 Laurens Stoop (UU), Data Driven Understanding of Energy-Meteorological Variability and its Impact on Energy System Operations
19 Azadeh Mozafari Mehr (TU/e), Multi-perspective Conformance Checking: Identifying and Understanding Patterns of Anomalous Behavior
20 Ritsart Anne Plantenga (UL), Omgang met Regels
21 Federica Vinella (UU), Crowdsourcing User-Centered Teams
22 Zeynep Ozturk Yurt (TU/e), Beyond Routine: Extending BPM for Knowledge-Intensive Processes with Controllable Dynamic Contexts
23 Jie Luo (VUA), Lamarck' s Revenge: Inheritance of Learned Traits Improves Robot Evolution
24 Nirmal Roy (TUD), Exploring the effects of interactive interfaces on user search behaviour
25 Alisa Rieger (TUD), Striving for Responsible Opinion Formation in Web Search on Debated Topics
26 Tim Gubner (CWI), Adaptively Generating Heterogeneous Execution Strategies using the VOILA Framework
27 Lincen Yang (UL), Information-theoretic Partition-based Models for Interpretable Machine Learning
28 Leon Helwerda (UL), Grip on Software: Understanding development progress of Scrum sprints and backlogs
29 David Wilson Romero Guzman (VUA), The Good, the Efficient and the Inductive Biases: Exploring Efficiency in Deep Learning Through the Use of Inductive Biases
30 Vijanti Ramautar (UU), Model-Driven Sustainability Accounting
31 Ziyu Li (TUD), On the Utility of Metadata to Optimize Machine Learning Workflows
32 Vinicius Stein Dani (UU), The Alpha and Omega of Process Mining
33 Siddharth Mehrotra (TUD), Designing for Appropriate Trust in Human-AI interaction

34 Robert Deckers (VUA), From Smallest Software Particle to System Specification - MuD-ForM: Multi-Domain Formalization Method
35 Sicui Zhang (TU/e), Methods of Detecting Clinical Deviations with Process Mining: a fuzzy set approach
36 Thomas Mulder (TU/e), Optimization of Recursive Queries on Graphs
37 James Graham Nevin (UvA), The Ramifications of Data Handling for Computational Models
38 Christos Koutras (TUD), Tabular Schema Matching for Modern Settings
39 Paola Lara Machado (TU/e), The Nexus between Business Models and Operating Models: From Conceptual Understanding to Actionable Guidance
40 Montijn van de Ven (TU/e), Guiding the Definition of Key Performance Indicators for Business Models
41 Georgios Siachamis (TUD), Adaptivity for Streaming Dataflow Engines
42 Emmeke Veltmeijer (VUA), Small Groups, Big Insights: Understanding the Crowd through Expressive Subgroup Analysis
43 Cedric Waterschoot (KNAW Meertens Instituut), The Constructive Conundrum: Computational Approaches to Facilitate Constructive Commenting on Online News Platforms
44 Marcel Schmitz (OU), Towards learning analytics-supported learning design
45 Sara Salimzadeh (TUD), Living in the Age of AI: Understanding Contextual Factors that Shape Human-AI Decision-Making
46 Georgios Stathis (Leiden University), Preventing Disputes: Preventive Logic, Law & Technology
47 Daniel Daza (VUA), Exploiting Subgraphs and Attributes for Representation Learning on Knowledge Graphs
48 Ioannis Petros Samiotis (TUD), Crowd-Assisted Annotation of Classical Music Compositions

---

2025 01 Max van Haastrecht (UL), Transdisciplinary Perspectives on Validity: Bridging the Gap Between Design and Implementation for Technology-Enhanced Learning Systems
02 Jurgen van den Hoogen (JADS), Time Series Analysis Using Convolutional Neural Networks
03 Andra-Denis Ionescu (TUD), Feature Discovery for Data-Centric AI
04 Rianne Schouten (TU/e), Exceptional Model Mining for Hierarchical Data
05 Nele Albers (TUD), Psychology-Informed Reinforcement Learning for Situated Virtual Coaching in Smoking Cessation
06 Daniël Vos (TUD), Decision Tree Learning: Algorithms for Robust Prediction and Policy Optimization
07 Ricky Maulana Fajri (TU/e), Towards Safer Active Learning: Dealing with Unwanted Biases, Graph-Structured Data, Adversary, and Data Imbalance
08 Stefan Bloemheuvel (TiU), Spatio-Temporal Analysis Through Graphs: Predictive Modeling and Graph Construction
09 Fadime Kaya (VUA), Decentralized Governance Design - A Model-Based Approach
10 Zhao Yang (UL), Enhancing Autonomy and Efficiency in Goal-Conditioned Reinforcement Learning
11 Shahin Sharifi Noorian (TUD), From Recognition to Understanding: Enriching Visual Models Through Multi-Modal Semantic Integration
12 Lijun Lyu (TUD), Interpretability in Neural Information Retrieval

13 Fuda van Diggelen (VUA), Robots Need Some Education: on the complexity of learning in evolutionary robotics
14 Gennaro Gala (TU/e), Probabilistic Generative Modeling with Latent Variable Hierarchies
15 Michiel van der Meer (UL), Opinion Diversity through Hybrid Intelligence
16 Monika Grewal (TU Delft), Deep Learning for Landmark Detection, Segmentation, and Multi-Objective Deformable Registration in Medical Imaging
17 Matteo De Carlo (VUA), Real Robot Reproduction: Towards Evolving Robotic Ecosystems
18 Anouk Neerincx (UU), Robots That Care: How Social Robots Can Boost Children's Mental Wellbeing
19 Fang Hou (UU), Trust in Software Ecosystems
20 Alexander Melchior (UU), Modelling for Policy is More Than Policy Modelling (The Useful Application of Agent-Based Modelling in Complex Policy Processes)
21 Mandani Ntekouli (UM), Bridging Individual and Group Perspectives in Psychopathology: Computational Modeling Approaches using Ecological Momentary Assessment Data
22 Hilde Weerts (TU/e), Decoding Algorithmic Fairness: Towards Interdisciplinary Understanding of Fairness and Discrimination in Algorithmic Decision-Making
23 Roderick van der Weerdt (VUA), IoT Measurement Knowledge Graphs: Constructing, Working and Learning with IoT Measurement Data as a Knowledge Graph

Curriculum Vitae

Zhong Li was born in Yunnan Province, China, in 1994. He lived in various places in Yunnan until 2012, when he moved to Shanghai. From 2012 to 2016, he pursued a Bachelor's degree in Statistics at Tongji University, graduating in 2016 with the highest distinction as an Excellent Graduate of Shanghai City. That same year, he was selected for the prestigious *Sino-French Dual Degree Program in Science and Engineering*, spending his first graduate year at Tongji University and the following two years at École Nationale de la Statistique et de l'Analyse de l'Information (ENSAI) in France. In 2017, Zhong relocated to Rennes, France, to pursue a diplôme d'ingénieur in Data Science at ENSAI. From May to August 2018, he completed a research internship at LUMC in Leiden, The Netherlands. In 2019, he moved to Paris to complete his Master's thesis at HSBC bank, where he interned from April to September. During his time at ENSAI, Zhong was awarded the prestigious Master Eiffel Scholarship by the French Ministry for Europe and Foreign Affairs, a recognition for which he remains deeply grateful. In 2020, Zhong returned to Shanghai and graduated with dual degrees: a Master's degree in Mathematics from Tongji University and a diplôme d'ingénieur in Data Science from ENSAI. Following his graduation, he worked full-time as a system engineer at SMEE from April to October. In November 2020, he moved to Leiden, the Netherlands, to pursue a PhD in Computer Science at Leiden University. From March to June 2024, Zhong worked as a visiting researcher at TUM, Germany.

During his PhD studies, he took courses in *Explainable AI*, *Responsible AI*, *Social AI*, and *Scientific Conduct*, among others. As the world changes, so do Zhong's research interests. But his curiosity about its evolution stands the test of time.