Utah State University

DigitalCommons@USU

All Graduate Theses and Dissertations, Fall 2023 to Present

Graduate Studies

5-2024

Achieving Responsible Anomaly Detection

Xiao Han
Utah State University, xiao.han@usu.edu

Follow this and additional works at: https://digitalcommons.usu.edu/etd2023

 Part of the Computer Sciences Commons

Recommended Citation

Han, Xiao, "Achieving Responsible Anomaly Detection" (2024). *All Graduate Theses and Dissertations, Fall 2023 to Present*. 126.
https://digitalcommons.usu.edu/etd2023/126

This Dissertation is brought to you for free and open access by the Graduate Studies at DigitalCommons@USU. It has been accepted for inclusion in All Graduate Theses and Dissertations, Fall 2023 to Present by an authorized administrator of DigitalCommons@USU. For more information, please contact digitalcommons@usu.edu.

ACHIEVING RESPONSIBLE ANOMALY DETECTION

by

Xiao Han

A dissertation submitted in partial fulfillment
of the requirements for the degree

of

DOCTOR OF PHILOSOPHY

in

Computer Science

Approved:

Shuhan Yuan, Ph.D. Major Professor	Soukaina Filali Boubrahimi, Ph.D. Committee Member
Curtis Dyreson, Ph.D. Committee Member	John Edwards, Ph.D. Committee Member
Luis Gordillo, Ph.D. Committee Member	D. Richard Cutler, Ph.D. Vice Provost of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2024

Copyright © Xiao Han 2024

All Rights Reserved

ABSTRACT

Achieving Responsible Anomaly Detection

by

Xiao Han, Doctor of Philosophy

Utah State University, 2024

Major Professor: Shuhan Yuan, Ph.D.
Department: Computer Science

This dissertation presents a comprehensive approach to achieving responsible anomaly detection, focusing on enhancing performance and ensuring explainability and fairness. The initial phase of the dissertation focuses on performance enhancement, where cutting-edge machine learning techniques are harnessed to improve the precision of anomaly detection systems. Through the development and implementation of LogGPT, LogTAD, FADS, and FADScr, this research demonstrates how few-shot learning, reinforcement learning, transfer learning, and the application of generative pre-trained transformer models can significantly bolster anomaly detection in diverse and challenging environments. These methodologies are not just theoretical contributions; they are practical innovations that address real-world limitations, such as sparse data availability and the need for cross-domain adaptability.

Moving beyond performance, the dissertation delves into the critical area of explainability, merging it with root cause analysis and mitigation. Here, the dissertation explores the imperative of making anomaly detection systems not only effective but also understandable and accountable. InterpretableSAD, RecAD, AERCA, and RootCLAM stand out as key contributions in this area, offering novel methodologies for dissecting and understanding the complexities behind detected anomalies. These frameworks enable users to not only identify

but also understand the causal factors of anomalies, facilitating more informed and effective mitigation strategies.

Additionally, the dissertation addresses the critical aspect of fairness through the introduction of the CFAD framework, which pioneers the incorporation of counterfactual fairness into anomaly detection. This dimension ensures that the developed systems operate without bias, offering equitable and ethical anomaly detection across diverse scenarios.

In summary, this body of work represents a significant leap forward in the field of anomaly detection. It not only advances the technical capabilities of detection systems through innovative machine learning techniques but also addresses the ethical implications of their deployment. By integrating performance enhancement with explainability, root cause analysis, mitigation strategies, and fairness considerations, this dissertation sets a new standard for the development and application of anomaly detection technologies, ensuring that they serve the broader goal of promoting a more secure, understandable, and equitable digital world.

(234 pages)

PUBLIC ABSTRACT

Achieving Responsible Anomaly Detection

Xiao Han

In the digital transformation era, safeguarding online systems against anomalies – unusual patterns indicating potential threats or malfunctions – has become crucial. This dissertation embarks on enhancing the accuracy, explainability, and ethical integrity of anomaly detection systems. By integrating advanced machine learning techniques, it improves anomaly detection performance and incorporates fairness and explainability at its core.

The research tackles performance enhancement in anomaly detection by leveraging few-shot learning, demonstrating how systems can effectively identify anomalies with minimal training data. This approach overcomes data scarcity challenges. Reinforcement learning is employed to iteratively refine models, enhancing decision-making processes. Transfer learning enables the application of insights across domains, improving system versatility. The integration of Generative Pre-trained Transformer (GPT) models marks a significant advancement, offering enhanced precision in anomaly identification through sophisticated language modeling techniques.

Exploring explainability and root cause analysis, the dissertation introduces advanced frameworks that shed light on the mechanisms behind anomaly detections across different data types. InterpretableSAD enhances sequential log data analysis, pinpointing specific anomalous events to clarify detection processes. RootCLAM addresses tabular data anomalies through causal inference, identifying root causes and suggesting actionable mitigation strategies. The narrative extends to time series data with RecAD and AERCA; RecAD offers algorithmic recourse by proposing minimal-cost actions for correcting anomalies, while AERCA utilizes an autoencoder-based framework to unravel Granger causality, illuminating

causative factors behind anomalies. These frameworks empower users with the knowledge and tools to understand and act upon findings, facilitating the identification of irregularities across diverse data landscapes.

Ethical integrity remains paramount, addressed through the CFAD framework, which ensures counterfactual fairness by embedding ethical principles directly into anomaly detection processes. This guarantees equitable treatment across scenarios, advocating for technologies that serve all users equitably and challenge inherent biases.

Extensive evaluations on various datasets demonstrate the proposed models' effectiveness in addressing anomaly detection challenges. This dissertation contributes to advancing techniques that are not only accurate but also interpretable and fair, promoting the responsible use of anomaly detection in real-world applications. This dissertation lays a solid foundation for further exploration into advanced anomaly detection techniques, promising to guide the development of even more robust, transparent, and equitable systems in the digital age.

To my family and parents, whose love and support have been my constant guide and strength.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to all those who have supported me throughout the journey of completing this dissertation. The completion of this work would not have been possible without their invaluable assistance, encouragement, and guidance.

First and foremost, I extend my sincere thanks to my advisor, Dr. Shuhan Yuan, for his unwavering support, insightful feedback, and constant encouragement. His expertise and guidance have been indispensable in shaping both the direction and substance of my research.

I am also grateful to all my dissertation committee members, Dr. Soukaina Filali Boubrahimi, Dr. Curtis Dyreson, Dr. John Edwards and Dr. Luis Gordillo, for their constructive critiques and valuable suggestions that greatly improved this dissertation.

Special thanks to He Cheng and Xingyi Zhao, whose companionship and intellectual exchanges have enriched my graduate experience. Their perspectives and camaraderie have been a source of motivation and joy throughout this process.

My appreciation also goes to the staff and faculty in the Department of Computer Science for providing a stimulating and supportive research environment. Their expertise and assistance have been invaluable.

To my friends and family, who have provided me with endless love and encouragement, thank you. Your belief in me has been a constant source of strength. A special mention goes to my family, whose sacrifices and unwavering belief in my abilities have made all of this possible.

This dissertation stands as a testament to the collaborative effort and spirit of all those mentioned and many unmentioned. Thank you.

This work was supported in part by the Utah State University Presidential Doctoral Research Fellowship program and the National Science Foundation through Grant 2103829.

Xiao Han

CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

Anomaly detection is a pivotal task in data analysis that has garnered increasing attention in recent years, driven by the exponential growth of big data and the escalating need to automatically identify unusual patterns or behaviors across various domains. An anomaly signifies an event or pattern deviating from the norm or expected behavior within a dataset. These deviations can arise from diverse factors, including errors, fraud, cyber-attacks, or outliers, and manifest in various data types, such as numerical and categorical, as well as in different data formats, including tabular and sequential. The detection of anomalies is crucial, given their potential significant implications across multiple sectors, including finance, healthcare, and cybersecurity. However, anomaly identification poses challenges due to their rarity, subtlety, and the inherent difficulty in distinguishing them from normal behavior.

Responsible anomaly detection involves multiple facets: performance, explainability, and fairness, all vital for developing trustworthy and efficient systems. Performance pertains to the accuracy of anomaly detection methods, while explainability relates to the methods' transparency and the interpretability of results. Fairness ensures unbiased and equitable treatment of all individuals or groups impacted by anomaly detection.

Despite its importance, responsible anomaly detection faces several challenges. A primary challenge is enhancing performance across various scenarios, including cold-start situations and few-shot anomaly detection and classification. Increasing explainability, especially for deep learning models, which often lack transparency, remains another hurdle. Traditional explainable anomaly detection approaches utilize attention mechanisms to provide explanations based on attention scores. However, these scores tend to explain "Why was the sample detected as an anomaly?" rather than "Why is the sample an anomaly?" Additionally, maintaining fairness to prevent bias and ensure equitable treatment across all cases is a critical challenge.

This dissertation addresses these challenges by proposing novel methods focusing on performance, explainability, and fairness. We investigate transfer learning and unsupervised learning techniques to develop adaptable anomaly detection methods suitable for cold-start scenarios with enhanced accuracy. We also explore reinforcement learning techniques [1–4] to improve anomaly detection and classification performance in few-shot and semi-supervised settings. Furthermore, we aim to develop deep learning models and other interpretable techniques to offer more transparent and understandable results. Our research also delves into ensuring fairness in anomaly detection, contributing to the development of more reliable and equitable methods. These methods lay a foundation for future research in anomaly detection, applicable across various domains, including finance, healthcare, and cybersecurity. By tackling these challenges, we aspire to facilitate better decision-making and improve outcomes, contributing significantly to society's ability to detect and prevent fraudulent or malicious activities.

1.1 Motivation

1.1.1 Why Is Anomaly Detection Important?

Anomaly detection stands as a pivotal area of research within a multitude of fields, including computer science, engineering, finance, and security. Its paramount importance is underscored by its capability to uncover unusual patterns or events that deviate from established norms or expected behaviors [5–7]. These deviations, if unnoticed, can inflict considerable damage on organizations or systems by undermining their integrity, efficiency, and security. By leveraging anomaly detection techniques, stakeholders can significantly enhance decision-making processes, improve system accuracy and efficiency, and unearth anomalous events within vast datasets.

In finance, anomaly detection plays a crucial role in safeguarding economic integrity and operational stability [8–11]. It empowers financial institutions to pinpoint fraudulent transactions and irregular financial patterns swiftly, thereby mitigating potential financial losses and preserving trust in financial systems. Techniques developed for anomaly detection

enable early identification of fraud, insider trading, or credit risk anomalies, which are essential for preempting financial misconduct and ensuring regulatory compliance.

In security, the deployment of anomaly detection methodologies is vital for maintaining the robustness of cybersecurity defenses [10, 12–16]. These techniques facilitate the early detection of cyber-attacks, including malware infiltration, unauthorized access attempts, and other security breaches, by analyzing deviations from normal network or system behaviors. By identifying these threats early, organizations can prevent potential data breaches, safeguard sensitive information, and maintain the continuity of their operations.

In the manufacturing industry, anomaly detection contributes to enhancing product quality and operational efficiency [17–21]. It enables the early identification of equipment malfunctions, process deviations, or quality defects, facilitating timely interventions that can prevent costly downtimes and ensure the reliability of manufacturing processes. This not only helps in reducing operational costs but also in maintaining competitive advantage through consistent product quality.

In healthcare, the application of anomaly detection is transforming patient care and disease management [22–24]. By analyzing medical data, such as patient records, imaging data, or genetic information, anomaly detection algorithms can reveal atypical patterns indicative of diseases or health conditions. This assists healthcare professionals in diagnosing conditions early, personalizing treatment plans, and ultimately improving patient outcomes. Moreover, in public health, anomaly detection can track the spread of infectious diseases by identifying outbreaks, enabling timely and effective responses to health crises.

As the amount of data generated by various systems continues to increase, anomaly detection is becoming increasingly crucial in many domains. As such, research in anomaly detection is critical in developing effective and efficient algorithms and techniques for identifying anomalous events in large datasets. With the potential to improve decision-making processes, enhance system accuracy and efficiency, and detect anomalous events in large datasets, doing research in anomaly detection is essential for organizations across various fields.

1.1.2 Why Is Responsible Anomaly Detection Important?

Responsible anomaly detection emerges as a critical frontier in the field, emphasizing the integration of social responsibility principles into the development of anomaly detection algorithms [25, 26]. These principles – performance, explainability, and fairness – are not just technical metrics but are foundational to ensuring that anomaly detection systems contribute positively to society.

The paramount importance of performance in anomaly detection systems is underscored by their need for accuracy and reliability. High-performing models are characterized by their ability to minimize false positives and negatives while maintaining superior accuracy. A substantial volume of research is dedicated to enhancing this accuracy [27–31], reflecting the field's recognition of its critical role. Additionally, it is essential for these models to adapt to new data and changing environments, thereby preserving their effectiveness over time. The primary challenge in this context arises from the dynamic nature of real-world data, which can fluctuate unpredictably and potentially compromise the accuracy of these models [32]. In response, the development and application of transferable anomaly detection methods have become crucial. These methods significantly improve the robustness and adaptability of the models, ensuring they deliver consistent performance across a variety of scenarios [33–37].

Explainable anomaly detection [38–45] refers to the ability to understand and interpret the reasons behind the detected anomalies. In some cases, black-box anomaly detection models may be effective at identifying anomalies, but the lack of transparency and interpretability can be problematic, particularly when trying to diagnose and resolve issues. By developing explainable anomaly detection methods, researchers can provide insights into the underlying causes of anomalies, leading to more informed decision-making.

Fairness in anomaly detection [46,47] is an emerging area of research that seeks to ensure that the detection of anomalies does not result in unjust or discriminatory outcomes. For example, if an anomaly detection model is used to identify fraudulent transactions, it may inadvertently target certain groups unfairly, resulting in higher false-positive rates or biased decision-making. By developing fairness-aware anomaly detection methods, researchers can

mitigate these risks and ensure that the detection of anomalies is done in a fair and equitable manner.

The significance of focusing on performance, explainability, and fairness extends beyond mere technical achievements; it's about fostering trust, accountability, and equity in the application of anomaly detection technologies. By navigating these challenges, the development of anomaly detection models can be steered towards ensuring not only technological efficacy but also social responsibility. These efforts are instrumental in crafting robust, reliable anomaly detection systems that offer valuable insights and bolster decision-making processes, all while upholding ethical standards and promoting fairness. As such, the exploration and advancement in these areas are not just beneficial but essential for the responsible application of anomaly detection in real-world settings.

1.1.3 Challenges in Responsible Anomaly Detection

Responsible anomaly detection is confronted with a set of multifaceted challenges that span technical, ethical, and operational dimensions. Addressing these challenges is crucial for developing systems that are not only effective in identifying anomalies but also adhere to principles of fairness, transparency, and accountability. The key challenges identified in this dissertation include:

- Performance Enhancement in Diverse Conditions: Achieving high accuracy and adaptability in anomaly detection across varying scenarios, especially in the face of sparse data and dynamically changing environments, remains a significant challenge. This issue underscores the need for advanced methodologies capable of maintaining robust performance over time.

- Explainability: Enhancing the transparency of anomaly detection models, particularly those employing complex algorithms, poses a challenge. It is essential to develop techniques that allow users to understand the rationale behind detected anomalies and to perform root cause analysis for deeper insights.

- Fairness in Anomaly Detection Processes: Ensuring that anomaly detection systems operate without bias and provide equitable outcomes across all users is a critical challenge. Developing fairness-aware approaches is necessary to prevent discriminatory practices and ensure ethical application.

1.2 Contributions and Findings

The main contributions and findings of this dissertation are the following:

Performance. Performance is a crucial aspect of responsible anomaly detection since poor performance could have negative consequences such as false alarms, missed anomalies, or unnecessary investigations. Our goal is to improve the performance of anomaly detection in various scenarios.

- We introduce LogGPT, a novel framework that employs the Generative Pre-trained Transformer (GPT) model for log anomaly detection, improving the performance of anomaly detection in log data. This task is crucial for maintaining system security and reliability but is complicated by the large volume and complex structure of log data. LogGPT leverages the power of generative log language models to predict the next log entry based on the preceding sequence, and enhances its performance with a reinforcement learning strategy specifically designed for anomaly detection. Experimental results on three datasets demonstrate that LogGPT significantly outperforms existing state-of-the-art approaches, showcasing its effectiveness in identifying anomalies in log data.

- We present LogTAD, a framework aimed at addressing the challenge of unsupervised log anomaly detection across different systems. The task focuses on detecting anomalous log records to ensure the stability of systems, which is complicated by the scarcity of anomalous samples and the impracticality of gathering extensive training data for newly deployed systems. Our method leverages a domain adversarial adaptation technique to transfer knowledge of log data from various systems, enabling the model to identify anomalies across multiple platforms effectively. By mapping log sequences

into a shared hypersphere and establishing a system-agnostic center, LogTAD can detect anomalies by their distance from this center. Experimental results demonstrate LogTAD's ability to achieve high accuracy in cross-system anomaly detection with a minimal set of logs from new systems, highlighting its efficacy and potential for practical applications.

- We develop the Few-shot Anomaly Detection and Classification model through Reinforced Data Selection (FADS), a novel framework that addresses the challenge of anomaly detection and classification with limited labeled samples and a large number of unlabeled samples. FADS iteratively improves anomaly detection and classification performance by exploring unlabeled datasets to augment the training set. By employing a reinforcement learning-based data selection strategy, FADS selects high-quality weakly-labeled samples from the unlabeled dataset, enhancing the training process. Experimental results demonstrate that FADS significantly outperforms traditional methods, achieving state-of-the-art performance in anomaly detection and classification with only a few labeled samples initially. We then present FADScr, an extension of the FADS, which further improves the performance in anomaly detection and classification by iteratively enhancing performance with a combinatorial reward system. Experimental results show the improvement by extending FADS to FADScr and FADScr achieves state-of-the-art performance in anomaly detection and classification tasks.

Explainability. Explainability is an essential aspect of responsible anomaly detection since the lack of transparency or interpretability can undermine the credibility of anomaly detection models.

- We introduce InterpretableSAD, an interpretable framework for anomaly detection in sequential log data, focusing on the identification of anomalous sequences and the fine-grained detection of anomalous events within those sequences. Anomaly detection in sequential log data presents a significant challenge due to the rarity of anomalies

and the difficulty of identifying anomalous events within sequences. InterpretableSAD addresses this challenge by employing a novel data augmentation strategy, generating anomalous sequences through negative sampling to train a binary classification model effectively. Post-training, we apply the interpretable machine learning technique of Integrated Gradients to identify and explain the contribution of individual events to the detection of anomalies. This approach allows for both high-level sequence anomaly detection and detailed event-level analysis. Experimental results on multiple log datasets demonstrate the effectiveness of InterpretableSAD.

- We present RecAD, an algorithmic recourse framework designed for anomaly detection in multivariate time series. This framework aims to recommend minimal-cost actions for correcting abnormal time series to facilitate domain experts in effectively understanding and rectifying abnormal behaviors. Utilizing UnSupervised Anomaly Detection (USAD) for identifying abnormal steps, RecAD innovatively predicts recourse actions that can reverse these abnormalities, ensuring a return to normalcy for subsequent time steps. It notably considers the downstream impact of interventions by generating counterfactual time series for the following steps. Experimental evaluations on two synthetic and one real-world dataset have demonstrated RecAD's efficacy in suggesting actionable recourses for anomaly detection, validating its practical utility and effectiveness.

- We develop AERCA, a novel autoencoder-based framework for root cause analysis of anomalies in multivariate time series. This work addresses the challenge of identifying the root cause of anomalies due to complex dependencies between time series. AERCA utilizes an encoder-decoder structure to capture the Granger causality and models the distributions of exogenous variables in normal conditions. By defining anomalies as interventions on exogenous variables, it highlights those significantly deviated from normal status. Experiments on various datasets show AERCA's effectiveness in capturing causal relationships and identifying anomaly root causes.

- We introduce RootCLAM, a framework designed to tackle the challenge of locating the root causes of anomalies and mitigating them effectively. This task focuses on identifying abnormal features caused by external interventions and mitigating such anomalies to restore normalcy. RootCLAM operates in two phases: initially, it localizes the root cause of an anomaly by identifying features affected by external interventions. Subsequently, it proposes mitigation actions targeted at these root causes, aiming to revert the affected features and outcomes to their normal states. The methodology leverages a Structural Causal Model (SCM) to understand the causal relationships and applies soft interventions for mitigation, using a continuous optimization-based algorithm to compute these actions. Experimental results on multiple datasets demonstrate RootCLAM's effectiveness in not only accurately localizing root causes but also in successfully mitigating anomalies with minimal perturbations compared to baseline methods.

Fairness. Fairness is another important factor in responsible anomaly detection as biased or unfair models can have significant negative impacts on certain individuals.

- We develop CFAD, which aims to ensure counterfactual fairness in anomaly detection models by maintaining consistent detection outcomes across factual and counterfactual worlds. The CFAD framework tackles the challenge of generating counterfactual data and ensuring fair anomaly detection through a two-phase process: firstly, by generating counterfactual examples that represent counterfactual scenarios where sensitive attributes are modified, thereby providing a basis for fairness comparison. Secondly, it leverages an adversarial training methodology that forces the model to minimize differences in detection outcomes between actual and counterfactual examples, thus promoting fairness. Experimental results on both synthetic and real datasets demonstrate that CFAD effectively detects anomalies while ensuring counterfactual fairness, outperforming several baseline methods in balancing effectiveness and fairness.

1.3 Organization of the Dissertation

The dissertation unfolds through eleven chapters, methodically covering performance, explainability, and fairness in anomaly detection. The first four chapters (2-4) delve into enhancing detection performance, showcasing LogGPT, LogTAD, FADS, and FADScr. Chapters 5-8 pivot to explainability, introducing InterpretableSAD, RecAD, AERCA, and RootCLAM for clearer anomaly detection insights. Chapter 9 focuses on fairness with the CFAD framework, ensuring unbiased detection. Chapter 10, the final section, outlines future research directions, acknowledging current limitations and envisioning the next steps in advancing anomaly detection.

- **Chapter 2 (LogGPT: Log Anomaly Detection via GPT)** focuses on the development and application of LogGPT for log data analysis, emphasizing its impact on improving anomaly detection accuracy.

- **Chapter 3 (LogTAD: Unsupervised cross-system log anomaly detection via domain adaptation)** details the innovative approach of LogTAD to overcome the cold-start problem in anomaly detection, utilizing domain adaptation techniques.

- **Chapter 4 (FADS: Few-shot anomaly detection and classification through reinforced data selection & FADScr: Few-shot Anomaly Detection and Classification Through Reinforced Data Selection with a Combinatorial Reward)** presents the FADS and FADScr frameworks, showcasing their contributions to the domain of few-shot learning in anomaly detection. FADS distinguishes itself by utilizing a reinforced data selection process, thereby improving the efficiency of anomaly detection with limited data. FADScr, on the other hand, advances this approach by introducing a combinatorial reward mechanism. This mechanism further refines the data selection process, leading to enhanced performance in few-shot anomaly detection scenarios.

- **Chapter 5 (InterpretableSAD: Interpretable anomaly detection in sequential log data)** discusses the InterpretableSAD framework, showcasing its approach to

making anomaly detection in log data interpretable.

- **Chapter 6 (RecAD: Algorithmic Recourse for Anomaly Detection in Multivariate Time Series)** delves into RecAD's development, emphasizing its novel framework for algorithmic recourse in anomaly detection that recommends minimal-cost actions to mitigate detected anomalies.

- **Chapter 7 (AERCA: Root Cause Analysis of Anomalies in Multivariate Time Series through Granger Causal Discovery)** focuses on AERCA's methodology for identifying causal relationships and root causes in time series anomalies, enhancing the interpretability of anomaly detection systems.

- **Chapter 8 (RootCLAM: On Root Cause Localization and Anomaly Mitigation through Causal Inference)** details RootCLAM's approach to employing causal inference for effective root cause analysis and anomaly mitigation in tabular data, complementing the time series focus of AERCA and RecAD.

- **Chapter 9 (CFAD: Achieving counterfactual fairness for anomaly detection)** introduces the CFAD framework, addressing fairness in anomaly detection by ensuring counterfactual fairness through the generation of counterfactual samples and adversarial training.

- **Chapter 10 (Conclusions and Future Work)** discusses the limitations of current works and outlines potential future research directions, aiming to further the field of anomaly detection.

CHAPTER 2

LOGGPT: LOG ANOMALY DETECTION VIA GPT

In this chapter, we present LogGPT, an innovative framework that employs the Generative Pre-trained Transformer (GPT) for log anomaly detection. By modeling log sequences as natural language, LogGPT leverages deep learning to predict the next log entry based on the preceding sequence, enhancing anomaly detection accuracy. A unique reinforcement learning strategy is introduced to fine-tune the model, focusing on anomaly detection tasks. This approach significantly outperforms existing models, as evidenced by extensive experimental validation across multiple datasets, marking a significant advancement in log anomaly detection methodologies.

2.1 Introduction

Effectively detecting abnormal events in online computer systems is critical to maintaining the security and reliability of the systems. Logs, which are a fundamental component of modern computer systems, serve as a critical source of information for system monitoring, debugging, and security auditing as they record the system status, offering valuable insights into system performance and potential issues. Anomalies in log data often signify system faults, security breaches, or operational failures, making their detection a crucial task [33, 48–52].

However, the task of anomaly detection in log data is challenging due to the nature of high dimensionality, large volume, and complex structure. Machine learning models have been extensively employed for anomaly detection in log data. Traditional models, such as Principal Component Analysis (PCA) [53], Isolation forest [54], and one-class Support Vector Machines (OCSVM) [55] have been widely used. However, these models often require manual feature engineering or assume linear relationships among log entries, which makes them less effective in handling the dynamic nature of log data.

Recently, deep learning models have emerged for log anomaly detection, such as LSTM-based models like DeepLog [48], LogAnomaly [56], and OC4Seq [57], and BERT-based models like LogBERT [49]. One commonly used strategy is to borrow the idea of language modeling in the natural language processing field to capture the sequential pattern of log data. In this work, we call this group of log anomaly detection models **log language model**-based approaches. Particularly, the log language model is first trained to predict the next or masked log entries given the normal sequences. Then, the anomalies can be detected if the observed log entry is not in the top-K list predicted by the log language model. The rationale is that if a log sequence follows normal patterns, the log language model should be able to predict the next or masked log entries. Therefore, when an observed log entry is not in the top-K list predicted by the log language model, it means that the log entry has a low ratio to be in this specific position given the context, indicating the abnormality.

Although empirical studies have demonstrated the effectiveness of leveraging language models for log anomaly detection, the current models still face some limitations. The traditional LSTM-based log language models [58], such as DeepLog, often fail to fully capture long-term dependencies in log sequences. Therefore, the recently developed models usually adopt the Transformer structure [59] to model the long log sequences, such as LogBERT [49]. However, the masked log language model adopted in LogBERT may not be able to capture the natural flow in log sequences. More importantly, there is a gap between log language modeling and anomaly detection. Technically, the log language model is usually trained to correctly predict the next log entry, while the current log anomaly detection models label the anomalies if the observed log entry is not in the Top-K list predicted by the log language model. In other words, there is a gap in the objective between the training phase and the testing phase for log anomaly detection.

Inspired by the training strategy for large language models, to fill up the gap, we introduce LogGPT, a novel framework for log anomaly detection that leverages the Generative Pre-trained Transformer (GPT) model. LogGPT still harnesses the power of generative log language models to capture the intricate patterns and dependencies in log data. Specifically,

LogGPT is pre-trained to predict the next log entry given the preceding sequence (prompt). More importantly, we further fine-tune LogGPT via reinforcement learning. Specifically, LogGPT employs a novel reward mechanism based on whether the observed log entry is within the Top-K predicted log entries from the log language model. If the observed log entry is found within the Top-K predictions, LogGPT will receive a positive reward; otherwise, it will receive a negative reward. Reinforced by this reward signal, we expect that for the normal sequences, LogGPT can ensure the log entry is within the Top-K predictions.

The contributions of this work are threefold. First, we propose LogGPT, a novel framework for anomaly detection in log data, which utilizes the generative log language model to capture the patterns of normal log sequences by training to predict the next log key given the previous sequence. This novel approach effectively addresses the limitations of both traditional machine learning models and deep learning models like DeepLog [48] and LogBERT [49], providing a more robust and effective solution for log anomaly detection. Second, we introduce a Top-K reward metric specifically designed for fine-tuning the log language model for anomaly detection. This reward metric gives a positive reward if the actual log key is in the Top-K predictions, and a negative reward otherwise, thereby guiding the model to focus on the most relevant parts of the log sequence and enhancing the accuracy of anomaly detection. Third, we conduct extensive experiments to validate the effectiveness of LogGPT in detecting anomalies in log data. Experimental results demonstrate that LogGPT outperforms state-of-the-art methods, underscoring its potential as a powerful tool for anomaly detection in log data.

2.2 Related Work

Log anomaly detection, a critical task for ensuring system security and reliability, has received extensive research. The methods for log anomaly detection can be broadly categorized into two phases: traditional machine learning models and deep learning models.

In the early phase, traditional machine-learning models were the primary tools for log anomaly detection. Models such as Principal Component Analysis (PCA) [53], Isolation forest [54], and one-class Support Vector Machines (OCSVM) [55] were commonly used.

Although these models are capable of identifying outliers in the log data, these models have several limitations. First, the traditional machine learning models usually require manual feature engineering, which is labor-intensive and might not capture the complex patterns in log data. Furthermore, these models struggle with capturing complex patterns in log sequences.

The advanced deep learning models have significantly improved the performance of log anomaly detection. In particular, Long Short-Term Memory Networks (LSTMs), known for their ability to model sequential data, have proven to be effective for log anomaly detection, such as DeepLog [48] and LogAnomaly [56]. DeepLog functions by predicting the next log key based on the preceding sequence, identifying anomalies when the actual next log key significantly deviates from the prediction. On the other hand, LogAnomaly models a log stream as a natural language sequence and develops template2vec to extract the semantic information hidden in log templates. Therefore, LogAnomaly can detect both sequential and quantitative log anomalies simultaneously. However, these models come with their own set of limitations. A primary challenge with LSTM is that this type of recurrent architecture struggles to encode very long or complex sequences due to its relatively simple structure. This issue is particularly pronounced in log anomaly detection, where the sequences can be quite long and complex.

To address the limitations of LSTM-based models, researchers have turned to the use of Transformer [60], which is a more powerful model to capture the long-term dependencies in the sequences, such as LogBERT [49] or CAT [61]. LogBERT is a self-supervised framework that learns the patterns of normal log sequences based on BERT [60]. Specifically, LogBERT takes normal log sequences with random masks as inputs and is trained to predict the randomly masked log entries. After training, LogBERT can encode the patterns of normal log sequences. One limitation is that the masked log language model may not always capture the natural flow of log sequences in some contexts. Moreover, the performance of LogBERT is sensitive to the mask ratio, a hyperparameter controlling how many tokens will be replaced with MASK tokens during both the training and testing phases. In this work, we propose

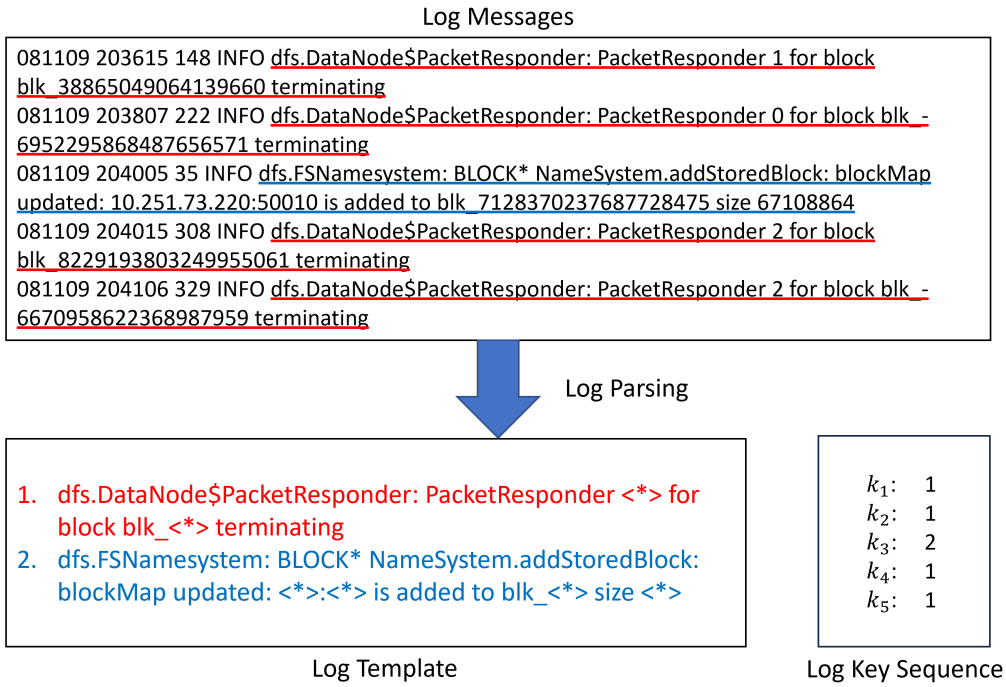


Fig. 2.1: Log key extraction from HDFS dataset messages via Log Parser. The message with a red/blue underscore indicates the detailed computational event for each log key separately.

LogGPT, which leverages the GPT model to learn patterns in normal log sequences by predicting the next log entries in a sequence, and further proposes a novel reinforcement learning mechanism to enhance the performance for anomaly detection.

2.3 Preliminary

In this section, we provide a detailed overview of two key components for log anomaly detection, log sequence preprocessing and log language model.

2.3.1 Log Sequence Preprocessing

The first step of log anomaly detection is to preprocess the log messages because it is hard to capture the sequential pattern from the raw text-based log messages. The major line of research in log anomaly detection is to first adopt a log parser, such as Drain [62], to extract the template from the log messages, as shown in Figure 2.1. Each template usually indicates one type of log message, called a log key.

After getting the log keys, the sequence of raw log messages can be transformed into a sequence of log keys. In this case, the log keys are similar to the vocabulary in natural language, while the sequence is like a sentence consisting of a sequence of log keys. Therefore, a language model can be leveraged to model the log sequences.

Formally, after preprocessing, the log messages with the same template are represented by a log key $k \in \mathcal{K}$, where $\mathcal{K}$ indicates the set of log keys extracted from the log messages. Then, a log sequence is organized as ordered log keys, denoted as $S = \{k_1, ..., k_t, ..., k_T\}$, where T indicates the length of the log sequence.

2.3.2 Log Language Model

We use DeepLog [48] to illustrate the concept of the log language model. DeepLog leverages Long Short-Term Memory networks (LSTMs) for log language modeling. The primary objective of DeepLog is to learn a probabilistic model of normal execution from log data and then detect anomalies as significant deviations from normal patterns.

DeepLog is trained on $\mathcal{D} = \{S^i\}_{i=1}^N$ consisting of normal log sequences. The LSTM network in DeepLog is trained to predict the next log key in a sequence based on the preceding sequence. Formally, given a sequence of log keys $S_{1:T} = \{k_1, ..., k_t, ..., k_T\}$, where k_t indicates the log key at the t-th position. DeepLog trains an LSTM to model the conditional probability $p(k_{t+m+1}|S_{t:t+m})$ for $t = 1, 2, ..., T - m - 1$, where m indicates the window size. Particularly, DeepLog adopts a sliding window with size m to split the sequences into a set of small windows and predict the next log key given the previous m log keys. The LSTM is trained to maximize the likelihood of the next log key given the preceding sequence, which can be formulated as the following objective function:

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T-m-1} \log p(k^i_{t+m+1}|S^i_{t:t+m}), \tag{2.1}$$

where θ denotes the parameters of LSTM.

During the anomaly detection phase, given a new sequence, DeepLog still splits the sequences into small windows and employs the trained LSTM model to predict the next log

key. The LSTM model predicts a probability distribution over all possible log keys in $\mathcal{K}$, ranking them based on their likelihood of being the next key in the sequence. Then, an abnormal sequence will be labeled as abnormal if the observed log key does not appear in the Top-K prediction list multiple times across all sliding windows in that sequence.

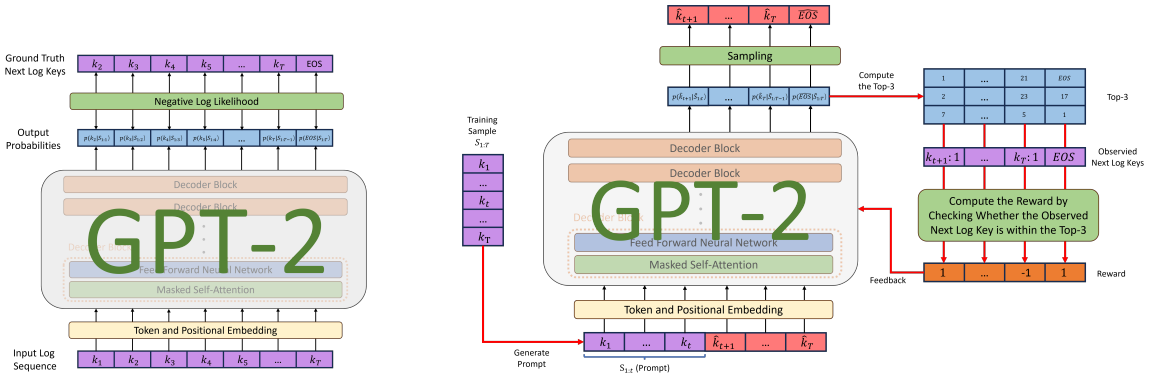
The concept of Top-K predictions is introduced to account for the inherent uncertainty and variability in log sequences. Even in normal operations, there can be multiple valid "next" log keys as the systems usually have multiple normal patterns. Therefore, during the anomaly detection phase, instead of predicting a single 'most likely' next log key, the model identifies the Top-K most probable next log keys. As long as the observed log key is in the Top-K list, we could consider the sequence normal.

The value of K, a tunable hyperparameter, determines the strictness of the model for anomaly detection. A smaller K results in a stricter model that allows fewer possibilities for the next log key, usually leading to high recall and low precision, while a larger K results in a more flexible model that considers a broader range of log keys as normal, usually resulting in high precision and low recall.

2.4 LogGPT

In this section, we introduce LogGPT, a novel log anomaly detection model based on GPT. Similar to DeepLog, LogGPT detects the log anomaly by examining whether the observed log key is in the Top-K prediction list. Because GPT is a more powerful structure compared to LSTM used by DeepLog, LogGPT does not need to further split the sequence into multiple small windows. Instead, LogGPT is trained to predict the next log key given the previous sequence, which intrinsically can capture the long-term dependence of log sequences. Moreover, besides leveraging the powerful GPT structure, we also propose a novel reinforcement learning strategy to further improve the performance of log anomaly detection.

The design of LogGPT is inspired by the training process of large language models, where the training process consists of two primary stages: pre-training and fine-tuning, as shown in Figure 2.2.



(a) Pre-training (b) Fine-tuning

Fig. 2.2: Framework of LogGPT.

In the pre-training stage (Figure 2.2a), a generative log language model $f_\theta(\cdot)$ is trained on a corpus of normal log sequences $\mathcal{D}$, which allows the model to learn the underlying patterns and structures of normal system behavior. After pre-training, LogGPT is capable of generating log sequences based on a given part of the log sequences.

The fine-tuning stage (Figure 2.2b) is designed to further refine the model's ability to distinguish between normal and abnormal log sequences. In this stage, we employ reinforcement learning techniques to finetune the pre-trained LogGPT. Borrowing the terminology from the large language model, we define a set of prompts $\mathcal{P} = \{S^i_{1:t}\}^N_{i=1}$, where $S^i_{1:t} \subseteq S^i_{1:T}$ and $S^i_{1:T} \in \mathcal{D}$. These prompts are fed into the LogGPT to generate the following sequence $\hat{S}^i_{t:T}$ step by step. We propose a novel reward, called the Top-K metric, to fine-tune LogGPT for anomaly detection.

2.4.1 Generative Log Language Model

LogGPT utilizes GPT-2 [63] for modeling the log sequences, which is based on Transformer decoder [59] that utilizes a self-attention mechanism to capture dependencies between log keys in the log sequence. LogGPT is trained to predict the next log key given the preceding log keys. The objective function for pretraining the LogGPT is defined as follows:

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T-1}\log p(k^i_{t+1}|S^i_{1:t}), \tag{2.2}$$

where θ denotes the parameters of LogGPT, N is the number of log sequences and T is the length of each sequence, $p(k^i_{t+1}|S^i_{1:t})$ indicates the probability of log key at the $t+1$-th position predicted by LogGPT given the sequence $S^i_{1:t}$.

Specifically, to derive $p(k^i_{t+1}|S^i_{1:t})$, the structure of LogGPT can be defined as:

$$\mathbf{h}^i_t = \mathsf{Transformer\_Decoder}(S^i_{1:t}) \tag{2.3a}$$

$$p(k^i_{t+1}|S^i_{1:t}) = \mathsf{Softmax}(\mathbf{h}^i_t\mathbf{W}), \tag{2.3b}$$

where $\mathbf{h}^i_t \in \mathbb{R}^d$ indicates the hidden representation derived from the Transformer decoder [59,63], and $\mathbf{W} \in \mathbb{R}^{d\times|\mathcal{K}|}$ is the parameter of the language model head that maps the hidden representation to a probability distribution of all log keys in $\mathcal{K}$.

By training the model to predict the next log key in normal log sequences, LogGPT encodes the normal system behavior. After pre-training, GPT-2 is capable of generating a log sequence $\hat{S}^i_{t+1:T} = \{\hat{k}^i_{t+1}, ..., \hat{k}^i_T\}$ based on a given part of the log sequence $S^i_{1:t}$. This capability is crucial for the subsequent fine-tuning stage, where the model is further refined to distinguish between normal and anomalous log sequences.

2.4.2 Reinforcement Learning for Log Anomaly Detection

In the context of LogGPT, we employ reinforcement learning to fine-tune the pre-trained GPT-2 model for the task of log anomaly detection. The reinforcement learning paradigm is particularly suitable for our task as it allows the model to learn from its predictions and adjust its behavior based on the feedback received, thereby enhancing its ability to detect anomalies. In the context of our framework, we define the following elements.

State: The state, denoted as $\tilde{S}^i_{1:t} = S^i_{1:t}$, is initially defined as the given part of a log sequence. As the model generates the log sequence $\hat{S}^i_{t+1:T}$ based on the given part, the state evolves dynamically. Specifically, for each step j where $t+1 \le j \le T-1$, the state $\tilde{S}^i_{1:j}$ becomes the concatenation of the given part of the log sequence $S^i_{1:t}$ and the generated part of the log sequence $\hat{S}^i_{t+1:j}$, denoted as $\tilde{S}^i_{1:j} = \{S^i_{1:t}, \hat{S}^i_{t+1:j}\}$. The sequence $\tilde{S}^i_{1:j}$ is further

transformed to a hidden representation $\tilde{\mathbf{h}}_j^i$ by the Transformer decoder shown in Equation 2.3a.

Action: An action is defined as sampling a log key from the K log keys with the highest probabilities predicted by LogGPT, denoted as $a_{j+1}^i \sim \text{Top-K}(p(\hat{k}_{j+1}^i | \tilde{S}_{1:j}^i))$.

Policy: A policy takes the form of LogGPT and is defined by its parameters. Specifically, given the current part of the sequence until the j-th position, the policy outputs a probability distribution over the action space, represented as $\pi_\theta(a_{j+1}^i | \tilde{\mathbf{h}}_j^i)$, where θ indicates the parameters of LogGPT.

Reward: The reward function provides feedback to the policy based on the quality of its actions. We propose a novel reward function to evaluate the predicted log key for anomaly detection, called the Top-K metric.

At each step, the Top-K metric checks whether the observed next log key is within the Top-K predicted log keys. If this is the case, the model receives a reward of 1; otherwise, it receives a reward of -1. Given a part of log sequence $S_{1:t}^i$, after an action is taken, the reward function is formulated as:

$$r_{j+1} = \begin{cases} 1, & \text{if } k_{j+1}^i \in \text{Top-K}(p(\hat{k}_{j+1}^i | \tilde{S}_{1:j}^i)) \\ -1, & \text{if } k_{j+1}^i \notin \text{Top-K}(p(\hat{k}_{j+1}^i | \tilde{S}_{1:j}^i)) \end{cases}. \tag{2.4}$$

Here, k_{j+1}^i refers to the actual next log key, and $p(\hat{k}_{j+1}^i | \tilde{S}_{1:j}^i)$ denotes the probability distribution predicted by LogGPT over the action space given the current state.

The Top-K metric promotes better generalization and robustness of LogGPT in anomaly detection. By encouraging the model to predict a set of likely next log keys rather than a single most likely log key, the Top-K metric helps LogGPT learn a more nuanced representation of the normal log patterns. This approach recognizes that log data may contain inherent variability even for the normal log sequences, and a broader range of acceptable candidates can still reflect normal system behavior. The Top-K metric, therefore, enhances the precision of anomaly detection by aligning the model's predictions with the complex nature of log data.

2.4.3 Policy Update

We adopt Proximal Policy Optimization (PPO) [64] for the policy update. PPO is a type of policy gradient method that optimizes the policy directly by maximizing the expected reward and can further maintain the stability of the learning process and prevent harmful updates. The objective function of PPO is defined as follows:

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{i=1}^{N}\sum_{j=t}^{T-1}\frac{\pi_\theta(a_{j+1}^i|\mathbf{h}_j^i)}{\pi_{\theta_{\text{old}}}(a_{j+1}^i|\mathbf{h}_j^i)}r_{j+1}\right], \tag{2.5}$$

where π_θ is the new policy, $\pi_{\theta_{\text{old}}}$ is the old policy, and r_{j+1} is the reward for an action.

The policy π_θ is updated by performing gradient ascent on the objective function $J(\theta)$:

$$\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta), \tag{2.6}$$

where α is the learning rate.

The policy update process is repeated for a number of iterations until the policy converges or a maximum number of iterations is reached. The Top-K metric encourages the model to recognize the inherent variability in normal log data by rewarding predictions that include the actual next log key within a broader set.

2.4.4 Anomaly Detection

After fine-tuning, LogGPT is deployed to detect abnormal log sequences. Given a new log sequence $S_{1:T}$, LogGPT iteratively predicts the next log key k_{t+1} given the preceding subsequence $S_{1:t}$ for $1 \leq t \leq T-1$.

For each predicted log key, the model generates a set of Top-K predicted log keys. This set represents the K most likely log keys at the current position. The actual next log key is then compared to this set. As long as one actual log key is not in the set of Top-K predicted log keys, the whole log sequence will be flagged as anomalous.

2.5 Experiments

Table 2.1: Statistics of the Datasets. The number in the parentheses indicates the unique log keys in the training set.

Dataset	# of Unique Log Keys	# of Log Sequences	Avg. Seq. Length	Training Data	Testing Data	
					Normal	Anomalous
HDFS	48 (15)	575,061	19	5,000	553,223	16,838
BGL	396 (160)	36,927	58	5,000	28,631	3,296
Thunderbird	7,703 (904)	112,959	166	5,000	67,039	40,920

2.5.1 Experimental Setup

Datasets. We evaluate LogGPT on three log datasets, namely HDFS, BGL, and Thunderbird. Table 2.1 shows the statistics of three datasets. For all the datasets, we randomly select 5000 normal log sequences as the training dataset.

- HDFS (Hadoop Distributed File System) [53]: This dataset is derived from Hadoop-based map-reduce jobs that were run on Amazon EC2 nodes. The anomalies within this dataset are identified through a manual labeling process based on a set of predefined rules. The log sequences are constructed based on the session ID present in each log message, resulting in an average sequence length of 19. The HDFS dataset consists of 575,061 log sequences, out of which 16,838 have been labeled as anomalous.

- BGL (BlueGene/L Supercomputer System) [65]: The BGL dataset originates from a BlueGene/L supercomputer system, located at the Lawrence Livermore National Labs (LLNL). It includes both alert and non-alert messages, with the alert messages being treated as anomalies. Log sequences are formed using a time sliding window of 1 minute, yielding an average sequence length of 58. The BGL dataset contains 36,927 log sequences, with 3,296 of them classified as anomalous.

- Thunderbird [65]: This dataset is collected from another supercomputer system. The dataset used in this study comprises the first 20,000,000 log messages from the original Thunderbird dataset that compose 112,959 log sequences, with 40,920 of them marked as anomalous. Log sequences are created using a time sliding window of 1 minute, leading to an average sequence length of 166.

Baselines. We compare LogGPT with a variety of baseline methods, consisting of both traditional machine learning models and deep learning models:

- PCA (Principal Component Analysis) [66]: This technique constructs a counting matrix based on the frequency of log key sequences. It then reduces this matrix into a lower-dimensional space to identify anomalies.

- iForest (Isolation Forest) [54]: iForest is an unsupervised learning algorithm, which also adopts a counting matrix as input. It isolates anomalies instead of profiling normal data points. It represents features as tree structures and anomalies are detected as instances with short average path lengths on the constructed isolation trees.

- OCSVM (One-Class Support Vector Machine) [67]: OCSVM is a variant of the Support Vector Machine algorithm that is designed for anomaly detection tasks [55, 68]. The model is trained on normal data and finds the maximum margin hyperplane that separates the normal data from the origin.

- LogCluster [69]: LogCluster is a density-based log clustering approach that groups similar log messages together. Anomalies are detected as log messages that do not belong to any cluster or belong to small clusters.

- DeepLog [48]: DeepLog is a deep learning-based approach for anomaly detection in log data. It uses a long short-term memory (LSTM) network to model the log sequences and detect anomalies based on the prediction errors.

- LogAnomaly [56]: LogAnomaly models a log stream as a natural language sequence, which can detect both sequential and quantitative log anomalies simultaneously.

- OC4Seq (Multi-Scale One-Class Recurrent Neural Networks) [57]: OC4Seq is designed to detect anomalies in discrete event sequences. Recognizing that an anomalous sequence could be caused by individual events, subsequences of events, or the entire sequence, OC4Seq employs a multi-scale RNN framework to capture different levels of sequential patterns simultaneously.

Table 2.2: Experimental Results on HDFS, BGL, and Thunderbird Datasets.

Method	HDFS			BGL			Thunderbird		
	Precision	Recall	F-1 score	Precision	Recall	F-1 score	Precision	Recall	F-1 score
PCA	$0.166_{\pm 0.008}$	$0.059_{\pm 0.003}$	$0.087_{\pm 0.002}$	$0.117_{\pm 0.023}$	$0.035_{\pm 0.007}$	$0.054_{\pm 0.010}$	$0.953_{\pm 0.004}$	$0.980_{\pm 0.005}$	$0.966_{\pm 0.003}$
iForest	$0.043_{\pm 0.010}$	$0.422_{\pm 0.224}$	$0.078_{\pm 0.021}$	$0.491_{\pm 0.364}$	$0.037_{\pm 0.052}$	$0.063_{\pm 0.090}$	$0.338_{\pm 0.128}$	$0.015_{\pm 0.011}$	$0.028_{\pm 0.020}$
OCSVM	$0.058_{\pm 0.012}$	$0.910_{\pm 0.089}$	$0.108_{\pm 0.021}$	$0.073_{\pm 0.003}$	$0.345_{\pm 0.010}$	$0.121_{\pm 0.004}$	$0.550_{\pm 0.004}$	$0.998_{\pm 0.000}$	$0.709_{\pm 0.003}$
LogCluster	$\mathbf{0.996_{\pm 0.003}}$	$0.368_{\pm 0.001}$	$0.538_{\pm 0.001}$	$\mathbf{0.941_{\pm 0.015}}$	$0.641_{\pm 0.033}$	$0.762_{\pm 0.021}$	$\mathbf{0.977_{\pm 0.005}}$	$0.291_{\pm 0.063}$	$0.445_{\pm 0.067}$
DeepLog	$0.793_{\pm 0.092}$	$0.863_{\pm 0.031}$	$0.824_{\pm 0.060}$	$0.792_{\pm 0.048}$	$0.946_{\pm 0.012}$	$0.861_{\pm 0.028}$	$0.864_{\pm 0.005}$	$0.997_{\pm 0.000}$	$0.926_{\pm 0.003}$
LogAnomaly	$0.907_{\pm 0.017}$	$0.369_{\pm 0.014}$	$0.524_{\pm 0.017}$	$0.884_{\pm 0.002}$	$0.850_{\pm 0.009}$	$0.867_{\pm 0.003}$	$0.873_{\pm 0.005}$	$0.996_{\pm 0.000}$	$0.931_{\pm 0.003}$
OC4Seq	$0.922_{\pm 0.059}$	$0.758_{\pm 0.227}$	$0.808_{\pm 0.157}$	$0.441_{\pm 0.045}$	$0.352_{\pm 0.044}$	$0.391_{\pm 0.041}$	$0.901_{\pm 0.046}$	$0.823_{\pm 0.232}$	$0.845_{\pm 0.177}$
LogBERT	$0.754_{\pm 0.142}$	$0.749_{\pm 0.037}$	$0.745_{\pm 0.082}$	$0.917_{\pm 0.006}$	$0.892_{\pm 0.006}$	$0.905_{\pm 0.005}$	$0.962_{\pm 0.019}$	$0.965_{\pm 0.008}$	$0.963_{\pm 0.007}$
CAT	$0.102_{\pm 0.022}$	$0.422_{\pm 0.082}$	$0.164_{\pm 0.034}$	$0.177_{\pm 0.122}$	$0.210_{\pm 0.184}$	$0.190_{\pm 0.148}$	$0.751_{\pm 0.072}$	$0.516_{\pm 0.124}$	$0.607_{\pm 0.120}$
LogGPT	$0.884_{\pm 0.030}$	$\mathbf{0.921_{\pm 0.066}}$	$\mathbf{0.901^{*}_{\pm 0.036}}$	$0.940_{\pm 0.010}$	$\mathbf{0.977_{\pm 0.018}}$	$\mathbf{0.958^{*}_{\pm 0.011}}$	$0.973_{\pm 0.004}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{0.986^{*}_{\pm 0.002}}$

The asterisk indicates that LogGPT significantly outperforms the best baseline at the 0.05 level, according to the paired t-test.

Table 2.3: Performance of LogGPT with or without reinforcement learning.

Metric	Approach	HDFS	BGL	Thunderbird
Precision	LogGPT w/o RL	$0.932_{\pm 0.015}$	$0.936_{\pm 0.011}$	$0.971_{\pm 0.004}$
	LogGPT	$0.884_{\pm 0.030}$	$0.940_{\pm 0.010}$	$0.973_{\pm 0.004}$
Recall	LogGPT w/o RL	$0.790_{\pm 0.101}$	$0.975_{\pm 0.018}$	$1.000_{\pm 0.000}$
	LogGPT	$0.921_{\pm 0.066}$	$0.977_{\pm 0.018}$	$1.000_{\pm 0.000}$
F-1 score	LogGPT w/o RL	$0.853_{\pm 0.065}$	$0.955_{\pm 0.010}$	$0.985_{\pm 0.002}$
	LogGPT	$0.901^{*}_{\pm 0.036}$	$0.958_{\pm 0.011}$	$0.986^{*}_{\pm 0.002}$

Significantly outperforms LogGPT w/o RL at the 0.05 level (paired t-test).

- LogBERT [49]: LogBERT is a BERT-based architecture to capture the patterns of normal log sequences via a log language model. LogBERT is trained to predict the masked log keys on normal log sequences and detects the abnormal log sequences based on the prediction errors.

- CAT (Content-Aware Transformer) [61]: CAT is a self-attentive encoder-decoder transformer framework designed for anomaly detection in event sequences. It incorporates the semantic information of event content by using a content-awareness layer to generate representations of each event. The encoder learns preamble event sequence representations with content awareness, and the decoder embeds sequences under detection into a latent space where anomalies are distinguishable.

Implementation Details. We first employ Drain [62] to parse raw log messages into log keys. For the baseline models, we utilize the Loglizer [70] package to evaluate PCA, OCSVM, iForest, and LogCluster for anomaly detection. DeepLog and LogAnomaly are

evaluated using the Deep-loglizer [71] package. For OC4Seq[1], LogBERT[2], and CAT[3], we use the open-source code provided by the authors separately.

As for LogGPT, we use a GPT model with 6 layers and 6 heads. The dimensions of the embeddings and hidden states are set to 60. The learning rate is set to 1e-4 for the pre-training phase and 1e-6 for the fine-tuning phase. To accommodate different datasets, we set the K in Top-K to 50% of the training log keys. It means during the test phase if an observed log key is not in the top 50% of the prediction list from the GPT, the sequence will be labeled as an anomaly. This allows us to maintain a high level of flexibility when dealing with datasets of varying sizes and characteristics. The batch size for the pre-training phase is set to 16, and we train the model for 100 epochs. The episode is set to 20 with early stop criteria to prevent overfitting and ensure efficient training. The code for LogGPT is publicly available[4].

2.5.2 Experimental Results

Performance on Log Anomaly Detection. Table 2.2 illustrates the results and standard deviation of LogGPT and various baselines over 10 runs on the HDFS, BGL, and Thunderbird datasets. The asterisk in the table indicates that LogGPT significantly outperforms the best baseline for each dataset at the 0.05 level, according to the paired t-test.

First, we can observe that PCA, iForest, and OCSVM perform poorly on the HDFS and BGL datasets, as indicated by their low F-1 scores. However, PCA's performance is notably better on the Thunderbird dataset, achieving a high F-1 score. This inconsistency in performance across datasets highlights the sensitivity of PCA to datasets.

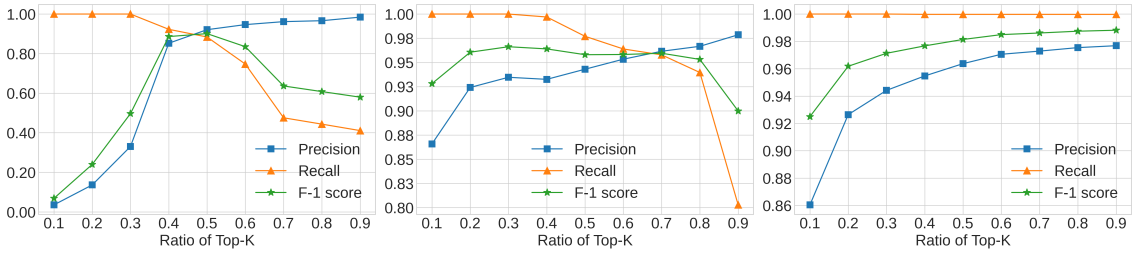
LogCluster, specifically designed for log anomaly detection, shows improved performance over other traditional machine learning models, i.e., PCA, iForest, and OCSVM, on the HDFS and BGL datasets but is outperformed by PCA on the Thunderbird dataset. This

[1] https://github.com/KnowledgeDiscovery/OC4Seq
[2] https://github.com/HelenGuohx/logbert
[3] https://github.com/mmichaelzhang/CAT
[4] https://github.com/nokia/LogGPT

pattern further emphasizes the importance of dataset-specific characteristics in determining the effectiveness of different methods.

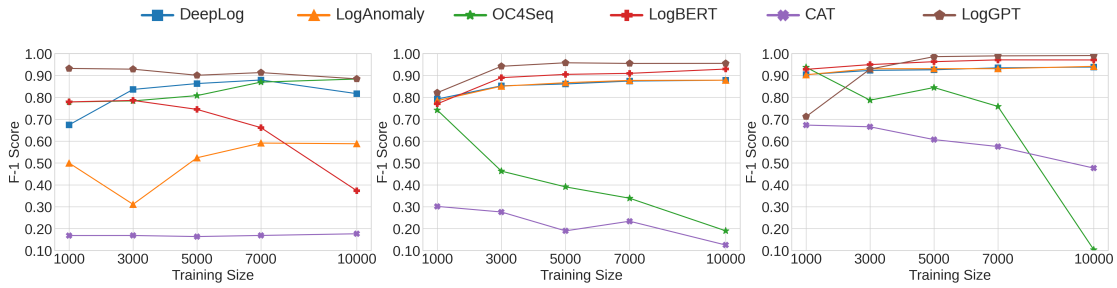
Deep learning-based approaches, such as DeepLog, LogAnomaly, OC4seq, LogBERT, and CAT, outperform traditional methods across all three datasets, which shows the advantages of utilizing deep learning to capture complex patterns in log sequences.

Our proposed model, LogGPT, stands out by consistently achieving the highest F-1 scores across all three datasets, with significant margins over all baselines.



(a) HDFS (b) BGL (c) Thunderbird

Fig. 2.3: Impact of the ratio of Top-K log keys.



(a) HDFS (b) BGL (c) Thunderbird

Fig. 2.4: Impact of the training size.

Ablation Studies. To investigate the contribution of reinforcement learning (RL) to the performance of LogGPT, we conducted an ablation study, comparing the performance of LogGPT with and without the RL component. The results are summarized in Table 2.3.

First, we can notice that on both HDFS and Thunderbird datasets, LogGPT significantly outperforms LogGPT without the RL component, which demonstrates that the RL component enhances the overall performance of LogGPT for log anomaly detection. Especially, on the HDFS dataset, by finetuning the GPT model with RL reward, the recall achieved by LogGPT is improved with a large margin with a little sacrifice on precision, leading to extensive improvement in the F-1 score. It also shows that fine-tuning the log language model with Top-K reward can identify more log anomalies. Meanwhile, on the BGL dataset, we can also notice a slight improvement in F-1 of LogGPT compared to the one without the RL component. Another interesting finding is that even the LogGPT without the RL component already outperforms all baselines (shown in Table 2.2) in three datasets, which also shows the advantage of leveraging the GPT model to capture the patterns of log sequences.

Parameter Analysis: Ratio of Top-K. LogGPT detects the anomalies by examining whether the observed log key is in the Top-K list predicted by GPT. Therefore, K is an important parameter to determine the anomalies. We first analyze the difference in the performance by tuning K for anomaly detection. By default, K is set as 50% of unique log keys. It means if the next log key falls into the top 50% of unique log keys predicted by GPT, the sequence is normal.

The impact of different top-K ratios on the precision, recall, and F-1 score for the HDFS, BGL, and Thunderbird datasets is illustrated in Figure 2.3. On both HDFS and BGL datasets, we have similar observations. With the increasing of ratios as normal log keys, the recall keeps decreasing when the ratio is greater than a threshold, such as 40% in HDFS and BGL. This happens because when we have a large ratio, most of the keys are considered normal. In this case, the recall will be low. On the other hand, if the observed log key is predicted with an extremely low probability at a specific position, with a high chance, this log key is abnormal. Therefore, we can observe the increase in precision along with the increase in ratios.

For the Thunderbird dataset, the precision increases as the top-K ratio increases, while

the recall remains almost constant, with a slight decrease at higher top-K ratios. The F-1 score increases steadily, reaching a peak at a specific top-K ratio. The reason for this behavior can be attributed to the inherent characteristics of the Thunderbird dataset. It is likely that the normal data within the Thunderbird dataset has high variability, which needs a broader range of acceptable continuations in the log sequences to reduce the false positive. As the top-K ratio increases, LogGPT becomes more selective in flagging anomalies, thereby increasing precision by reducing false positives.

Overall, a low top-K ratio tends to lead to high recall but low precision, while a high top-K ratio leads to high precision but potentially lower recall. The optimal top-K ratio varies across datasets, reflecting the unique characteristics of each dataset.

Scalability Analysis: Training Size. It is well known that deep learning models usually require a sufficient number of training samples. The impact of training size on the performance of log anomaly detection models is critical. By analyzing the F-1 scores of various models across different training sizes, we can gain insights into their effectiveness and efficiency. In this experiment, we compare LogGPT with other deep learning-based baselines, across three datasets by varying the training size. Figure 2.4 shows the experimental results.

The effect of the training size on the HDFS dataset reveals distinct patterns across different models (shown in Figure 2.4a). LogGPT demonstrates consistent performance across various training sizes, highlighting its robustness and ability to generalize well. OC4Seq shows a consistent increase in performance with the training size, indicating that it benefits from more extensive training data. DeepLog and LogAnomaly exhibit fluctuations in performance, which may be attributed to the sensitivity to training size. The decline in performance for LogBERT and stability for CAT may reflect limitations in their ability to leverage additional training data without changing other hyper-parameters. The varying behaviors of these models underscore the importance of carefully selecting the training size based on the model's characteristics.

We have similar observations on BGL and Thunderbird datasets. First, with larger training sizes, the performance of LogGPT, DeepLog, LogAnomaly, and LogBERT keep

improving, which shows that these models can benefit from additional training data. Meanwhile, LogGPT can outperform those baselines in most cases. However, the sharp decline for OC4Seq and overall downward trend for CAT may indicate overfitting or challenges in generalizing from larger training sets.

Overall, LogGPT can achieve very good performance in three datasets. More training samples can further boost the performance of LogGPT.

2.6 Summary

In this chapter, we introduced LogGPT, a novel approach to log anomaly detection that builds upon GPT models, further enhanced by a reinforcement learning strategy. Through modeling log sequences as natural language, LogGPT innovatively adapts GPT for log anomaly detection. More importantly, recognizing the existing gap between language modeling and anomaly detection, LogGPT integrates a fine-tuning process guided by a novel Top-K reward metric for anomaly detection. Extensive experiments conducted across various datasets demonstrated the effectiveness of LogGPT, showcasing significant improvements over existing state-of-the-art methods. The early version of this work is published at BigData 2023 [72].

CHAPTER 3

UNSUPERVISED CROSS-SYSTEM LOG ANOMALY DETECTION VIA DOMAIN ADAPTATION

In this chapter, we introduce a transferable log anomaly detection approach, called LogTAD, to improve performance and address the cold-start problem of anomaly detection in online systems. Existing transfer learning-based approaches for cross-system log anomaly detection require labeled anomalous data from both source and target systems to build a classifier, which is often infeasible to collect effectively. In contrast, LogTAD only uses the normal samples from both systems and does not require any labeled anomalous data from the target system. The core idea of LogTAD is to derive a system invariant center representation based on the normal data in both source and target systems using the domain adversarial technique. By mapping both source and target log sequences into the same hypersphere with a similar distribution, LogTAD derives a shared center for both systems. Anomalous sequences from both systems can then be detected by having large distances to the center. LogTAD addresses the cold-start problem by requiring only a small number of normal samples from the target system. Experimental results on two log datasets show that LogTAD can achieve good performance on both source and target systems.

3.1 Introduction

Online services, e.g. cloud computing, web platforms, and remote databases, have become essential in our daily life. How to maintain reliability and stability is a major challenge of operating successful online services, since a small glitch would cause a crash of the whole system that affects users' experience or even leads to financial loss. Therefore, effectively detecting anomalous status is critical to building reliable online systems.

In practice, system logs, which are extensively adopted for recording states of online systems, take a critical part in detecting the anomalous status of online systems. In recent

years, many machine learning-based log anomaly detection approaches are proposed to detect anomalous log sequences [48, 53, 56, 65, 70, 73–78]. Especially, due to limited anomalies, many unsupervised machine learning approaches are proposed to identify the anomalous log sequences, such as Principal Component Analysis (PCA) [53], DeepLog [48], and LogBERT [49]. The key idea of these models is to capture normal patterns from a large number of normal logs. Then, the models can detect anomalous log sequences based on the obtained normal patterns. However, online systems are deployed constantly. Given a newly deployed system, it takes some time to collect enough logs to train an unsupervised anomaly detection model. On the other hand, a newly deployed system usually has an urgent requirement to automatically identify the abnormal status.

To tackle this cold-start problem, several transfer learning-based approaches are proposed to detect anomalies in different systems, such as LogTransfer [79]. The basic assumption of these approaches is that the anomalous log sequences in different online systems usually share similar patterns. For example, the words in log messages from different systems usually have some overlaps. Meanwhile, the normal workflows of different systems also have some similarities. The idea is to build a cross-system classifier to detect anomalies in both source and target systems. The major disadvantage of existing models is that they usually require both normal and abnormal log data from both source and target systems to build the classifier. However, in practice, it is often infeasible to collect sufficient anomalous data from the target system effectively.

To address the limitation of current transfer learning approaches for cross-system log anomaly detection, in this work, we propose a transferable log anomaly detection approach, called LogTAD, which does not require the labeled anomalous data from both source and target systems. Inspired by the DeepSVDD approach [80], which makes the normal data close to a center of a hypersphere, the core idea of our approach is to derive a system invariant center representation based on the normal data in both source and target systems. Specifically, we leverage the domain adversarial technique to map both source and target log sequences into the same hypersphere with a similar distribution and derive a shared center

for both systems. Then, the anomalous sequences from both systems can be detected by having large distances to the center.

The main contributions of this work are as follows. First, we develop a cross-system anomaly detection model that only using the normal samples. Second, the proposed model only requires a small number of samples from the target system so that it can address the cold-start problem of anomaly detection. Third, the experimental results on two log datasets show that LogTAD can achieve good performance on both source and target systems.

3.2 LogTAD

In this work, we assume the existence of one source system S and one target system T. A log sequence generated by a system is denoted as $\mathcal{X} = \{x_n\}_{n=1}^{N}$, where x_n indicates the n-th log message. Given a dataset $\mathcal{D}$ consisting of normal log sequences from the source system $\mathcal{D}^S = \{\mathcal{X}_i^S\}_{i=1}^{M_S}$ and a small number of normal log sequences from the target system $\mathcal{D}^T = \{\mathcal{X}_i^T\}_{i=1}^{M_T}$ ($M_T << M_S$), we aim to build an unsupervised and transferable log anomaly detection model (LogTAD), which is able to detect the anomalous log sequences from both source and target systems. Our framework consists of two parts. One is an encoder to map log sequences from both systems into a low dimensional space, while another part is to map the representations of log sequences from both systems into one hypersphere with a shared center. Specifically, we leverage the idea of DeepSVDD [80] to map normal sequences close to a center of a hypersphere. Meanwhile, a domain adversarial framework is proposed to ensure the representations of sequences from both systems are close to one center. Then, we can detect the anomalies in both systems that have large distances to the center. Figure 3.1 shows the framework of LogTAD.

3.2.1 Log Sequence Representation

Log messages generated by the online systems are descriptive texts. Many existing log anomaly approaches adopt tools to generate templates from log messages and transfer raw log sequences into sequences of templates. However, under the setting of cross-system anomaly detection, the templates from source and target systems can be totally different. Then,

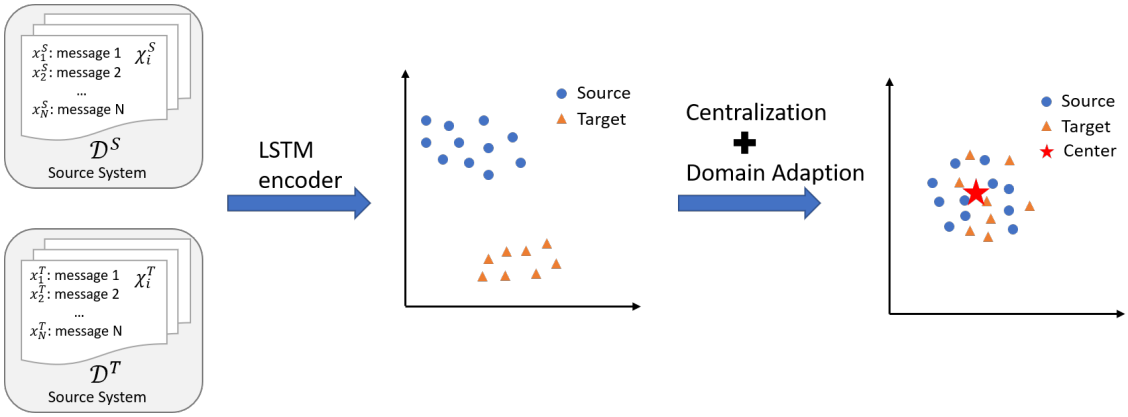


Fig. 3.1: Framework of LogTAD

it is infeasible to use the knowledge from the source system to improve the performance of anomaly detection in the target system. Hence, in this work, we propose to use the information from the raw messages. The intuition is that the descriptive words from different systems should share some similarities, i.e., there are some overlaps in terms of words from source and target systems. Then, we are able to transfer the knowledge from the source system to the target system. Specifically, we first use word2vec to represent words in each log message as embedding vectors and then adopt the mean operation over the word embeddings in a log message to derive the representation of one message. Formally, we now represent a log sequence as $\mathcal{X} = \{\mathbf{x}_n\}_{n=1}^{N}$ where $\mathbf{x}_n \in \mathbb{R}^d$ is the feature vector of n-th message.

Then, an encoder encodes the log messages in a sequence to a sequence representation with an LSTM model:

$$\mathbf{h}_n = LSTM(\mathbf{x}_n, \mathbf{h}_{n-1}), \tag{3.1}$$

where $\mathbf{x}_n$ is the feature vector of the n-th message; $\mathbf{h}_n$ indicates the n-th hidden vector of LSTM. The last hidden vector $\mathbf{h}_N$ captures the information of an entire sequence and is considered as the log sequence representation $\mathbf{v} = \mathbf{h}_N$. Here, we use a shared LSTM model to get the sequence representations from both source and target systems. Specifically, we denote the representation of the i-th sequence from the source system as $\mathbf{v}_i^S$, while the representation of i-th sequence from the target system as $\mathbf{v}_i^T$. One advantage of using one shared LSTM model is that using one LSTM to model the sequences from both systems

can reduce the requirement on the samples from the target system that has limited samples available.

3.2.2 Log Sequence Centralization

Inspired by the DeepSVDD that the normal log sequences should be in a hypersphere and close to the center in the embedding space, we first derive the center of all the log sequences from both source and target systems as the mean over all log sequences in both datasets, i.e., $\mathbf{c} = Mean(\mathbf{v}_i^\epsilon)$, where $\epsilon \in \{S, T\}$ indicates the source or target dataset. To make the representation of normal log sequences close to the center $\mathbf{c}$, we develop the objective function as

$$\mathcal{L}_{en} = \sum_{\epsilon \in \{S,T\}} \sum_{i=1}^{M_\epsilon} \|\mathbf{v}_i^\epsilon - \mathbf{c}\|^2. \tag{3.2}$$

By minimizing the Equation 3.2, we expect all normal log sequences are close to the center.

3.2.3 System-agnostic Representation via Domain Adversarial Training

Although we adopt one shared LSTM model to map log sequences into a hypersphere, the representations of log sequences from different systems can be still located in different regions of the hypersphere instead of clustering around the center $\mathbf{c}$. Hence, we propose an adversarial training method for cross-system (domain adaptation) data mapping, which encourages the representations of log sequences from different systems close to each other.

We formulate the adversarial training as follows. A discriminator D is used to distinguish whether the representations of log sequences are from the source or target system. We adopt the logistic function to make the prediction, i.e., $D(\mathbf{v}^\epsilon) = \sigma(\mathbf{w}^T\mathbf{v}^\epsilon + b)$, where $\sigma(\cdot)$ indicates the logistic function, $\mathbf{w}$ and b are the trainable parameters. The shared LSTM as a generator, i.e., $\mathbf{v}^\epsilon = G(\mathcal{X}^\epsilon)$, is trained to make representations of log sequences from different systems have similar distributions so that the discriminator is unable to distinguish the source samples from the target samples. Specifically, the objective function is

$$\mathcal{L}_{adv} = \min_{G} \max_{D} \big(\mathbb{E}_{\mathcal{X}^S \sim P_{\text{source}}} [\log D(G(\mathcal{X}^S))] + \mathbb{E}_{\mathcal{X}^T \sim P_{\text{target}}} [\log(1 - D(G(\mathcal{X}^T)))] \big), \tag{3.3}$$

where $\mathcal{X}^S$ and $\mathcal{X}^T$ indicate the source and target log sequences. By training through the minmax optimization, the representations of log sequences generated by the shared LSTM model are able to mislead the discriminator, which means the distributions of source and target log sequences are mixed. Finally, we train our framework by the following objective function:

$$\mathcal{L} = \mathcal{L}_{en} + \lambda \mathcal{L}_{adv}, \tag{3.4}$$

where λ is a hyperparameter to balance the two parts. After the training phase, the encoder gains the ability to embed both source and target samples close to the center $\mathbf{c}$ so that the abnormal samples from both systems should have large distances to the center.

Log Anomaly detection. To detect the anomalies, we need to set a radius γ as a decision boundary that separates normal and anomalous sequences. The samples with distances to the center greater than γ will be labeled as anomalous sequences. We adopt a validation set in the source and target systems respectively to find the best radius γ^S and γ^T. In our experiments, we randomly select one-day log sequences from each system as the validation set.

3.3 Experiments

3.3.1 Experimental Setup

Table 3.1: Statistics of the Datasets

Dataset	# of Logs	# of Log Sequences	
		Normal	Anomalous
BGL	1,212,150	265,583	37,450
TB	3,737,209	565,817	368,481

Datasets. We adopt the following two datasets: (1) BlueGene/L supercomputer system (**BGL**) dataset [65], which is collected from a BlueGene/L supercomputer system at Lawrence Livermore National Labs; (2) Thunderbird (**TB**) dataset [65], which is collected from a Thunderbird supercomputer system at Sandia National Labs. The statistics of the two datasets are shown in Table 3.1. Table 3.2 further shows the numbers of shared words across different datasets. The normal log sequences in the BGL and Thunderbird datasets have 664 and 1753 unique words, respectively, while the number of shared words is 254. The anomalous log sequences in the BGL and Thunderbird datasets have 195 and 54 unique words, respectively, while the number of shared words is 16.

Table 3.2: Statistics of Shared Words

	BGL Normal	BGL Anomalous	TB Normal	TB Anomalous
BGL Normal	664	133	254	25
BGL Anomalous	133	195	99	16
TB Normal	254	99	1753	49
TB Anomalous	25	16	49	54

Baselines. We compare our framework with the following anomaly detection methods to evaluate the performance of LogTAD: (1) Principal Component Analysis (*PCA*), which is a dimensional reduction based anomaly detection approach; (2) *LogCluster* [74], which is a clustering-based log anomaly detection approach; (3) *DeepLog* [48], which is a deep learning based log anomaly detection method; (4) Deep Support Vector Data Description (*DeepSVDD*) [80], which is a deep learning based one-class classification model; (5) *LogTransfer* [79], which can achieve cross-system anomaly detection but requires labeled data from both systems to train a classifier.

Implementation Details. We adopt a sliding window with size 20 to cut log files into short sequences and leverage the Drain package [62] to extract templates from raw log messages. The LSTM encoder consists of two hidden layers with 128 dimensions. For the domain discriminator, we use a two-layer fully connected neural network with 64 hidden dimensions. We use 100,000 normal sequences from the source system and 1,000 normal

Table 3.3: Anomaly Detection on Source and Target Systems

	Method	Source		Target	
		F1	AUC	F1	AUC
BGL → TB	PCA w/ TB	0.322	0.587	0.731	0.776
	PCA w/o TB	0.642	0.816	0.558	0.504
	LogCluster w/ TB	0.530	0.746	0.677	0.716
	LogCluster w/o TB	0.713	0.829	0.559	0.504
	DeepLog w/ TB	0.662	0.854	0.590	0.619
	DeepLog w/o TB	0.678	0.867	0.556	0.500
	DeepSVDD w/ TB	0.499	0.725	0.567	0.616
	DeepSVDD w/o TB	0.566	0.789	0.577	0.646
	LogTransfer	0.971	0.972	0.792	0.828
	LogTAD	0.926	0.964	0.758	0.804
TB → BGL	PCA w/ BGL	0.789	0.798	0.577	0.773
	PCA w/o BGL	0.760	0.779	0.229	0.658
	LogCluster w/ BGL	0.708	0.688	0.697	0.886
	LogCluster w/o BGL	0.724	0.716	0.223	0.500
	DeepLog w/ BGL	0.687	0.701	0.527	0.843
	DeepLog w/o BGL	0.660	0.677	0.223	0.500
	DeepSVDD w/ BGL	0.660	0.699	0.196	0.537
	DeepSVDD w/o BGL	0.794	0.808	0.195	0.497
	LogTransfer	0.995	0.995	0.788	0.833
	LogTAD	0.788	0.797	0.845	0.909

sequences from the target system in the training stage. The λ defined in Equation 3.4 is 0.1. Our code is available online [1].

3.3.2 Experimental Results

We evaluate the cross-system log anomaly detection on two scenarios, i.e., BGL → TB, where BGL is the source system while Thunderbird is the target system, and TB → BGL, where Thunderbird is the source system while BGL is the target system. We evaluate the performance in terms of F1 score and AUC. Table 3.3 shows the performance of LogTAD as well as baselines.

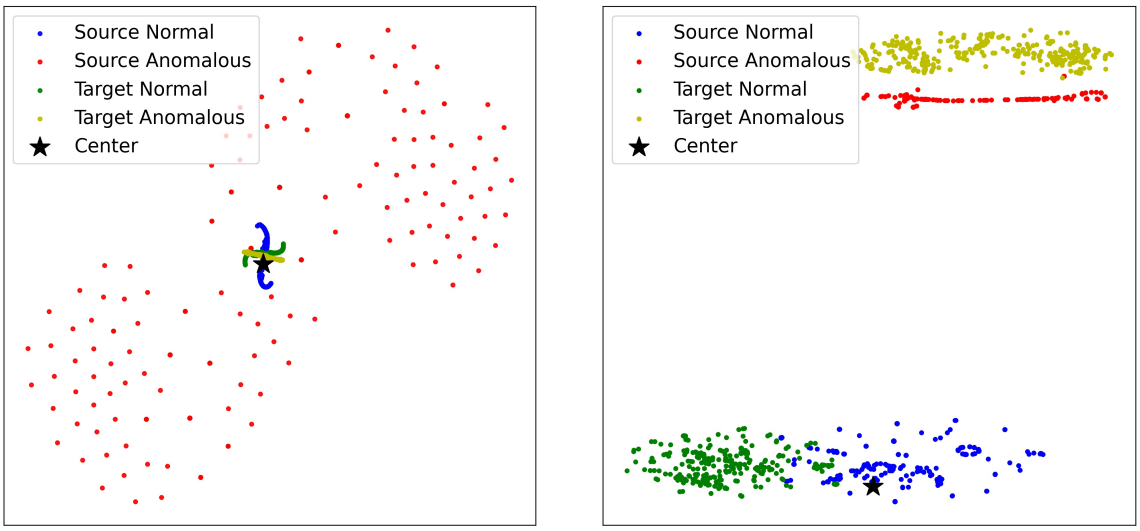
LogTAD v.s. Unsupervised Anomaly Detection Approaches. As PCA, LogCluster, and DeepLog are unsupervised models and not designed for domain adaptation, we evaluate these models in two scenarios, i.e., training with or without using the log sequences from the target system. First, for both BGL → TB and TB → BGL scenarios,

[1] https://github.com/hanxiao0607/LogTAD

PCA, LogCluster, and DeepLog can achieve reasonable F1 scores and AUC values in the source system with no training samples from the target system. However, when given log sequences from the target system in the training dataset, PCA, LogCluster, and DeepLog gain better F1 scores and AUC values in the target system with worse results in the source system. It indicates that the existing state-of-the-art unsupervised log anomaly detection models do not have the capability for cross-system adaptation. Using the log sequences from both source and target systems can only make the detection models confused about the distribution of normal samples, which leads to poor performance. On the other hand, LogTAD achieves better performance on both source and target systems under BGL $\rightarrow$ TB and TB $\rightarrow$ BGL scenarios compared with PCA, LogCluster, and DeepLog.

LogTAD v.s. DeepSVDD. DeepSVDD gets a lower F1 score in both source and target systems when given target sequences in the training dataset. This is because mixed training data lead to diverse data distribution. The center derived from such distribution is not useful for detecting anomalies in the target system since we only use a small number of samples from the target system. On the other hand, without domain adaptation, the samples from the target systems are more like outliers to the source system. As a result, the performance of DeepSVDD is also not good on the source system. Hence, using adversarial domain adaptation to make samples have similar distribution is key to achieve cross-system anomaly detection. Another observation is that when only using data from the source system, the performance of DeepSVDD is still not as good as LogTAD. This could be because LogTAD derives the center by using more samples with the same distribution.

LogTAD v.s. LogTransfer. LogTransfer is a supervised transfer learning method that adopts normal and anomalous data from the source and target systems to train a cross-system anomaly detection classifier. LogTransfer can achieve promising performance when sufficient labeled data are available. In this experiment, we aim to identify how many labeled anomalous samples are required to train the LogTransfer so that it can have a similar performance as LogTAD. Especially, we use 100 anomalous sequences from the source system and 10 anomalous sequences from the target system to train LogTransfer.

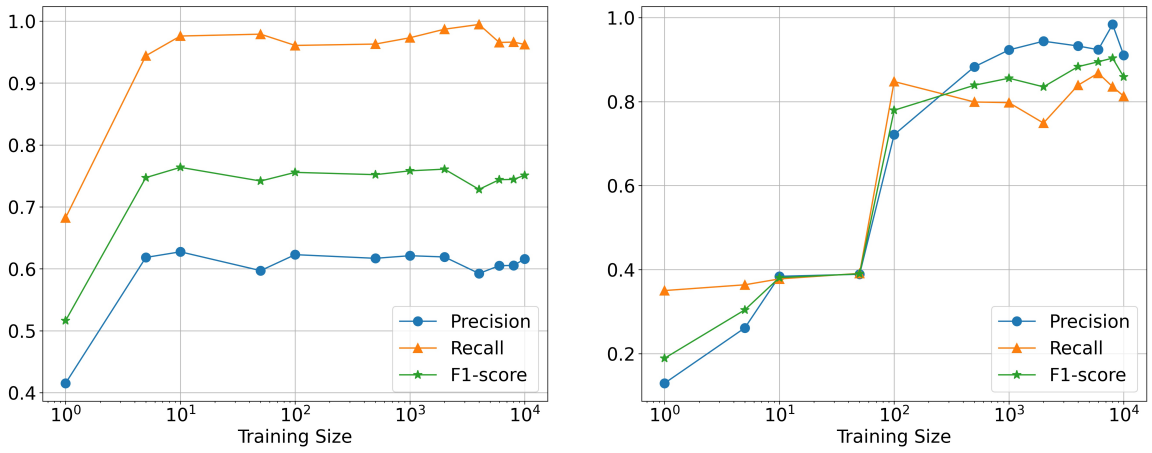


(a) Without domain adaptation (b) With domain adaptation

Fig. 3.2: Log Sequence Visualization (BGL → Thunderbird)

LogTransfer can achieve the best performance in the source system. For the scenario BGL → TB, LogTransfer also achieves slightly better performance than LogTAD, while for the scenario TB → BGL, using 10 anomalous sequences from the target system is not sufficient to outperform LogTAD. Therefore, in the cases where labeled anomalous samples are hard to obtain, LogTAD can still provide good performance by only using normal data.

Visualization. We show the effectiveness of our cross-system adaptation approach by visualizing sequence representations $\mathbf{v}$. We adopt the t-SNE algorithm to map the sequence representations into a two-dimensional space. For each dataset, we randomly select 300 normal and anomalous sequences, respectively, and consider the scenario BGL → TB. Figure 3.2 compares the sequence representations derived with and without domain adversarial training in the latent space. As shown in Figure 3.2a, for the model without domain adversarial training, normal sequences from the source and target systems and anomalous sequences from the target system are highly mixed. Meanwhile, these three types of sequences keep a small distance from the center in the latent space. Only anomalous sequences from the source system keep away from the center region. On the other hand, as shown in Figure 3.2b, with domain adversarial training, normal sequences from both source and target sys-



(a) BGL → Thunderbird

(b) Thunderbird → BGL

Fig. 3.3: LogTAD performance with different training sizes from the target system. (x-axis is in log-scale)

tems are close to the center, while anomalous sequences from both systems are out of the center region. Hence, LogTAD can detect anomalous sequences from both source and target systems.

Sample Size. We further evaluate the performance of LogTAD with various amounts of sequences from the target system. Figure 3.3 shows the experimental results. In general, we can notice that LogTAD can achieve a relatively high anomaly detection performance on the target system by using a small number of normal sequences from the target system. Especially, for the scenario BGL → TB, only using 10 normal log sequences from the Thunderbird dataset is sufficient to achieve good performance. When considering the scenario TB → BGL, using around 100 samples can achieve reasonable performance, and the performance can keep improving when more samples are available. In general, even the novel online system is deployed for a short term, a handful of sequences from the target system can be easily obtained, so LogTAD has strong feasibility and accuracy in detecting anomalies in the novel system.

3.4 Summary

In this work, we present a cross-system log anomaly detection framework, LogTAD.

It makes the normal log data in a hypersphere close to a center and utilizes the domain adversarial adaptation to make the log data from different systems follow similar distributions. Then, we can detect anomalies in different systems with large distances to the center. The experiment results show the effectiveness of our framework. One limitation of this work is that we assume the source and target systems should share similar patterns in terms of descriptive words. In the future, we plan to alleviate this assumption to improve the generalization of the approach. The early version of this work is published at CIKM 2021 [81].

CHAPTER 4

FEW-SHOT ANOMALY DETECTION AND CLASSIFICATION THROUGH REINFORCED DATA SELECTION AND FEW-SHOT ANOMALY DETECTION AND CLASSIFICATION THROUGH REINFORCED DATA SELECTION WITH A COMBINATORIAL REWARD

In this chapter, we introduce a novel framework called Few-shot Anomaly Detection and Classification model through Reinforced Data Selection (FADS). FADS is designed to tackle the challenge of detecting and classifying fine-grained anomalies in situations where there are many labeled normal samples but only a few labeled abnormal samples in each anomaly class, along with a large number of unlabeled samples. The core idea of FADS is to use a small set of labeled anomalies to train a few-shot learning model and then progressively improve this model by adding more potential anomalies from the unlabeled dataset. This is done through a reinforcement learning-based strategy that picks out the best samples from the unlabeled samples to add to the training set. These selected samples are high-quality, meaning they're likely to be anomalies with correct pseudo-labels, which helps the model get better over time. Our experiments show that FADS achieves top-notch performance in anomaly detection and classification.

Building on FADS, we then introduce an enhancement called FADScr. FADScr takes everything good about FADS and adds a novel combinatorial reward system to choose which samples to add from the unlabeled dataset. This new system makes FADScr even better at finding the right samples to improve the model's learning, making it more effective at detecting and classifying anomalies. This means FADScr not only overcomes the limitations of FADS but also sets a new standard in few-shot anomaly detection and classification, showing significant improvements in our tests.

4.1 Introduction

Anomaly detection indicates the detection of data samples that significantly deviate from the majority of data [51, 82]. Because of the extensive demand in a wide spectrum of applications, such as external and internal threats in cyberspace, anomaly detection has become an increasingly important research task.

Due to the small number of anomalies, classical supervised learning algorithms cannot be employed. Currently, the majority of approaches are trained in the unsupervised or semi-supervised learning manner [51,82,83]. However, one limitation of existing anomaly detection approaches is that existing approaches cannot further classify anomalies into specific anomaly categories. In many real-world scenarios, understanding the types of anomalies is critical. For example, for the malicious insider threat detection task, the malicious insider conducts a variety of malicious activities, such as using a thumb drive to steal data before leaving the company, logging into another user's machine and searching for interesting files, or uploading documents to Dropbox for personal gain. Anomaly detection aims to label all malicious activities as anomalies, which is good. Meanwhile, anomaly classification further predicts the malicious activities into corresponding categories, which can provide more insight to the domain experts so that defense strategies can be designed precisely.

However, due to the scarcity of anomalies, the number of anomalies in each class is even smaller. Hence, it is extremely hard to train a multiclass classifier in a traditional supervised manner. Recently, to learn from a limited number of labeled samples, few-shot learning as a special type of machine learning has become an emerging research topic, which aims to learn classifiers given only a few labeled samples of each class [84]. One common strategy of few-shot learning is to project limited samples into a smaller embedding space so that similar samples are grouped together while dissimilar samples are separated [84–87]. However, in the anomaly detection scenario, such cluster assumption only holds for normal samples since normal data are similar. For anomalies, by only having a few samples, it is hard to build a cluster to represent a class of anomalies, especially considering that anomalies are much more diverse. As a result, current few-shot anomaly detection approaches only work on

distinguishing anomalies from normal samples and cannot further divide the anomalies into fine-grained classes [88–91]. Hence, how to leverage the powerful few-shot learning models for anomaly detection and classification is still under-exploited.

In this work, we address the challenge of fine-grained anomaly detection and classification within real-world scenarios, characterized by an abundance of labeled normal samples, a scarcity of labeled abnormal samples across anomaly classes, and a vast amount of unlabeled data. To navigate these conditions, we initially introduced the Few-shot Anomaly Detection and Classification model with Reinforced Data Selection (FADS). FADS leverages the extensive unlabeled dataset to progressively refine the few-shot learner's capability for anomaly detection and classification by assuming that the unlabeled dataset mirrors the real-world distribution of a large number of normal samples interspersed with a few anomalies.

The core methodology of FADS involves a two-step iterative process of data selection and model retraining. Initially, a few-shot learning model is trained on a limited set of labeled anomalies. This model is then applied to the unlabeled dataset to predict anomalies, generating weakly-labeled samples. These predictions serve as a preliminary filter, after which our proposed reinforcement learning-based data selection strategy identifies high-quality samples. These samples, deemed likely to be correctly labeled, augment the existing labeled training set, creating a new, enhanced dataset for retraining the model. This cycle is expected to continuously improve the model's performance.

Building on the foundational principles of FADS, we further develop this concept into the Few-shot Anomaly Detection and Classification model through Reinforced Data Selection with a Combinatorial Reward (FADScr). FADScr enhances the original framework by incorporating a novel reinforcement learning strategy for data selection, which evaluates potential training samples not only for their ability to improve anomaly detection performance but also for their capacity to reduce uncertainty in model predictions. This dual-criteria evaluation is encapsulated in a combinatorial reward function, a key innovation of FADScr that distinguishes it from its predecessor.

The primary contributions of FADS and its extension, FADScr, can be summarized

as follows: First, we offer a novel framework that uses reinforced data selection from an unlabeled dataset to progressively enhance the few-shot learner for more accurate anomaly detection and classification, even with a minimal number of labeled anomalies per class. Second, the introduction of a reinforcement learning-based data selection strategy that includes a combinatorial reward mechanism for selecting high-quality samples represents a significant advancement in this domain. This strategy improves the model's performance by ensuring that selected samples contribute positively to both detection accuracy and prediction confidence. Lastly, experimental results validate that both FADS and FADScr achieve state-of-the-art performance in anomaly detection and few-shot anomaly classification, demonstrating their effectiveness in leveraging large-scale unlabeled datasets for fine-grained anomaly detection.

In essence, while FADS achieves better performance by enhancing few-shot learning models for anomaly detection and classification through reinforced data selection, FADScr builds upon and extends this foundation by introducing a more sophisticated approach to sample selection. This approach, characterized by its combinatorial reward function, further refines the model's ability to accurately identify and classify anomalies, thereby advancing the field of few-shot learning in complex, real-world scenarios.

4.2 Related Work

Anomaly detection. The tasks of anomaly detection have been studied for decades. In recent years, various learning-based anomaly detection approaches are proposed [51,82,83]. As anomalies are scarce, unsupervised and semi-supervised approaches are widely used to detect anomalies because manually labeling a large number of abnormal samples is time-consuming and labor-intensive [48,56,92–95]. The unsupervised anomaly detection approaches usually assume the availability of normal samples. The basic idea is to capture the normal patterns, and the anomalies can then be detected with highly deviant patterns. However, only observing the normal samples for training could lead to a high false-positive rate due to a lack of prior knowledge of true anomalies [40,96]. In many real-world scenarios, a small number of labeled abnormal samples, e.g., from domain experts are also available. Hence, semi-

supervised anomaly detection approaches, which assume the availability of labeled samples as well as large-scale unlabeled samples, are commonly used to boost the performance of anomaly detection by leveraging the limited anomalies [94, 95, 97, 98].

In literature, a few studies also leverage the reinforcement learning technique for anomaly detection [40, 93, 96, 99, 100]. Research in [96] considers anomaly detection in a data stream setting as a sequential decision process. The agent aims to select true anomalies from data streams for human judgment within a budget limit. Research in [93] proposes an inverse reinforcement learning model for sequential anomaly detection in an unsupervised setting by learning an implicit reward function to guide the agent. Research in [40] proposes a reinforcement learning framework to identify unknown anomalies by training an anomaly detection agent based on a few classes of known anomalies. In this work, we propose a reinforcement learning framework to augment the training dataset by exploring large-scale unlabeled samples. The agent in our approach is trained to select anomalies that can improve the performance of the few-shot learning model for anomaly detection and classification.

Few-shot learning. Few-shot classification models, which usually learn classifiers given only a few labeled samples of each class, have demonstrated great performance in computer vision and natural language processing [84, 101–104]. Few-shot learning is able to learn models for rare cases and can reduce data-collecting efforts. Hence, applying the few-shot learning techniques for anomaly detection is a natural fit. Recently, several few-shot learning approaches for anomaly detection have been proposed [88, 91, 105, 106]. However, these works can only detect anomalies but cannot achieve fine-grained classification. In this work, we propose a reinforced data selection technique to progressively improve the few-shot learner for anomaly detection and classification.

4.3 FADS

4.3.1 Problem Definition

Let $\mathcal{L} = \{(x_i, y_i)\}_{i=1}^{N_L}$ be a labeled dataset, where x_i indicates the i-th sample, while $y_i \in \{0, 1, ..., K\}$ indicates the corresponding label. Particularly, $y_i = 0$ indicates a normal

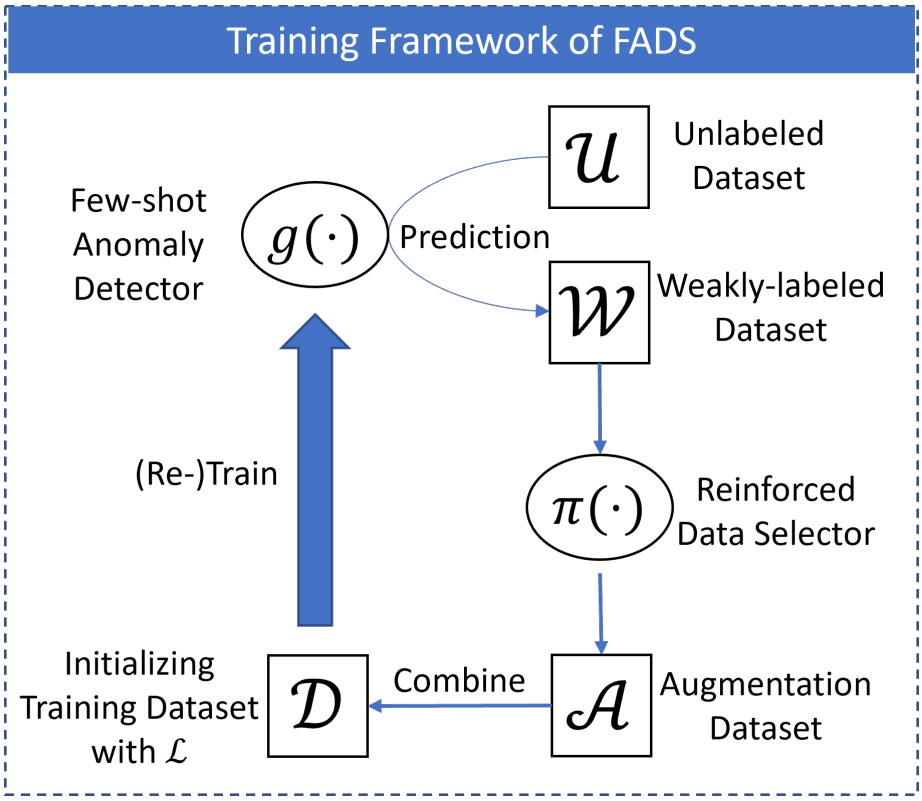


Fig. 4.1: The training framework of FADS

sample while $y_i \in \{1, ..., K\}$ denotes a class of anomalies. Consider that it is usually feasible to have a large number of normal samples in the anomaly detection scenario, the labeled dataset can be decomposed as $\mathcal{L} = \mathcal{L}_N \bigcup \mathcal{L}_A$ with $|\mathcal{L}_N| > |\mathcal{L}_A|$, where $\mathcal{L}_N$ only consists of normal samples and $\mathcal{L}_A$ includes few samples in each anomaly class.

With such a small amount of abnormal samples, it is hard to train an accurate anomaly detection and classification model. To tackle this challenge, in this work, besides a labeled dataset $\mathcal{L}$, we further leverage an unlabeled dataset $\mathcal{U} = \{x_j\}_{j=1}^{N_U}$. The goal is to learn a few-shot anomaly detection and classification model that can leverage the knowledge of the unlabeled dataset $\mathcal{U}$, especially the potential anomalies in $\mathcal{U}$, to maximally improve the performance of the model on anomaly detection and classification.

4.3.2 Framework Overview

In this work, we propose a few-shot anomaly detection and classification model through

reinforced data selection (FADS), which is able to gradually enhance the performance of the few-shot learning model by exploiting the unlabeled dataset $\mathcal{U}$.

In particular, FADS first trains a few-shot learning model on an initial training dataset with only labeled samples $\mathcal{D} = \mathcal{L}$. Then, FADS iteratively improves the model by selecting samples from unlabeled dataset $\mathcal{U}$ to augment the training dataset. We employ the reinforcement learning technique for data selection. In each training iteration, we first use the current few-shot model to predict the label $\hat{y}_j$ for each sample x_j in $\mathcal{U}$, thus producing a weakly-labeled dataset $\mathcal{W} = \{(x_j, \hat{y}_j)\}_{j=1}^{N_U}$. Then, we train a reinforcement learning agent to select the weakly-labeled samples that have high chances to be accurate into an augmentation set $\mathcal{A} = \{(x_j, \hat{y}_j)|a_j = 1\}_{j=1}^{N_U}$, where $a_j \in \{0, 1\}$ indicates whether the sample $(x_j, \hat{y}_j)$ is selected by the agent or not. We combine the augmentation dataset $\mathcal{A}$ with the existing training dataset to compose the new training dataset, i.e., $\mathcal{D} = \mathcal{D} \cup \mathcal{A}$, and then re-train the few-shot model on the new training dataset. We expect that the performance of the few-shot model will be improved with those augmented samples. Therefore, as the training iteration moves forward, the few-shot anomaly detection model can be improved progressively. The overview of FADS is shown in Figure 4.1.

4.3.3 Prototypical Network

In this work, we adopt the prototypical network [85] as our base few-shot learning model. Following the typical process for training the few-shot learning model, we first randomly select N_S and N_Q samples from the training dataset $\mathcal{D}$ to compose the support set $\mathcal{S}$ and query set $\mathcal{Q}$, respectively.

The prototypical network learns an embedding function $g(\cdot)$ to map each sample x_i to an embedding space, denoted as $\mathbf{x}_i = g(x_i)$. Based on the support set, the prototype representation for each class can be derived by a mean operation:

$$\mathbf{c}_k = \frac{1}{|\mathcal{S}_k|} \sum_{(x_i, y_i) \in \mathcal{S}_k} g(x_i), \tag{4.1}$$

where $\mathcal{S}_k$ indicates the subset of samples in the support set $\mathcal{S}$ with the class k. Then, given

a distance function $d(\cdot)$, the prototypical network predicts the distribution of classes for a query point $x \in \mathcal{Q}$ based on a softmax over distances to the prototypes:

$$p(y = k|\mathbf{x}) = \frac{exp(-d(g(\mathbf{x}), \mathbf{c}_k))}{\sum_{k'} exp(-d(g(\mathbf{x}), \mathbf{c}_{k'}))}. \tag{4.2}$$

The objective function of the prototypical network is the negative log-probability $L = -\log p(y = k|\mathbf{x})$ of the true class k. The training objective is to make the samples from the same class have small distances to the class prototype $\mathbf{c}_k$. A test sample can be labeled based on the label of its nearest prototype in the embedding space.

4.3.4 Reinforced Data Selection

As the labeled dataset $\mathcal{L}$ only has a very small number of anomalies, the performance of the initial prototypical network trained on $\mathcal{L}$ still has room for improvement if more training samples can be incorporated. To this end, we propose to exploit the unlabeled dataset $\mathcal{U}$. We first use the current prototypical network to predict the labels of samples in $\mathcal{U}$ and get a weakly-labeled dataset $\mathcal{W}$, which consists of both correctly and incorrectly labeled samples. If we simply adopt $\mathcal{W}$ to augment the training set, due to the noisy supervised signals, the performance of the prototypical network could be worse. Hence, we propose a reinforcement learning-based data selection method to select samples from $\mathcal{W}$ based on their reliability to compose an augmentation set $\mathcal{A}$. Once the reinforced data selection agent is capable of selecting the correctly labeled samples to augment the training set, the performance of the few-shot learner can be improved. Therefore, the reward to the data selection agent is designed based on the performance of the few-shot learner on an unobserved validation set. The key components of the reinforcement learning framework are described as below.

State. For each state s_j with weakly-labeled sample $(x_j, \hat{y}_j) \in \mathcal{W}$, its state representation $\mathbf{s}_j$ is defined as the concatenation of the embedding vector $\mathbf{x}_j = g(x_j)$ and the distance value d_j to the closest prototype, i.e., $\mathbf{s}_j = [\mathbf{x}_j, d_j]$.

Action. The data selection agent then needs to take an action whether to select this sample $(x_j, \hat{y}_j)$ into the augmentation set $\mathcal{A}$ based on the state representation $\mathbf{s}_j$. In specific, $a_j = 0$

means the sample $(x_j, \hat{y}_j)$ will be rejected, while $a_j = 1$ indicates the sample will be selected and added to the augmentation set $\mathcal{A}$.

Policy Network. The data selection agent makes decisions about whether to select a weakly-labeled sample based on a policy network $\pi_\theta(\cdot)$. We adopt a neural network to parameterize the policy network that takes the state representation as input and outputs the probability of the action, $p(a_j|s_j) = \pi_\theta(\mathbf{s}_j)$. The action a_j is then sampled based on $p(a_j|s_j)$.

Rewards. The policy network is trained with guidance from the reward function. We define the reward based on the performance of the few-shot learning model on an unseen validation set. As in our task, we aim to detect anomalies as well as classify abnormal samples into different classes. Therefore we design the reward function based on the performance of anomaly detection and classification, i.e., an anomaly detection reward r^d and a classification reward r^c. Specifically, we adopt the F1 score as the metric to evaluate the performance of anomaly detection and the macro F1 score over all classes to evaluate the performance of classification.

In our scenario, if wrongly-labeled samples are selected to compose the training dataset, we can expect that the performance of the few-shot learning model will be damaged. On the other hand, if the samples with correct labels are selected, the performance of the model can be improved. Hence, the reward is designed based on the performance change after the model is trained on an augmentation set.

Specifically, in each episode, we first calculate the F1 and macro F1 scores of the current prototypical network. We denote the prototypical network at the beginning of each episode as $g_0(\cdot)$ and the corresponding F1 and macro F1 scores achieved by $g_0(\cdot)$ as $F1_0$ and $MarcoF1_0$. Then, we work on improving $g_0(\cdot)$ by composing an augmentation set. To this end, first, we randomly sample a set of batches $\mathcal{B} = \{B_l\}_{l=1}^{L}$ from $\mathcal{W}$, where each batch B_l consists of a number of samples. Given a batch B_l, for each sample in B_l, we sample an action a_j (select to the augmentation set or not) based on the policy $p(a|s_j) = \pi_\theta(\mathbf{s}_j)$. Then, we can compose the augmentation set $\mathcal{A}_l$ from the batch B_l. After combining $\mathcal{A}_l$ with $\mathcal{D}$, we

update the prototypical network, denoted as $g_l(\cdot)$, and further evaluate $g_l(\cdot)$ on a validation set to derive F1 and macro F1 scores, denoted as $F1_l$ and $MacroF1_l$. Finally, the reward for the agent in a batch can be calculated as the difference of F1 scores, $r_l^d = F1_l - F1_0$ and $r_l^c = MacroF1_l - MacroF1_0$. The overall reward function in a batch is formulated as:

$$r_l = r_l^d + \alpha \cdot r_l^c, \tag{4.3}$$

where α is a hyperparameter to balance anomaly detection and classification tasks. In the experiments, we set $\alpha = 1$.

Optimization. The data selection agent is trained based on the actor-critic algorithm [2]. The output of the actor is the probability of two actions (select or not), while the output of the critic is the predicted reward value based on the current state. Both actor and critic are parameterized by feed-forward neural networks. The goal of the agent is to maximize the rewards, which is defined as:

$$\mathcal{J}(\theta) = \mathbb{E}_{\pi_\theta}[r_l]. \tag{4.4}$$

The parameter θ in policy network π_θ is trained based on policy gradient [107]:

$$\theta \leftarrow \theta + \eta \bigtriangledown_\theta \mathcal{J}(\theta), \tag{4.5}$$

where η indicates the learning rate. The gradient for a batch of weakly-labeled samples can be approximated by [2]

$$\bigtriangledown_\theta \mathcal{J}(\theta) = \frac{1}{|B_l|} \sum_{j=1}^{|B_l|} \bigtriangledown_\theta log \pi_\theta(\mathbf{s}_j)(r_l - V_\varphi(\mathbf{s}_j)), \tag{4.6}$$

where $|B_l|$ indicates the number of samples in a batch B_l; $V_\varphi(\mathbf{s}_j)$ is the expected reward from a critic network $V_\varphi(\cdot)$ parameterized by φ. The structure of the critic network is similar to the policy network with the last layer as a regression function. The critic network is designed to estimate the expected reward and hence updated according to the cumulative

difference between the real reward r_l and the predicted value $V_\varphi(\mathbf{s}_j)$,

$$\mathcal{L}(\varphi) = \frac{1}{|B_l|} \sum_{j=1}^{|B_l|} |r_l - V_\varphi(\mathbf{s}_j)|. \tag{4.7}$$

The parameters φ in $V_\varphi(s_j)$ can be optimized by:

$$\varphi = \varphi - \eta' \bigtriangledown_\varphi \mathcal{L}(\varphi), \tag{4.8}$$

where η' indicates the learning rate for updating the critic network.

4.3.5 Training Details

General Training Process. Algorithm 1 shows the pseudo-code of the training process. Given a small labeled dataset $\mathcal{L}$ and an unlabeled dataset $\mathcal{U}$, we first initialize an augmentation dataset $\mathcal{A}$ as an empty set (Line 1) and a training dataset $\mathcal{D}$ as the small labeled dataset $\mathcal{L}$ (Line 2). In each episode, we split $\mathcal{D}$ into two parts, $\mathcal{D}^{tr}$ as the real training set and $\mathcal{D}^{val}$ as the unobserved validation set (Line 4), and use $\mathcal{D}^{tr}$ to train the prototypical network $g(\cdot)$ (Line 5). We deploy the prototypical network to predict labels of samples in $\mathcal{U}$ and get a weakly labeled data set $\mathcal{W}$ (Line 6). We then apply a reinforced data selection algorithm to generate a batch of samples as the augmentation set $\mathcal{A}$ from $\mathcal{W}$ (Line 7). After obtaining $\mathcal{A}$, we remove $\mathcal{A}$ from the unlabeled dataset $\mathcal{U}$ (Line 8) and add $\mathcal{A}$ to $\mathcal{D}$ to form a new training data for next iteration (Line 9).

With the increasing of training episodes, the size of the training dataset $\mathcal{D}^{tr}$ keeps increasing and the performance of the prototypical network becomes better. After finishing all the training iterations, we further tune the prototypical network based on the whole dataset $\mathcal{D}$ (Line 10) and deploy it for anomaly detection and classification.

Reinforced Data Selection. The key step for improving the performance of the prototypical network is the reinforced data selection algorithm (Lines 13 – 29), which is to select the reliable weakly-labeled samples from $\mathcal{W}$ to augment the training set. We first apply the current prototypical network $g_0(\cdot)$ on the validation set $\mathcal{D}^{val}$ to get F1 and macro F1 scores

Algorithm 1: The training process of FADS

Input : Labeled set $\mathcal{L}$, unlabeled set $\mathcal{U}$
Output: Prototypical network $g(\cdot)$

Initialize an augmentation set $\mathcal{A} \leftarrow \emptyset$
Initialize a training dataset $\mathcal{D} \leftarrow \mathcal{L}$
for $t = 0$ **to** T **do**
    Split $\mathcal{D}$ to $\mathcal{D}^{tr}$ and $\mathcal{D}^{val}$
    Train prototypical network $g(\cdot)$ on $\mathcal{D}^{tr}$
    Predict labels on $\mathcal{U}$ and get $\mathcal{W}$ based on $g(\cdot)$
    $\mathcal{A} \leftarrow$ Data_Selection ($g(\cdot)$, $\mathcal{W}$, $\mathcal{D}^{tr}$, $\mathcal{D}^{val}$)
    Remove selected samples $\mathcal{U} \leftarrow \mathcal{U}/\mathcal{A}$
    Augmentation dataset $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{A}$
Train prototypical network $g(\cdot)$ on $\mathcal{D}$
return prototypical network $g(\cdot)$

Function `Data_Selection`($g_0(\cdot)$,$\mathcal{W}$,$\mathcal{D}^{tr}$,$\mathcal{D}^{val}$)`:`
    Apply $g_0(\cdot)$ on $\mathcal{D}^{val}$ to derive $(F1_0, MacroF1_0)$
    Generate subsets $\mathcal{B} = \{B_l\}_{l=1}^{L}$ from $\mathcal{W}$
    $\mathcal{R} = \emptyset$, $\mathcal{A} = \emptyset$
    foreach $B_l \in \mathcal{B}$ **do**
        Agent selects samples from B_l to generate $\mathcal{A}_l$
        $\mathcal{D}_l^{tr} \leftarrow \mathcal{D}^{tr} \cup \mathcal{A}_l$
        Train prototypical network $g_l(\cdot)$ on $\mathcal{D}_l^{tr}$
        Apply $g_l(\cdot)$ on $\mathcal{D}^{val}$ to derive $(F1_l, MacroF1_l)$
        Compute the reward r_l based on Eq. (4.3)
        $\mathcal{R} = \mathcal{R} \cup \{r_l\}$
        Update the policy network π_θ based on Eq. (4.5)
        Update the critic network V_φ based on Eq. (4.8)
    $l \leftarrow \arg\max \mathcal{R}$
    if $F1_l > F1_0$ *and* $MacroF1_l > MacroF1_0$ **then**
        $\mathcal{A} \leftarrow \mathcal{A}_l$
    return $\mathcal{A}$

$(F1_0, MacroF1_0)$ as the baseline performance (Line 14). We then derive a set of sample batches $\mathcal{B} = \{B_l\}_{l=1}^{L}$ from $\mathcal{W}$. Then, for each batch B_l, we select samples based on the prediction of the policy network $p(a_j|s_j) = \pi_\theta(\mathbf{s}_j)$ and generate $\mathcal{A}_l$ with selected samples. After getting the augmented training set $\mathcal{D}_l^{tr}$ by combining $\mathcal{D}^{tr}$ with $\mathcal{A}_l$ (Line 19), we train the prototypical network $g_l(\cdot)$ on $\mathcal{D}_l^{tr}$ again (Line 20) and apply $g_l(\cdot)$ on $\mathcal{D}^{val}$ to derive F1 and macro F1 scores, $(F1_l, MacroF1_l)$, (Line 21). The reward is then calculate based on the difference between $(F1_l, MacroF1_l)$ and $(F1_0, MacroF1_0)$. We further update the policy network π_θ and the critic network V_φ (Lines 24 – 25).

After going through all batches, we get a set of rewards $\mathcal{R} = \{r_l\}_{l=0}^{L}$. We only select $\mathcal{A}_l$ leading to the maximum reward r_l in $\mathcal{R}$ (Line 26) and also ensure that the prototypical network $g_l(\cdot)$ trained on $\mathcal{D}_l^{tr}$ can get better performance in terms of F1 and Macro F1 scores compared with the original one $g_0(\cdot)$ (Lines 27-28). Then, we consider $\mathcal{A}_l$ as the augmentation set $\mathcal{A}$ at the current episode (Line 29).

Warm-up Phase. In the early stage of training, the data selection agent takes actions based on randomly initialized parameters. The selected samples could include misclassified instances due to the uncertainty of the policy network. Then, both $\mathcal{D}^{tr}$ and $\mathcal{D}^{val}$ could contain mislabeled samples, which could damage the performance of FADS in the long turn. Hence, we set up a warm-up phase for the training. After the warm-up phase, all the samples selected during the warm-up phase will be removed from $\mathcal{D}$ and added back to $\mathcal{U}$. The agent will select samples from scratch again. We expect the performance of the data selection agent will become stable with the convergence of parameters during the warm-up phase and can select reliable weakly-labeled samples after that. In our experiments, the warm-up phase is 10 episodes.

4.3.6 Experimental Setup

Datasets. We apply FADS to detect and classify anomalies on the following three datasets.

- **UNSW-NB15** [108]. The UNSW-NB15 dataset was generated by the IXIA PerfectStorm tool in the Cyber Range Lab of UNSW Canberra. It consists of normal and five synthetic attacks including suspension caused by feeding random data (Fuzzers), denial-of-

Table 4.1: Statistics of datasets in experiments

Dataset	Class	Training		Testing
		$\mathcal{L}$	$\mathcal{U}$	
UNSW-NB15 (293 features)	Normal	100	30,000	20,000
	Fuzzers	10	3,000	2,000
	DoS	10	3,000	2,000
	Exploits	10	3,000	2,000
	Generic	10	3,000	2,000
	Reconnaissance	10	3,000	2,000
IDS2018 (77 features)	Normal	100	30,000	20,000
	Bot	10	3,000	2,000
	DoS	10	3,000	2,000
	Bruteforce	10	3,000	2,000
CERT (various length)	Normal	100	2,000	2,000
	DataUploading	10	27	28
	DataStealing	10	201	201
	MassEmail	10	10	10
	UnauthdLogin	10	21	21

services by overloading the server or network (DoS), attacks with known security problems (Exploits), attack block-ciphers without information about the structure (Generic), and attacks for collecting information (Reconnaissance).

• **IDS2018** [109]. IDS2018 contains detailed network traffic and log files of five victim organizations with 420 PCs and 30 servers. It consists of normal network traffic and three different types of attacking traffic including automatically synthesized requests of upload, download, screenshots, and keylogging (Bot), denial-of-service by overloading the server or network (DoS), and password crack by brute-force (Bruteforce).

• **CERT** [110]. CERT is an insider threat dataset that contains a collection of synthetic normal and malicious user activities. We use CERT V5.2, which consists of four types of insider threats including unauthorized uploading data (DataUploading), using a thumb drive to steal data (DataStealing), sending out mass emails causing panic (MassEmail), unauthorized log into other computers (UnauthdLogin). Due to the extremely small number of insiders, the number of samples for each anomaly class is small.

Data Preprocessing. Both UNSW-NB15 and IDS2018 are multidimensional datasets, where each instance consists of multiple features to describe the basic information. After typical preprocessing steps, each sample in UNSW-NB15 consists of 293 features, while each sample in IDS2018 consists of 77 features. We randomly select 100 normal samples as

$\mathcal{L}_N$ and 10 samples from each anomaly class as $\mathcal{L}_A$. To construct an unlabeled dataset $\mathcal{U}$, we randomly select 30,000 normal samples and 3,000 abnormal samples from each anomaly class. The testing dataset contains 20,000 normal samples and 2,000 abnormal samples from each anomaly class.

CERT is a sequential dataset, where each entry indicates one user activity. For CERT V5.2, there are 23 different types of activities, such as logon, logoff, web visiting, and file editing. We group user activities into sessions as data samples, and each session consists of all user activities between logon and logoff operations on a computer. Similarly, we randomly select 100 normal samples as $\mathcal{L}_N$ and 10 samples from each anomaly class as $\mathcal{L}_A$. The unlabeled dataset $\mathcal{U}$ is composed of randomly selected 2,000 normal sessions and half of the remaining abnormal sessions (excluding the 10 sessions used in $\mathcal{L}_A$) from each anomaly class. The testing dataset consists of 2,000 normal sessions and the other half of the remaining data in each anomalous class. Table 4.1 summarizes the statistics of datasets used in our experiments.

Baselines. We use two sets of baselines to evaluate the performance of FADS for anomaly detection and classification, respectively.

Baselines for anomaly detection. We adopt the following six baselines to evaluate the performance of anomaly detection.

• **One-Class SVM (OCSVM)** [111] is a one-class classification model that can detect outliers based on normal samples.

• **Isolation Forest (iForest)** [54] is a widely used tree-based anomaly detection model.

• **Label Random PU Learning (LRPU)** [112] is a classical positive-unlabeled learning approach. It trains a PU learner to predict probabilities that differ by only a constant factor from the true conditional probabilities of being positive.

• **PU learning with a Selection Bias (PUSB)** [113] is an advanced positive-unlabeled learning approach. It tackles the challenge when positive samples include noise.

• **Deep Support Vector Data Description (DeepSVDD)** [80] is a deep learning-based one-class anomaly detection method. The method deploys a neural network to map

normal data into a hypersphere with minimum volume. Samples that fall outside of the hypersphere are labeled as anomalies.

• **Deep Semi-Supervised Anomaly Detection (Deep SAD)** [94] is a semi-supervised anomaly detection model. Deep SAD utilizes a small number of labeled normal and anomalous samples as well as an unlabeled dataset to define a hypersphere that can efficiently cover most of the normal samples and further identify the anomalies outside the hypersphere.

OCSVM, iForest, and DeepSVDD as one-class models are trained on the normal set $\mathcal{L}_N$. LRPU and PUSB as positive-unlabeled models are trained on the anomalous set $\mathcal{L}_A$ and unlabeled set $\mathcal{U}$. Deep SAD as a semi-supervised model is trained on the labeled set $\mathcal{L}$ and unlabeled set $\mathcal{U}$.

Baselines for anomaly classification. We adopt the following two baselines to evaluate the performance of anomaly classification.

• **Support Vector Machine (SVM) [114]** is a classification method. We adopt the one-vs-rest strategy for multiclass anomaly detection.

• **Multilayer Perceptron (MLP) [115]** is a classical neural network for multiclass classification.

We train both baselines on the labeled set $\mathcal{L}$ and also adopt the synthetic minority oversampling technique (SMOTE) [116] for oversampling the data in anomaly classes.

Evaluation Metrics. For anomaly detection, we consider all anomaly classes as one anomaly class and adopt the Area Under Precision-Recall Curve (AUC-PR), the Area Under Receiver Operating Characteristic Curve (AUC-ROC), and False Positive Rate (FPR) at 95% True Positive Rate (TPR) to measure the performance of FADS. Especially, FPR at 95% TPR can be considered as the probability that an anomaly is misclassified as normal when the true positive rate is 95%. For anomaly classification, Precision, Recall, and F1 score are used to evaluate the performance of FADS on each abnormal class. We also derive the Macro-F1 score to evaluate the performance over all classes. In our experiments, we report the results over 10 runs. The paired t-test is adopted to examine the statistical significance of the performance of FADS over the best baseline.

4.3.7 Implementation Details

For UNSW-NB15 and IDS2018 datasets, because the input samples are multidimensional data, we adopt a feedforward neural network with three fully connected layers as the implementation of prototypical network $g(\cdot)$, where the dimension of output embedding is 128. For the CERT dataset, because the input samples are sequential data about user activities in a session, we adopt a GRU neural network [117] to map sequences into an embedding space with the dimension of 64. For all three datasets, as we only have 10 labeled samples in each anomaly class in the initial training set $\mathcal{L}_A$, we initially assign 6 samples to $\mathcal{D}^{tr}$ to train the prototypical network with 3 samples as the support set $\mathcal{S}$ and 3 samples as the query set $\mathcal{Q}$ and the rest 4 samples to $\mathcal{D}^{val}$ for evaluation. With the training episode increasing, we assign more samples selected by the policy network to $\mathcal{D}^{tr}$ and $\mathcal{D}^{val}$ with a fixed ratio as 30% of samples in $\mathcal{S}$, 30% of samples in $\mathcal{Q}$, and 40% of samples in $\mathcal{D}^{val}$ to retrain the prototypical network.

In the implementation of actor-critic reinforcement learning, both actor and critic are three-layer feedforward neural networks. In the training stage, we set the number of training episodes $T = 20$, while in each episode, we generate a set of sample batches $\{B_l\}_{l=1}^{L}$ with $L = 50$. For each batch B_l, we randomly select samples from each predicted class in $\mathcal{W}$ and compose a balanced set for the actor to select. Especially, for UNSW-NB15 and IDS2018 datasets, we randomly select 10 samples from each class, so the sizes of each batch are $|B_l| = 60$ for the UNSW-NB15 dataset and $|B_l| = 40$ for the IDS2018 dataset, respectively. For the CERT dataset, we select 2 samples from each class with the batch size $|B_l| = 10$. The **code and datasets** used in the experiments are available online [1].

4.3.8 Experimental Results

Anomaly Detection. We first evaluate the performance of FADS for anomaly detection. In particular, after the prototypical network predicts each anomaly into a specific anomaly class, we consider all detected samples in anomaly classes as anomalies. We compare FADS for anomaly detection with two traditional one-class classification models (OCSVM and

[1] https://github.com/hanxiao0607/FADS

]

Table 4.2: Results on anomaly detection (mean ± std.). ↑ indicates larger value is better; ↓ indicates lower value is better.

		UNSW-NB15	IDS2018	CERT
AUC-PR ↑	OCSVM	0.5001 ± 0.0020	0.3026 ± 0.0137	0.1233 ± 0.0199
	iForest	0.7564 ± 0.0321	0.2936 ± 0.0093	0.1331 ± 0.0218
	LRPU	0.6542 ± 0.0049	0.4393 ± 0.0689	0.1395 ± 0.0362
	PUSB	0.6108 ± 0.0372	0.3741 ± 0.0155	0.1147 ± 0.0000
	DeepSVDD	0.6334 ± 0.0163	0.3516 ± 0.0759	0.1149 ± 0.0087
	DeepSAD	0.8203 ± 0.1976	0.5959 ± 0.0905	0.1269 ± 0.0103
	FADS	$\mathbf{0.9178 \pm 0.0214}$	$\mathbf{0.6791 \pm 0.0279}^{*}$	$\mathbf{0.2480 \pm 0.0260}^{**}$
AUC-ROC ↑	OCSVM	0.7501 ± 0.0020	0.6447 ± 0.0202	0.5122 ± 0.0910
	iForest	0.9131 ± 0.0181	0.5513 ± 0.0233	0.5751 ± 0.0827
	LRPU	0.7407 ± 0.0037	0.6828 ± 0.0894	0.5404 ± 0.0623
	PUSB	0.7081 ± 0.0279	0.5932 ± 0.0101	0.5000 ± 0.0000
	DeepSVDD	0.7932 ± 0.0298	0.6235 ± 0.0709	0.5032 ± 0.0409
	DeepSAD	0.9042 ± 0.1219	0.8310 ± 0.0553	0.5510 ± 0.0419
	FADS	$\mathbf{0.9751 \pm 0.0080}^{*}$	$\mathbf{0.9249 \pm 0.0134}^{**}$	$\mathbf{0.7286 \pm 0.0286}^{**}$
FPR (95% TPR) ↓	OCSVM	0.9938 ± 0.0007	0.9819 ± 0.0031	0.9046 ± 0.0484
	iForest	0.1941 ± 0.0459	0.9796 ± 0.0228	0.7732 ± 0.1477
	LRPU	0.4557 ± 0.1765	0.6235 ± 0.0776	0.8872 ± 0.0489
	PUSB	0.9923 ± 0.0041	0.8291 ± 0.2636	0.9532 ± 0.0070
	DeepSVDD	0.7081 ± 0.1564	0.8515 ± 0.1457	0.8694 ± 0.0572
	DeepSAD	0.2884 ± 0.4027	0.4002 ± 0.3597	0.7645 ± 0.1141
	FADS	$\mathbf{0.0375 \pm 0.0180}^{*}$	$\mathbf{0.2380 \pm 0.1108}$	$\mathbf{0.7036 \pm 0.1119}$

Significantly outperforms DeepSAD at the: * 0.05 and ** 0.01 level, paired t-test.

Table 4.3: Results on anomaly classification (mean ± std.)

Dataset	Class	Precision			Recall			F1		
		SVM	MLP	FADS	SVM	MLP	FADS	SVM	MLP	FADS
UNSW-NB15	Normal	0.8902 ± 0.0418	0.9953 ± 0.0042	0.9960 ± 0.0034	0.9839 ± 0.0067	0.9043 ± 0.0261	0.9581 ± 0.0112	0.9356 ± 0.0219	0.9474 ± 0.0146	$\mathbf{0.9766 \pm 0.0070}$
	Fuzzers	0.4120 ± 0.0374	0.3724 ± 0.0315	0.3913 ± 0.0491	0.3098 ± 0.1452	0.0459 ± 0.0877	0.3812 ± 0.0812	0.3369 ± 0.1012	$\mathbf{0.4079 \pm 0.0471}$	0.3808 ± 0.0530
	DoS	0.4407 ± 0.1241	0.4758 ± 0.0814	0.4227 ± 0.1007	0.5985 ± 0.2082	0.4995 ± 0.1538	0.4150 ± 0.1159	$\mathbf{0.4996 \pm 0.1607}$	0.4797 ± 0.1040	0.4130 ± 0.1036
	Exploits	0.3517 ± 0.1029	0.3631 ± 0.0814	0.3606 ± 0.0671	0.1026 ± 0.1465	0.4191 ± 0.1213	0.3987 ± 0.1123	0.1174 ± 0.1338	$\mathbf{0.3809 \pm 0.0835}$	0.3720 ± 0.0809
	Generic	0.9810 ± 0.0069	0.6091 ± 0.1201	0.9332 ± 0.0653	0.9697 ± 0.0109	0.9732 ± 0.0022	0.9655 ± 0.0152	$\mathbf{0.9752 \pm 0.0049}$	0.7428 ± 0.0841	0.9478 ± 0.0362
	Reconnaissance	0.3713 ± 0.1090	0.3088 ± 0.0580	0.2809 ± 0.0483	0.1991 ± 0.1215	0.2475 ± 0.0896	0.3437 ± 0.0925	0.2236 ± 0.1037	0.2663 ± 0.0683	$\mathbf{0.3052 \pm 0.0585}$
	macro-average	0.5745 ± 0.0393	0.5208 ± 0.0367	0.5641 ± 0.0313	0.5272 ± 0.0455	0.5838 ± 0.0331	0.5770 ± 0.0367	0.5147 ± 0.0482	0.5375 ± 0.0353	$\mathbf{0.5659 \pm 0.0348}^{*}$
IDS2018	Normal	0.9903 ± 0.0172	0.9935 ± 0.0091	0.9954 ± 0.0087	0.5607 ± 0.0402	0.7329 ± 0.0940	0.8635 ± 0.0186	0.7150 ± 0.0331	0.8398 ± 0.0653	$\mathbf{0.9246 \pm 0.0102}$
	Bot	0.5910 ± 0.2390	0.6445 ± 0.2386	0.4570 ± 0.0203	0.7960 ± 0.2489	0.7946 ± 0.2478	0.9992 ± 0.0008	0.5972 ± 0.0373	$\mathbf{0.6298 \pm 0.0303}$	0.6269 ± 0.0191
	DoS	0.2019 ± 0.0425	0.3792 ± 0.1892	0.8529 ± 0.1446	0.7191 ± 0.0074	0.7186 ± 0.0078	0.7337 ± 0.0150	0.3126 ± 0.0525	0.4693 ± 0.1540	$\mathbf{0.7817 \pm 0.0668}$
	Bruteforce	0.5226 ± 0.0838	0.5830 ± 0.1111	0.7583 ± 0.0235	0.9507 ± 0.0987	0.9199 ± 0.1038	0.9392 ± 0.0939	0.6684 ± 0.0813	0.7053 ± 0.0952	$\mathbf{0.8371 \pm 0.0465}$
	macro-average	0.5765 ± 0.0459	0.6501 ± 0.0345	0.7659 ± 0.0345	0.7566 ± 0.0683	0.7915 ± 0.0819	0.8839 ± 0.0206	0.5733 ± 0.0292	0.6610 ± 0.0628	$\mathbf{0.7926 \pm 0.0197}^{**}$
CERT	Normal	0.8854 ± 0.0000	0.9681 ± 0.0147	0.9452 ± 0.0081	1.0000 ± 0.0000	0.4819 ± 0.0739	0.8294 ± 0.0356	0.9392 ± 0.0000	0.6397 ± 0.0653	$\mathbf{0.8830 \pm 0.0196}$
	DataUploading	0.0000 ± 0.0000	0.2030 ± 0.0878	0.3112 ± 0.1011	0.0000 ± 0.0000	0.8630 ± 0.0739	0.8222 ± 0.1058	0.0000 ± 0.0000	0.3196 ± 0.1032	$\mathbf{0.4440 \pm 0.1228}$
	DataStealing	0.0000 ± 0.0000	0.2746 ± 0.0310	0.4129 ± 0.0813	0.0000 ± 0.0000	0.7657 ± 0.0859	0.5055 ± 0.0809	0.0000 ± 0.0000	0.4023 ± 0.0382	$\mathbf{0.4477 \pm 0.0568}$
	MassEmail	0.0000 ± 0.0000	0.0221 ± 0.0045	0.4129 ± 0.0813	0.0000 ± 0.0000	0.7100 ± 0.1375	0.6600 ± 0.1356	0.0000 ± 0.0000	0.0428 ± 0.0083	$\mathbf{0.1179 \pm 0.0553}$
	UnauthdLogin	0.0000 ± 0.0000	0.0784 ± 0.0134	0.0659 ± 0.0813	0.0000 ± 0.0000	0.8238 ± 0.0604	0.7762 ± 0.0675	0.0000 ± 0.0000	0.1428 ± 0.0229	$\mathbf{0.4755 \pm 0.0860}$
	macro-average	0.1771 ± 0.0000	0.3092 ± 0.0205	0.4172 ± 0.0287	0.2000 ± 0.0000	0.7289 ± 0.0385	0.7187 ± 0.0399	0.1787 ± 0.0000	0.3095 ± 0.0331	$\mathbf{0.4736 \pm 0.0344}^{**}$

Significantly outperforms MLP in terms of macro-F1 at the: * 0.05 and ** 0.01 level, paired t-test.

iForest), two positive-unlabeled learning models (LRPU and PUSB), and two advanced deep anomaly detection models (DeepSVDD and DeepSAD). As shown in Table 4.2, FADS achieves better performance than all baselines with a large margin on all three datasets. Especially, although DeepSAD as a semi-supervised model also uses both labeled and unlabeled datasets, FADS significantly outperforms DeepSAD in most cases and has a much smaller

standard deviation. It means actively identifying the potential anomalies in the unlabeled dataset can improve the performance of anomaly detection. Meanwhile, DeepSAD still outperforms other baselines, which shows the advantage of using a small set of labeled samples for anomaly detection. LRPU as a PU learner achieve better performance than three one-class models (OCSVM, iForest, and DeepSVDD) on IDS2018 and CERT datasets, meaning that in some cases, leveraging the unlabeled set can improve the performance. However, due to the limited anomalies (positive samples), both LRPU and PUSB underperform FADS.

Anomaly Classification. FADS can achieve fine-grained anomaly classification based on the prototypical networks, so we further evaluate FADS for the anomaly classification task. We compare FADS with two classification models, SVM and MLP. Table 4.3 shows the results on anomaly classification. In short, FADS achieves the highest macro-F1 score over all classes, meaning that FADS can achieve the best performance for anomaly classification. Meanwhile, on the UNSW-NB15 and IDS2018 datasets, FADS achieves the best performance in most classes in terms of F1 score compared with SVM and MLP. CERT is the most challenging dataset because of extremely limited labeled samples. On the CERT dataset, SVM predicts all anomalies as normal, while MLP has a low recall in the normal set indicating a high misclassification rate on normal samples. On the other hand, FADS achieves a significant improvement compared with baselines on CERT. It indicates for the challenging anomaly detection task with extremely limited samples, leveraging anomalies in the unlabeled dataset is critical to boosting the performance.

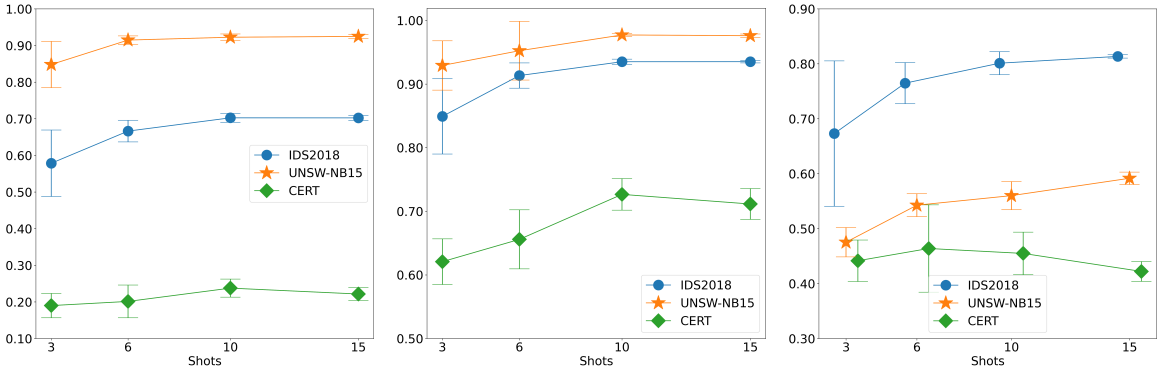
Table 4.4: Performance of FADS with or without reinforced data selection (mean ± std.)

Dataset	Approach	Anomaly Detection			Anomaly Classification		
		AUC-PR ↑	AUC-ROC ↑	FPR at 95% TPR ↓	Precision	Recall	F1
UNSW-NB15	ProtoNet	0.8639 ± 0.0461	0.9542 ± 0.0199	0.0781 ± 0.0411	0.5257 ± 0.0398	0.5586 ± 0.0434	0.5307 ± 0.0388
	ProtoNet*	0.7868 ± 0.0222	0.9299 ± 0.0099	0.1123 ± 0.0319	0.4636 ± 0.0437	0.5374 ± 0.0486	0.4819 ± 0.0449
	ProtoNet*(10%)	0.7578 ± 0.0288	0.9132 ± 0.0168	0.1340 ± 0.0099	0.4200 ± 0.0324	0.4838 ± 0.0423	0.4384 ± 0.0355
	FADS	$\mathbf{0.9178 \pm 0.0214}$	$\mathbf{0.9751 \pm 0.0080}$	$\mathbf{0.0375 \pm 0.0180}$	$\mathbf{0.5641 \pm 0.0313}$	$\mathbf{0.5770 \pm 0.0367}$	$\mathbf{0.5659 \pm 0.0348}$
IDS2018	ProtoNet	0.6086 ± 0.0643	0.8866 ± 0.0281	0.4524 ± 0.2453	0.7026 ± 0.0561	0.8373 ± 0.0655	0.7349 ± 0.0431
	ProtoNet*	0.5751 ± 0.0713	0.8708 ± 0.0403	0.4321 ± 0.3258	0.6824 ± 0.0593	0.8283 ± 0.0649	0.7138 ± 0.0523
	ProtoNet*(10%)	0.4825 ± 0.0818	0.8053 ± 0.0687	0.4438 ± 0.2178	0.5313 ± 0.0822	0.6371 ± 0.1247	0.5474 ± 0.0880
	FADS	$\mathbf{0.6791 \pm 0.0279}$	$\mathbf{0.9249 \pm 0.0134}$	$\mathbf{0.2380 \pm 0.1108}$	$\mathbf{0.7659 \pm 0.0345}$	$\mathbf{0.8839 \pm 0.0206}$	$\mathbf{0.7926 \pm 0.0197}$
CERT	ProtoNet	0.1818 ± 0.0314	0.6605 ± 0.0532	0.8839 ± 0.0631	0.3494 ± 0.0544	0.5341 ± 0.0666	0.3595 ± 0.0606
	ProtoNet*	0.1747 ± 0.0194	0.6695 ± 0.0392	0.7390 ± 0.0908	0.3380 ± 0.0367	0.6184 ± 0.0574	0.3525 ± 0.0434
	ProtoNet*(10%)	0.1892 ± 0.0267	0.6880 ± 0.0400	0.7192 ± 0.1043	0.3671 ± 0.0504	0.6953 ± 0.0438	0.3905 ± 0.0589
	FADS	$\mathbf{0.2480 \pm 0.0260}$	$\mathbf{0.7286 \pm 0.0286}$	$\mathbf{0.7036 \pm 0.1119}$	$\mathbf{0.4172 \pm 0.0287}$	$\mathbf{0.7187 \pm 0.0399}$	$\mathbf{0.4736 \pm 0.0344}$

FADS with and without Data Selection. We further evaluate the performance improvement of FADS over the simple prototypical networks without reinforced data selection. We consider three settings as baselines. First, we train a prototypical network on the initially labeled dataset, denoted as *ProtoNet*. Second, after we train the prototypical network on the initial dataset, we apply the network to label the unlabeled dataset $\mathcal{U}$ and use the whole weakly-labeled dataset $\mathcal{W}$ as the augmentation set ($\mathcal{A} \leftarrow \mathcal{W}$) to retrain the prototypical network. We evaluate the performance of the prototypical network after re-training on the test set, denoted as *ProtoNet**. The third one is similar to ProtoNet*, but we retrain the prototypical network based on samples in $\mathcal{W}$ having top 10% of the highest probabilities in each class, which means high confidence belonging to a class, denoted as *ProtoNet* (10%)*. We evaluate three models for anomaly detection and classification. For the anomaly classification, we report the macro precision, recall, and F1 score here.

Table 4.4 shows that FADS achieves unanimously better performances over baselines on three datasets for both anomaly detection and classification tasks. Meanwhile, we can notice that in general, both ProtoNet* and ProtoNet* (10%) cannot beat regular ProtoNet, which shows that simply using weakly labeled samples for training will hurt the performance due to incorrect labels. Furthermore, FADS still has the largest performance gains on the CERT dataset in terms of macro-F1 for classification, which shows the criticalness of augmenting the training set with reliable samples in the case of limited samples. Another advantage of FADS is that it can have a lower standard deviation compared with traditional prototypical networks that do not have the augmentation dataset from the reinforced data selection.

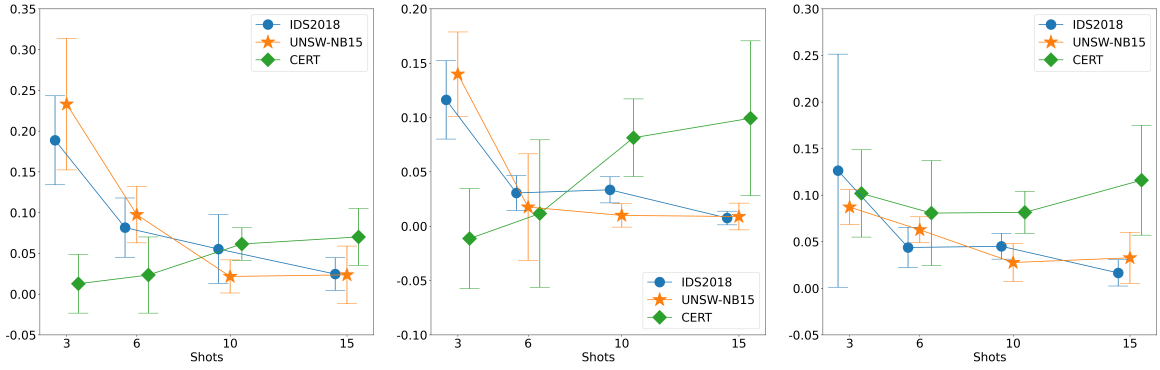
Performance of FADS with Different Numbers of Initially Labeled Anomalies. Table 4.4 indicates that the reinforced data selection strategy for training data augmentation can improve the performance of anomaly detection and classification. We then evaluate the performance under the settings that different numbers of labeled anomalies are available at the beginning of the training phase. Figure 4.2 shows the experimental results in terms of AUC-PR, AUC-ROC (anomaly detection) and Macro-F1 (anomaly classification). We can notice that on IDS2018 and UNSW-NB15 datasets, with more labeled samples available,



(a) AUC-PR (b) AUC-ROC (c) Macro-F1

Fig. 4.2: The performance of FADS with different numbers of initially labeled anomalies in one class. (a),(b) for anomaly detection and (c) for anomaly classification

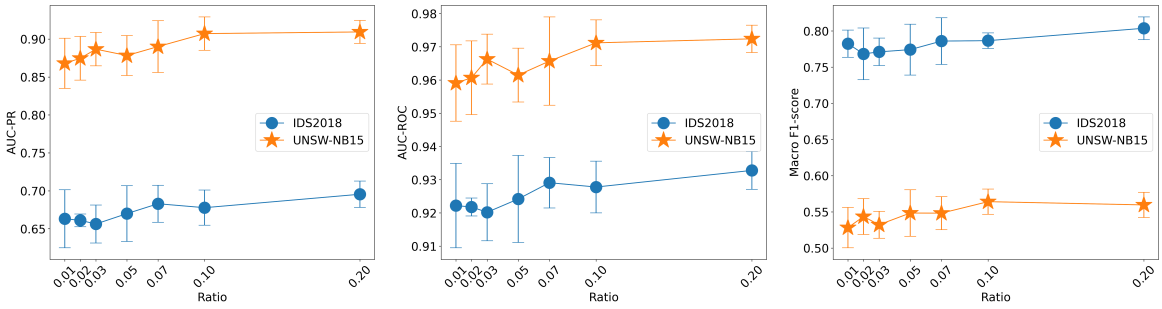
the performance of FADS keeps improving. Furthermore, the standard deviations are also decreasing with more available samples on IDS2018 and UNSW-NB15. It means with more anomalies available initially, the performance of anomaly detection and classification will become more stable. The performance of anomaly detection on CERT dataset also basically follows the similar trend, i.e., more available anomalies leading to better performance. On the other hand, the macro-f1 scores for anomaly classification keep stable with more samples. This could be because the fine-grained classification task on CERT requires more anomalies to further improve the performance due to its difficulty.



(a) AUC-PR (b) AUC-ROC (c) Macro-F1

Fig. 4.3: The performance gains of FADS over ProtoNet with different numbers of initially labeled anomalies.

Performance Gains of FADS over ProtoNet with Different Numbers of Initially Labeled Anomalies. Figure 4.3 shows that, on IDS2018 and UNSW-NB15 datasets, the performance gains compared with ProtoNet without reinforced data selection is decreasing when more labeled samples are available. This indicates that once prototypical networks have sufficient samples for training, incorporating more samples will not change the results significantly. However, different from the results on UNSW-NB15 and IDS2018, the performance gains on CERT increase when the number of available anomalies in each class increases. It could be because three samples are not sufficient to train prototypical networks for insider threat detection due to its difficulty. As a result, the weakly-labeled samples are full of noise, and no correctly labeled samples are available for re-training. Meanwhile, we can observe that FADS always achieves positive gains over ProtoNet.



(a) AUC-PR (b) AUC-ROC (c) Macro F1 score

Fig. 4.4: Performance of FADS with various anomaly ratios in the unlabeled set.

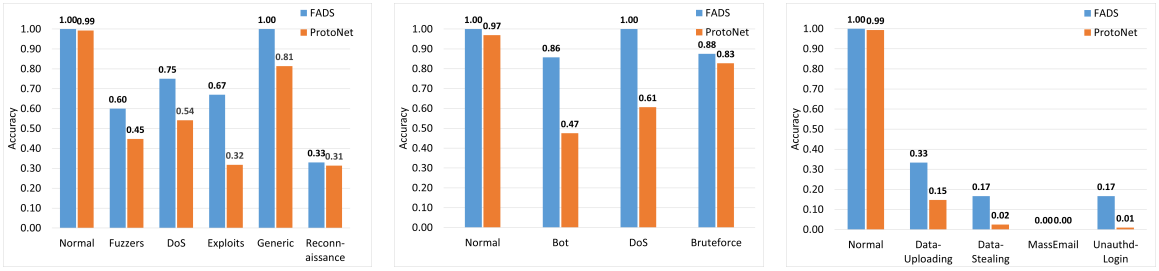
Performance of FADS with Various Anomaly Ratios in the Unlabeled Dataset. For the UNSW-NB15 and IDS2018 datasets, we set the ratio of anomalies in the unlabeled dataset $\mathcal{U}$ as 0.1 by default. We further tune this ratio to check the performance change of FADS. Note that because of limited labeled anomalies, we do not conduct this experiment on CERT. Figure 4.4 shows the results of different ratios of anomalies in $\mathcal{U}$ from 0.01 to 0.2. First, we can observe that with more anomalies in $\mathcal{U}$, the performance of FADS on anomaly detection and classification keeps improving. Especially, for the IDS2018 dataset, in terms of AUC-PR and macro F1 score, the performance of FADS keeps improving with

more anomalies in the unlabeled dataset. For the UNSW-NB15 dataset, the performance of FADS is improved when the ratios of anomalies increase from 0.01 to 0.1 and keeps stable even we include more anomalies in the unlabeled dataset. This could be because for the UNSW-NB15 dataset, the 10% of anomalies already cover the majority of anomaly types, and adding more anomalies in the unlabeled dataset can only reduce the standard deviation.

Table 4.5: Impact of each reward component

		AUC-PR	AUC-ROC	Macro F1
r^d	UNSW-NB15	0.9105 ± 0.0266	0.9721 ± 0.0087	0.5591 ± 0.0333
	IDS2018	0.6787 ± 0.0372	0.9243 ± 0.0166	0.7867 ± 0.0328
	CERT	0.2554 ± 0.0265	0.7408 ± 0.0294	0.4671 ± 0.0397
r^c	UNSW-NB15	0.9030 ± 0.0345	0.9692 ± 0.0126	0.5604 ± 0.0277
	IDS2018	0.6787 ± 0.0360	0.9207 ± 0.0164	0.7897 ± 0.0243
	CERT	0.2260 ± 0.0230	0.7033 ± 0.0233	0.4685 ± 0.0343
Combine	UNSW-NB15	0.9178 ± 0.0214	0.9751 ± 0.0080	0.5659 ± 0.0348
	IDS2018	0.6791 ± 0.0279	0.9249 ± 0.0134	0.7926 ± 0.0197
	CERT	0.2480 ± 0.0260	0.7286 ± 0.0286	0.4736 ± 0.0344

Ablation Study on Reward Components. The reward function defined in Equation 4.3 consists of two parts, rewards for anomaly classification and detection. In this experiment, we check the performance of FADS if we keep only one reward component. Table 4.5 indicates that incorporating the reward for both anomaly detection and classification can improve the performance slightly compared with the model with only one reward component. Meanwhile, FADS with one reward component (shown in Table 4.5) already achieves better performance than ProtoNet (shown in Table 4.4).



(a) UNSW-NB15 (b) IDS2018 (c) CERT

Fig. 4.5: Accuracy of the data selection agent

Accuracy of the Reinforced Data Selection Agent. We further evaluate whether the reinforced data selection agent is able to select the high-quality samples from the weakly-labeled dataset $\mathcal{W}$. We calculate the accuracy based on all selected samples when the training procedure for data selection is finished. Figure 4.5 shows the results, where the blue bar indicates the accuracy of the FADS agent while the orange bar indicates the accuracy of initial prototypical networks. We can notice that the initial prototypical networks cannot achieve good performance on anomaly classification, which means the weakly-labeled dataset $\mathcal{W}$ consists of a lot of mislabeled samples. However, the samples selected from the agent have much higher chances to be correct, indicating that the agent is able to identify high quality weakly-labeled samples from $\mathcal{W}$. Therefore, using the selected samples to augment the training dataset can improve the performance of the few-shot anomaly detection model.

4.4 FADScr

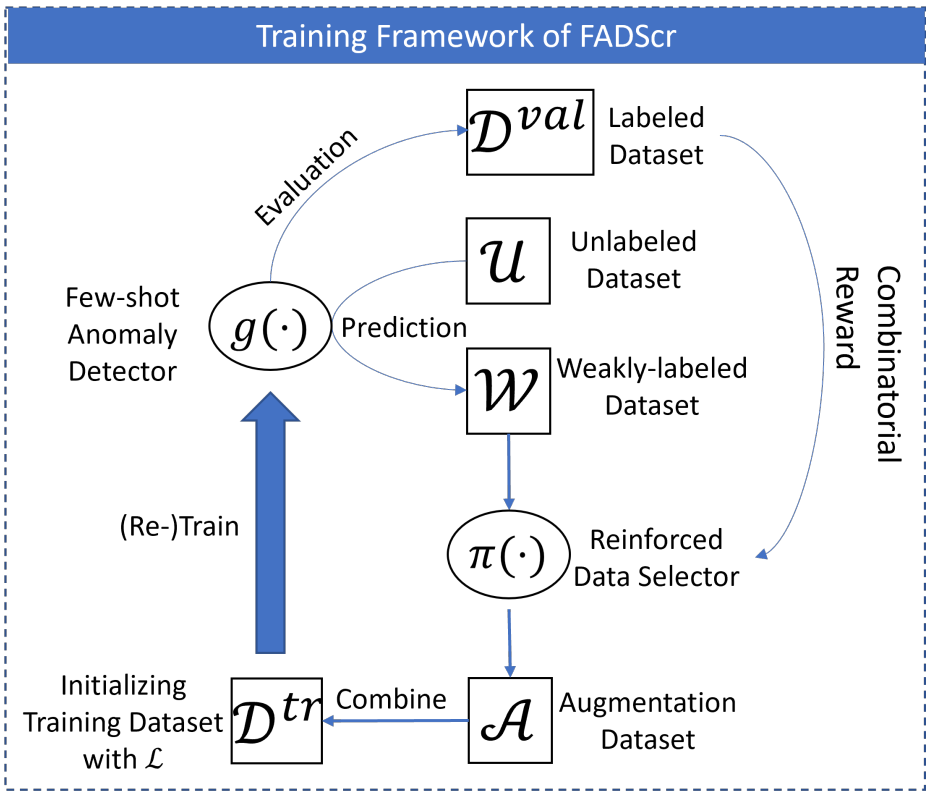


Fig. 4.6: The training framework of FADScr

4.4.1 Framework Overview

We further extend FADS to a few-shot anomaly detection and classification model through reinforced data selection with a combinatorial reward (FADScr), which is able to gradually enhance the performance of the few-shot learning model by exploiting the unlabeled dataset $\mathcal{U}$.

In particular, FADScr first trains a few-shot learning model on an initial training dataset with only labeled samples $\mathcal{D} = \mathcal{L}$. Then, FADScr iteratively improves the model by selecting samples from unlabeled dataset $\mathcal{U}$ to augment the training dataset. We employ the reinforcement learning technique for data selection. In each training iteration, we first use the current few-shot model to predict the label $\hat{y}_j$ for each sample x_j in $\mathcal{U}$, thus producing a weakly-labeled dataset $\mathcal{W} = \{(x_j, \hat{y}_j)\}_{j=1}^{N_U}$. Then, we train a reinforcement learning agent to select the weakly-labeled samples that have high chances to be accurate into an augmentation set $\mathcal{A} = \{(x_j, \hat{y}_j)|a_j = 1\}_{j=1}^{N_U}$, where $a_j \in \{0, 1\}$ indicates whether the sample $(x_j, \hat{y}_j)$ is selected by the agent or not. We combine the augmentation dataset $\mathcal{A}$ with the existing training dataset to compose the new training dataset, i.e., $\mathcal{D} = \mathcal{D} \cup \mathcal{A}$, and then re-train the few-shot model on the new training dataset. We expect that the performance of the few-shot model will be improved with those augmented samples. Therefore, as the training iteration moves forward, the few-shot anomaly detection model can be improved progressively. The overview of FADScr is shown in Figure 4.6.

4.4.2 Reinforced Data Selection

As the labeled dataset $\mathcal{L}$ only has a very small number of anomalies, the performance of the initial prototypical network trained on $\mathcal{L}$ still has room for improvement if more training samples can be incorporated. To this end, we propose to exploit the unlabeled dataset $\mathcal{U}$. We first use the current prototypical network to predict the labels of samples in $\mathcal{U}$ and get a weakly-labeled dataset $\mathcal{W}$, which consists of both correctly and incorrectly labeled samples. If we simply adopt $\mathcal{W}$ to augment the training set, due to the noisy supervised signals, the performance of the prototypical network could be worse. Hence, we propose a reinforcement learning-based data selection method to select samples from $\mathcal{W}$ based on their reliability

to compose an augmentation set $\mathcal{A}$. Once the reinforced data selection agent is capable of selecting the correctly labeled samples to augment the training set, the performance of the few-shot learner can be improved. Therefore, the reward to the data selection agent is designed based on the performance of the few-shot learner on an unobserved validation set. The key components of the reinforcement learning framework are described below.

State. For each state s_j with weakly-labeled sample $(x_j, \hat{y}_j) \in \mathcal{W}$, its state representation $\mathbf{s}_j$ is defined as the concatenation of the embedding vector $\mathbf{x}_j = g(x_j)$ and the distance value d_j to the closest prototype, i.e., $\mathbf{s}_j = [\mathbf{x}_j, d_j]$.

Action. The data selection agent then needs to take an action whether to select this sample $(x_j, \hat{y}_j)$ into the augmentation set $\mathcal{A}$ based on the state representation $\mathbf{s}_j$. In specific, $a_j = 0$ means the sample $(x_j, \hat{y}_j)$ will be rejected, while $a_j = 1$ indicates the sample will be selected and added to the augmentation set $\mathcal{A}$.

Policy Network. The data selection agent makes decisions about whether to select a weakly-labeled sample based on a policy network $\pi_\theta(\cdot)$. We adopt a neural network to parameterize the policy network that takes the state representation as input and outputs the probability of the action, $p(a_j|s_j) = \pi_\theta(\mathbf{s}_j)$. The action a_j is then sampled based on $p(a_j|s_j)$.

Rewards. The policy network is trained with guidance from the reward function. We define the reward based on the performance of the few-shot learning model on an unseen validation set. As in our task, we aim to detect anomalies as well as classify abnormal samples into different classes. Therefore we design the reward function based on the performance of anomaly detection and classification, i.e., an anomaly detection reward r^d and a classification reward r^c. Specifically, we adopt the F1 score as the metric to evaluate the performance of anomaly detection and the macro F1 score over all classes to evaluate the performance of classification. In our scenario, if wrongly-labeled samples are selected to compose the training dataset, we can expect that the performance of the few-shot learning model will be damaged. On the other hand, if the samples with the correct labels are selected, the performance of the model can be improved. Hence, the reward is designed based on the

performance change after the model is trained on an augmentation set.

Moreover, inspired by active learning with the strategy of labeling uncertain samples, we expect the agent can select the samples with high uncertainties to the augmentation set so that the performance of the few-shot learning model for anomaly detection and classification can be further improved. Here, the uncertainty of a sample is quantified by its predicted probabilities $p(y = k|\mathbf{x})$. When the predicted probability of a sample for the true class is less than a threshold, i.e., $p(y = k|\mathbf{x}) < \tau$, we consider the sample as a hard sample. Otherwise, the samples are easy samples. If the agent only selects easy samples into the augmentation set, for prototypical networks, the center $\mathbf{c}_k$ of the class (defined in Eq. **??**) cannot be updated significantly, meaning that the hard samples still have low predicted probabilities. Therefore, we also design a reward r^p based on probability changes on hard samples after the model is trained on an augmentation set.

Specifically, in each episode, we first calculate the F1, macro F1 scores, and predicted probabilities $p(y = k|\mathbf{x})$ on validation samples by the current prototypical network. We denote the prototypical network at the beginning of each episode as $g_0(\cdot)$ and the corresponding F1, macro F1 scores achieved by $g_0(\cdot)$ as $F1_0$ and $MarcoF1_0$. Meanwhile, to evaluate whether the prototypical network can reduce the prediction uncertainties, we also select hard samples, i.e., $p(y = k|\mathbf{x}) < \tau$, from the validation set and use the mean probabilities as the uncertainty score, denoted as Pr_0.

Then, we work on improving $g_0(\cdot)$ by composing an augmentation set. To this end, first, we randomly sample a set of batches $\mathcal{B} = \{B_l\}_{l=1}^{L}$ from $\mathcal{W}$, where each batch B_l consists of a number of samples. Given a batch B_l, for each sample in B_l, we sample an action a_j (select to the augmentation set or not) based on the policy $p(a|s_j) = \pi_\theta(\mathbf{s}_j)$. Then, we can compose the augmentation set $\mathcal{A}_l$ from the batch B_l. After combining $\mathcal{A}_l$ with $\mathcal{D}$, we update the prototypical network, denoted as $g_l(\cdot)$, and further evaluate $g_l(\cdot)$ on a validation set to derive F1, macro F1 scores, and probabilities, denoted as $F1_l$, $MacroF1_l$, Pr_l.

Finally, the reward for the agent in a batch can be calculated as the differences of F1 scores and probabilities, $r_l^d = F1_l - F1_0$, $r_l^c = MacroF1_l - MacroF1_0$, and $r_l^p = Pr_l - Pr_0$.

The overall reward function in a batch is formulated as:

$$r_l = r_l^d + \alpha \cdot r_l^c + \beta \cdot r_l^p, \tag{4.9}$$

where α and β are hyperparameters to balance anomaly detection, classification tasks, and the reward for reducing uncertain samples.

Optimization. The data selection agent is trained based on the actor-critic algorithm [2]. The output of the actor is the probability of two actions (select or not), while the output of the critic is the predicted reward value based on the current state. Both actor and critic are parameterized by feed-forward neural networks. The goal of the agent is to maximize the rewards, which is defined as:

$$\mathcal{J}(\theta) = \mathbb{E}_{\pi_\theta}[r_l]. \tag{4.10}$$

The parameter θ in policy network π_θ is trained based on policy gradient [107]:

$$\theta \leftarrow \theta + \eta \bigtriangledown_\theta \mathcal{J}(\theta), \tag{4.11}$$

where η indicates the learning rate. The gradient for a batch of weakly-labeled samples can be approximated by [2]

$$\bigtriangledown_\theta \mathcal{J}(\theta) = \frac{1}{|B_l|} \sum_{j=1}^{|B_l|} \bigtriangledown_\theta log \pi_\theta(\mathbf{s}_j)(r_l - V_\varphi(\mathbf{s}_j)), \tag{4.12}$$

where $|B_l|$ indicates the number of samples in a batch B_l; $V_\varphi(\mathbf{s}_j)$ is the expected reward from a critic network $V_\varphi(\cdot)$ parameterized by φ. The structure of the critic network is similar to the policy network with the last layer as a regression function. The critic network is designed to estimate the expected reward and hence updated according to the cumulative difference between the real reward r_l and the predicted value $V_\varphi(\mathbf{s}_j)$,

$$\mathcal{L}(\varphi) = \frac{1}{|B_l|} \sum_{j=1}^{|B_l|} |r_l - V_\varphi(\mathbf{s}_j)|. \tag{4.13}$$

The parameters φ in $V_\varphi(s_j)$ can be optimized by:

$$\varphi = \varphi - \eta' \bigtriangledown_\varphi \mathcal{L}(\varphi), \tag{4.14}$$

where η' indicates the learning rate for updating the critic network.

Algorithm 2: The training process of FADScr

Input : Labeled set $\mathcal{L}$, unlabeled set $\mathcal{U}$
Output: Prototypical network $g(\cdot)$

Initialize an augmentation set $\mathcal{A} \leftarrow \emptyset$
Initialize a training dataset $\mathcal{D} \leftarrow \mathcal{L}$
for $t = 0$ **to** T **do**
    Split $\mathcal{D}$ to $\mathcal{D}^{tr}$ and $\mathcal{D}^{val}$
    $\mathcal{D}^{tr} \leftarrow \mathcal{D}^{tr} \cup \mathcal{A}$
    Train prototypical network $g(\cdot)$ on $\mathcal{D}^{tr}$
    Predict labels on $\mathcal{U}$ and get $\mathcal{W}$ based on $g(\cdot)$
    $\mathcal{A}_{ep} \leftarrow$ Data_Selection ($g(\cdot)$, $\mathcal{W}$, $\mathcal{D}^{tr}$, $\mathcal{D}^{val}$)
    Remove selected samples $\mathcal{U} \leftarrow \mathcal{U}/\mathcal{A}_{ep}$
    Increase the augmentation dataset $\mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{A}_{ep}$
Train prototypical network $g(\cdot)$ on $\mathcal{D} \cup \mathcal{A}$
return prototypical network $g(\cdot)$
Function `Data_Selection`($g_0(\cdot)$,$\mathcal{W}$,$\mathcal{D}^{tr}$,$\mathcal{D}^{val}$)`:`
    Apply $g_0(\cdot)$ on $\mathcal{D}^{val}$ to derive $(F1_0, MacroF1_0, Pr_0)$
    Generate subsets $\mathcal{B} = \{B_l\}_{l=1}^{L}$ from $\mathcal{W}$
    $\mathcal{R} = \emptyset$, $\mathcal{A}_{ep} = \emptyset$
    foreach $B_l \in \mathcal{B}$ **do**
        Agent selects samples from B_l to generate $\mathcal{A}_l$
        $\mathcal{D}_l^{tr} \leftarrow \mathcal{D}^{tr} \cup \mathcal{A}_l$
        Train prototypical network $g_l(\cdot)$ on $\mathcal{D}_l^{tr}$
        Apply $g_l(\cdot)$ on $\mathcal{D}^{val}$ to derive $(F1_l, MacroF1_l, Pr_l)$
        Compute the reward r_l based on Eq. (4.9)
        $\mathcal{R} = \mathcal{R} \cup \{r_l\}$
        Update the policy network π_θ based on Eq. (4.11)
        Update the critic network V_φ based on Eq. (4.14)
    $l \leftarrow \arg\max \mathcal{R}$
    if $F1_l > F1_0$, $MacroF1_l > MacroF1_0$, *and* $Pr_l > Pr_0$ **then**
        $\mathcal{A}_{ep} \leftarrow \mathcal{A}_l$
    return $\mathcal{A}_{ep}$

4.4.3 Training Details

General Training Process. Algorithm 2 shows the pseudo-code of the training process. Given a small labeled dataset $\mathcal{L}$ and an unlabeled dataset $\mathcal{U}$, we first initialize an augmentation dataset $\mathcal{A}$ as an empty set (Line 1) and a training dataset $\mathcal{D}$ as the small labeled dataset $\mathcal{L}$ (Line 2). In each episode, we split $\mathcal{D}$ into two parts, $\mathcal{D}^{tr}$ as the real training set and $\mathcal{D}^{val}$ as the unobserved validation set (Line 4), combine $\mathcal{D}^{tr}$ with $\mathcal{A}$ as the new training set (Line 5), and use it to train the prototypical network $g(\cdot)$ (Line 6). We deploy the prototypical network to predict labels of samples in $\mathcal{U}$ and get a weakly labeled data set $\mathcal{W}$ (Line 7). We then apply a reinforced data selection algorithm to generate a batch of samples as the augmentation set $\mathcal{A}_{ep}$ from $\mathcal{W}$ (Line 8). After obtaining $\mathcal{A}$, we remove $\mathcal{A}_{ep}$ from the unlabeled dataset $\mathcal{U}$ (Line 9) and add $\mathcal{A}_{ep}$ to $\mathcal{A}$ to form a new augmentation set for next iteration (Line 10).

With the increasing of training episodes, the size of the augmentation dataset $\mathcal{A}$ keeps increasing and the performance of the prototypical network becomes better. After finishing all the training iterations, we further combine labeled dataset $\mathcal{D}$ with the augmentation dataset $\mathcal{A}$, retrain the prototypical network from scratch based on the combined dataset (Line 11), and deploy it for anomaly detection and classification.

Reinforced Data Selection. The key step for improving the performance of the prototypical network is the reinforced data selection algorithm (Lines 13 – 29), which is to select the reliable weakly-labeled samples from $\mathcal{W}$ to augment the training set. We first apply the current prototypical network $g_0(\cdot)$ on the validation set $\mathcal{D}^{val}$ to get F1, macro F1 scores, and sample prediction probabilities $(F1_0, MacroF1_0, Pr_0)$ as the baseline performance (Line 14). We then derive a set of sample batches $\mathcal{B} = \{B_l\}_{l=1}^{L}$ from $\mathcal{W}$.

Then, for each batch B_l, we select samples based on the prediction of the policy network $p(a_j|s_j) = \pi_\theta(\mathbf{s}_j)$ and generate $\mathcal{A}_l$ with selected samples. After getting the augmented training set $\mathcal{D}_l^{tr}$ by combining $\mathcal{D}^{tr}$ with $\mathcal{A}_l$ (Line 19), we train the prototypical network $g_l(\cdot)$ on $\mathcal{D}_l^{tr}$ again (Line 20) and apply $g_l(\cdot)$ on $\mathcal{D}^{val}$ to derive F1, macro F1 scores, and sample prediction probabilities $(F1_l, MacroF1_l, Pr_l)$, (Line 21). The reward is then calcu-

lated based on the difference between $(F1_l, MacroF1_l, Pr_l)$ and $(F1_0, MacroF1_0, Pr_0)$. We further update the policy network π_θ and the critic network V_φ (Lines 24 – 25).

After going through all batches, we get a set of rewards $\mathcal{R} = \{r_l\}_{l=0}^{L}$. We only select $\mathcal{A}_l$ leading to the maximum reward r_l in $\mathcal{R}$ (Line 26) and also ensure that the prototypical network $g_l(\cdot)$ trained on $\mathcal{D}_l^{tr}$ can get better performance in terms of F1, Macro F1 scores, and sample prediction probabilities compared with the original one $g_0(\cdot)$ (Lines 27-28). Then, we consider $\mathcal{A}_l$ as the augmentation set $\mathcal{A}_{ep}$ at the current episode (Line 29).

Warm-up Phase. In the early stage of training, the data selection agent takes actions based on randomly initialized parameters. The selected samples could include misclassified instances due to the uncertainty of the policy network. Then, $\mathcal{D}^{tr}$ could contain mislabeled samples, which could damage the performance of FADScr in the long turn. Hence, we set up a warm-up phase for the training. After the warm-up phase, all the samples selected during the warm-up phase will be removed from $\mathcal{D}$ and added back to $\mathcal{U}$. The agent will select samples from scratch again. We expect the performance of the data selection agent will become stable with the convergence of parameters during the warm-up phase and can select reliable weakly-labeled samples after that. In our experiments, the warm-up phase is 10 episodes.

4.4.4 Implementation Details

For UNSW-NB15 and IDS2018 datasets, because the input samples are multidimensional data, we adopt a feedforward neural network with three fully connected layers as the implementation of prototypical network $g(\cdot)$, where the dimension of output embedding is 128. For the CERT dataset, because the input samples are sequential data about user activities in a session, we adopt a GRU neural network [118] to map sequences into an embedding space with a dimension of 64. For all three datasets, as we only have 10 labeled samples in each anomaly class in the initial training set $\mathcal{L}_A$, we initially assign 6 samples to $\mathcal{D}^{tr}$ to train the prototypical network with 3 samples as the support set $\mathcal{S}$ and 3 samples as the query set $\mathcal{Q}$ and the rest 4 samples to $\mathcal{D}^{val}$ for evaluation. In the experiments, we set α and β in Equation 4.9 as $\alpha = 1$ and $\beta = 0.01$.

In the implementation of actor-critic reinforcement learning, both actor and critic are three-layer feedforward neural networks. In the training stage, we set the number of training episodes $T = 30$, while in each episode, we generate a set of sample batches $\{B_l\}_{l=1}^{L}$. For each batch B_l, we select samples from each predicted class in $\mathcal{W}$ based on their closeness to the prototype representations. Especially, for UNSW-NB15 and IDS2018 datasets, in each batch, we select the top 10 samples that are close to the corresponding prototype representation from each class as well as 50 samples that are not close to any prototype representations as hard samples, leading to the sizes of a batch are $|B_l| = 110$ for UNSW-NB15 and $|B_l| = 90$ for IDS2018, respectively. For the CERT dataset, we select the closest 1 sample from each class and 5 hard samples, leading to the batch size $|B_l| = 10$. The **code and datasets** used in the experiments are available online [2].

4.4.5 Experimental Results

Anomaly Detection. We first evaluate the performance of FADScr for anomaly detection. In particular, after the prototypical network predicts each anomaly into a specific anomaly class, we consider all detected samples in anomaly classes as anomalies. We compare FADScr for anomaly detection with two traditional one-class classification models (OCSVM and iForest), two positive-unlabeled learning models (LRPU and PUSB), and two advanced deep anomaly detection models (DeepSVDD and DeepSAD). As shown in Table 4.6, FADScr achieves better performance than all baselines with a large margin on all three datasets. Especially, although DeepSAD as a semi-supervised model also uses both labeled and unlabeled datasets, FADScr significantly outperforms DeepSAD in most cases and has a much smaller standard deviation. It means actively identifying the potential anomalies in the unlabeled dataset can improve the performance of anomaly detection. Meanwhile, DeepSAD still outperforms other baselines, which shows the advantage of using a small set of labeled samples for anomaly detection. LRPU as a PU learner achieves better performance than three one-class models (OCSVM, iForest, and DeepSVDD) on IDS2018 and CERT

[2] https://github.com/hanxiao0607/FADScr

]

Table 4.6: Results on anomaly detection (mean ± std.). ↑ indicates larger value is better; ↓ indicates lower value is better.

		UNSW-NB15	IDS2018	CERT
AUC-PR ↑	OCSVM	0.4999 ± 0.0029	0.3069 ± 0.0115	0.1161 ± 0.0170
	iForest	0.7337 ± 0.0537	0.2914 ± 0.0151	0.1321 ± 0.0198
	LRPU	0.6559 ± 0.0084	0.4365 ± 0.0444	0.1380 ± 0.0520
	PUSB	0.6119 ± 0.0246	0.3774 ± 0.0158	0.1147 ± 0.0000
	DeepSVDD	0.6088 ± 0.0359	0.4704 ± 0.1126	0.1192 ± 0.0068
	DeepSAD	0.8103 ± 0.1958	0.5945 ± 0.1064	0.1304 ± 0.0176
	FADScr	$\mathbf{0.9249 \pm 0.0206}^{*}$	$\mathbf{0.7121 \pm 0.0592}^{**}$	$\mathbf{0.2531 \pm 0.0397}^{**}$
AUC-ROC ↑	OCSVM	0.7499 ± 0.0029	0.6517 ± 0.0187	0.5671 ± 0.0436
	iForest	0.8991 ± 0.0315	0.5489 ± 0.0280	0.5937 ± 0.0572
	LRPU	0.7420 ± 0.0063	0.6338 ± 0.0289	0.5432 ± 0.0482
	PUSB	0.7089 ± 0.0184	0.5953 ± 0.0103	0.5000 ± 0.0000
	DeepSVDD	0.7436 ± 0.0316	0.7128 ± 0.0905	0.5279 ± 0.0204
	DeepSAD	0.9024 ± 0.1497	0.7840 ± 0.0896	0.5601 ± 0.0558
	FADScr	$\mathbf{0.9783 \pm 0.0066}^{*}$	$\mathbf{0.9299 \pm 0.0219}^{**}$	$\mathbf{0.7406 \pm 0.0357}^{**}$
FPR (95% TPR) ↓	OCSVM	0.9941 ± 0.0007	0.9149 ± 0.0140	0.9138 ± 0.0483
	iForest	0.2093 ± 0.0660	0.9549 ± 0.0452	0.7526 ± 0.1303
	LRPU	0.3738 ± 0.0675	0.5236 ± 0.1379	0.8827 ± 0.0643
	PUSB	0.9842 ± 0.0328	0.8691 ± 0.1974	0.9522 ± 0.0107
	DeepSVDD	0.6664 ± 0.1704	0.7159 ± 0.2406	0.8619 ± 0.0678
	DeepSAD	0.2222 ± 0.3990	0.4326 ± 0.3343	0.6997 ± 0.0938
	FADScr	$\mathbf{0.0333 \pm 0.0133}^{*}$	$\mathbf{0.1659 \pm 0.0986}^{**}$	$\mathbf{0.6849 \pm 0.0854}$

Significantly outperforms DeepSAD at the: * 0.05 and ** 0.01 level, paired t-test.

datasets, meaning that in some cases, leveraging the unlabeled set can improve the performance. However, due to the limited anomalies (positive samples), both LRPU and PUSB underperform FADScr.

Table 4.7: Results on anomaly classification (mean ± std.)

Dataset	Class	Precision			Recall			F1		
		SVM	MLP	FADScr	SVM	MLP	FADScr	SVM	MLP	FADScr
UNSW-NB15	Normal	0.8135 ± 0.0089	0.8399 ± 0.0099	0.9979 ± 0.0038	0.9873 ± 0.0406	0.9874 ± 0.0358	0.9607 ± 0.0129	0.8957 ± 0.0055	0.9110 ± 0.0056	$\mathbf{0.9789 \pm 0.0066}$
	Fuzzers	0.3858 ± 0.0791	0.3538 ± 0.0720	0.3824 ± 0.0469	0.1121 ± 0.0790	0.1244 ± 0.0476	0.4335 ± 0.0962	0.1502 ± 0.0268	0.1726 ± 0.0310	$\mathbf{0.4036 \pm 0.0635}$
	DoS	0.5341 ± 0.0389	0.5002 ± 0.0525	0.4602 ± 0.0726	0.3729 ± 0.1242	0.3335 ± 0.1077	0.4532 ± 0.0917	0.4302 ± 0.0933	0.3884 ± 0.0839	$\mathbf{0.4539 \pm 0.0744}$
	Exploits	0.3946 ± 0.0936	0.4224 ± 0.0748	0.3749 ± 0.0650	0.1862 ± 0.0849	0.2513 ± 0.0635	0.3900 ± 0.0769	0.2336 ± 0.0432	0.3048 ± 0.0599	$\mathbf{0.3782 \pm 0.0591}$
	Generic	0.9762 ± 0.0197	0.9579 ± 0.0247	0.9217 ± 0.0912	0.9666 ± 0.0253	0.9662 ± 0.0161	0.9649 ± 0.0389	$\mathbf{0.9710 \pm 0.0153}$	0.9618 ± 0.0148	0.9414 ± 0.0640
	Reconnaissance	0.3561 ± 0.1102	0.3865 ± 0.0864	0.3212 ± 0.0614	0.1392 ± 0.0976	0.2258 ± 0.0751	0.3631 ± 0.0710	0.1740 ± 0.0404	0.2696 ± 0.0497	$\mathbf{0.3391 \pm 0.0604}$
	macro-average	0.5767 ± 0.0281	0.5768 ± 0.0251	0.5764 ± 0.0290	0.4607 ± 0.0463	0.4815 ± 0.0258	0.5942 ± 0.0303	0.4758 ± 0.0199	0.5014 ± 0.0190	$\mathbf{0.5825 \pm 0.0288}^{**}$
IDS2018	Normal	0.9904 ± 0.0180	0.9945 ± 0.0098	0.9923 ± 0.0172	0.5657 ± 0.1111	0.8392 ± 0.0363	0.8853 ± 0.0391	0.6941 ± 0.0454	0.9019 ± 0.0293	$\mathbf{0.9350 \pm 0.0164}$
	Bot	0.6482 ± 0.2551	0.5348 ± 0.1855	0.5134 ± 0.1452	0.7578 ± 0.2499	0.9477 ± 0.1526	0.9711 ± 0.1110	0.6311 ± 0.0961	$\mathbf{0.6538 \pm 0.0846}$	0.6513 ± 0.0635
	DoS	0.1934 ± 0.0456	0.6615 ± 0.1765	0.8520 ± 0.1090	0.6914 ± 0.1263	0.7047 ± 0.0997	0.8136 ± 0.0579	0.3201 ± 0.1082	0.6796 ± 0.1179	$\mathbf{0.8258 \pm 0.0429}$
	Bruteforce	0.4814 ± 0.0965	0.6722 ± 0.0773	0.7807 ± 0.0779	0.9299 ± 0.1292	0.9629 ± 0.0886	0.8444 ± 0.1376	0.6452 ± 0.1215	0.8018 ± 0.0797	$\mathbf{0.8062 \pm 0.0903}$
	macro-average	0.5784 ± 0.0517	0.7157 ± 0.0217	0.7846 ± 0.0558	0.7362 ± 0.0759	0.8636 ± 0.0515	0.8786 ± 0.0436	0.5726 ± 0.0659	0.7593 ± 0.0452	$\mathbf{0.8046 \pm 0.0403}^{**}$
CERT	Normal	0.9599 ± 0.0188	0.9016 ± 0.0032	0.9354 ± 0.0108	0.4883 ± 0.2146	0.9464 ± 0.0207	0.8874 ± 0.0215	0.6205 ± 0.1991	$\mathbf{0.9243 \pm 0.0102}$	0.9097 ± 0.0097
	DataUploading	0.2076 ± 0.1397	0.3474 ± 0.1479	0.4684 ± 0.1665	0.8722 ± 0.0911	0.7222 ± 0.1464	0.8125 ± 0.1094	0.3161 ± 0.1713	0.4503 ± 0.1174	$\mathbf{0.5787 \pm 0.1128}$
	DataStealing	0.3283 ± 0.1198	0.0506 ± 0.1264	0.4044 ± 0.0525	0.7124 ± 0.1206	0.0100 ± 0.0306	0.4042 ± 0.1173	$\mathbf{0.4374 \pm 0.1264}$	0.0159 ± 0.0480	0.3983 ± 0.0725
	MassEmail	0.0223 ± 0.0108	0.1436 ± 0.2089	0.1111 ± 0.0546	0.6450 ± 0.2417	0.4250 ± 0.2468	0.6418 ± 0.1638	0.0428 ± 0.0202	$\mathbf{0.1949 \pm 0.1669}$	0.1813 ± 0.0724
	UnauthdLogin	0.2152 ± 0.1901	0.3456 ± 0.2120	0.5245 ± 0.1700	0.6929 ± 0.1758	0.5143 ± 0.1222	0.7524 ± 0.0935	0.2987 ± 0.2198	0.3753 ± 0.1337	$\mathbf{0.6010 \pm 0.1259}$
	macro-average	0.3467 ± 0.0784	0.3578 ± 0.0629	0.4888 ± 0.0419	0.6822 ± 0.1017	0.5236 ± 0.0521	0.6997 ± 0.0550	0.3432 ± 0.1205	0.3922 ± 0.0468	$\mathbf{0.5338 \pm 0.0388}^{**}$

Significantly outperforms MLP in terms of macro-F1 at the: * 0.05 and ** 0.01 level, paired t-test.

Anomaly Classification. FADScr can achieve fine-grained anomaly classification based

on the prototypical networks, so we further evaluate FADScr for the anomaly classification task. We compare FADScr with two classification models, SVM and MLP. Table 4.7 shows the results of anomaly classification. In short, FADScr achieves the highest macro-F1 score over all classes, meaning that FADScr can achieve the best performance for anomaly classification. Meanwhile, on the UNSW-NB15 and IDS2018 datasets, FADScr achieves the best performance in most classes in terms of F1 score compared with SVM and MLP. CERT is the most challenging dataset because of extremely limited labeled samples. On the CERT dataset, SVM predicts all anomalies as normal, while MLP has a low recall in the normal set indicating a high misclassification rate on normal samples. On the other hand, FADScr achieves a significant improvement compared with baselines on CERT. It indicates for the challenging anomaly detection task with extremely limited samples, leveraging anomalies in the unlabeled dataset is critical to boosting the performance.

Table 4.8: Performance of FADScr with or without reinforced data selection (mean ± std.)

Dataset	Approach	Anomaly Detection			Anomaly Classification		
		AUC-PR ↑	AUC-ROC ↑	FPR at 95% TPR ↓	Precision	Recall	F1
UNSW-NB15	ProtoNet	0.8522 ± 0.0424	0.9504 ± 0.0155	0.0873 ± 0.0405	0.5180 ± 0.0301	0.5503 ± 0.0359	0.5205 ± 0.0322
	ProtoNet*	0.7245 ± 0.0988	0.8865 ± 0.0796	0.1821 ± 0.1649	0.4153 ± 0.0556	0.4673 ± 0.0713	0.4262 ± 0.0634
	ProtoNet*(10%)	0.7111 ± 0.0685	0.8908 ± 0.0374	0.2153 ± 0.0862	0.4595 ± 0.0420	0.5194 ± 0.0517	0.4645 ± 0.0467
	FADS	0.9050 ± 0.0232	0.9693 ± 0.0102	0.0398 ± 0.0121	0.5357 ± 0.0841	0.5813 ± 0.0296	0.5676 ± 0.0302
	FADScr	$\mathbf{0.9249 \pm 0.0206}^{**}$	$\mathbf{0.9783 \pm 0.0066}^{**}$	$\mathbf{0.0333 \pm 0.0133}^{*}$	$\mathbf{0.5764 \pm 0.0290}$	$\mathbf{0.5942 \pm 0.0303}^{*}$	$\mathbf{0.5825 \pm 0.0288}^{**}$
IDS2018	ProtoNet	0.6894 ± 0.0433	0.9180 ± 0.0276	0.2758 ± 0.1406	0.7104 ± 0.0595	0.8396 ± 0.0588	0.7388 ± 0.0415
	ProtoNet*	0.5081 ± 0.0722	0.8208 ± 0.0669	0.4142 ± 0.2097	0.5329 ± 0.0819	0.6317 ± 0.1166	0.5471 ± 0.0767
	ProtoNet*(10%)	0.5288 ± 0.0670	0.8555 ± 0.0386	0.3360 ± 0.1740	0.6371 ± 0.0776	0.7620 ± 0.0808	0.6400 ± 0.0627
	FADS	0.6780 ± 0.0343	0.9219 ± 0.0231	0.2835 ± 0.1209	0.7657 ± 0.0403	$\mathbf{0.8800 \pm 0.0385}$	0.7902 ± 0.0275
	FADScr	$\mathbf{0.7121 \pm 0.0592}^{**}$	$\mathbf{0.9299 \pm 0.0219}^{**}$	$\mathbf{0.1659 \pm 0.0986}^{**}$	$\mathbf{0.7846 \pm 0.0558}^{*}$	0.8786 ± 0.0436	$\mathbf{0.8046 \pm 0.0403}$
CERT	ProtoNet	0.1784 ± 0.0569	0.6314 ± 0.0729	0.9777 ± 0.0637	0.3474 ± 0.0499	0.5501 ± 0.0724	0.3613 ± 0.0697
	ProtoNet*	0.1788 ± 0.0305	0.6738 ± 0.0564	0.7316 ± 0.1062	0.3418 ± 0.0443	0.6141 ± 0.1017	0.3551 ± 0.0632
	ProtoNet*(10%)	0.2367 ± 0.0347	0.7066 ± 0.0404	0.7651 ± 0.1248	0.3847 ± 0.0465	0.7160 ± 0.0725	0.4353 ± 0.0627
	FADS	0.2397 ± 0.0383	0.7161 ± 0.0451	0.7083 ± 0.1322	0.4119 ± 0.0379	$\mathbf{0.7243 \pm 0.0453}$	0.4694 ± 0.0449
	FADScr	$\mathbf{0.2531 \pm 0.0397}$	$\mathbf{0.7406 \pm 0.0357}^{*}$	$\mathbf{0.6849 \pm 0.0854}$	$\mathbf{0.4888 \pm 0.0419}^{**}$	0.6997 ± 0.0550	$\mathbf{0.5338 \pm 0.0388}^{**}$

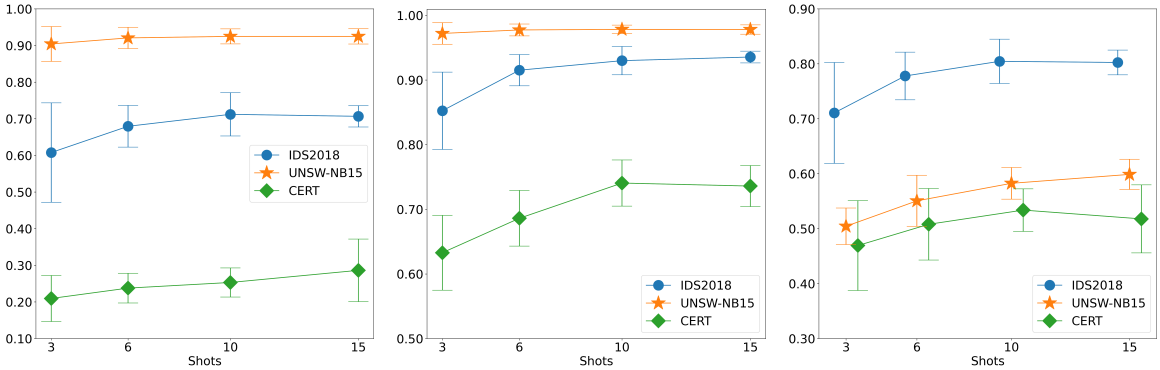
Significantly outperforms FADS at the: * 0.05 and ** 0.01 level, paired t-test.

FADScr with and without Data Selection. We evaluate the performance improvement of FADScr over the simple prototypical networks without reinforced data selection. We consider three settings as baselines. First, we train a prototypical network on the initially labeled dataset, denoted as *ProtoNet*. Second, after we train the prototypical network on the initial dataset, we apply the network to label the unlabeled dataset $\mathcal{U}$ and use the whole weakly-labeled dataset $\mathcal{W}$ as the augmentation set ($\mathcal{A} \leftarrow \mathcal{W}$) to retrain the prototypical network. We evaluate the performance of the prototypical network after re-training on the test set, denoted as *ProtoNet**. The third one is similar to ProtoNet*, but we retrain the

prototypical network based on samples in $\mathcal{W}$ having top 10% of the highest probabilities in each class, which means high confidence belonging to a class, denoted as *ProtoNet* (10%)*.

We also compare FADScr with our previous work *FADS* [119] which does not have the reward to choose the hard samples for performance improvement. We compare four models for anomaly detection and classification. For the anomaly classification, we report the macro precision, recall, and F1 score here.

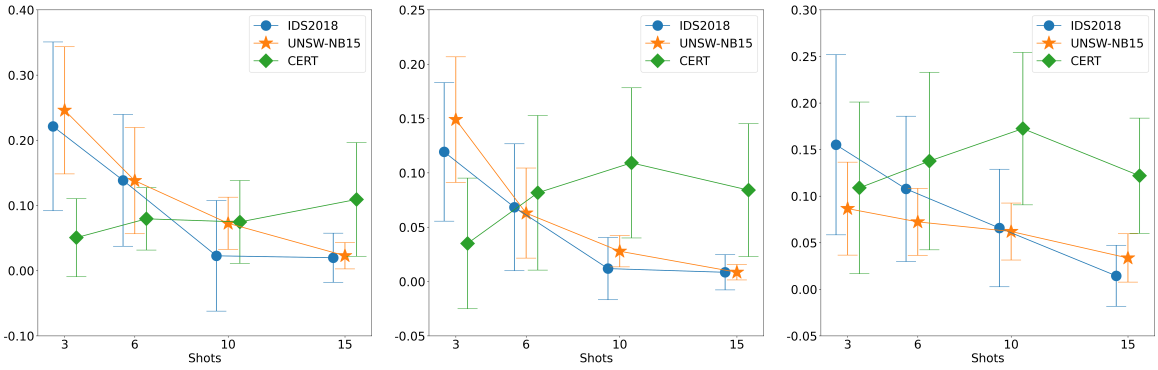
Table 4.8 shows that FADScr achieves unanimously better performances over baselines on three datasets for both anomaly detection and classification tasks. Meanwhile, we can notice that in general, both ProtoNet* and ProtoNet* (10%) cannot beat regular ProtoNet, which shows that simply using weakly labeled samples for training will hurt the performance due to incorrect labels. Furthermore, FADScr still has the largest performance gains on the CERT dataset in terms of macro-F1 for classification, which shows the criticalness of augmenting the training set with reliable samples in the case of limited samples. When comparing the performance of FADS and FADScr, we can notice FADScr with a combinatorial reward can further improve the performance in terms of AUC-PR by a large margin, and also achieve much better F1 scores, especially on the CERT and UNSW-NB15 datasets. Another advantage of FADScr is that in most cases, it can have a lower standard deviation compared with all other baselines.



(a) AUC-PR (b) AUC-ROC (c) Macro-F1

Fig. 4.7: The performance of FADScr with different numbers of initially labeled anomalies in one class. (a),(b) for anomaly detection and (c) for anomaly classification

Performance of FADScr with Different Numbers of Initially Labeled Anomalies. Table 4.8 indicates that the reinforced data selection strategy for training data augmentation can improve the performance of anomaly detection and classification. We then evaluate the performance under the settings where different numbers of labeled anomalies are available at the beginning of the training phase. Figure 4.7 shows the experimental results in terms of AUC-PR, AUC-ROC (anomaly detection), and Macro-F1 (anomaly classification). We can notice that on IDS2018 and UNSW-NB15 datasets, with more labeled samples available, the performance of FADScr keeps improving. Furthermore, the standard deviations are also decreasing with more available samples on IDS2018 and UNSW-NB15. It means with more anomalies available initially, the performance of anomaly detection and classification will become more stable. The performance of anomaly detection on the CERT dataset also basically follows a similar trend, i.e., more available anomalies leading to better performance. On the other hand, the Macro-F1 scores for anomaly classification keep stable with more samples. This could be because the fine-grained classification task on CERT requires more anomalies to further improve the performance due to its difficulty.

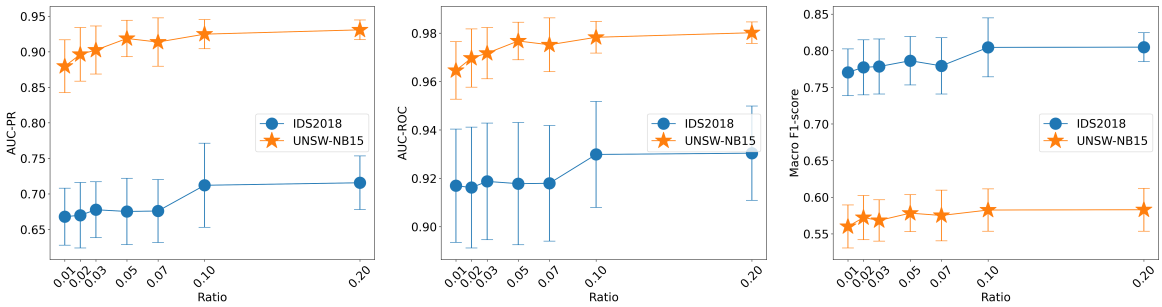


(a) AUC-PR (b) AUC-ROC (c) Macro-F1

Fig. 4.8: The performance gains of FADScr over ProtoNet with different numbers of initially labeled anomalies.

Performance Gains of FADScr over ProtoNet with Different Numbers of Initially Labeled Anomalies. Figure 4.8 shows that, on IDS2018 and UNSW-NB15 datasets, the performance gains compared with ProtoNet without reinforced data selection are decreasing

when more labeled samples are available. This indicates that once prototypical networks have sufficient samples for training, incorporating more samples will not change the results significantly. However, different from the results on UNSW-NB15 and IDS2018, the performance gains on CERT increase when the number of available anomalies in each class increases. It could be because three samples are not sufficient to train prototypical networks for insider threat detection due to its difficulty. As a result, the weakly-labeled samples are full of noise, and no correctly labeled samples are available for re-training. Meanwhile, we can observe that FADScr always achieves positive gains over ProtoNet.



(a) AUC-PR (b) AUC-ROC (c) Macro F1 score

Fig. 4.9: Performance of FADScr with various anomaly ratios in the unlabeled set.

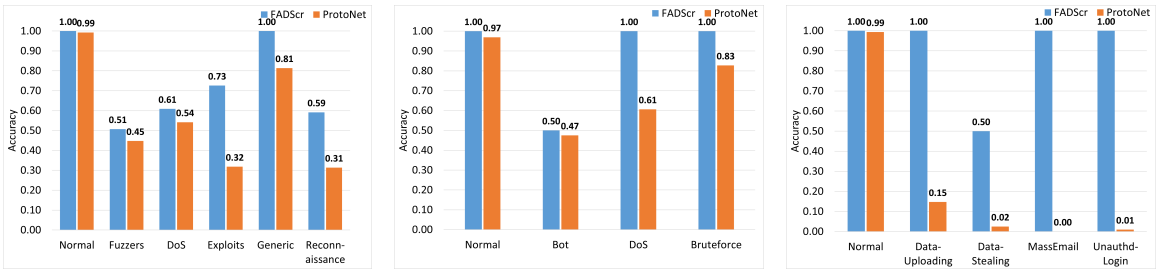
Performance of FADScr with Various Anomaly Ratios in the Unlabeled Dataset. For the UNSW-NB15 and IDS2018 datasets, we set the ratio of anomalies in the unlabeled dataset $\mathcal{U}$ as 0.1 by default. We further tune this ratio to check the performance change of FADScr. Note that because of limited labeled anomalies, we do not conduct this experiment on CERT. Figure 4.9 shows the results of different ratios of anomalies in $\mathcal{U}$ from 0.01 to 0.2. First, we can observe that with more anomalies in $\mathcal{U}$, the performance of FADScr on anomaly detection and classification keeps improving. Especially, for the IDS2018 dataset, in terms of AUC-PR and macro F1 score, the performance of FADScr keeps improving with more anomalies in the unlabeled dataset. For the UNSW-NB15 dataset, the performance of FADScr is improved when the ratios of anomalies increase from 0.01 to 0.1 and then keeps stable even including more anomalies in the unlabeled dataset. This could be because

for the UNSW-NB15 dataset, the 10% of anomalies already cover the majority of anomaly types, and adding more anomalies in the unlabeled dataset can only reduce the standard deviation.

Table 4.9: Impact of each reward component

		AUC-PR	AUC-ROC	Macro F1
r^d	UNSW-NB15	0.9050 ± 0.0182	0.9585 ± 0.0053	0.5587 ± 0.0293
	IDS2018	0.6714 ± 0.0269	0.9237 ± 0.0140	0.7873 ± 0.0241
	CERT	0.2432 ± 0.0379	0.7111 ± 0.0336	0.4685 ± 0.0540
r^c	UNSW-NB15	0.9002 ± 0.0259	0.9574 ± 0.0078	0.5622 ± 0.0235
	IDS2018	0.6656 ± 0.0325	0.9216 ± 0.0150	0.7901 ± 0.0282
	CERT	0.2382 ± 0.0338	0.6978 ± 0.0369	0.4707 ± 0.0409
r^h	UNSW-NB15	0.9025 ± 0.0169	0.9578 ± 0.0049	0.5612 ± 0.0270
	IDS2018	0.6651 ± 0.0355	0.9161 ± 0.0233	0.7874 ± 0.0309
	CERT	0.2456 ± 0.0390	0.7042 ± 0.0446	0.4712 ± 0.0414
Combine	UNSW-NB15	0.9249 ± 0.0206	0.9783 ± 0.0066	0.5825 ± 0.0288
	IDS2018	0.7121 ± 0.0592	0.9299 ± 0.0219	0.8046 ± 0.0403
	CERT	0.2531 ± 0.0397	0.7406 ± 0.0357	0.5338 ± 0.0388

Ablation Study on Reward Components. The reward function defined in Equation 4.9 consists of three parts, rewards for anomaly classification and detection as well as the improvement on hard samples. In this experiment, we check the performance of FADScr if we keep only one reward component. Table 4.9 indicates that incorporating all three reward components can improve performance compared with the model with only one reward component. Meanwhile, FADScr with one reward component (shown in Table 4.9) already achieves better performance than ProtoNet (shown in Table 4.8).



(a) UNSW-NB15 (b) IDS2018 (c) CERT

Fig. 4.10: Accuracy of the data selection agent

Accuracy of the Reinforced Data Selection Agent. We further evaluate whether the reinforced data selection agent is able to select the high-quality samples from the weakly-labeled dataset $\mathcal{W}$. We calculate the accuracy based on all selected samples when the training procedure for data selection is finished. Figure 4.10 shows the results, where the blue bar indicates the accuracy of the FADScr agent while the orange bar indicates the accuracy of initial prototypical networks. We can notice that the initial prototypical networks cannot achieve good performance on anomaly classification, which means the weakly-labeled dataset $\mathcal{W}$ consists of a lot of mislabeled samples. However, the samples selected from the agent have much higher chances to be correct, indicating that the agent is able to identify high quality weakly-labeled samples from $\mathcal{W}$. Therefore, using the selected samples to augment the training dataset can improve the performance of the few-shot anomaly detection model.

4.5 Summary

In this chapter, we detail the progression and enhancement of anomaly detection and classification methodologies through the development of the Few-shot Anomaly Detection and Classification model with Reinforced Data Selection (FADS), and its subsequent extension, the Few-shot Anomaly Detection and Classification model through Reinforced Data Selection with a Combinatorial Reward (FADScr). Our work initially focuses on the FADS framework, which pioneers the use of a large-scale unlabeled dataset to iteratively refine a few-shot learning model's capability for accurately detecting and classifying anomalies. FADS employs a reinforcement learning-based strategy to select potential anomalies from the unlabeled dataset, thereby augmenting the training dataset. This process begins with the training of a prototypical network on a limited set of labeled anomalies, followed by the application of this network to generate a weakly-labeled dataset from the unlabeled samples.

A key innovation in FADS is the introduction of a reinforcement learning-based data selection agent, trained to identify and select the most reliable weakly-labeled samples. These samples are then used to augment the training set, aiming to progressively enhance the prototypical network's performance in anomaly detection and classification. This iterative enhancement process underscores the core methodology of FADS, setting a foundational

approach for anomaly detection and classification through few-shot learning.

Building on the principles established by FADS, FADScr introduces a significant advancement in the form of a combinatorial reward mechanism within the data selection process. This extension refines the selection criteria for weakly-labeled samples by evaluating their contribution not only to the reliability of the training set but also to the reduction of uncertainty in the model's predictions. The incorporation of this combinatorial reward mechanism allows for a more nuanced and effective augmentation of the training set, which in turn facilitates a more precise and efficient anomaly detection and classification by the prototypical network.

Our experimental results demonstrate the effectiveness of both frameworks, highlighting the superior performance of FADScr as an extension of FADS in the realm of few-shot anomaly detection and classification. Looking forward, we aim to expand the capabilities of these models by exploring the reinforcement learning framework's potential to detect and classify new, previously unidentified classes of anomalies. This future direction promises to further enhance the adaptability and precision of few-shot learning models in navigating the complexities of real-world datasets. The early version of this work is published at ICDM 2022 [119].

CHAPTER 5

INTERPRETABLESAD: INTERPRETABLE ANOMALY DETECTION IN SEQUENTIAL LOG DATA

In this chapter, we propose a novel framework called InterpretableSAD for detecting anomalous log sequences and identifying anomalous events. The framework employs a negative sampling algorithm to generate potential anomalous sequences, which enables training a binary classifier for anomaly detection. Additionally, we introduce Integrated Gradients, a model interpretation approach, for identifying the events in a sequence that significantly contribute to the anomalous result, thereby marking them as anomalous events. Most existing anomaly detection models only achieve the detection of anomalous sequences, but InterpretableSAD goes further to identify the anomalous events within the anomalous sequences. To achieve high performance in anomalous event detection, a novel baseline generation algorithm is proposed to ensure an appropriate baseline input for feature attributions. Experimental results show that InterpretableSAD performs better than existing models in detecting anomalous log sequences and identifying anomalous events with high accuracy.

5.1 Introduction

Anomaly detection in sequential log data, which aims to identify sequences that deviate from the expected behavior or patterns, has received much attention due to its broad application [48, 51, 56, 82, 92, 93]. For example, online services generate large amounts of log messages that record states of systems, where the log messages can be modeled as an event sequence [65]. Because online services are everywhere in our daily life, and a little jitter of the services could cause severe consequences, such as financial losses, it is crucial to detect anomalous states in a timely manner to ensure the reliability of the online services and mitigate the losses.

Traditional approaches for sequential anomaly detection are strictly rule-based and dependent on domain knowledge about the patterns of sequences [73]. Although the rule-based approaches can achieve good performance for specific anomalies, the limitations are still obvious, i.e., it is hard to extend to detect new types of anomalies. Currently, many machine learning-based approaches are proposed. Considering the lack of anomalous samples, many unsupervised learning models, such as Principal Component Analysis (PCA) [53], or one class classification models, such as one-class SVM [55], are used to detect anomalies. In order to further model the temporal information of sequential data, the state-of-the-art anomaly detection approaches are mainly based on deep learning models. For example, DeepLog [48] and LogAnomaly [56] utilize long-short term memory (LSTM) network to capture normal sequential patterns from normal samples and detect anomalies.

Because the anomalous samples are rare due to the nature of anomalies, most of the existing approaches are trained in an unsupervised learning manner. These approaches usually assume the normal samples are concentrated in a hypersphere, while the anomalous samples are outside the hypersphere [67, 80]. However, due to the dynamics of sequential data, such assumption can be easily violated for sequential anomaly detection. In this work, different from the existing work, we propose a data augmentation strategy to generate the anomalous samples by negative sampling. Based on negative sampling, we can generate sufficient anomalous samples to train a binary classification model without making an assumption about normal data distribution. When the generated samples are large enough to cover the common anomalous scenarios, the classifier trained on generated anomalous samples can detect the real anomalies as well.

Furthermore, the existing log anomaly detection approaches can only detect anomalous log sequences and cannot identify anomalous events in the sequence. Specifically, if a log sequence is detected as anomalous, there must be one or more events in the sequence that deviate from the expected patterns. For example, if a system is under attack, the operations conducted by the attacker are anomalous events in a log sequence. As an online service system can generate hundreds of logs per second, the capacity of distinguishing anomalous

events from normal ones in a sequence is also critical for system administrators to locate the anomalous operations besides detecting the anomalous sequences. However, detecting the anomalous events in a sequence faces several challenges. First, because the information of a single event is very limited, it is hard to identify useful features to represent an event. Second, an anomalous event could be caused by the broken of correlation among the events. It means we still need the information of the whole sequence to identify the anomalous events. To tackle these challenges, we creatively apply an interpretable machine learning technique, Integrated Gradients (IG) [120], for anomalous event detection. The motivation is that if a classifier predicts a sequence as anomalous, the model interpretation technique should be able to identify which part of the input sequence leads to the anomalous outcome. Then, the events that are responsible for the anomalous result are anomalous events.

In this work, we propose a framework, called InterpretableSAD, for detecting anomalous log sequences as well as anomalous events. Specifically, we propose a negative sampling algorithm to generate potential anomalous sequences automatically so that we can train a binary classification model through the observed normal samples and the generated anomalous samples. We further propose to apply Integrated Gradients for anomalous event detection. The events in a sequence that significantly contribute to the anomalous result are marked as anomalous events.

The main contributions of this work are as follows. First, we propose a novel negative sampling strategy to generate potential anomalous samples based on the observed normal samples, and then we can train a binary classifier for anomaly detection. While data augmentation techniques are widely used in computer vision and natural language processing to improve model performance, it is under-exploited in the area of anomaly detection. Second, most existing anomaly detection models only achieve the anomalous sequence detection and cannot identify the anomalous events in the sequence. We novelly apply a model interpretation approach, Integrated Gradients (IG), to achieve anomalous event detection. Third, because IG relies on an appropriate baseline input for feature attributions, we further propose a novel baseline generation algorithm to improve the performance of anomalous

event detection. Experimental results show that InterpretableSAD can achieve state-of-the-art performance on anomalous log sequence detection and further identify the anomalous events in the anomalous sequences with high accuracy.

5.2 Related Work

Anomaly Detection in Sequential Log Data. Many sequential anomaly detection approaches have been proposed in recent years. Several traditional anomaly detection approaches are based on supervised learning, such as logistic regression, decision tree [70], and Support Vector Machines (SVM) [75]. However, the major limitation of the supervised approaches is that they require an enormous number of labeled data for training, which is usually unavailable in anomaly detection scenarios. Hence, the unsupervised learning approaches have received more attention in the anomaly detection field, such as the dimensionality reduction-based approaches and clustering-based approaches [53, 74]. However, these approaches cannot capture the order information of sequence data.

In recent years, deep learning based sequential anomaly detection models are proposed to detect anomalies by checking differences between normal and anomalous patterns [48, 56, 92, 121, 122]. Specifically, when anomalous events occur, the pattern of sequences will be changed as well, and the models can detect the changes and then flag anomalies. Most of the existing approaches adopt recurrent neural networks to detect anomalous sequences in an unsupervised manner by modeling the patterns of normal sequences [48, 56]. These models adopt long-short term memory (LSTM) [123] to predict the next possible events based on previous events in a sequence. An anomalous sequence will be detected if the actual event is out of a candidate set of expected normal events. However, the existing cutting edge unsupervised approaches are only effective on sequence level anomaly detection and unable to provide detailed information of anomalies on the sub-sequence or event level.

Data Augmentation. Modern machine learning models usually require a large amount of labeled data for training. Unfortunately it is sometimes hard to achieve in reality due to cost and time factors. Data augmentation technique is to tackle the scarcity of labeled data issue by artificially expanding the labeled dataset. Currently, data augmentation is extensively

used in image classification and natural language processing [124–126] to generate auxiliary training data. In practice, data augmentation is conducted by a set of transformation functions, such as rotation and flip for image data or synonym replacement for text data, on the existing dataset. The data generated by carefully designed transformation functions is beneficial to improve the performance of machine learning models.

Negative sampling as a special data augmentation technique is used to generate negative samples instead of positive ones when the negative samples are not available. Negative sampling is a key step in various applications, such as training word embeddings, knowledge graph embeddings, as well as recommender systems [127–130]. Since the main challenge of anomaly detection is the scarcity of anomalous samples, we propose the negative sampling strategy in our work to generate the potential abnormal sequences from the normal ones. Then, we can build a binary classification model to detect anomalies.

Interpretable Machine Learning. Although modern deep learning models have achieved great success in many applications, non-transparency is still a big issue for deploying highly complex models in production environments. As a consequence, interpretable machine learning, which aims at providing human understandable explanations about the decisions made by the models, has become an active research area [131–138]. Interpretable machine learning techniques can generally be categorized into two groups: intrinsic interpretability and post-hoc interpretability. Intrinsic interpretability indicates the models are interpretable due to simple structures, such as decision tree or linear regression, while the post-hoc interpretability implies creating a second model to provide explanations for an existing model [131, 132].

The interpretable anomaly detection models are very limited in the literature [137, 139]. Research in [139] focuses on detecting the outliers in attributed networks, while research in [137] also targets the interpretable anomaly detection in sequential data that leverages the attention mechanism to provide an attention score of each event in a sequence. However, the proposed model is trained to capture the normal patterns based on LSTM by predicting the next event given the previous events in a sequence. The attention scores, which are used for interpretation, are more about the correlation among events in a sequence instead of the

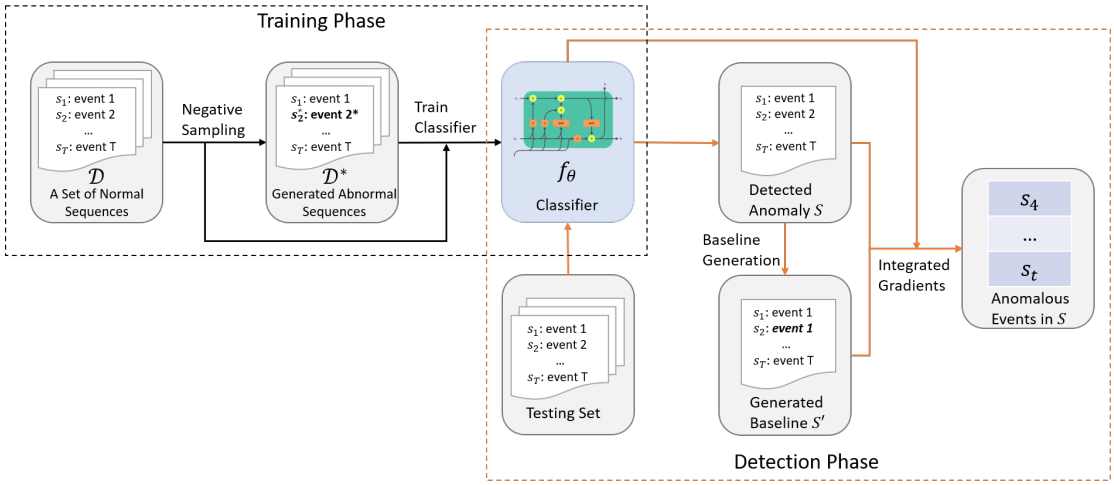


Fig. 5.1: Framework of InterpretableSAD

correlation between events and the label (normal or anomalous). In our work, after training a classification model based on sequences generated by negative sampling, our framework can model the correlation between events and prediction outcome directly based on the interpretable machine learning technique.

5.3 InterpretableSAD

Consider a log sequence of discrete events $S = \{s_1, ..., s_t, ..., s_T\}$, where $s_t \in \mathcal{E}$ indicates the event at the t-th position, and $\mathcal{E}$ is a set of unique events. In this work, we aim at predicting whether a log sequence S is anomalous based on a training dataset $\mathcal{D} = \{S^i\}_{i=1}^N$ that consists of only normal sequences. Meanwhile, for a sequence predicted as anomalous, we further target to identify anomalous events in the sequence so that the domain users can get insights about the detection model as well as anomalous sequences. To achieve the above two goals, in the training phase, assuming that we only have normal samples in our training dataset, we propose a negative sampling approach to generate the potential anomalous sequences so that we can train a classification model based on both positive and negative samples. After training, in the detection phase, once a sequence is predicted as anomalous by the classifier, we novelly leverage an interpretable machine learning technique to identify the anomalous events in the sequence. Specifically, we adopt the Integrated

Gradients approach, which can explain the relationship between the prediction results and input features, to identify the anomalous events. Figure 5.1 shows our sequential data anomaly detection framework.

5.3.1 Data Augmentation via Negative Sampling

In the anomaly detection field, in most cases, we have plenty of normal samples but only observe a small number of anomalous samples. Hence, most anomaly detection approaches are based on one-class classification models. Different from the existing studies, in this work, we propose a data augmentation approach via negative sampling to generate the potential anomalous log sequences based on the observed normal sequences. Then, we can train a binary classification model based on two classes of samples.

In order to train an accurate binary classifier, we aim to generate a dataset $\mathcal{D}^*$ with sufficient anomalous samples that can cover common anomalous scenarios. We consider two anomalous scenarios for anomalous log sequence generation. First, there are some rare events in the sequences. For example, if an online system is compromised by an attacker, the attacker could conduct some uncommon events on the system. Second, some regular events happen in an unusual context. For example, an attacker aims to evade detection by performing some regular activities, but these activities happen at the wrong time, which means that these activities are suspicious based on their context. To simulate these two scenarios, we propose an algorithm (shown in Algorithm 3) to generate the potential anomalous log sequences.

Given the training set $\mathcal{D}$, in order to consider both the event and its context information, we generate a set of bigram events, where each bigram is a sub-sequence of two adjacent events, e.g., (s_t, s_{t+1}), in the sequence S. We then build a bigram event dictionary $\mathcal{B}$, where the key is the bigram, and the value is the corresponding frequency of the bigram in $\mathcal{D}$. Then, given a normal sequence $S \in \mathcal{D}$, we use it as a template to generate the potential anomalous sample by randomly replace r number of events in S. Specifically, for a randomly selected event s_t, we will replace the event s_{t+1} with another event s^*_{t+1} so that the bigram (s_t, s^*_{t+1}) is rare or never observed in the training set $\mathcal{D}$. Due to the scarcity of anomalous events, the

bigram with a low frequency is suspicious. Since we replace r events with low-frequency, we expect that there is a high possibility that the generated sequences are anomalous.

Algorithm 3: Negative Sampling

Input : Training set $\mathcal{D}$, Negative sample size M
Output: Negative sample set $\mathcal{D}^*$

```
Generate a bigram event dictionary B based on D
for i = 0 to M do
    Randomly select S from D
    ind ← Randomly select r indices of events from S
    for t in ind do
        (s_t, s*_{t+1}) ← randomly select or generate a rare or never observed bigram in
         B
        (s_t, s_{t+1}) ← (s_t, s*_{t+1})
    S* ← S,   D*+ = S*
return D*
```

5.3.2 Training a Classification Model

After generating a set of anomalous sequences $\mathcal{D}^*$, we use both $\mathcal{D}$ and $\mathcal{D}^*$ to train a binary classification model $f : S \rightarrow [0, 1]$. In particular, we first adopt word2vec [127] to get representations of events in $\mathcal{E}$ by training on the dataset $\mathcal{D}$. After mapping the events in a sequence to the embedding space, we train a neural network f_θ to predict whether a sequence is normal or anomalous:

$$\hat{y} = f_\theta(S), \tag{5.1}$$

where $\hat{y}$ indicates the predicted label. Because we have two classes of samples on hand, we further adopt the cross-entropy loss to train the neural network. The objective function is defined as:

$$\mathcal{L} = \sum_{j \in \mathcal{D}^* \cup \mathcal{D}} -y_j \log \hat{y}_j - (1 - y_j) \log(1 - \hat{y}_j). \tag{5.2}$$

It is worth noting that any neural network, which can model the sequential data, is able to be used in our framework for anomalous sequence detection.

5.3.3 Anomalous Event Detection via Integrated Gradients

After the classification model is trained based on the normal and generated anomalous sequences, we can deploy the model for detecting the anomalous samples for real. However, in practice, only detecting the anomalous sequences is far from sufficient. For example, because online service systems can generate a huge amount of messages per minute, only identifying anomalous sequences is not sufficient to help the system administrator locate the anomalous operations from attackers. Hence, we further aim at detecting the anomalous events in sequences. There are two key challenges in detecting anomalous events. First, an independent event in a sequence does not contain enough information to support anomaly detection. Second, whether an event is anomalous also depends on its context. To tackle these challenges, in this work, instead of designing a traditional detection model built on the event information, we consider the anomalous event detection in a sequence as a model interpretation problem. The motivation is that when a classification model predicts a sequence as anomalous, the model should detect some anomalous patterns in the sequence. By leveraging the model interpretation techniques, we can further identify the anomalous patterns, i.e., anomalous events, in the sequence.

In our framework, we adopt Integrated Gradients (IG) [120] to derive feature attributions of each sequence. IG is a model interpretable technique that can interpret prediction results by attributing input features in a human-understandable way for various classification tasks, such as image or text classification. For example, in an image or text classification task, IG can show which pixels or words are responsible for a certain label. In this work, we leverage IG to identify anomalous events that cause the sequential anomalies.

Formally, given a neural network $f_\theta : S \to [0, 1]$, integrated gradients are attributions of the prediction at input S relative to a baseline input S' as a vector $A_{f_\theta}(S, S') = (a_1, ..., a_T)$, where a_t is the contribution of s_t to the prediction $f_\theta(S)$. A large positive a_t indicates that feature strongly increases the network output f_θ, while a_t close to zero indicates that the feature did not influence f_θ. Hence, we consider the importance scores $A_{f_\theta}(S, S')$ as anomalous scores to detect the anomalous events.

Specifically, in our scenario, let S be a sequence, and S' be a baseline sequence. The integrated gradient for the t-th event for sequence S and baseline S' is defined as follows.

$$IG_t(S) \equiv (s_t - s'_t) \times \int_{\alpha=0}^{1} \frac{\partial f_\theta(S' + \alpha \times (S - S'))}{\partial s_t} d\alpha. \tag{5.3}$$

The integrated gradients have a property called completeness axiom, which indicates that the sum of integrated gradients over the whole sequence is the difference between the output of classification model f_θ on the input sequence and its baseline, i.e.,

$$\sum_{t=1}^{T} A_{f_\theta}(S_t, S'_t) = f_\theta(S) - f_\theta(S'). \tag{5.4}$$

The completeness axiom of IG ensures that the anomalous score of each event is proportional to the contribution of making S as an anomaly [120].

As shown in Equation 5.3, IG gets the importance score for each event in S by integrating the gradient for each event from a baseline S' to the sequence S, where the baseline is supposed to represent "absence" of features [140]. Hence, finding a reasonable baseline is an essential step for applying the IG method. For image classification models, the black image is widely used as a baseline, while the zero-embedding matrix is a common baseline for the text classification task. However, it is not straightforward to find a single baseline for anomaly detection on sequential data. Different from the text classification task, say sentiment analysis, where the key words contributed to the sentiment are usually positive or negative words, the anomalous events in sequences are related to the context of the sequences, and no widely accepted criteria can be used to quantify the abnormality. Therefore, we propose to generate a unique baseline for each sequence. Meanwhile, based on the completeness axiom shown in Equation 5.4, the sum of importance scores over events in a sequence is the prediction difference between the original sequence and the baseline. In order to have a reasonable IG value, the generated baseline is expected to be a positive sequence so that the sum of importance scores can be in a reasonable scale.

Specifically, to generate the baseline sequence for an anomalous sequence, we first sort

Algorithm 4: Baseline Generation

Input : Neural network f_θ, Anomalous sample S, Training set $\mathcal{D}$, Replacement Threshold τ

Output: Baseline S'

```
i = 0
while f_θ(S) is not normal & i < τ do
    s_t ← Select the event in S with the lowest frequency based on D
    s_t ← s_{t-1}, i+ = 1
S' ← S
return S'
```

the events based on their frequencies in the training set $\mathcal{D}$ and then replace the lowest frequent event s_t with its preceding event s_{t-1} to generate a new sequence. We evaluate whether this new sequence is a normal sequence or not by the neural network f_θ. If this new sequence is still predicted as anomalous, we further replace the currently lowest frequent event with its preceding event until the generated sequence is predicted as normal based on f_θ. We consider the generated sequence as a baseline S' for the original sequence S to derive the IG values. The motivation of replacing the low-frequency events with their preceding events is that a normal sequence should have some sort of integrity at the event level. If there is a low-frequency event in the sequence, we aim at replacing that event with a normal event and keep the integrity of the sequence.

Meanwhile, followed by the strategy proposed in [141], we expect the generated baseline has a short distance to the original sequence in the embedding space. Hence, we set a maximum replacement number τ as a threshold. Once we place τ events in the original sequence, we will stop the replacement disregard the predicted label of the current generated baseline. Algorithm 4 shows the procedure of baseline generation for an anomalous sequence. It is worth noting that the purpose of the above procedure is to generate baselines instead of detecting anomalous events. We will show in our experiments that simply labeling the low-frequency events as anomalous events cannot achieve good performance.

After generating the baseline S', we can derive the anomalous scores of events in a sequence S. Based on the definition of IG, if we consider the anomalous sequence as a positive class, the events with positive scores are anomalous, i.e., making positive contributions to

the prediction. Hence, by default, we can set a threshold $\eta = 0$ to identify the anomalous events. Moreover, if we have a small validation set consisting of anomalous sequences with fine-grained labeled information, we can further leverage the validation set to fine-tune the detection threshold η to identify an optimal value that can lead to better performance on anomalous event detection.

5.4 Experiments

5.4.1 Experimental Setup

Datasets

We apply our framework on detecting the anomalous log sequences on the following three log datasets.

- Hadoop Distributed File System (HDFS) [53]. HDFS dataset is generated by running Hadoop-based map-reduce jobs on Amazon EC2 nodes and manually labeled through handcrafted rules to identify anomalies. HDFS dataset consists of 11,172,157 log messages.

- BlueGene/L Supercomputer System (BGL) [65]. BGL dataset contains 4,747,963 log messages that are collected from a BlueGeme/L supercomputer system at Lawrence Livermore National Labs. The log messages can be categorized into alert and not-alert messages. There are 348,460 alert messages that are labeled as anomalous.

- Thunderbird [65]. Thunderbird dataset is another large-scale system log dataset that is collected from a Thunderbird supercomputer system at Sandia National Labs. We select the first 5,000,000 log messages from the original dataset for our experiment.

Both BGL and Thunderbird datasets provide the fine-grained label for each log message, so we adopt these two datasets to evaluate our framework for anomalous sequence and event

detection. The HDFS dataset only has labels in the sequence level, so we only use the HDFS dataset to evaluate the anomalous sequence detection. All datasets are available online [1].

Log Data Preprocessing. The raw log messages in the three datasets are unstructured text data. Following the typical preprocessing approach, we first we adopt the log parser, Drain [62], to extract log keys (string templates) from log messages (shown in Figure 5.2).

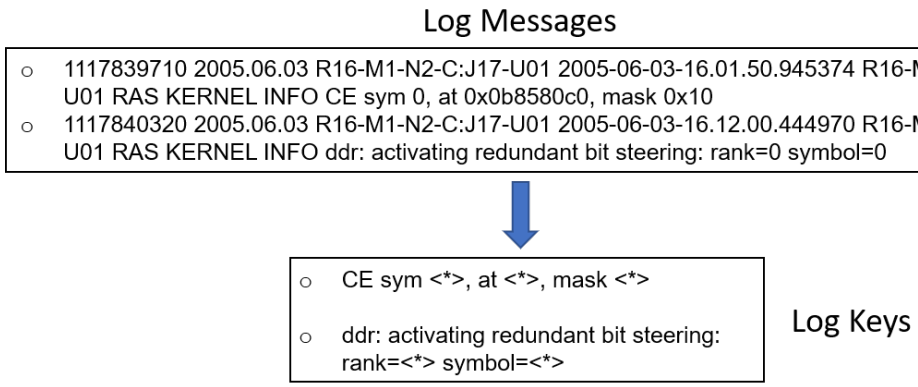


Fig. 5.2: Log messages and corresponding log keys

Then, similar to previous studies [57], for BGL and Thunderbird datasets, we adopt a sliding window to generate appropriate sequences. Especially for our experiment, we define the sliding window with a window size of 100 and a step size of 20. For HDFS, we group log keys into log sequences based on the session ID in the log messages. We compose a training dataset $\mathcal{D}$ that consists of 100,000 normal log sequences from each log dataset. Without a particular note, we generate an anomalous dataset $\mathcal{D}^*$ with 2,000,000 anomalous sequences, which is 20 times larger than the training dataset from each log dataset. When generating $\mathcal{D}^*$, the number of replaced log keys r in Algorithm 3 for each sequence is randomly set with the range from zero to the length of original sequence. The testing dataset consists of both normal and anomalous sequences. The statistics of test datasets are listed in Table 5.1. The number in the brackets under the column "# of Unique Log Keys" indicates the number of unique log keys in the training dataset. Since in this work, we focus on the scenario that the anomalous events are rare, for BGL and Thunderbird datasets, we only select the anomalous

[1] https://github.com/logpai/loghub/

Table 5.1: Statistics of Test Datasets

Dataset	# of Unique Log Keys	# of Log Sequences		# of Log Keys in Anomalous Sequences	
		Normal	Anomalous	Normal	Anomalous
HDFS	48 (19)	458,223	16,838	N/A	N/A
BGL	396 (318)	19,430	4,190	326,491	7,139
Thunderbird	806 (774)	22,538	76,189	6,866,417	479,883

Table 5.2: Results on Anomalous Log Sequence Detection

Method	BGL			Thunderbird			HDFS		
	Precision	Recall	F-1 score	Precision	Recall	F-1 score	Precision	Recall	F-1 score
PCA	67.91	99.79	80.82	94.83	84.43	89.33	97.77	42.12	58.88
iForest	73.13	38.19	50.17	95.06	17.92	30.15	41.59	58.80	48.72
OCSVM	24.60	100	39.49	87.13	100	93.12	6.68	90.58	12.44
LogCluster	8.03	15.97	10.69	86.56	22.94	36.26	98.37	67.45	80.03
DeepLog	42.39	52.08	46.74	82.42	81.36	81.89	56.98	48.37	52.32
LogAnomaly	42.58	53.17	47.29	81.69	82.11	81.90	55.85	48.03	51.65
InterpretableSAD	94.25	88.47	**91.27**	97.31	96.42	**96.86**	92.31	87.04	**89.60**

sequences with anomalous log keys less than or equal to 10% in the testing sets.

Baselines

We use two sets of baselines to evaluate the performance of InterpretableSAD for anomalous sequence and event detection, respectively. Note that to distinguish the terminology "baseline" used as a benchmark for experiments and the generated input sequence for the IG method, in this section, we call the baseline used in IG as "feature attribution baseline".

Baselines for Anomalous Log Sequence Detection

- Principal Component Analysis (PCA) [53]. PCA builds a counting matrix based on the frequency of log keys and then map the original counting matrix into a low dimensional space. PCA-based anomaly detection can efficiently detect extreme values.

- One-Class SVM (OCSVM) [67]. One-Class SVM is a one-class classification model that can detect anomalies based on the observed normal samples.

- Isolation Forest (iForest) [54]. Isolation forest is a tree-based anomaly detection method. It constructs trees based on the features in normal samples and captures the anomalies that deviate from normal samples.

- LogCluster [74]. LogCluster is a clustering-based one-class approach, which groups normal samples into clusters and detects the anomalies based on distances to the clusters.

- DeepLog [48]. DeepLog is a deep learning-based log anomaly detection approach. DeepLog utilizes LSTM to model the patterns of normal log sequences by training on a normal dataset and detects the anomalous sequences based on the log key prediction. If DeepLog cannot correctly predict the next log key in a sequence, the sequence will be labeled as anomalous.

- LogAnomaly [56]. LogAnomaly is another deep learning approach for anomaly detection. It combines sequential and quantitative patterns to discover the anomalous log sequences. Similarly to DeepLog, the anomalous sequence is detected based on whether the LSTM model, which is trained on the normal samples, can correctly predict the next log key.

Baselines for Anomalous Event Detection

- Anchors [133]. Anchors is a model-agnostic algorithm for interpretation of any black-box classification model. Anchors discovers a decision rule (anchors) for each input sample, and identified anchors contain essential parts of the input that determine the prediction. To conduct a fair comparison for anomalous event detection, we adopt our neural network model f_θ trained based on normal and generated anomalous samples as the black-box classification model.

- Low-Freq. The baseline for deriving the integrated gradients of each anomalous sequence is generated by replacing the low frequency events with high frequency events in the context. We further evaluate the performance of only considering the low frequency events in the sequences as anomalous events.

- Integrated Gradients (IG). We also evaluate the performance of IG without using our feature attribution baseline generation algorithm (shown in Algorithm 4) for anomalous event detection. We adopt the zero embedding matrix as the feature attribution baseline, which is widely used in text classification tasks.

Evaluation Metrics

We consider the anomalous class as the target class and adopt Precision, Recall, and F1 score to measure the performance of our framework.

Implementation Details

Regarding baselines, we leverage the package *Loglizer* [70] to evaluate PCA, OCSVM, iForest as well as LogCluster, and adopt the open source deep learning-based log analysis toolkit *LogDeep* to evaluate DeepLog and LogAnomaly [2]. We use the open source repository of Anchor to evaluate its performance on anomalous event detection [3].

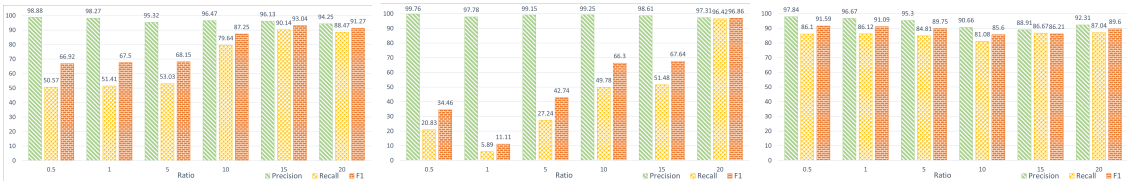
Regarding our model, We adopt the long short-term memory (LSTM) network as the neural network model f_θ for anomaly detection. For BGL and Thunderbird datasets, the embedding size of log keys is 8, while for the HDFS dataset, the embedding size is 4 due to the small number of unique log keys (19 in the training set). Regarding the LSTM structures, we set different hyper-parameters on the basis of the characteristics of each dataset. For the BGL dataset, we use a single-direction LSTM with the hidden size of 128; for the Thunderbird dataset, we use a bidirectional LSTM with the hidden size of 256; for the HDFS dataset, we use a single-direction LSTM with the hidden size of 64. The number of training epochs is set as 10 for all datasets. Our code is available online [4].

5.4.2 Experimental Results on Anomalous Log Sequence Detection

Table 5.2 shows the performance of our model as well as baselines for anomalous sequence detection on three datasets. On the BGL dataset, only PCA can achieve reason-

[2] https://github.com/donglee-afar/logdeep
[3] https://github.com/marcotcr/anchor
[4] https://github.com/hanxiao0607/InterpretableSAD

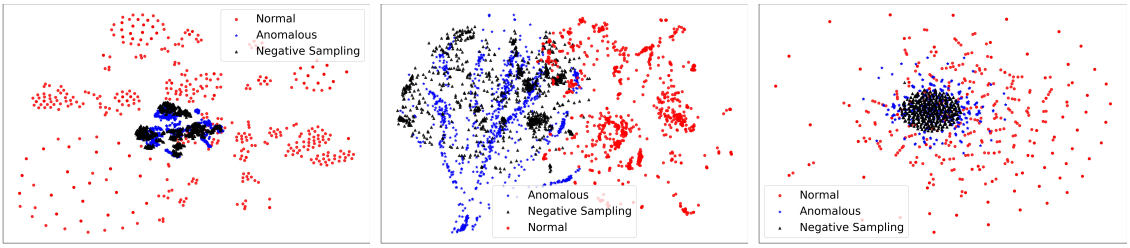
able performance, while all other baselines have poor F-1 scores. PCA can achieve an extremely high recall value, but cannot find a good balance between precision and recall on the anomalous sequence detection. On the Thunderbird dataset, all the baselines can achieve reasonable values in terms of precision, while the iForest and LogCluster fail to gain an acceptable performance on recall values. It means that they can only detect a small number of anomalous sequences. On the HDFS dataset, most baselines cannot achieve good performance. Meanwhile, surprisingly, on all three datasets, the deep learning-based approaches, DeepLog and LogAnomaly, cannot achieve remarkable performance even compared with the traditional anomaly detection models, like PCA. This could be because for BGL and Thunderbird, we focus on a more challenging scenario that aims at detecting the anomalous log sequences with small ratios of anomalous log keys (less than 10%). When only having a small number of anomalous events in a log sequence, the anomalous signal is not strong enough to make the models label it as anomalous. For the HDFS dataset, we generate the log sequences based on session IDs, which leads to long sequences, while DeepLog and LogAnomaly detect anomalous sequences based on the prediction accuracy of the last log keys, which is insufficient for long sequences. On the other hand, Interpretable-SAD achieves the best performance in terms of F-1 scores on all three datasets. It means that the negative samples generated based on the Algorithm 3 represent the true anomalous log sequences in real datasets. Meanwhile, the good performance also show that once we can generate appropriate negative samples, a classification model that is trained on two classes of log sequences can achieve better performance compared with one-class models.



(a) BGL (b) Thunderbird (c) HDFS

Fig. 5.3: Impact of the negative sampling ratio on the anomalous sequence detection

Sensitivity analysis on the size of generated anomalous sequences. In our work,



(a) BGL (b) Thunderbird (c) HDFS

Fig. 5.4: Visualization of the normal, anomalous, and generated anomalous sequences.

because we adopt negative sampling to generate potential anomalous sequences, technically, we can generate an infinite number of anomalous sequences. We further investigate the impact of generated anomalous sample size on anomaly detection performance. In particular, we generate six anomalous datasets with different sizes, where the ratios of generated anomalous datasets $|\mathcal{D}^*|$ to the training dataset $|\mathcal{D}|$ are 0.5, 1, 5, 10, 15, 20, respectively. Figure 5.3 shows the performance of anomaly detection on three datasets by training on different sizes of datasets. We have the following observations. First, for all the datasets, the precision values are high for different training sizes. Second, for the BGL and Thunderbird datasets, the recall values almost keep increasing along with the increase of sizes of anomalous datasets. For example, for the BGL dataset (shown in Figure 5.3a), the best performance is achieved when the size of generated anomalous dataset is 15 times larger than the training dataset, while for the Thunderbird dataset (shown in Figure 5.3b), a good performance is achieved only when the anomalous dataset is 20 times larger than the training dataset. It indicates that after training on a set of generated anomalous samples, the classification model can always detect some anomalies based on the observed samples. Hence, the precision values are high even with a small set of anomalous samples. It also shows the effectiveness of the negative sampling algorithm. However, in order to detect more anomalies (increasing recall), we need to generate more anomalous samples to cover various anomalous scenarios. Once we have sufficient anomalous samples to train the classification model, we can get good recall values as well as the F-1 scores. For HDFS, we notice that the overall performance keeps stable over different sizes of anomalies. This is because HDFS only has 19 unique log

keys in the training set, which means the search space is relatively small. Comparing with BGL and Thunderbird, which have 318 and 774 unique log keys, respectively, the number of potential anomalous scenarios in HDFS is much smaller. As a consequence, generating 50,000 anomalies is sufficient enough to cover most of the anomalous scenarios.

Visualization. We consider the last hidden state in the LSTM model as the sequence representation and adopt the t-SNE algorithm [142] to map the sequence representations into a two-dimensional space. For each dataset, we randomly select 1000 normal, anomalous, and generated samples, separately. As shown in Figure 5.4, for all datasets, the generated samples via negative sampling can cover the space of real anomalous samples. Especially, for BGL and HDFS datasets (Figures 5.4a and 5.4c), the points of generated anomalous samples and true anomalous samples are highly overlapped, while the majority of normal samples are outside the regions of anomalous samples. For the Thunderbird dataset (Figure 5.4b), the generated samples and abnormal samples are on left side of the space, while the normal samples are on the right side. Based on the visualization results, it is straightforward to notice that the LSTM model trained on the normal sequences and generated anomalous sequences can detect the real anomalous sequences for all three datasets.

5.4.3 Experimental Results on Anomalous Event Detection

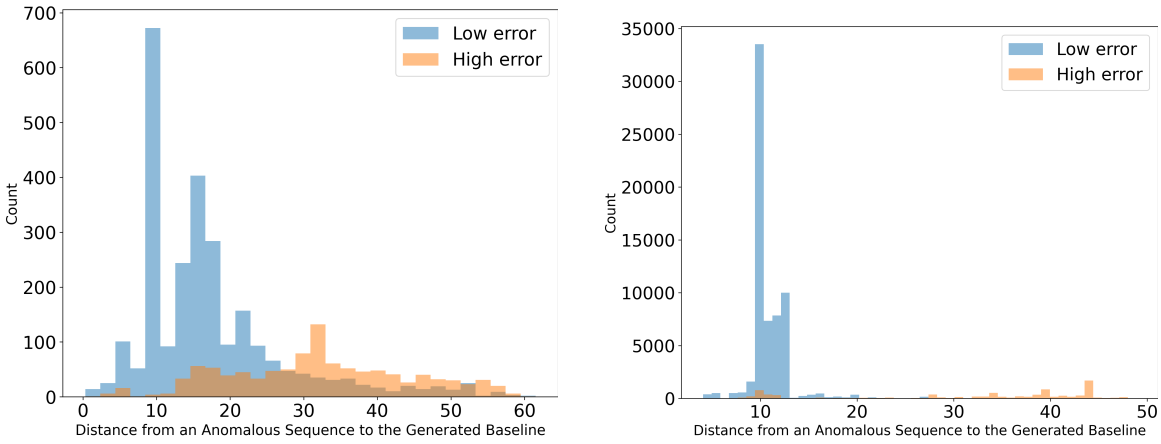
Table 5.3: Results on Anomalous Event Detection

Method	BGL			Thunderbird		
	Precision	Recall	F-1 score	Precision	Recall	F-1 score
Anchors	0.31	8.56	0.60	4.58	14.62	6.98
Low-Freq	38.76	93.59	54.82	52.61	99.00	68.70
IG w/o val	6.56	90.27	12.23	10.36	85.65	18.49
IG w/ val	42.43	73.83	53.89	20.92	44.48	28.45
InterpretableSAD w/o val	50.87	89.23	64.80	94.98	86.79	90.70
InterpretableSAD w/ val	68.92	82.53	**75.11**	93.84	98.31	**96.02**

We then study the performance of InterpretableSAD on anomalous event detection.

When evaluating InterpretableSAD and IG with the zero embedding matrix as the feature attribution baseline, we consider two scenarios, with or without a validation set consisting of 10% anomalous sequences in the testing datasets to tune a detection threshold η. Recall that we only consider the events with anomalous scores greater than η are anomalous. The default value of η is 0 without tuning on a validation set. As shown in Table 5.3, InterpretableSAD with a validation set achieves the best performance for anomalous event detection on both datasets. Meanwhile, even we use the default threshold $\eta = 0$ to detect the anomalous events, the performance is still good. The performance of IG with the zero embedding matrix as feature attribution baseline is poor, even we use a validation set to tune η. It indicates the importance of designing a good feature attribution baseline for the IG model, and the zero embedding matrix that is widely used as the feature attribution baseline in interpreting text classification models is not suitable for sequential anomaly detection. Moreover, we notice that simply labeling low frequent events as anomalous cannot achieve good results in terms of precision and F-1 score, even though the recall values are high on both datasets. It indicates that anomalous events are usually low frequent in the training dataset, but many normal events could also have low frequent, which will lead to low precision. It is hard to balance the precision and recall simply based on the frequency of events. For Anchors, it cannot achieve reasonable performance on both datasets. This could be because long sequences have huge search spaces to locate the anchors.

Sensitivity analysis on the distances between sequences and the feature attribution baselines. The performance of Integrated Gradients heavily relies on the feature attribution baselines. When we generate the feature attribution baselines, we expect the baseline has a short distance to the original sequence. To further show the impact of baselines on anomalous event detection, we consider the anomalous event detection results into two categories, low error and high error. If InterpretableSAD correctly detects at least 80% of anomalous events in an anomalous sequence, we consider the prediction result as low error; otherwise, we consider the prediction result as high error. Then, we explore the correlation between the performance of anomalous event detection and the distances from sequences to



(a) BGL (b) Thunderbird

Fig. 5.5: The correlation between the performance of anomalous event detection and the distances from sequences to corresponding baselines. Low error indicates given an anomalous sequence, InterpretableSAD correctly detect at least 80% of anomalous events.

corresponding feature attribution baselines. The distance is the L2 distance between the embedding matrices of the original sequence and baseline sequence. As shown in Figure 5.5, for both datasets, if baseline sequences have small distances to the original anomalous sequences, in most cases, we can achieve low error for anomalous event detection. For example, because we achieve a high F-1 score (96.02%) on the Thunderbird dataset, most sequences have small distances to the corresponding feature attribution baselines (shown in Figure 5.5b), while only a few sequences have large distances to the baselines. Similar observations on Figure 5.5a, the majority of sequences with high errors have larger distances to the feature attribution baselines. Hence, based on Figure 5.5, we have two findings. First, IG is sensitive to the feature attribution baselines, so choosing a good baseline is critical for anomalous event detection. Second, in practice, a good feature attribution baseline should meet the following requirements: 1) the feature attribution baseline can be predicted as normal sequence; 2) the distance from the original sequence to the feature attribution baseline should be small.

Case Study. For the HDFS dataset, we do not have the fine-grained event labels. We apply the case study to show the effectiveness of InterpretableSAD on anomalous event detection. Figure 5.6 shows an example of the model prediction on an anomalous sequence in the

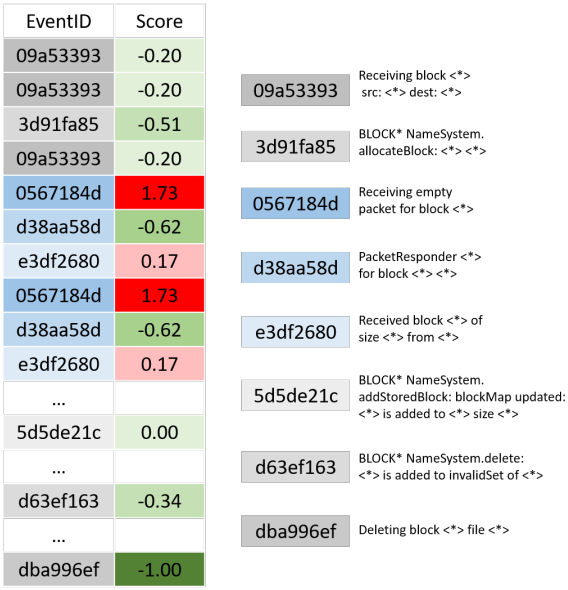


Fig. 5.6: An anomalous sequence in the HDFS dataset and the corresponding anomalous scores

HDFS dataset. The model predicts a sequence (session ID: "blk_-4364732810285057372") with 22 events as anomalous. Besides detecting the anomalous sequence, InterpretableSAD further derives the anomalous score for each event in the sequence. Specifically, this log sequence records a set of operations about failing to create a block. We notice that the event "0567184d" has a high anomalous score (1.73), which means it is responsible for the anomalous prediction outcome. "0567184d" indicates receiving an empty packet for the block. Based on the highlighted event "0567184d", we can understand that the failure of creating a block is caused by receiving empty packets for the block several times. Meanwhile, for all other operations in this sequence, such as block allocation, adding the block to an invalid set, and block deletion, InterpretableSAD assigns negative scores, which means these operations are common in a block creating procedure and not anomalous. Based on this case study, we show that according to the derived anomalous scores, system administrators can quickly locate the exactly anomalous events without manually examining each event in a sequence. This improvement can further help the system administrators to effectively mitigate the system failure.

5.5 Summary

In this work, we have developed InterpretableSAD for anomalous log sequence and more fine-grained anomalous log event detection. Considering the rare of anomalous samples, InterpretableSAD leverages the data augmentation strategy to generate anomalous samples by proposing a novel negative sampling algorithm. Then, a binary classification model can be trained on observed normal and generated anomalous sequences. A well-trained classifier is able to detect the real anomalous sequences. Since an anomalous log sequence usually consists of a large number of events, only detecting anomalous sequences is not sufficient to help domain experts locate the exact anomalies. InterpretableSAD further applies an interpretable machine learning technique, Integrated Gradients (IG), to detect the potential anomalous events in sequences. IG is able to show the importance of each feature to the prediction outcome of the classifier. We consider the importance scores derived from IG as anomalous scores to detect anomalous events. To apply IG for anomalous event detection, we propose a novel feature attribution baseline generation algorithm because a good baseline is critical for IG to derive reasonable scores of events. Experimental results on three log datasets show that our model can achieve state-of-the-art performance on the anomalous sequence and event detection. In the future, we plan to study how to efficiently generate negative samples so that a small ratio of generated samples can still cover the majority anomalous scenarios and also explore the baseline generation algorithms for anomaly detection with theoretical guarantees. Another direction of future work is to study an intrinsically interpretable model that is able to detect anomalous sequences and events in an end-to-end manner. The early version of this work is published at IEEE Big Data 2021 [143].

CHAPTER 6

ALGORITHMIC RECOURSE FOR ANOMALY DETECTION IN MULTIVARIATE TIME SERIES

In this chapter, we introduce RecAD, a novel framework for algorithmic recourse in time series anomaly detection. This work addresses the critical need for not just detecting anomalies in multivariate time series, which often indicate significant events like system faults or external attacks, but also recommending effective mitigation actions. RecAD leverages Unsupervised Anomaly Detection (USAD) and a counterfactual reasoning process to recommend minimal-cost actions that correct abnormalities. The framework accounts for the downstream impact of interventions, ensuring that the normality is maintained in subsequent time steps. Experimental results on synthetic and real-world datasets demonstrate RecAD's effectiveness in not only detecting anomalies but also in suggesting actionable recourse to restore normalcy.

6.1 Introduction

Multivariate time series is a collection of observations that are recorded chronologically and have correlations in time. Due to the ubiquitous of multivariate time series, anomaly detection in time series data has received a large number of studies [144] and has a wide spectrum of applications, such as detecting abnormal behaviors in online services [145, 146].

Algorithmic recourse is to provide recommendations to flip unfavorable outcomes by automated decision-making systems with minimum cost [147]. For example, if a loan application is denied by a system, algorithmic recourse is to figure out how to flip the decision (loan approved) with minimum cost. In the area of time series anomaly detection, after receiving an alert about a potential abnormal behavior detected by an anomaly detection model, algorithmic recourse is to predict recourse actions to fix such abnormal behavior.

For example, Figure 6.1 presents the usage of two control nodes, nodes 117 and 124,

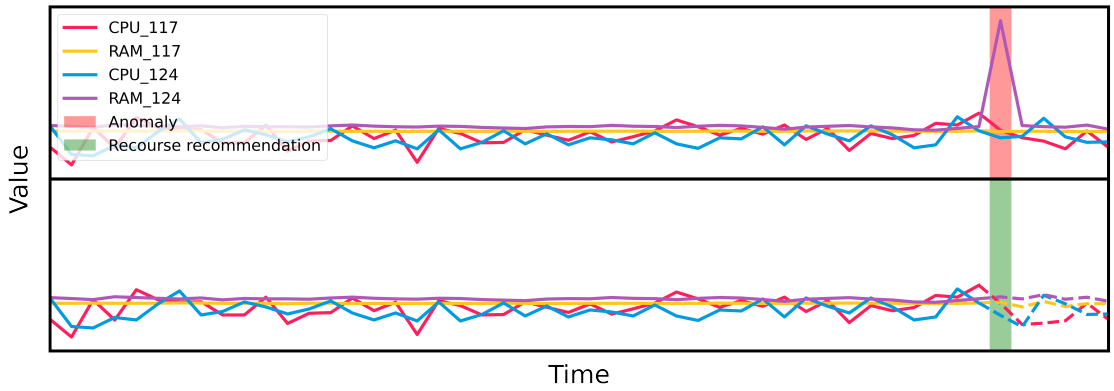


Fig. 6.1: Recourse recommendations for flipping an abnormal status of a distribution system to a normal status.

in an OpenStack testbed [148]. Each node's performance includes the CPU time spent on running the user's programs and memory usage. When an anomaly detection model triggers an anomaly alert (the red area in the top figure), a recourse action, which aims to flip the abnormal behavior, is recommended to free up memory usage on node 124. After taking the recourse action (the green area in the bottom figure), the system returns to normal (dashed lines in the bottom figure). Therefore, if there is abnormal behavior in a distributed system detected by an anomaly detection model, algorithmic recourse can help the domain expert quickly fix the issue with minimum cost.

In this work, we aim to recommend recourse actions to flip abnormal outcomes predicted by an anomaly detection model. The key challenge to recommending proper actions is that due to the nature of multivariate time series, any recourse actions on the current time step has a downstream impact governed by causal relationships. Therefore, the recourse actions in an abnormal time step should not only fix the current step but also ensure the normality in the following time steps.

We propose a framework for algorithmic **Rec**ourse in time series **A**nomaly **D**etection (RecAD), which is able to recommend recourse actions to flip the abnormal outcome. We first leverage UnSupervised Anomaly Detection for multivariate time series (USAD) [149] as the base anomaly detection model, which detects the abnormal time step based on the reconstruction error of an autoencoder model. Once an abnormal time step is detected, RecAD

can predict recourse actions to flip the abnormal outcomes by minimizing the reconstruction error of the abnormal time step. More importantly, due to the interdependence of multivariate time series, the recourse actions on the abnormal time step have downstream impacts and make the following time series unobservable in the counterfactual world. To quantify the downstream impact, RecAD derives the counterfactual time series after recourse based on the Abduction-Action-Prediction process [150] governed by the causal relationships in the multivariate time series. Then, the training objective of RecAD is to ensure after applying the recourse actions, the current and the downstream counterfactual time series are predicted as normal.

The contribution of this work can be summarized as follows. 1) We propose a novel framework for algorithmic recourse in time series anomaly detection, called RecAD. To the best of our knowledge, this is the first work on this topic. 2) RecAD considers the downstream impact of the intervention on the abnormal time step by deriving the counterfactual time series after the intervention. The goal is to ensure the following time series after intervention should also be normal. 3) The empirical studies on two synthetic and one real-world datasets show the effectiveness of RecAD for recommending recourse in time series anomaly detection.

6.2 Related Work

6.2.1 Time Series Anomaly Detection

A time series anomaly is defined as a sequence of data points that deviates from frequent patterns in the time series [151]. Recently, a large number of deep learning-based approaches have been developed for time series anomaly detection [144, 151]. Most of the approaches are trained in the semi-supervised setting, which assumes the availability of normal time series. Then, in the test phase, the anomaly detection model can mark anomalies that are different from normal behavior measured by an anomaly score. In this work, given a detected abnormal time series, we would further like to recommend recourse actions to flip the abnormal outcome.

6.2.2 Algorithmic Recourse

Algorithmic recourse is to provide explanations and recommendations to flip unfavorable outcomes by an automated decision-making system [147]. Specifically, given a predictive model and a sample having an unfavorable prediction from the model, algorithmic recourse is to identify the minimal consequential recommendation that leads to a favorable prediction from the model. The key challenge of identifying the minimal consequential recommendation is to consider the causal relationships governing the data. Any recommended actions on a sample should be carried out via structural interventions leading to a counterfactual instance. Multiple algorithmic recourse algorithms on binary classification models have developed [152–159]. Recently, algorithmic recourse for anomaly detection on tabular data is also discussed [160]. However, the existing study also does not consider causal relationships when generating counterfactuals. In this work, we focus on addressing the algorithmic recourse for anomaly detection in multivariate time series with the consideration of causal relationships.

6.3 Preliminary

6.3.1 Granger Causality

Granger causality [161, 162] is commonly used for modeling causal relationships in multivariate time series. The key assumption is that if the prediction of the future value Y can be improved by knowing past elements of X, then X "Granger causes" Y. Let a stationary time-series as $\mathcal{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_T)$, where $\mathbf{x}_t \in \mathbb{R}^d$ is a d-dimensional vector (e.g., d-dimensional time series data from d sensors) at a specific time t. Suppose that the true data generation mechanism is defined in the form of

$$x_t^{(j)} := f^{(j)}(\mathbf{x}_{\leq t-1}^{(1)}, \cdots, \mathbf{x}_{\leq t-1}^{(d)}) + u_t^{(j)}, \text{ for } 1 \leq j \leq d, \tag{6.1}$$

where $\mathbf{x}_{\leq t-1}^{(j)} = [\cdots, x_{t-2}^{(j)}, x_{t-1}^{(j)}]$ denotes the present and past of series j; $u_t^{(j)}$ indicates exogenous variable of time series j at time step t; $\mathcal{F} = \{f^{(1)}, ..., f^{(d)}\}$ is a set of nonlin-

ear functions, and $f^{(j)}(\cdot) \in \mathcal{F}$ is a nonlinear function for time series j that captures how the past values impact the future values of $\mathbf{x}^{(j)}$. Then, the time series i Granger causes j, if $f^{(j)}$ depends on $\mathbf{x}^{(i)}_{\leq t-1}$, i.e., $\exists \mathbf{x'}^{(i)}_{\leq t-1} \neq \mathbf{x}^{(i)}_{\leq t-1} : f^{(j)}(\mathbf{x}^{(1)}_{\leq t-1}, \cdots, \mathbf{x'}^{(i)}_{\leq t-1}, \cdots, \mathbf{x}^{(d)}_{\leq t-1}) \neq f^{(j)}(\mathbf{x}^{(1)}_{\leq t-1}, \cdots, \mathbf{x}^{(i)}_{\leq t-1}, \cdots, \mathbf{x}^{(d)}_{\leq t-1})$.

6.3.2 Generalised Vector Autoregression (GVAR)

Granger causal inference has been extensively studied [163–166]. Recently, a generalized vector autoregression (GVAR) is developed to model nonlinear Granger causality in time series by leveraging neural networks [165]. GVAR models the Granger causality of the t-th time step given the past K lags by

$$\mathbf{x}_t = \sum_{k=1}^{K} g_k(\mathbf{x}_{t-k})\mathbf{x}_{t-k} + \mathbf{u}_t, \tag{6.2}$$

where $g_k(\cdot) : \mathbb{R}^d \to \mathbb{R}^{d\times d}$ is a feedforward neural network predicting a coefficient matrix at time step $t-k$; $\mathbf{u}_t$ is the exogenous variable for time step t. The element (i,j) of the coefficient matrix from $g_k(\mathbf{x}_{t-k})$ indicates the influence of $x^{(j)}_{t-k}$ on $x^{(i)}_t$. Meanwhile, K neural networks are used to predict $\mathbf{x}_t$. Therefore, relationships between d variables over K time lags can be explored by inspecting K coefficient matrices. The K neural networks are trained by the objective function: $\mathcal{L} = \frac{1}{T-K}\sum_{t=K+1}^{T} \|\mathbf{x}_t - \hat{\mathbf{x}}_t\|_2 + \frac{\lambda}{T-K}\sum_{t=K+1}^{T} R(\mathcal{M}_t) + \frac{\gamma}{T-K-1}\sum_{t=K+1}^{T-1} \|\mathcal{M}_{t+1} - \mathcal{M}_t\|_2$, where $\hat{\mathbf{x}}_t = \sum_{k=1}^{K} g_k(\mathbf{x}_{t-k})\mathbf{x}_{t-k}$ indicates the predicted value GVAR; $\mathcal{M}_t := [g_K(\mathbf{x}_{t-K}) : g_{K-1}(\mathbf{x}_{t-K+1}) : \cdots : g_1(\mathbf{x}_{t-1})]$ indicates the concatenation of generalized coefficient matrices over the past the K time steps; $R(\cdot)$ is the penalty term for sparsity, such as L1 or L2 norm; the third term is a smooothness penalty; λ and γ are hyperparameters. After training, the generalized coefficient predicted by $g_k(\mathbf{x}_{t-k})$ indicates the causal relationships between time series at the time lag k.

6.4 RecAD

In this work, we aim to achieve algorithmic recourse for anomaly detection in multivariate time series. To this end, UnSupervised Anomaly Detection for multivariate time series

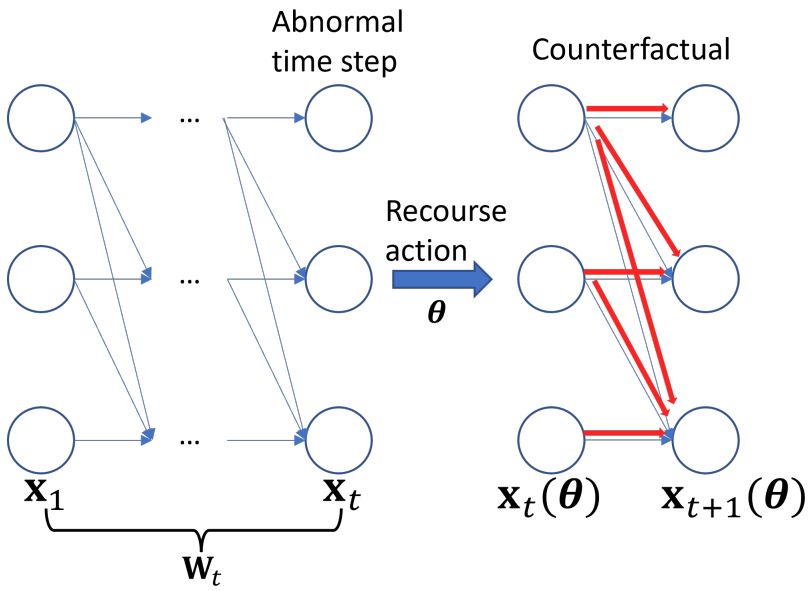


Fig. 6.2: Algorithmic Recourse on Multivariate Time Series

(USAD) [149] is adopted as a base anomaly detection model. After detecting the abnormal time steps, we propose to recommend recourse actions to flip the abnormal outcome, where the action values can fix the abnormal behavior with the minimum cost. Because the variables in a time series have causal connections through time, when recommending actions, we should consider the downstream impact on other variables. Therefore, we develop a framework for algorithmic **Rec**ourse in time series **A**nomaly **D**etection (RecAD), which is able to predict the recourse actions that fix the abnormal time series.

Anomaly in multivariate time series. Based on the structural equation of multivariate time series, we propose to describe the anomaly from the perspective of causal relationships in multivariate time series:

$$x_t^{(j)} := f^{(j)}(\mathbf{x}_{\leq t-1}^{(1)}, \cdots, \mathbf{x}_{\leq t-1}^{(d)}) + u_t^{(j)} + \epsilon_t^{(j)}, \text{ for } 1 \leq j \leq d. \tag{6.3}$$

The anomaly term $\epsilon_t^{(j)}$ can be due to either an external intervention or a structural intervention. The external intervention (i.e., **non-causal anomaly**) indicates a significantly deviating value in its exogenous variable $\tilde{u}_t^{(j)}$ and can be defined as: $x_t^{(j)} = f^{(j)}(\mathbf{x}_{\leq t-1}^{(1)}, \cdots, \mathbf{x}_{\leq t-1}^{(d)}) + \tilde{u}_t^{(j)} = f^{(j)}(\mathbf{x}_{\leq t-1}^{(1)}, \cdots, \mathbf{x}_{\leq t-1}^{(d)}) + u_t^{(j)} + \epsilon_t^{(j)}$, for $1 \leq j \leq d$, where $\tilde{u}_t^{(j)} = u_t^{(j)} + \epsilon_t^{(j)}$. The struc-

tural intervention (i.e., **causal anomaly**) indicates the replacement of the structural functions $\mathcal{F}$ with abnormal functions $\tilde{\mathcal{F}}$ and can be defined as: $x_t^{(j)} = \tilde{f}^{(j)}(\mathbf{x}_{\leq t-1}^{(1)}, \cdots, \mathbf{x}_{\leq t-1}^{(d)}) + u_t^{(j)} = f^{(j)}(\mathbf{x}_{\leq t-1}^{(1)}, \cdots, \mathbf{x}_{\leq t-1}^{(d)}) + u_t^{(j)} + \epsilon_t^{(j)}$, for $1 \leq j \leq d$, where $\tilde{f}^{(j)}(\mathbf{x}_{\leq t-1}^{(1)}, \cdots, \mathbf{x}_{\leq t-1}^{(d)})$ is an abnormal function for the time series j at time t. The anomaly term caused by the change of causal relationships is given by $\epsilon_t^{(j)} = \tilde{f}^{(j)}(\mathbf{x}_{\leq t-1}^{(1)}, \cdots, \mathbf{x}_{\leq t-1}^{(d)}) - f^{(j)}(\mathbf{x}_{\leq t-1}^{(1)}, \cdots, \mathbf{x}_{\leq t-1}^{(d)})$ which is time-dependent.

Equation (6.3) also follows the intuitive definition of an anomaly as an observation that deviates from some concepts of normality [167,168]. Here, normality indicates the structural equation without the anomaly term $\epsilon_t^{(j)}$.

6.4.1 Problem Formulation

Denote a multivariate time series as $\mathcal{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_T)$, where each time step $\mathbf{x}_t \in \mathbb{R}^d$ indicates an observation measured at time step t. Following the common setting for time series anomaly detection [149,169], given a time series $\mathcal{X}$, to model the local dependence between a time step $\mathbf{x}_t$ and past lags, we first define a local window with length K as $\mathbf{W}_t = (\mathbf{x}_{t-K+1}, ..., \mathbf{x}_t)$ and convert a time series $\mathcal{X}$ to a sequence of sliding windows $\mathcal{W} = (\mathbf{W}_K, \mathbf{W}_{K+1}, ..., \mathbf{W}_T)$. The multivariate time series anomaly detection approaches aim to label whether a time step $\mathbf{x}_t$ is abnormal based on a score function $s(\cdot)$ given the time window $\mathbf{W}_t$. If $s(\mathbf{W}_t) > \tau$, then the last time step $\mathbf{x}_t$ will be labeled as abnormal.

When a sliding window $\mathbf{W}_t$ is detected as abnormal, we would like to recommend recourse actions $\boldsymbol{\theta}_t$ on the actionable variables at the time step $\mathbf{x}_t$ to reverse the abnormal outcome. Meanwhile, as the time steps keep coming in, if the following sliding window is still abnormal, we would keep recommending recourse actions until the time series is normal.

As shown in Figure 6.2, in the training phase, once we intervene in a time step $\mathbf{x}_t$, the following time series is in the counterfactual world as the intervention has the downstream impact, which is unobservable. To properly train the model for action recommendations, the key is to derive the counterfactual time series, denoted as $\mathbf{W}_{t+1}^{CF} = (\mathbf{x}_2, ..., \mathbf{x}_t + \boldsymbol{\theta}_t, \mathbf{x}_{t+1}(\boldsymbol{\theta}_t))$, where $\mathbf{x}_{t+1}(\boldsymbol{\theta}_t)$ indicates the counterfactual time step $t+1$ after conducting intervention on $\mathbf{x}_t$. If $\mathbf{W}_{t+1}^{CF}$ is still detected as abnormal, further recourse actions will be recommended on

the counterfactual data $\mathbf{x}_{t+1}(\boldsymbol{\theta}_t)$.

6.4.2 Anomaly Detection for Time Series

In this work, we adopt UnSupervised Anomaly Detection for multivariate time series (USAD) [149] which is a state-of-the-art autoencoder-based anomaly detection model. USAD consists of two autoencoders, i.e., AE_1 and AE_2, with a shared encoder and two independent decoders, and derives the anomaly score function $s(\cdot)$ based on the reconstruction errors of two autoencoders. In the training phase, given a set of normal sliding windows, USAD combines traditional reconstruction-based training with adversarial training to capture the normal patterns of time series. Specifically, reconstruction-based training will let both autoencoders learn how to reproduce the input window $\mathbf{W}$, i.e., $\mathcal{L}_{AE_*} = \|\mathbf{W} - AE_*(\mathbf{W})\|_2$, where AE_* indicates either AE_1 or AE_2. The goal of adversarial training is for AE_1 to deceive AE_2, while AE_2 learns to distinguish between real data and reconstructed data from AE_1, i.e., $\mathcal{L}_{AD} = \min_{AE_1} \max_{AE_2} \|\mathbf{W} - AE_2(AE_1(\mathbf{W}))\|_2$.

After training, the anomaly score of a new window $\mathbf{W}^*$ is then calculated using the combination of reconstruction errors of two autoencoders,

$$s(\mathbf{W}^*) = \alpha\|\mathbf{W} - AE_1(\mathbf{W}^*)\|_2 + \beta\|\mathbf{W}^* - AE_2(AE_1(\mathbf{W}^*))\|_2, \tag{6.4}$$

where α and β are hyperparameters.

6.4.3 Algorithmic Recourse

When a sliding window $\mathbf{W}_t$ is detected as abnormal, we then recommend recourse actions on $\mathbf{x}_t$ with the consideration of downstream impacts.

Recourse on the abnormal time step

Given an abnormal point $\mathbf{x}_t$ and the previous $K-1$ time lags in $\mathbf{W}_t = (\mathbf{x}_{t-K+1}, \ldots, \mathbf{x}_{t-1}, \mathbf{x}_t)$, we formulate the recourse on $\mathbf{x}_t$ as soft intervention,

$$\mathbf{x}_t(\boldsymbol{\theta}_t) = \mathbf{x}_t + \boldsymbol{\theta}_t, \tag{6.5}$$

where $\boldsymbol{\theta}_t$ is the action values on $\mathbf{x}_t$ derived by a function, i.e., $\boldsymbol{\theta}_t = h_\phi(\cdot)$ parameterized by ϕ.

In order to successfully flip the abnormal outcomes, we consider two types of information for recourse prediction via $h_\phi(\cdot)$, time lag exclusion term Δ_t and the past window $\mathbf{W}_{t-1}$. As shown in Equation (6.3), the anomaly in multivariate time series is due to an additional anomaly term. Therefore, we derive the time lag independent term Δ_t at time t as $\Delta_t = \mathbf{x}_t - \hat{\mathbf{x}}_t$, where $\hat{\mathbf{x}}_t$ is the expected values given $\mathbf{W}_{t-1}$ derived by GVAR. As GVAR can simulate the nonlinear multivariate GC functions $\mathcal{F}$, Δ_t contains only independent noise term and anomaly term at time t.

Let $\boldsymbol{\theta}_t = h_\phi(\mathbf{W}_{t-1}, \Delta_t)$ be the function for predicting the recourse action given the previous K time lags $\mathbf{W}_{t-1}$ and Δ_t at time step t parameterized by ϕ, which is defined below:

$$\begin{aligned} \mathbf{z}_{t-1} = LSTM(\mathbf{W}_{t-1}) \quad \mathbf{z}_\Delta = FFNN(\Delta_t) \\ \boldsymbol{\theta}_t = FFNN(\mathbf{z}_{t-1} \oplus \mathbf{z}_\Delta), \end{aligned} \tag{6.6}$$

where $LSTM(\cdot)$ is the long short-term memory (LSTM) neural network; $FFNN(\cdot)$ is a feedforward neural network; and $\oplus$ indicates the vector concatenation operation. In a nutshell, to predict the recourse action, first, we adopt LSTM that takes the past K time lags $\mathbf{W}_{t-1}$ as input and derives a hidden representation $\mathbf{z}_{t-1}$ of the last time step to represent $\mathbf{W}_{t-1}$. Similarly, we adopt a feedforward neural network that takes Δ_t as input to derive the hidden representation $\mathbf{z}_\Delta$. Finally, we use another feedforward neural network for recourse prediction by concatenating $\mathbf{z}_{t-1}$ and $\mathbf{z}_\Delta$ as input.

By applying the action $\boldsymbol{\theta}_t$ on $\mathbf{x}_t$, the counterfactual time step can be computed as $\mathbf{x}_t(\boldsymbol{\theta}_t) = \mathbf{x}_t + \boldsymbol{\theta}_t$. The counterfactual window $\mathbf{W}_t(\boldsymbol{\theta}_t)$ is derived by replacing $\mathbf{x}_t$ with $\mathbf{x}_t(\boldsymbol{\theta}_t)$ in $\mathbf{W}_t$. To train the recourse prediction functions, the objective function is defined as:

$$\mathcal{L}_t(\phi) = \max\{s(\mathbf{W}_t(\boldsymbol{\theta}_t)) - \alpha\tau, 0\} + \lambda\|\mathbf{c} \cdot \boldsymbol{\theta}_t\|_2, \tag{6.7}$$

where $s(\mathbf{W}_t(\boldsymbol{\theta}_t))$ indicates the anomaly score defined in Equation (6.4); λ is a hyperparameter balancing the action values on the anomaly and the flipping of abnormal outcome, α is another hyperparameter controlling how close the anomaly score of the counterfactual sample should be to the threshold τ, $\mathbf{c} \in \mathbb{R}^d$ is a hyperparameter, describing the costs of revising time series (cost vector). Because in USAD, the anomaly is labeled due to a large reconstruction error on the input sample, the first term in the objective function is to ensure the counterfactual variant has a small reconstruction error. The second term, as a regularization term, ensures the minimum **action cost** on the original values.

Inferring the downstream impact. Based on the assumption of Granger causality, the recourse on $\mathbf{x}_t$ leads to the counterfactual variants of the following time steps $\mathbf{x}_{t'}(\boldsymbol{\theta}_t)$, where $t' \geq t$.

To evaluate the impact of the intervention, assuming that $\mathbf{x}_t(\boldsymbol{\theta}_t), \mathbf{x}_{t+1}(\boldsymbol{\theta}_t), \ldots, \mathbf{x}_{t'}(\boldsymbol{\theta}_t)$ is known, where $t' \geq t$, we further derive the counterfactual quantity of the next step $\mathbf{x}_{t'+1}$ by the Abduction-Action-Prediction (AAP) process [150]: 1) Abduction: update the probability $P(u^{(i)}_{t'+1})$ to obtain $P(u^{(i)}_{t'+1}|e)$, where u indicates the exogenous variables and e indicates propositional evidence; 2) Action: variables are intervened to reflect the counterfactual assumption; 3) Prediction: counterfactual reasoning occurs over the new model using updated knowledge.

Formally, based on the causal relationships learned by GVAR, the Abduction-Action-Prediction process to compute the counterfactual value in the $t'+1$-th time step can be described below.

Step 1 (abduction):

$$\mathbf{u}_{t'+1} = \mathbf{x}_{t'+1} - \sum_{k=1}^{K} g_k(\mathbf{x}_{t'+1-k})\mathbf{x}_{t'+1-k} \tag{6.8}$$

Step 2 (action):

$$\mathbf{a}_{t'} = \mathbf{x}_{t'} + \boldsymbol{\theta}_t \text{ if } t' = t \quad \mathbf{a}_{t'} = \mathbf{x}_{t'}(\boldsymbol{\theta}_t) \text{ if } t' > t \tag{6.9}$$

Step 3 (prediction):

$$\mathbf{x}_{t'+1}(\boldsymbol{\theta}_t) = \sum_{k=2}^{K} g_k(\mathbf{x}_{t'+1-k}(\boldsymbol{\theta}_t))\mathbf{x}_{t'+1-k}(\boldsymbol{\theta}_t) + g_1(\mathbf{a}_{t'})\mathbf{a}_{t'} + \mathbf{u}_{t'+1}. \tag{6.10}$$

Equations (6.8)-(6.10) provide the recursive equations for computing the counterfactual time series for L ($L < K$) steps based on the AAP process. The closed form formula for computing counterfactual value $\mathbf{x}_{t+L}(\boldsymbol{\theta}_t)$ can be derived as.

$$\begin{aligned} \mathbf{x}_{t+L}(\boldsymbol{\theta}_t) = &\sum_{l=0}^{L-1} g_{L-l}(\mathbf{x}_{t+l}(\boldsymbol{\theta}_t))\mathbf{x}_{t+l}(\boldsymbol{\theta}_t) \\ &+ \sum_{n=1+L}^{K} g_n(\mathbf{x}_{t+L-n})\mathbf{x}_{t+L-n} + \mathbf{u}_{t+L}, \end{aligned} \tag{6.11}$$

where $\mathbf{u}_{t+L}$ can be derived similar to Equation (6.8).

Because the intervention on the t-th step has the downstream impact, besides ensuring the counterfactual window $\mathbf{W}_t(\boldsymbol{\theta}_t)$ be normal, we would like to make sure that the following L steps are also normal. Therefore, we update the objective function in Equation (6.7) by considering the normality of following L steps,

$$\mathcal{L}(\phi) = \sum_{t'=t}^{t+L} \max\{s(\mathbf{W}_{t'}(\boldsymbol{\theta}_t)) - \alpha\tau, 0\} + \lambda\|\mathbf{c} \cdot \boldsymbol{\theta}_t\|_2. \tag{6.12}$$

Extend to multiple recourse predictions over time series. In practice, a time series could have multiple abnormal time steps, so the algorithmic recourse algorithm should be

able to keep flipping abnormal behavior. However, the challenge during the training phase is that we only observe a sequence of time series with abnormal time steps, but after conducting intervention at a time step t, the following time series in the counterfactual world $\mathcal{X}^{CF}$ is unobservable. Therefore, during the training phase, we need to derive the counterfactual time step, denoted as $\mathbf{x}_{t+1}^{CF}$, based on the AAP process. Then, if the counterfactual window $\mathbf{W}_{t+1}^{CF}$ is still detected as abnormal, i.e., $s(\mathbf{W}_{t+1}^{CF}) > \tau$, meaning that $\mathbf{x}_{t+1}^{CF}$ is still abnormal. We would further train the model $h_\phi(\cdot)$ to recommend recourse actions θ_{t+1} to flip $\mathbf{x}_{t+1}^{CF}$.

Algorithm 5 shows the pseudo-code of the training process for recourse predictions with multiple abnormal time steps over time series. Let $\tilde{\mathcal{X}}^{-}$ be a set of abnormal time series detected by USAD, where each abnormal time series has at least one abnormal time step. For each abnormal time series $\mathcal{X}$ in $\tilde{\mathcal{X}}^{-}$, initializing the counterfactual time series the same as the factual data (Line 2). Note that if there is no intervention conducted yet, the counterfactual time series keeps the same as the observed time series.

Suppose a recourse action is conducted before time step t, we first need to derive the counterfactual time series $\mathcal{X}^{CF}$ (Line 4). To this end, we first get the sliding window $\mathbf{W}_t$ that contains the current time step and the sliding window $\mathbf{W}_{t-1}$ that only contains the previous time steps from the observed time series (Line 15). We further get the sliding window $\mathbf{W}_t^{CF}$ and $\mathbf{W}_{t-1}^{CF}$ from the counterfactual time series (Line 16). Then we can compute the excepted normal values of the factual world for the current time step t by GVAR according to Equation (6.2) (Line 17). The time lag exclusion term $\mathbf{u}_t$ can be derived by Equation (6.8) (Line 18). Similarly, we further compute the excepted normal values of the counterfactual world for the current time step t by GVAR (Line 19). As $\hat{\mathbf{x}}_t^{CF}$ only includes the effects of the previous time steps without any time lag exclusion values (i.e., noise term and anomaly term) for the current time step, the accurate counterfactual values for the current time step t needs to consider the time lag exclusion term $\mathbf{u}_t$ (Line 20). The $\mathcal{X}^{CF}$ and $\mathbf{W}_t^{CF}$ values need to update each time step after a recourse action is conducted (Line 21).

If the updated sliding window $\mathbf{W}_t^{CF}$ is still detected as an anomaly (Line 5), we need to keep predicting the recourse action (Line 6. Specifically, we first compute the action

values with $h_\phi(\cdot)$ (Line 27). The counterfactual values of the current time step t can be calculated by Equation (6.9) (Line 28). Then we further update the values at time step t of counterfactual time series $\mathbf{x}_t^{CF}$ with values $\mathbf{x}_t(\theta_t)$, meaning that another recourse action is conducted (Line 8).

During the training phase, we expect $h_\phi(\cdot)$ can recommend the recourse action with the consideration of its downstream impact on future time steps. Therefore, we compute the counterfactual values for the following L steps according to Equation (6.11) (Line 9). Then, we update the parameters in $h_\phi(\cdot)$ based on the objective function defined in Equation (6.12).

Recourse prediction in the test phase

After completing the training process, the recourse prediction function $h_\phi(\cdot)$ is capable of predicting the appropriate recourse recommendations for each detected anomaly. In the test phase, the multivariate time series is considered as streaming data that is continuously fed into our framework. USAD starts by analyzing the first K time steps to detect any anomalies. If an anomaly is detected with the K time step, RecAD will utilize the information gathered up to that point $\mathbf{W}_{K-1}$ to make a recourse recommendation for the last time step $\mathbf{x}_K$. Then, different from the training phase, where we can only derive the counterfactual time series after intervention based on the AAP process, as the time series comes in as a stream, we can directly observe the following time series after the intervention. We will continue monitoring the incoming data to detect any anomalies and further recommend recourse actions once an abnormal time step is detected. This process will continue as long as the system is receiving input data.

6.5 Experiments

6.5.1 Experimental Setups

Algorithm 5: Training Procedure of RecAD

Input : Pretrained GVAR $g_k(\cdot)$, pretrained anomaly detector $s(\cdot)$, anomaly set $\tilde{\mathcal{X}}^-$

Output: $h_\phi(\cdot)$ for action prediction

foreach $\mathcal{X} \in \tilde{\mathcal{X}}^-$ **do**
  $t \leftarrow K$; $\mathcal{X}^{CF} \leftarrow \mathcal{X}$; $T \leftarrow$ length of $\mathcal{X}$
  while $t \leq T$ **do**
    $\mathcal{X}^{CF}$, $\mathbf{W}_t^{CF}$, $\mathbf{W}_t \leftarrow$ Counterfactual_Generation($\mathcal{X}$, $\mathcal{X}^{CF}$, $g_k(\cdot)$)
    if $s(\mathbf{W}_t^{CF}) > \tau$ **then**
      $\mathbf{x}_t(\theta_t)$, $h_\phi(\cdot) \leftarrow$ Action_Prediction($\mathbf{W}_t$, $\mathbf{W}_t^{CF}$, $g_k(\cdot)$, $h_\phi(\cdot)$)
      Update $\mathbf{x}_t^{CF}$ with $\mathbf{x}_t(\theta_t)$
      Update $\mathcal{X}^{CF}$ with $\mathbf{x}_t^{CF}$
      Compute $\mathbf{x}_{t+1+L}(\theta_t)$ with Eq. (6.11)
      Compute $\mathcal{L}(\phi)$ according to Eq. (6.12)
      Update $\phi = \phi - \eta \frac{\partial \mathcal{L}(\phi)}{\partial \phi}$
    t ++
Return $h_\phi(\cdot)$
Function `Counterfactual_Generation`($\mathcal{X}$, $\mathcal{X}^{CF}$, $g_k(\cdot)$)**:**
  Get $\mathbf{W}_t$ from $\mathcal{X}$, $\mathbf{W}_{t-1} = \mathbf{W}_t \setminus \{\mathbf{x}_t\}$
  Get $\mathbf{W}_t^{CF}$ from $\mathcal{X}^{CF}$, $\mathbf{W}_{t-1}^{CF} = \mathbf{W}_t^{CF} \setminus \{\mathbf{x}_t^{CF}\}$
  Compute $\hat{\mathbf{x}}_t = \sum_{k=1}^{K-1} g_k(\mathbf{x}_{t-k})\mathbf{x}_{t-k}$ with $\mathbf{W}_{t-1}$
  Compute $\mathbf{u}_t = \mathbf{x}_t - \hat{\mathbf{x}}_t$ ▷ `Abduction`
  Compute $\hat{\mathbf{x}}_t^{CF} = \sum_{k=1}^{K-1} g_k(\mathbf{x}_{t-k}^{CF})\mathbf{x}_{t-k}^{CF}$ with $\mathbf{W}_{t-1}^{CF}$
  $\mathbf{x}_t^{CF} = \hat{\mathbf{x}}_t^{CF} + \mathbf{u}_t$ ▷ `Prediction`
  Update $\mathcal{X}^{CF}$ and $\mathbf{W}_t^{CF}$ with $\hat{\mathbf{x}}_t^{CF}$
  Return $\mathcal{X}^{CF}$, $\mathbf{W}_t^{CF}$, $\mathbf{W}_t$
Function `Action_Prediction`($\mathbf{W}_t$, $\mathbf{W}_t^{CF}$, $g_k(\cdot)$, $h_\phi(\cdot)$)**:**
  $\mathbf{W}_{t-1} \leftarrow \mathbf{W}_t \setminus \{\mathbf{x}_t\}$; $\mathbf{W}_{t-1}^{CF} \leftarrow \mathbf{W}_t^{CF} \setminus \{\mathbf{x}_t^{CF}\}$
  Compute $\hat{\mathbf{x}}_t = \sum_{k=1}^{K-1} g_k(\mathbf{x}_{t-k})\mathbf{x}_{t-k}$ with $\mathbf{W}_{t-1}$
  Compute $\Delta_t = \mathbf{x}_t - \hat{\mathbf{x}}_t$
  Compute $\boldsymbol{\theta}_t = h_\phi(\mathbf{W}_{t-1}^{CF}, \Delta_t)$
  $\mathbf{x}_t(\theta_t) = \mathbf{x}_t^{CF} + \boldsymbol{\theta}_t$ ▷ `Action`
  Return $\mathbf{x}_t(\theta_t)$, $h_\phi(\cdot)$

Datasets

We conduct experiments on two semi-synthetic datasets and one real-world dataset. The purposes of using semi-synthetic datasets are as follows. 1) We can derive the ground truth downstream time series after the intervention on the abnormal time step based on the data generation equations in the test phase. 2) We can evaluate the fine-grained performance of RecAD by injecting different types of anomalies.

Linear Dataset [165] is a **synthetic** time series dataset with linear interaction dynamics. We adopt the structural equations defined in [165] that are

defined as:

$$\begin{aligned} x_t^{(1)} &= a_1 x_{t-1}^{(1)} + u_t^{(1)} + \epsilon_t^{(1)}, \\ x_t^{(2)} &= a_2 x_{t-1}^{(2)} + a_3 x_{t-1}^{(1)} + u_t^{(2)} + \epsilon_t^{(2)}, \\ x_t^{(3)} &= a_4 x_{t-1}^{(3)} + a_5 x_{t-1}^{(2)} + u_t^{(3)} + \epsilon_t^{(3)}, \\ x_t^{(4)} &= a_6 x_{t-1}^{(4)} + a_7 x_{t-1}^{(2)} + a_8 x_{t-1}^{(3)} + u_t^{(4)} + \epsilon_t^{(4)}, \end{aligned} \tag{6.13}$$

where coefficients $a_i \sim \mathcal{U}([-0.8, -0.2] \cup [0.2, 0.8])$, additive innovation terms $u_t^{(\cdot)} \sim \mathcal{N}(0, 0.16)$, and anomaly term $\epsilon_t^{(\cdot)}$.

Abnormal behavior injection. For non-causal point anomalies, the anomaly term is single or multiple extreme values for randomly selected time series variables at a specific time step t. For example, a point anomaly at time step t can be generated with an abnormal term $\boldsymbol{\epsilon}_t = [0, 2, 4, 0]$, which means the second and third time series have extreme values.

For non-causal sequence anomalies, the anomaly terms are function-generated values in a given time range. For instance, setting $\epsilon_{t+i}^{(1)} = 0.1 \times i$, for $0 \leq i \leq n$, will cause a trend anomaly for time series variable $x^{(1)}$; setting $\epsilon_{t+i}^{(1)} \sim \mathcal{N}(0, 0.16)$, for $0 \leq i \leq n$, will cause a shapelet anomaly; and setting $\epsilon_{t+i}^{(1)} = (a_1 x_{t+2i-1}^{(1)} + u_{t+2i}^{(1)}) + (a_1 x_{t+2i-2}^{(1)} + u_{t+2i-1}^{(1)}) - (a_1 x_{t+i-1}^{(1)} + u_{t+i}^{(1)})$, for $0 \leq i \leq n$, will cause a seasonal anomaly.

For causal sequence anomalies, we consider two scenarios: 1) changing the coefficients $\mathcal{A} = \{a_1, a_2, \cdots, a_8\}$ from a normal one to a different one in a time range t to $t + n$; 2)

changing generative functions from the original equation to the following equation:

$$\begin{aligned} x_t^{(1)} &= a_1 x_{t-1}^{(1)} + a_2 x_{t-1}^{(3)} + a_3 x_{t-1}^{(4)} + u_t^{(1)} + \epsilon_t^{(1)}, \\ x_t^{(2)} &= a_4 x_{t-1}^{(2)} + a_5 x_{t-1}^{(1)} + u_t^{(2)} + \epsilon_t^{(2)}, \\ x_t^{(3)} &= a_6 x_{t-1}^{(3)} + u_t^{(3)} + \epsilon_t^{(3)}, \\ x_t^{(4)} &= a_7 x_{t-1}^{(4)} + u_t^{(4)} + \epsilon_t^{(4)}. \end{aligned} \tag{6.14}$$

Lotka-Volterra [170] is another **synthetic** time series model that simulates a prairie ecosystem with multiple species. We follow the structure from [165], which defines as:

$$\begin{aligned} \frac{d\mathbf{x}^{(i)}}{dt} &= \alpha \mathbf{x}^{(i)} - \beta \sum_{j \in Pa(\mathbf{x}^{(i)})} \mathbf{y}^{(j)} - \eta (\mathbf{x}^{(i)})^2, \text{ for } 1 \leq j \leq p, \\ \frac{d\mathbf{y}^{(j)}}{dt} &= \delta \mathbf{y}^{(j)} \sum_{k \in Pa(\mathbf{y}^{(j)})} \mathbf{x}^{(k)} - \rho \mathbf{y}^{(j)}, \text{ for } 1 \leq j \leq p, \\ x_t^{(i)} &= x_t^{(i)} + \epsilon_t^{(i)}, \text{ for } 1 \leq j \leq p, \\ y_t^{(j)} &= y_t^{(j)} + \epsilon_t^{(j)}, \text{ for } 1 \leq j \leq p, \end{aligned} \tag{6.15}$$

where $\mathbf{x}^{(i)}$ and $\mathbf{y}^{(j)}$ denote the population sizes of prey and predator, respectively; $\alpha, \beta, \eta, \delta, \rho$ are parameters that decide the strengths of interactions, $Pa(\mathbf{x}^{(i)})$ and $Pa(\mathbf{y}^{(j)})$ correspond the Granger Causality between prey and predators for $\mathbf{x}^{(i)}$ and $\mathbf{y}^{(j)}$ respectively, and $\epsilon_t^{(\cdot)}$ is the abnormal term. We adopt 10 prey species and 10 predator species.

Abnormal behavior injection. We adopt similar strategies as used in the Linear Dataset to inject abnormal behavior.

For point anomalies and non-causal sequence anomalies, we perform a similar procedure as the linear dataset, i.e., randomly select time series variables at a specific time step t and assign single or multiple extreme values as point anomalies, and assign function-generated abnormal terms for a time range from t to $t + n$ as sequence anomalies.

For causal sequence anomalies, we still consider two scenarios: 1) changing the coefficients $\alpha, \beta, \eta, \delta, \rho$ to different values than the normal ones; 2) changing $Pa(\mathbf{x}^{(i)})$ and $Pa(\mathbf{y}^{(j)})$ to different ones from the original generative functions Equation (6.15).

Multi-Source Distributed System (MSDS) [148] is a **real-world** dataset that contains distributed traces, application logs, and metrics from an OpenStack testbed. MSDS consists 10-dimensional time series. The fault injections are treated as anomalies. The first half of MSDS without fault injection is used as a training set, while the second half includes 5.37% time steps as fault injections, which is used as a test set. As the real-world dataset, we cannot observe the downstream time series after the intervention. Therefore, in the test phase, we use GVAR and AAP to generate the counterfactual time series for evaluation.

Table 6.1: Statistics of three datasets for anomaly detection.

Dataset	Dim.	Train	Test (Anomalies %)		
			Point	Non-causal Seq.	Causal Seq.
Linear	4	50,000	250,000 (2%)	250,000 (6%)	250,000 (6%)
Lotka-Volterra	20	100,000	500,000 (1%)	500,000 (3%)	500,000 (3%)
MSDS	10	146,340	146,340 (5.37%)		

Table 6.1 shows the statistics of three datasets. Training datasets only consist of normal time series. Note that the test sets listed in Table 6.1 are used for evaluating the performance of anomaly detection. After detecting the abnormal time series in the test set, for the synthetic datasets, we use 50% of abnormal time series for training RecAD and another 50% for evaluating the performance of RecAD on recourse prediction, while for the MSDS dataset, we use 80% of abnormal time series for training RecAD and the rest 20% for evaluation.

Baselines

To our best knowledge, there is no causal algorithmic recourse approach in time series anomaly detection. We compare RecAD with the following baselines: 1) Multilayer perceptron (MLP) which is trained with the normal flattened sliding windows to predict the normal values for the next step; 2) LSTM which can capture the information over long periods of time and learn complex temporal dependencies to make predictions for the next step; 3) Vector Autoregression (VAR) is a statistical model that used to analyze GC within multivariate time series data and predict future values; 4) Generalised Vector Autoregression (GVAR) [165] is an extension of self-explaining neural network that can infer nonlinear

multivariate GC and predict values of the next step.

For all the baselines, in the training phase, we train them to predict the last value in a time window on the normal time series so that they can capture the normal patterns. In the testing phase, when a time window is detected as abnormal by USAD for the first time, indicating the last time step $\mathbf{x}_t$ is abnormal, we use baselines to predict the expected normal value in the last time step $\tilde{\mathbf{x}}_t$. Then, the recourse action values can be derived as $\theta_t = \tilde{\mathbf{x}}_t - \mathbf{x}_t$. For the sequence anomalies, we keep using the baselines to predict the expected normal values and derive the action values by comparing them with the observed values.

Evaluation Metrics

We evaluate the performance of algorithmic recourse based on the following three metrics.

1) **Flipping ratio**, which is to show the effectiveness of algorithms for algorithmic recourse.

$$\text{Flipping Ratio} = \frac{\text{Number of flipped time steps}}{\text{All detected abnormal time steps}}$$

2) **Action cost** per multivariate time series, which is to check the efficiency of predicted actions.

$$\text{Action Cost} = \frac{\text{Total action cost}}{\#\text{ of abnormal multivariate time series}},$$

where the "Total action cost" indicates the action cost ($\|\mathbf{c} \cdot \boldsymbol{\theta}_t\|_2$) to flip all the abnormal data in the test set.

3) **Action step** per multivariate time series, which is to show how many action steps are needed to flip the abnormal time series.

$$\text{Action Step} = \frac{\text{Total number of action time steps}}{\#\text{ of abnormal multivariate time series}}$$

Implementation Details

Similar to [149], we adopt a sliding window with sizes 5, 5, and 10 for the Linear, Lotka-Volterra, and MSDS datasets, respectively. We set the hyperparameters for GVAR

by following [165]. When training $h_\phi(\cdot)$, we set L in the objective function as $L = 1$, which is to ensure the following one time step should be normal. The cost vector $\mathbf{c}$ can be changed according to the requirements or prior knowledge. Because the baseline models are prediction-based models that cannot take the cost into account, to be fair, we use $\mathbf{1}$ as the cost vector. For baselines, MLP is a feed-forward neural network with a structure of $((K-1)*d)$-100-100-100-d that the input is the flattened vector of $K-1$ time steps with d dimensions and the output is the predicted value of the next time step. The LSTM model consists of one hidden layer with 100 dimensions and is connected with a fully connected layer with a structure of 100-d. We use statsmodels[1] to implement the VAR model. The baseline GVAR model is the same as GVAR within our framework. To implement $h_\phi(\cdot)$ in RecAD, we utilize an LSTM that consists of one hidden layer with 100 dimensions and a feed-forward network with structure d-100. Then we use another feed-forward network with a structure of 200-d to predict the intervention values. Our code is available online[2].

6.5.2 Experimental Results

Evaluation Results on Synthetic Datasets.

We first report the experimental results with standard deviations over 10 runs on synthetic datasets as we can conduct more fine-grained evaluations on synthetic datasets.

Table 6.2: Anomaly detection on synthetic datasets.

Anomaly Types	Metrics	Linear	Lotka-Volterra
Non-causal Point	F1	$0.749_{\pm 0.022}$	$0.787_{\pm 0.106}$
	AUC-PR	$0.619_{\pm 0.024}$	$0.840_{\pm 0.116}$
	AUC-ROC	$0.816_{\pm 0.016}$	$0.851_{\pm 0.068}$
Non-causal Seq.	F1	$0.878_{\pm 0.011}$	$0.677_{\pm 0.061}$
	AUC-PR	$0.798_{\pm 0.015}$	$0.519_{\pm 0.026}$
	AUC-ROC	$0.914_{\pm 0.010}$	$0.794_{\pm 0.089}$
Causal Seq.	F1	$0.756_{\pm 0.003}$	$0.714_{\pm 0.020}$
	AUC-PR	$0.604_{\pm 0.004}$	$0.559_{\pm 0.016}$
	AUC-ROC	$0.877_{\pm 0.002}$	$0.824_{\pm 0.078}$

[1] https://www.statsmodels.org/
[2] https://www.tinyurl.com/RecAD2023

The performance of anomaly detection. We evaluate the performance of USAD for anomaly detection in terms of the F1 score, the area under the precision-recall curve (AUC-PR), and the area under the receiver operating characteristic (AUC-ROC) on two synthetic datasets. Table 6.2 shows the evaluation results. Overall, USAD can achieve promising performance on different types of anomalies, which lays a solid foundation for recourse prediction.

Table 6.3: The performance of recourse prediction on non-causal anomaly.

Model	Linear						Lotka-Volterra					
	Point			Seq.			Point			Seq.		
	Flipping Ratio ↑	Action Cost ↓	Action Step ↓	Flipping Ratio ↑	Action Cost ↓	Action Step ↓	Flipping Ratio ↑	Action Cost ↓	Action Step ↓	Flipping Ratio ↑	Action Cost ↓	Action Step ↓
MLP	$0.778_{\pm 0.054}$	$8.406_{\pm 0.257}$	$1.188_{\pm 0.051}$	$0.867_{\pm 0.048}$	$22.394_{\pm 0.545}$	$2.261_{\pm 0.027}$	$0.741_{\pm 0.126}$	$277.580_{\pm 117.126}$	$1.237_{\pm 0.180}$	$0.688_{\pm 0.259}$	761.550 ± 64.422	$2.195_{\pm 0.757}$
LSTM	$0.807_{\pm 0.045}$	$8.383_{\pm 0.253}$	$1.170_{\pm 0.040}$	$0.878_{\pm 0.044}$	$22.439_{\pm 0.553}$	$2.248_{\pm 0.031}$	$0.893_{\pm 0.087}$	$313.510_{\pm 124.921}$	$1.096_{\pm 0.061}$	$0.889_{\pm 0.097}$	$1590.483_{\pm 85.126}$	$1.339_{\pm 0.145}$
VAR	$0.676_{\pm 0.063}$	$8.841_{\pm 0.224}$	$1.311_{\pm 0.077}$	$0.765_{\pm 0.061}$	$23.696_{\pm 0.511}$	$2.434_{\pm 0.037}$	$0.558_{\pm 0.166}$	$326.967_{\pm 126.322}$	$1.57_{\pm 0.324}$	$0.504_{\pm 0.151}$	$1445.084_{\pm 189.943}$	$2.570_{\pm 0.676}$
GVAR	$0.775_{\pm 0.053}$	$8.446_{\pm 0.249}$	$1.207_{\pm 0.052}$	$0.848_{\pm 0.051}$	$22.415_{\pm 0.547}$	$2.287_{\pm 0.029}$	$0.493_{\pm 0.332}$	$270.335_{\pm 117.223}$	$2.020_{\pm 1.049}$	$0.606_{\pm 0.266}$	$\mathbf{749.792_{\pm 105.656}}$	$2.547_{\pm 0.905}$
RecAD	$\mathbf{0.901_{\pm 0.035}}$	$\mathbf{8.201_{\pm 0.176}}$	$\mathbf{1.104_{\pm 0.024}}$	$\mathbf{0.944_{\pm 0.041}}$	$\mathbf{21.264_{\pm 0.947}}$	$\mathbf{2.193_{\pm 0.046}}$	$\mathbf{0.915_{\pm 0.088}}$	$\mathbf{262.759_{\pm 99.008}}$	$\mathbf{1.085_{\pm 0.055}}$	$\mathbf{0.972_{\pm 0.016}}$	$1374.112_{\pm 343.470}$	$\mathbf{1.329_{\pm 0.192}}$

The performance of recourse prediction on non-causal anomaly. The non-causal anomaly encompasses both point and sequential anomalies. Table 6.3 shows the performance of RecAD for recourse prediction on non-causal anomaly. First, in all settings, RecAD can achieve the highest flipping ratios, which shows the effectiveness of RecAD on flipping abnormal behavior. Meanwhile, RecAD can achieve low or comparable action costs and action steps compared with other baselines. Although some baselines can achieve lower action costs in some settings, this could be due to the low flipping ratios. More importantly, all baselines do not consider the downstream impact of recourse actions.

Table 6.4: The performance of recourse prediction on causal anomaly.

Dataset	Model	Flipping Ratio ↑	Action Cost ↓	Action Step ↓
Linear	MLP	$0.884_{\pm 0.023}$	$41.387_{\pm 2.151}$	$2.359_{\pm 0.035}$
	LSTM	$0.903_{\pm 0.021}$	$42.412_{\pm 2.002}$	$2.398_{\pm 0.043}$
	VAR	$0.782_{\pm 0.035}$	$59.346_{\pm 3.285}$	$2.883_{\pm 0.062}$
	GVAR	$0.874_{\pm 0.024}$	$39.474_{\pm 2.116}$	$2.415_{\pm 0.041}$
	ReccAD	$\mathbf{0.919_{\pm 0.037}}$	$\mathbf{38.917_{\pm 6.247}}$	$\mathbf{2.169_{\pm 0.206}}$
Lotka-Volterra	MLP	$0.665_{\pm 0.277}$	$\mathbf{1578.597_{\pm 49.253}}$	$2.247_{\pm 0.744}$
	LSTM	$0.890_{\pm 0.099}$	$3159.333_{\pm 173.463}$	$2.310_{\pm 0.100}$
	VAR	$0.488_{\pm 0.178}$	$3088.788_{\pm 435.034}$	$2.630_{\pm 0.640}$
	GVAR	$0.584_{\pm 0.277}$	$1846.415_{\pm 347.144}$	$2.618_{\pm 0.858}$
	ReccAD	$\mathbf{0.970_{\pm 0.012}}$	$2767.819_{\pm 581.046}$	$\mathbf{1.386_{\pm 0.120}}$

The performance of recourse prediction on causal anomaly. We examine the performance of RecAD on causal anomaly. The results are shown in Table 6.4. First, RecAD can achieve the highest flipping ratio compared with baselines on both Linear and Lotka-Volterra datasets. High flipping ratios on both datasets indicate that the majority of causal anomalies can be successfully flipped. Meanwhile, RecAD can also achieve low action costs and action steps with high flipping ratios. Although MLP can achieve the lowest action cost on the Lotka-Volterra dataset, the flipping ratio achieved by MLP is much lower than RecAD. Overall, RecAD meets the requirement of algorithmic recourse, i.e., flipping the abnormal outcome with minimum costs, on causal anomalies.

Therefore, based on Tables 6.3 and 6.4, we can demonstrate that RecAD can provide recourse prediction on different types of anomalies in multivariate time series.

Evaluation Results on Real Dataset

We further report the experimental results with standard deviations over 10 runs on MSDS.

The performance of anomaly detection. We first evaluate the performance of USAD for anomaly detection. USAD can achieve $0.888_{\pm 0.097}$, $0.996_{\pm 0.001}$, and $0.985_{\pm 0.003}$ in terms of F1 score, AUC-PR, and AUC-ROC, respectively. It means USAD can find most of the anomalies in the MSDS dataset.

Table 6.5: The performance of recourse prediction in MSDS.

Model	Flipping Ratio ↑	Action Cost ↓	Action Step ↓
MLP	$0.687_{\pm 0.282}$	$6.848_{\pm 2.506}$	$1.443_{\pm 0.680}$
LSTM	$0.830_{\pm 0.211}$	$6.798_{\pm 2.604}$	$1.279_{\pm 0.493}$
VAR	$0.704_{\pm 0.273}$	$6.759_{\pm 2.821}$	$1.432_{\pm 0.596}$
GVAR	$0.712_{\pm 0.211}$	$8.923_{\pm 3.258}$	$1.425_{\pm 0.466}$
RecAD	$\mathbf{0.841_{\pm 0.080}}$	$\mathbf{6.747_{\pm 1.543}}$	$\mathbf{1.249_{\pm 0.068}}$

The performance of recourse prediction. Because for the real-world dataset, we do not know the types of anomalies, we report the performance of recourse prediction on any detected anomalies. As shown in Table 6.5, RecAD achieves the flipping ratio of 0.84,

meaning that RecAD can flip 84.1% of detected abnormal time steps, much higher than all baselines. Regarding the average action cost per time series and the average action step, RecAD also outperforms the baselines by registering the lowest values. This suggests that, by incorporating Granger causality, RecAD is capable of identifying recourse actions that minimize both cost and the number of action steps.

Table 6.6: The performance of recourse prediction using different components of RecAD.

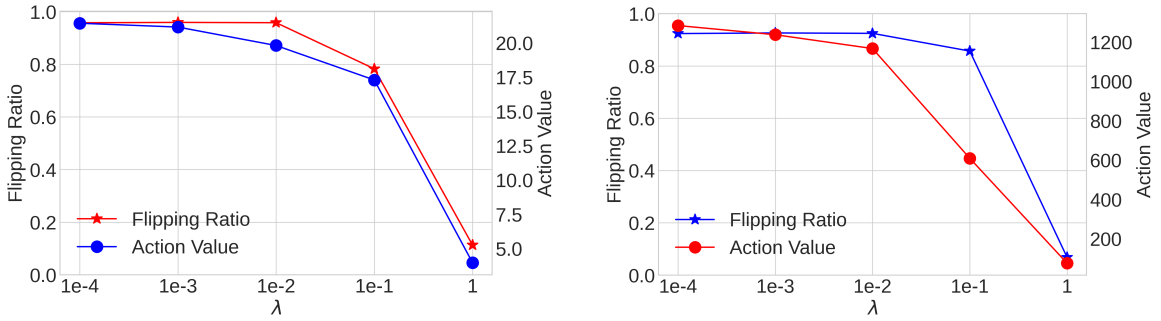
Dataset	Metric	RecAD w/o FFNN	RecAD w/o LSTM	RecAD
Linear	Flipping Ratio ↑	$0.340_{\pm 0.191}$	$0.676_{\pm 0.085}$	$\mathbf{0.922_{\pm 0.040}}$
	Action Cost ↓	$119.286_{\pm 174.173}$	$23.589_{\pm 23.453}$	$\mathbf{22.794_{\pm 13.054}}$
	Action Step ↓	$2.905_{\pm 0.882}$	$2.276_{\pm 0.939}$	$\mathbf{1.822_{\pm 0.521}}$
Lotka-Volterra	Flipping Ratio ↑	$0.353_{\pm 0.210}$	$0.523_{\pm 0.090}$	$\mathbf{0.952_{\pm 0.054}}$
	Action Cost ↓	$\mathbf{876.107_{\pm 810.047}}$	$1035.192_{\pm 881.242}$	$1468.230_{\pm 1086.090}$
	Action Step ↓	$2.706_{\pm 1.068}$	$2.304_{\pm 0.646}$	$\mathbf{1.266_{\pm 0.180}}$
MSDS	Flipping Ratio ↑	$0.228_{\pm 0.147}$	$0.697_{\pm 0.214}$	$\mathbf{0.841_{\pm 0.080}}$
	Action Cost ↓	$\mathbf{3.494_{\pm 2.665}}$	$4.136_{\pm 0.727}$	$6.747_{\pm 1.543}$
	Action Step ↓	$2.048_{\pm 0.607}$	$1.581_{\pm 0.525}$	$\mathbf{1.249_{\pm 0.068}}$

Ablation Study

We evaluate the performance of using different parts of RecAD (i.e., FFNN and LSTM) for recourse prediction. As RecAD contains an LSTM to catch the previous $K-1$ time lags and a feedforward neural network (FFNN) to include the time lag exclusion term Δ_t, we then test the performance of these two parts separately. Table 6.6 shows the average flipping ratio, action cost, and action step for three types of anomalies for the synthetic datasets and results for the real-world dataset MSDS. We can notice that RecAD achieves higher flipping ratios and lower action steps than the one only using a part of RecAD. It shows the importance of considering both information for reasonable action value prediction.

Sensitivity Analysis

The objective function (Equation (6.12)) for training RecAD employs the hyperparameter λ to balance the flipping ratio and action value. As shown in Figure 6.3, on both synthetic datasets, we have similar observations that with the increasing of λ, both action



(a) Linear (b) Lotka-Volterra

Fig. 6.3: Effects of the hyperparameter λ in Eq. (6.12).

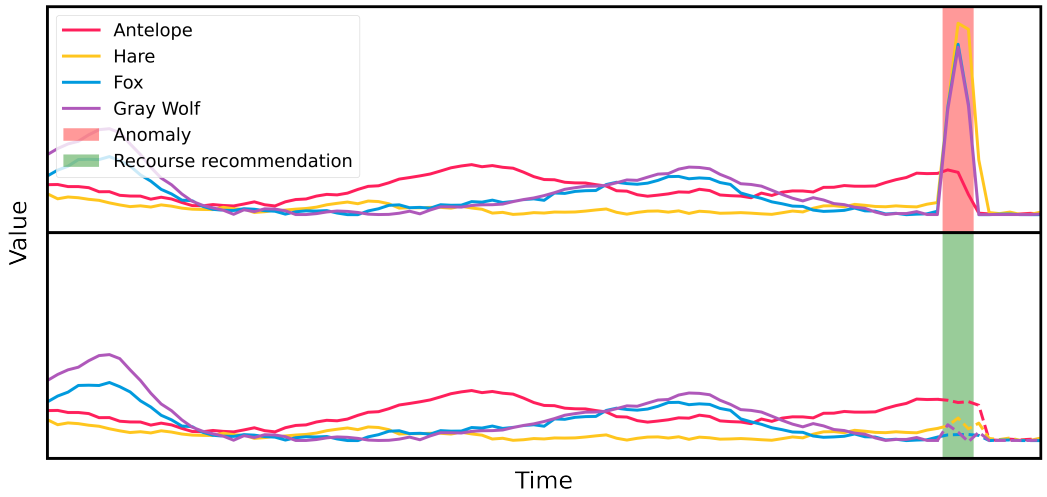


Fig. 6.4: Recourse recommendations for intervening in an imbalanced ecosystem to restore balance.

value and flipping ratio decrease. A large λ indicates a large penalty for high action values, which could potentially hurt the performance of flipping abnormal time steps as small action values may not be sufficient to flip the anomalies.

Case Study

We further conduct case studies to show how to use the recourse action predicted by RecAD as an explanation for anomaly detection in multivariate time series.

Case study on the Lotka-Volterra dataset. Figure 6.4 shows a simulation of a prairie ecosystem that contains antelope, hare, fox, and gray wolf based on the Lotka-Volterra model [170], where each time series indicates the population of a species. As shown in the

top figure, in most of the time steps, the numbers of carnivores (fox and gray wolf) and herbivores (antelope and hare) keep stable in a balanced ecosystem, say 0.1k-1k antelopes, 1k-10k hares, 0.1k-1k foxes, and 0.1k-1k gray wolves. After detecting abnormal behavior at a specific time step (red area in the top figure), the algorithmic recourse aims to provide recourse actions to flip the abnormal outcome. In this case, the algorithmic recourse model recommends the intervention of reducing the populations of hares, foxes, and gray wolves by 100.1k, 9.3k, and 7.5k, respectively. After applying the recourse actions (green area in the bottom figure), we can notice the populations of four species become stable again (the dashed line in the bottom figure). Therefore, the recourse actions can provide recommendations to restore the balance of the prairie ecosystem.

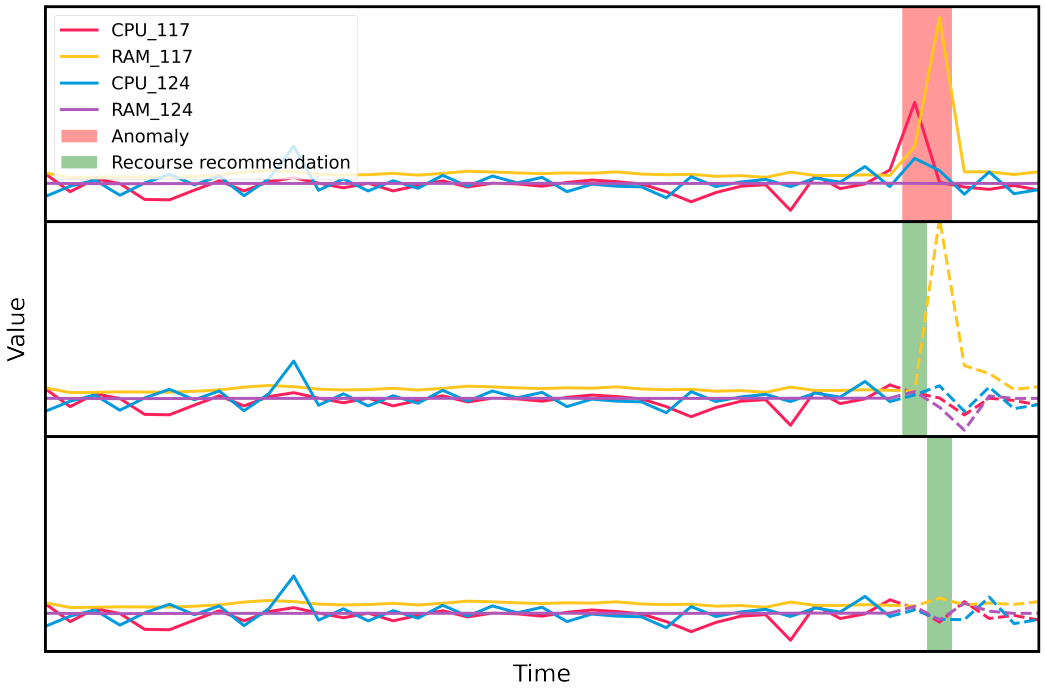


Fig. 6.5: Recourse recommendations for restoring the abnormal CPU and RAM usages in MSDS.

Case study on the MSDS dataset. Figure 6.5 depicts a case study on MSDS with control nodes 117 and 124. USAD detects a subsequence of anomaly consisting of two abnormal time steps from two time series (CPU and RAM usages on node 117), highlighted in the red

area of the top figure.

When the first abnormal time step is detected, RecAD suggests releasing the CPU usage by 6.7 on node 117 (the green area in the middle figure). In other words, it also means the anomaly here is due to the higher CPU usage than normal with a value of 6.7. After taking this action, the following time steps are affected by this action. A counterfactual time series is then generated using the AAP process, which is shown as the dashed lines in the middle of Figure 6.5. RecAD continues to monitor subsequent time steps for any abnormalities.

The following time step is still detected as abnormal in the time series of memory usage of node 117. RecAD recommends releasing the RAM usage by 13.39 on node 117 (the green area in the bottom figure), meaning that the abnormal time step here is due to high memory usage in a margin of 13.39. After taking the recourse action, the counterfactual time series is then generated (the dashed lines in the bottom figure). We can then observe that the entire time series returns to normal.

In summary, recourse actions recommended by RecAD can effectively flip the outcome and lead to a normal counterfactual time series. Meanwhile, based on the recourse actions, the domain expert can understand why a time step is abnormal.

6.6 Summary

In this work, we have developed a novel framework for algorithmic recourse in time series anomaly detection, called RecAD, which can recommend recourse actions to fix anomalies with the minimum cost. To recommend proper actions with the consideration of the downstream impact of the intervention on the current time step, we leverage Granger causality to model the interdependence in multivariate time series and derive the counterfactual time series based on the Abduction-Action-Prediction process. The empirical studies have demonstrated the effectiveness of RecAD for recommending recourse actions in time series anomaly detection. The early version of this work is preprinted at arXiv [171].

CHAPTER 7

ROOT CAUSE ANALYSIS OF ANOMALIES IN MULTIVARIATE TIME SERIES THROUGH GRANGER CAUSAL DISCOVERY

In this chapter, we propose a novel autoencoder-based framework for root cause analysis (AERCA). AERCA captures the Granger causality in multivariate time series via an encoder-decoder structure, which not only learns the causal dependency between time series but also explicitly models the distributions of exogenous variables of time series in the normal status. By defining the anomaly as an intervention on exogenous variables of time series, AERCA identifies the root cause of abnormal time series by highlighting the exogenous variables that are significantly deviated from their normal status. Experiments on multiple synthetic and real-world datasets demonstrate that AERCA can accurately capture the causal relationships between time series and further identify the root cause of abnormal time series.

7.1 Introduction

Root cause analysis, which is to identify the underlying causes of an anomaly, has a wide spectrum of applications in various domains, such as diagnosing the fault of online cloud-based systems or cyber-physical systems [38, 172–174]. For example, cyber-physical systems, such as water plant systems, are usually equipped with multiple sensors to monitor the system status of different entities, e.g., water tanks, and detect abnormal status. Once an abnormal status of the cyber-physical systems is detected, root cause analysis is to effectively locate the abnormal entities leading to the abnormal behavior of the whole system. For instance, the overflow of a downstream water tank in a water plant could be due to an issue of upstream actuators. The importance of root cause analysis lies in its ability to determine why a system fails, enabling domain users to take further actions to mitigate the abnormal status.

When an abnormal status is detected, the traditional approach for detecting the root cause is to manually trace the root cause based on the topological structure of the systems. However, due to the increasing complexity of the system, such an approach becomes challenging and time-consuming. Therefore, developing data-driven approaches for root cause analysis has received a lot of attention [38]. Systems, such as cyber-physical systems, are usually equipped with sensors to monitor their status via various metrics such that the sensor data can be treated as multivariate time series. Analyzing the multivariate time series data can provide insightful information for root cause analysis.

Recently, one line of research has been proposed that first learns the dependency graph from the time series data and then locates the root causes via exploring the dependency graph [175–179]. Building a high-fidelity dependency graph from observational data is crucial for this approach to achieve accurate root cause identification. Typically, the dependency graph is a directed acyclic graph (DAG), where each node indicates an entity in the system, such as a water tank, and a directed edge from node A to node B indicates that node A impacts node B, such as the upstream water tank impacts the downstream water tank. The DAG, a.k.a., the causal graph, can depict the causal relationship between time series to help understand how the abnormal behavior propagates through the system and further improve the effectiveness of locating the root cause. Specifically, the causal graph is learned from the time series data and causal inference techniques are leveraged for root causal analysis. Equipped with a structural causal model (SCM), the root cause is usually caused by the change of the SCM, as illustrated in Figure 7.1.

The current causal discovery approaches for root cause analysis mainly focus on identifying the causal structures among time series/endogenous variables (solid circle in Figure 7.1) without explicitly modeling the impact of exogenous variables (dotted circle in Figure 7.1) in the SCM. However, we argue that it is crucial to explicitly consider exogenous variables in causal discovery for root cause analysis due to the external nature of the factors that cause anomalies. In SCM, an exogenous variable is one whose variation is not accounted for within the model but is considered to be external. For this reason, we model the root

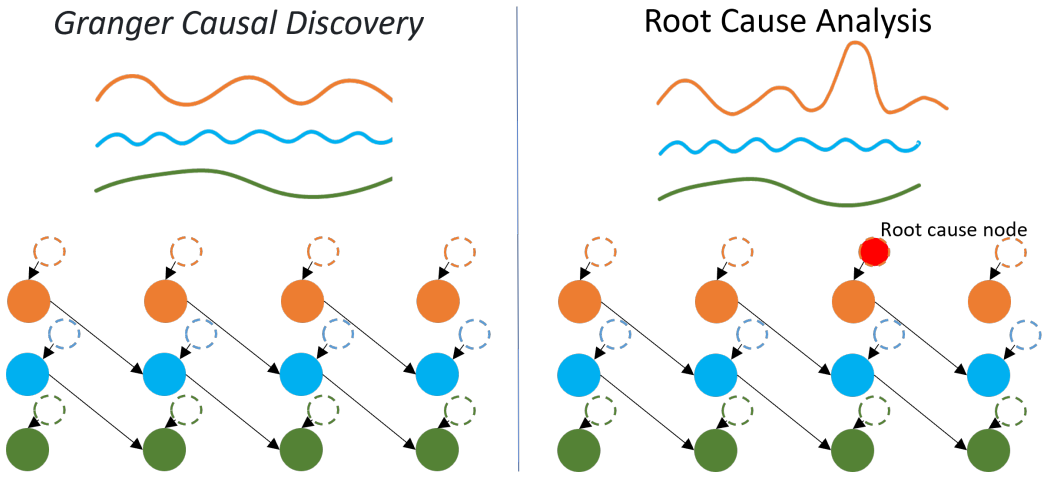


Fig. 7.1: Root cause analysis through Granger causality.

cause of the anomaly as an intervention to the exogenous variables, which we refer to as the **exogenous intervention**. As a result, we assume that the exogenous variables follow a stable distribution in the normal data but undergo exogenous interventions when anomalies occur. For example, if an attacker directly manipulates the sensor reading of the water tank in a water plant, it can be considered an exogenous intervention since all underlying causal relationships among entities in the water plant remain unchanged.

Based on the definition of the exogenous intervention, identifying the root cause is equivalent to detecting the exogenous intervention. Therefore, the key to root cause identification is to model the normality of exogenous variables for all time series and then highlight abnormal exogenous variable values. To this end, we propose a novel autoencoder-based framework for root cause analysis, denoted as AERCA, which can discover both the Granger causal relationship in time series and the distribution of exogenous variables. To model the data generation process, i.e., the causal relationships as well as the distributions of exogenous variables, the encoder aims to model the abductive reasoning process to derive the exogenous variable for each time series. We assume that the exogenous variables are mutually independent and impose constraints to ensure this independence. Meanwhile, the decoder learns a deductive reasoning process to infer the input based on the exogenous variables. We show that to predict the input at time t, rather than using exogenous variables

of all time steps before t, the decoder only needs to take in the exogenous variables and observed time series from a window prior to t. We train AERCA on the normal data. Then, upon deployment, if the encoder-derived values of exogenous variables significantly deviate from the norm, the corresponding time series are highly likely to be the root cause of the anomaly.

The contributions of this work are as follows: 1) we propose a novel encoder-decoder structure for Granger causal discovery, which can not only learn the causal relationships between time series but also capture the distribution of exogenous variables; 2) based on the learned structural causal model, AERCA can not only identify the root cause time series but also highlight the root cause time steps; 3) experimental results on multiple datasets show that AERCA can achieve state-of-the-art performance on both Granger causal discovery and root cause identification.

7.2 Related Work

Understanding the root cause of an anomaly has received increasing attention because of wide real-world applications. Accurate root cause localization can help domain users understand and mitigate abnormal behaviors.

Traditionally, root cause diagnosis is achieved by finding variables that are changed significantly during the failure period [180]. However, this line of approaches cannot consider the impact of other variables, leading to a high false positive.

Recently, the mainstream approaches in root cause analysis follow a two-step framework: identifying the dependency between variables from observational data and then localizing the root cause by exploring the dependency graph. Therefore, the key step is to build the dependency graph. In some scenarios, domain knowledge or a systems tool can be leveraged to build the dependency graph. For example, in a microservice system, a directed edge between two nodes usually indicates a system call [174, 181–183]. After obtaining the dependency graph, when a system metric indicates an abnormal status, the random walk technique is adopted to localize the root cause, where the nodes with high probabilities of visiting have a high chance of being the root cause nodes [183].

However, there are some limitations to leveraging the domain knowledge or a systems tool to build the dependency graph. First, as the system becomes sophisticated, it becomes impractical to build the dependency graph based on domain knowledge. Second, the call graph learned by system tools may not represent the true dependency between sensors [181]. Therefore, data-driven approaches are now commonly used for learning the dependency between variables. For example, various deep neural networks are developed to capture the temporal and spatial correlations in the multivariate time series for root cause analysis [169, 184]. The graph neural network, such as the graph attention network (GAT), is also adopted to learn the dependencies between components, and attention weights learned by GAT represent the dependent strength between components [185].

Recently, causal inference-based root cause analysis has received increasing attention, which models the anomaly as data under intervention [175, 176]. Under this assumption, root cause localization is to identify the intervention on observational data [176]. Several approaches leverage the PC algorithm [186] or its variance to build the causal graph by using the conditional independent test [177, 178]. Some approaches also leverage the graph neural networks to learn the causal relationships between nodes by simulating the data generation process [179, 187].

In this work, we propose a novel approach for Granger causal discovery via explicitly modeling the distribution of exogenous variables in the structural causal model. For the task of root cause identification, most of the existing studies can only locate the root cause time series but cannot further indicate the abnormal time step. In contrast, our approach locates the root cause as the time series receiving exogenous intervention at specific time steps, which is much more informative.

7.3 Preliminary

7.3.1 Structural Causal Model (SCM)

We adopt Pearl's Structural Causal Model (SCM) [150], which is defined below.

Definition 1 *An SCM is a triple* $\mathcal{M} = \{U, V, F\}$ *where*

1) U *is a set of exogenous variables that are determined by factors outside the model and assumed to be unobserved in the model.*

2) V *is a set of endogenous variables/time series that are determined by variables in* $U \cup V$*.*

3) F *is a set of functions* $\{f^{(1)}, \ldots, f^{(d)}\}$*; for each* $X^{(j)} \in V$*, a corresponding function* $f^{(j)}$ *is a mapping from* $U \cup (V \setminus \{X^{(j)}\})$ *to* $X^{(j)}$*, where a set of time series* $X_{\mathsf{PA}^{(j)}} \subseteq V \backslash \{X^{(j)}\}$ *are called the parents of* $X^{(j)}$*.*

An SCM is often illustrated by a causal graph $\mathcal{G}$ where a node represents each observed variable, and the causal relationships are represented by directed edges.

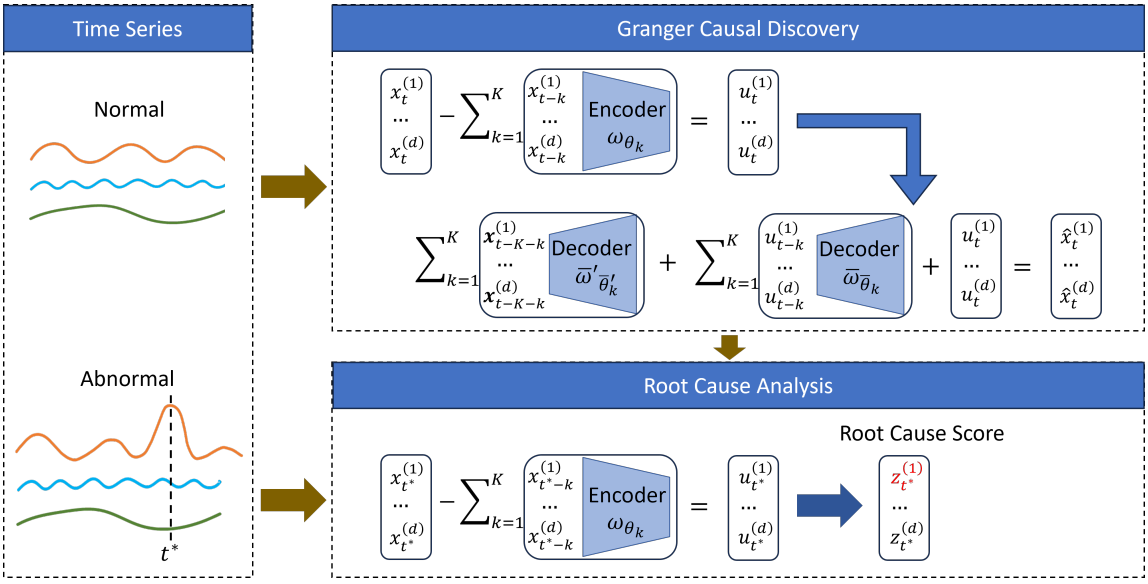


Fig. 7.2: The overview of the proposed AERCA.

7.3.2 Granger Causality

Granger causality [161, 162] is commonly used for modeling causal relationships in multivariate time series. The key assumption is that if the prediction of the future value Y can

be improved by knowing past elements of X, then X "Granger causes" Y. Granger causality was originally defined for linear relationships, while recently, the non-linear Granger causality has been proposed [164, 188]:

Let a stationary time-series as $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_T)$, where $\mathbf{x}_t \in \mathbb{R}^d$ is a d-dimensional vector (e.g., d-dimensional time series data from d sensors) at a specific time t. Suppose that the true data generation mechanism is defined in the form of

$$x_t^{(j)} := f^{(j)}(\mathbf{x}_{\leq t-1}^{(1)}, \cdots, \mathbf{x}_{\leq t-1}^{(d)}) + u_t^{(j)}, \text{ for } 1 \leq j \leq d, \tag{7.1}$$

where $\mathbf{x}_{\leq t-1}^{(j)} = [\cdots, x_{t-2}^{(j)}, x_{t-1}^{(j)}]$ denotes the past of series j; $u_t^{(j)} \in \mathbf{u}^{(j)}$ indicates exogenous variable for time series j at time step t; $\mathcal{F} = \{f^{(1)}, ..., f^{(d)}\}$ is a set of non-linear functions, and $f^{(j)}(\cdot) \in \mathcal{F}$ is a nonlinear function for time series j that captures how the past values impact the future values of $\mathbf{x}^{(j)}$.

The time series i Granger causes j, if $f^{(j)}$ depends on $\mathbf{x}_{\leq t-1}^{(i)}$, i.e., $\exists \mathbf{x}'^{(i)}_{\leq t-1} \neq \mathbf{x}_{\leq t-1}^{(i)}$: $f^{(j)}(\mathbf{x}_{\leq t-1}^{(1)}, \cdots, \mathbf{x}'^{(i)}_{\leq t-1}, \cdots, \mathbf{x}_{\leq t-1}^{(d)}) \neq f^{(j)}(\mathbf{x}_{\leq t-1}^{(1)}, \cdots, \mathbf{x}_{\leq t-1}^{(i)}, \cdots, \mathbf{x}_{\leq t-1}^{(d)})$.

7.4 Problem Formulation

Based on the structural equation of multivariate time series defined in Equation 7.1, in this work, we focus on the anomaly caused by exogenous interventions on a single or multiple time series, e.g., due to attacks on a sensor, leading to a significantly deviating value in its exogenous variable $\hat{u}_t^{(j)}$, which can be defined as

$$\begin{aligned} \tilde{x}_t^{(j)} &= f^{(j)}(\mathbf{x}_{\leq t-1}^{(1)}, \cdots, \mathbf{x}_{\leq t-1}^{(d)}) + \hat{u}_t^{(j)} \\ &= f^{(j)}(\mathbf{x}_{\leq t-1}^{(1)}, \cdots, \mathbf{x}_{\leq t-1}^{(d)}) + u_t^{(j)} + \epsilon_t^{(j)}, \text{ for } 1 \leq j \leq d, \end{aligned} \tag{7.2}$$

where $\hat{u}_t^{(j)} = u_t^{(j)} + \epsilon_t^{(j)}$ with an anomaly term $\epsilon_t^{(j)}$. Note that the abnormal time series caused by exogenous interventions can be either a point anomaly or a sequential anomaly. The point anomaly can be due to an exogenous intervention on a specific time series at a time step. In contrast, a sequence anomaly can be caused by the propagation of an exogenous

intervention through time by following the causal structural model or a consistent exogenous intervention over time steps.

Therefore, an informative root cause analysis shows not just the time series but also the time steps receiving the exogenous intervention. Based on this motivation, we define the task of root cause identification below.

Definition 2 *The root cause identification is to locate the time series/variables* (j) *at specific time step(s)* t *with the abnormal exogenous variable* $\hat{u}_t^{(j)}$.

7.5 AERCA

For the anomaly caused by the exogenous interventions, to achieve the root cause analysis, we model the Granger causality in multivariate time series by explicitly modeling the distribution of exogenous variables. To this end, we develop an encoder-decoder structure for root cause analysis, called AERCA, which can calculate the exogenous variable for each time series at a specific time step. Especially, AERCA explicitly computes the exogenous variables via the encoder, and the decoder predicts the current value by simulating the data generation mechanism defined by the Granger causality. By training the encoder-decoder structure on the normal time series, the model can capture the distribution of exogenous variables in the normal status. When an exogenous intervention occurs, the derived exogenous variables should significantly differ from the normal ones. Meanwhile, because we explicitly derive the exogenous variables at each time step, even if the time series is still abnormal due to the error propagation through time, AERCA can distinguish the root cause from the downstream impact. Figure 7.2 shows the framework of AERCA.

7.5.1 Granger Causal Discovery

Motivation. As shown in Equation 7.1, based on the Granger causality, the value of the time series at step t is a function of past time series plus an exogenous term at the current step, i.e., $\mathbf{x}_t := f(\mathbf{x}_{\leq t-1}) + \mathbf{u}_t$ for simplicity. Then, by recursively resolving the

previous time step, say $\mathbf{x}_{t-1}$, with their previous time step, i.e., $\mathbf{x}_{t-2}$ until the first time step, we can rewrite the Granger causal model as a function of exogenous variables:

$$\mathbf{x}_t = f(\mathbf{u}_{\leq t-1}) + \mathbf{u}_t. \tag{7.3}$$

Equation 7.3 indicates that the observed data at step t is a function of all previous exogenous variables. In the context of the encoder-decoder structure, the observed data can be reconstructed from the exogenous variables, which matches the purpose of the decoder.

On the other hand, we can rewrite Equation 7.1 as

$$\mathbf{u}_t := \mathbf{x}_t - f(\mathbf{x}_{\leq t-1}). \tag{7.4}$$

It means we can derive the exogenous variables based on the observed data, leading to the design of the encoder.

Therefore, we develop an encoder-decoder structure, where the encoder learns Granger causal relationships $f(\cdot)$ by taking the previous time series values as input to compute the exogenous variables by simulating the Equation 7.4, while the decoder takes exogenous variables derived from the encoder as inputs to reconstruct the value of the current time step $\mathbf{x}_t$ by simulating the Equation 7.3.

Encoder-decorder Structure. Given normal multivariate time series $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_T)$, we define a window with length K as $\mathbf{W}_t = (\mathbf{x}_{t-K+1}, ..., \mathbf{x}_t)$ and convert a time series $\mathbf{X}$ to a sequence of sliding windows $\mathcal{W} = (\mathbf{W}_K, \mathbf{W}_{K+1}, ..., \mathbf{W}_T)$. We first aim to learn the Granger causality of time series in a window, i.e., a window causal graph [188].

Given a time series window, the encoder parameterizes the Granger causality in time series defined in Equation 7.1 as

$$\mathbf{x}_t = \sum_{k=1}^{K} \omega_{\theta_k}(\mathbf{x}_{t-k})\mathbf{x}_{t-k} + \mathbf{u}_t, \tag{7.5}$$

where $\omega_{\theta_k}(\mathbf{x}_{t-k})$ indicates the k-th neural network to predict the Granger causal relationship

between x_{t-k} and x_t. The output of $\omega_{\theta_k}(\mathbf{x}_{t-k})$ can be reshaped as a $d \times d$ coefficient matrix, where the entry element (i, j) indicates the influence of $x_{t-k}^{(j)}$ on $x_t^{(i)}$. As shown in Equation 7.5, K neural networks are used to predict the weights of past K time legs on deriving $\mathbf{x}_t$. Therefore, relationships between d time series over K time lags can be explored by inspecting K coefficient matrices.

Because the goal of the encoder is to derive the exogenous variables, we can rewrite the Equation 7.5 as

$$\mathbf{u}_t = \mathbf{x}_t - \sum_{k=1}^{K} \omega_{\theta_k}(\mathbf{x}_{t-k})\mathbf{x}_{t-k}. \tag{7.6}$$

Therefore, given a time series window $\mathbf{W}_t$, we apply the encoder K times to derive the exogenous variables in a window, denoted as $\mathbf{U}_t = (\mathbf{u}_{t-K+1}, \ldots, \mathbf{u}_t)$.

To enforce independence between the derived exogenous variables, we ensure that the distribution of $\mathbf{U}_t$ adheres to an isotropic standard Gaussian distribution Q. By assuming that the exogenous variables follow a multivariate Gaussian distribution and applying the KL divergence to quantify the distribution difference, we formulate the independence constraint as

$$D_t^{KL}(P(\mathbf{U}_t)\|Q) = \mu_t^T \mu_t + tr\{\Sigma_t\} - d - \log|\Sigma_t| \tag{7.7}$$

where μ_t and Σ_t are the mean and covariance matrix of $\mathbf{U}_t$.

The decoder is to reconstruct the input $\mathbf{x}_t$ based on the exogenous variables $\mathbf{U}_t$. One challenge is that theoretically, the value $\mathbf{x}_t$ at the current time step is computed by the exogenous variables of all the previous time steps. However, considering the potential infinite length of the time series, it is impractical to reconstruct $\mathbf{x}_t$ by using all the previous time steps. To tackle this challenge, we iteratively replace the $\mathbf{x}_{t-k}$ with $\mathbf{x}_{t-(k+1)}$ for a subsequence with length n and derive the following equation.

$$\begin{aligned}\mathbf{x}_t = &\sum_{m=2}^{n+1} \mathbf{y}_{n+1-m}\mathbf{u}_{t-(n+1-m)} \\ &+ \mathbf{y}_n\mathbf{x}_{t-n} + \sum_{m=2}^{n+1} \mathbf{y}_{n+1-m} \sum_{k=m}^{K} \omega_k \mathbf{x}_{t-k-(n+1-m)},\end{aligned} \tag{7.8}$$

where ω_k indicates the parameter of Granger causality, and y_n is computed by a recursive equation: $\mathbf{y}_0 = \mathbb{1}$, $\mathbf{y}_1 = \omega_1\mathbf{y}_0$, $\mathbf{y}_2 = \omega_1\mathbf{y}_1 + \omega_2\mathbf{y}_0$, ..., $\mathbf{y}_n = \omega_1\mathbf{y}_{n-1} + \omega_2\mathbf{y}_{n-2} + ... + \omega_n\mathbf{y}_0$.

Equation 7.8 indicates that the value at the current time step $\mathbf{x}_t$ can be derived by the exogenous variables from a previous window $[\mathbf{u}_{t-1}, ..., \mathbf{u}_{t-K}]$ and the observed time series from the immediate previous window $[\mathbf{x}_{t-K-1}, ..., \mathbf{x}_{t-2K}]$.

Inspired by the Equation 7.8, we propose a decoder structure that combines both observed time series and exogenous variables. Specifically, we parameterize the impact of exogenous variable $\mathbf{u}_{t-k}$ on $\mathbf{x}_t$ by a neural network $\bar{\omega}_{\bar{\theta}_k}$ and the impact of observed time series $\mathbf{x}_{t-K-k}$ on $\mathbf{x}_t$ by another neural network $\bar{\omega}'_{\bar{\theta}'_k}$. Then, the decoder computes $\mathbf{x}_t$ based on the following equation.

$$\hat{\mathbf{x}}_t = \sum_{k=1}^{K} \bar{\omega}_{\bar{\theta}_k}(\mathbf{u}_{t-k})\mathbf{u}_{t-k} + \sum_{k=1}^{K} \bar{\omega}'_{\bar{\theta}'_k}(\mathbf{x}_{t-K-k})\mathbf{x}_{t-K-k} + \mathbf{u}_t, \tag{7.9}$$

where $\hat{\mathbf{x}}_t$ indicates the reconstructed value at time step t, and $\mathbf{u}_{t-k}$ is computed by encoder defined in Equation 7.6.

The whole encoder-decoder structure can be defined as $\hat{\mathbf{x}}_t = AE_{\theta_k,\bar{\theta}_k,\bar{\theta}'_k}(\mathbf{x}_{<t})$. Given a time series with length T, the objective function to train the encoder neural netework ω_{θ_k} and decoder neural networks $\bar{\omega}_{\bar{\theta}_k}$, $\omega'_{\bar{\theta}'_k}$ is defined as:

$$\begin{aligned}\mathcal{L} = &\sum_{t=K+1}^{T} \Big\{ \|\hat{\mathbf{x}}_t - \mathbf{x}_t\|_2 + \beta D_t^{KL} + \lambda_{en} R(\Omega_t) + \lambda_{de} R(\bar{\Omega}_t) + \lambda_{de} R(\bar{\Omega}'_t) \Big\} \\ &+ \sum_{t=K+1}^{T-1} \Big\{ \gamma_{en} S(\Omega_{t+1}, \Omega_t) + \gamma_{de} S(\bar{\Omega}_{t+1}, \bar{\Omega}_t) + \gamma_{de} S(\bar{\Omega}'_{t+1}, \bar{\Omega}'_t) \Big\},\end{aligned} \tag{7.10}$$

where D_t^{KL} indicates the independence constraint on $\mathbf{U}_t$ defined in Equation 7.7; $\Omega_t :=$

$[\omega_{\theta_K}(\mathbf{x}_{t-K}) : \cdots : \omega_{\theta_1}(\mathbf{x}_{t-1})]$ indicates the concatenation of coefficient matrices over the past K time steps; similarly, we have $\bar{\Omega}_t := [\bar{\omega}_{\theta_K}(\mathbf{u}_{t-K}) : \cdots : \bar{\omega}_{\theta_1}(\mathbf{u}_{t-1})]$ and $\bar{\Omega}'_t := [\bar{\omega}'_{\theta'_K}(\mathbf{x}_{t-2K}) : \cdots : \bar{\omega}'_{\theta'_1}(\mathbf{x}_{t-K-1})]$; $R(\cdot)$ indicates the L1 and L2 norm penalty for sparsity of the coefficient matrices from the encoder and decoder; the $S(\cdot, \cdot)$ is a smoothness penalty, defined as $S(\Omega_{t+1}, \Omega_t) = \|\Omega_{t+1} - \Omega_t\|_2$; λ and γ are hyperparameters.

Granger Causal Discovery. As the encoder-decoder is proposed to simulate the data generation process governed by Granger causality, we expect the function ω_{θ_k} can capture the causal relationships in time series. To further summarize the Granger causal relationships between variables as a summary causal graph, similar to [165], we aggregate the output from ω_{θ_k} into a summarized coefficient matrix as

$$S_{i,j} = \max_{1 \leq k \leq K} \{\text{median}_{K+1 \leq t \leq T}(|(\omega_{\theta_k}(\mathbf{x}_{t-k}))_{i,j}|), \text{ for } 1 \leq i, j \leq d,$$

where $S_{i,j}$ indicates the strength of the Granger causal effect from $\mathbf{x}^{(i)}$ on $\mathbf{x}^{(j)}$. To further derive the adjacency matrix A, we set a threshold τ, if the value $S_{i,j} > \tau$, then $A_{i,j} = 1$. In experiments, the threshold is set based on the quantile of the coefficient matrix S.

7.5.2 Root Cause Localization

After training on the normal time series, we expect that the exogenous variables can be approximated by the encoder. When deploying the model for root cause localization, we assume the time series is arrived in a streaming manner. When a new time step t^* is arrived, we first adopt the encoder to derive the exogenous variables $\mathbf{u}_{t^*}$ based on Equation 7.6. Then, for each time series, $u^{(j)}_{t^*}$, we compute the z-score as the root cause score $z^{(j)}_{t^*} = \frac{u^{(j)}_{t^*} - \mu^{(j)}}{\sigma^{(j)}}$, where $\mu^{(j)}$ and $\sigma^{(j)}$ indicate the mean and standard deviation of the exogenous variable for the j-th time series in normal data. We then adopt streaming peaks-over-threshold (SPOT) [189] to dynamically determine the threshold of labeling the potential root cause.

7.6 Experiments

Table 7.1: Statistics of Datasets

Dataset	Training	Test		
	# of Time Step	# of Sequences ($\|\mathcal{X}\|$)	Avg. Len. ($\mathbf{T}$)	Avg. # of Root Variables
Linear (4)	5,000	100	500	3.75
Nonlinear (6)	5,000	100	500	5.25
Lotka-Volterra (40)	40,000	100	2,000	30.75
Lorenz 96 (20)	200,000	100	2,000	15.75
SWaT (51)	49,500	20	51	13.35
MSDS (10)	29,268	4,255	21	3.05

7.6.1 Experimental Setup

Datasets. We conduct experiments on four synthetic and two real-world datasets. By using the synthetic datasets, we have the ground truth about the structural causal models as well as the root cause of anomalies. For the real-world datasets, we only have information about root cause variables. Therefore, we use the real-world dataset only for evaluating the root cause identification.

Linear Dataset [165] is a **synthetic** time series dataset with linear interaction dynamics. The structural equations are defined as:

$$
\begin{aligned}
x_t^{(1)} &= a_1 x_{t-1}^{(1)} + u_t^{(1)} + \epsilon_t^{(1)}, \\
x_t^{(2)} &= a_2 x_{t-1}^{(2)} + a_3 x_{t-1}^{(1)} + u_t^{(2)} + \epsilon_t^{(2)}, \\
x_t^{(3)} &= a_4 x_{t-1}^{(3)} + a_5 x_{t-1}^{(2)} + u_t^{(3)} + \epsilon_t^{(3)}, \\
x_t^{(4)} &= a_6 x_{t-1}^{(4)} + a_7 x_{t-1}^{(2)} + a_8 x_{t-1}^{(3)} + u_t^{(4)} + \epsilon_t^{(4)},
\end{aligned}
$$

where coefficients $a_i \sim \mathcal{U}([-0.8, -0.2] \cup [0.2, 0.8])$, additive innovation terms $u_t^{(\cdot)} \sim \mathcal{N}(0, 0.16)$, and anomaly term $\epsilon_t^{(\cdot)}$.

Nonlinear Dataset [190] is a **synthetic** time series dataset with non-linear interaction dynamics, of which the structural equation is defined as:

$$
\mathbf{X}_t = A^T \sum_{m=1}^{t} \beta_m \cos(\mathbf{X}_{t-m} + 1) + \epsilon,
$$

where β is the regression coefficient, and ϵ represents standard Gaussian noise. The noise scale is kept below 1 and is proportional to the value of d. The non-linear relationship

between time series is introduced through the cosine function. The adjacency matrix A of the underlying causal graph is generated using the Erdős–Rényi model [191].

Lotka-Volterra [165] is a **synthetic** time series model that simulates a prairie ecosystem with multiple species. The structural equations are defined as:

$$\begin{aligned}
\frac{d\mathbf{x}^{(i)}}{dt} &= \alpha\mathbf{x}^{(i)} - \beta \sum_{j \in Pa(\mathbf{x}^{(i)})} \mathbf{y}^{(j)} - \eta(\mathbf{x}^{(i)})^2, \text{ for } 1 \leq j \leq p, \\
\frac{d\mathbf{y}^{(j)}}{dt} &= \delta\mathbf{y}^{(j)} \sum_{k \in Pa(\mathbf{y}^{(j)})} \mathbf{x}^{(k)} - \rho\mathbf{y}^{(j)}, \text{ for } 1 \leq j \leq p, \\
x_t^{(i)} &= x_t^{(i)} + \epsilon_t^{(i)}, \; y_t^{(j)} = y_t^{(j)} + \epsilon_t^{(j)}, \text{ for } 1 \leq j \leq p,
\end{aligned}$$

where $\mathbf{x}^{(i)}$ and $\mathbf{y}^{(j)}$ denote the population sizes of prey and predator, respectively; $\alpha, \beta, \eta, \delta, \rho$ are parameters that decide the strengths of interactions, $Pa(\mathbf{x}^{(i)})$ and $Pa(\mathbf{y}^{(j)})$ correspond the Granger Causality between prey and predators for $\mathbf{x}^{(i)}$ and $\mathbf{y}^{(j)}$ respectively, and $\epsilon_t^{(\cdot)}$ is the abnormal term. We simulate 20 prey species and 20 predator species.

Lorenz 96 [165] a **synthetic** time series data, where the i-th variable is defined by the following nonlinear differential equations:

$$\frac{dx^{(i)}}{dt} = (x^{(i+1)} - x^{(i-2)})x^{(i-1)} - x^{(i)} + F, \text{ for } 1 \leq i \leq d,$$

where $x^{(0)} := x^{(d)}$, $x^{(-1)} := x^{(d-1)}$, and $x^{(d+1)} := x^1$; and F is a constant controlling the nonlinearity of the data.

Abnormal behavior injection to the synthetic datasets. For point anomalies, the anomaly term is single or multiple extreme values for randomly selected time series variables at a specific time step t. For example, a point anomaly at time step t can be generated with an abnormal term $\boldsymbol{\epsilon}_t = [0, 2, 4, 0]$, which means the second and third time series have extreme values.

For sequential anomalies, the anomaly terms are function-generated values in a given time range. For instance, setting $\epsilon_{t+i}^{(1)} = 0.1 \times i, \text{ for } 0 \leq i \leq n$, will cause a trend anomaly for time series variable $x^{(1)}$; setting $\epsilon_{t+i}^{(1)} \sim \mathcal{N}(0, 0.16), \text{ for } 0 \leq i \leq n$, will cause a shapelet

anomaly; and setting $\epsilon_{t+i}^{(1)} = (a_1 x_{t+2i-1}^{(1)} + u_{t+2i}^{(1)}) + (a_1 x_{t+2i-2}^{(1)} + u_{t+2i-1}^{(1)}) - (a_1 x_{t+i-1}^{(1)} + u_{t+i}^{(1)})$, for $0 \leq i \leq n$, will cause a seasonal anomaly.

SWaT [192] is a **real-world** dataset collected from a testbed that simulates a real-world water treatment plant. The dataset consists of both normal operations and attack scenarios within the water treatment process.

Multi-Source Distributed System (MSDS) [148] is a **real-world** dataset developed on an OpenStack testbed. Instances of fault injections are identified as anomalies.

Table 7.1 shows the statistics of datasets for training and evaluation. The number in the parenthesis indicates the dimensions of multivariate time series. The training set only has the normal time series. In the test set, for each sequence in the synthetic datasets, we conduct one or multiple exogenous interventions randomly to generate the anomalies. The last column shows the average number of variables receiving the exogenous interventions. Note that because Lorenz 96 [165] has the most complicated interdependencies between time series, we use more training samples to train AERCA for causal discovery.

Evaluation Metrics. In this work, the root cause identification is achieved based on the learned causal models. Therefore, we evaluate AERCA in both causal discovery and root cause identification.

Causal Discovery. We adopt the commonly used metrics to evaluate the performance of causal discovery [163, 188, 193–196], including F1, AUC-ROC, AUC-PR, and Hamming distance (HD). In the context of causal discovery, F1, AUC-ROC, and AUC-PR quantify the correctness of edge discovery, while Hamming distance calculates the proportion of disagreeing edges between the learned causal graph and the ground truth causal graph.

Root Cause Identification. Following the existing work [174, 176, 178, 197], we adopt the "recall at top-k" to evaluate the performance of root cause identification, denoted as $AC@K$. This metric quantifies the probability of identifying the correct root cause in the list of variables with the top-k highest root cause scores. Given a set of time series $\mathcal{X}$, the definition of $AC@K$ is shown below.

$$AC@K = \frac{1}{|\mathcal{X}|}\sum_{\mathbf{X}\in\mathcal{X}}\frac{|V_{\mathbf{X}}^{(RC)} \cap \{R_{\mathbf{X}}[k]|k = 1, 2, ...K\}|}{min(K, |V_{\mathbf{X}}^{(RC)}|)},$$

where $R_{\mathbf{X}}[k]$ indicates the time series at the k-th rank for the sequence $\mathbf{X}$, and $V_{\mathbf{X}}^{(RC)}$ indicates a set of root cause variables over the whole time series $\mathbf{X}$. Note that if a time series receives multiple exogenous interventions, it only counts as one root cause time series in $V_{\mathbf{X}}^{(RC)}$. We further compute the overall performance by computing the average $AC@K$, denoted as $Avg@K = \frac{1}{K}\sum_{k=1}^{K} AC@k$.

AC@k quantifies the performance of root cause analysis as long as the approach finds the root cause time series. However, in some cases, a time series can receive exogenous interventions for multiple time steps. To quantify the effectiveness of approaches for locating the root cause time series at specific time steps, we further develop the metric "recall at top-k over all time steps" below.

$$AC^*@K = \frac{1}{|\mathcal{X}|}\sum_{\mathbf{X}\in\mathcal{X}}\frac{|\bigcup_{t\in T} V_{\mathbf{x}_t}^{(RC)} \cap \{R_{\mathbf{X}}^*[k]|k = 1, 2, ..., K\}|}{min(K, |\bigcup_{t\in T} V_{\mathbf{x}_t}^{(RC)}|)},$$

where $V_{\mathbf{x}_t}^{(RC)}$ indicates the set of root cause time series at the t-th time step; $R_{\mathbf{X}}^*[k]$ indicates the time series at the k-th rank over all time steps. Similarly, we also compute the overall performance by computing the average $AC^*@K$, denoted as $Avg^*@K = \frac{1}{K}\sum_{k=1}^{K} AC^*@k$.

The experimental results are reported as the average of five independent runs. For baselines that cannot identify the root cause at specific time steps, we consider the root cause predicted by the baselines indicating the abnormal time series at the last time step in a sliding window.

Baselines. We choose two sets of baselines for comparing the performance of causal discovery and root cause identification, respectively.

Causal Discovery. We compare AERCA for the causal discovery with the following baselines. 1) **VAR** (Vector AutoRegressive) [166] is a linear model to analyze and predict the temporal interdependencies between multiple time series datasets; 2) **cMLP** [164] indicates structured multilayer perceptrons (MLPs) combined with sparsity penalties on the weights

for Granger causal discovery; 3) **cLSTM** [164] leverages recurrent neural networks (RNNs) for Granger causal discovery; 4) **TCDF** (Temporal Causal Discovery Framework) [163] uses attention-based convolutional neural networks for causal discovery from time series data; 5) **eSRU** (economy-Statistical Recurrent Units) [198] leverages a special type of RNN called the statistical recurrent unit (SRU) for inferring the Granger causality; 6) **PCMCI** [199] combines linear or nonlinear conditional independence tests with a causal discovery algorithm to estimate causal networks; 7) **GVAR** [165] is a vector autoregression with generalized coefficient matrices predicted by neural networks; 8) **CUTS** [200] is a neural Granger causal discovery algorithm building a causal adjacency matrix with imputed data under sparse penalty.

Root Cause Identification. We compare AERCA with the following baselines. 1) **ϵ-Diagnosis** [180] assumes that the root cause nodes have significantly changed between the abnormal and normal periods and conducts pair-wise significant tests to locate the root cause; 2) **RCD** (Root Cause Discovery) [178] learns the partial causal graph related to the root cause and locate the root cause as the interventional targets; 3) **CIRCA** (Causal Inference-based Root Cause Analysis) [176] builds structural causal graph via domain knowledge and locates the root cause in anomalies as the nodes with significant distribution changes given its parents. All baselines are implemented by PyRCD [201].

Implementation Details. We implement distinct neural network configurations tailored to the complexity of the dataset at hand. Specifically, for synthetic datasets, we employ a two-layer feedforward neural network architecture with a hidden dimension of 50, whereas for real-world datasets, the architecture is expanded to eight layers, each boasting a hidden dimension of 1000. Preprocessing of data is standardized across datasets using a MinMax scaler, with further efficiency measures including downsampling of the SWaT dataset at intervals of every 10 seconds and the MSDS dataset every 5 steps. The training framework is anchored by a learning rate of 1×10^{-6}, with the Adam optimizer facilitating parameter optimization. Hyperparameters β, λ, and γ are initially set to 1, ensuring a balanced approach to regularization and loss function adjustment. The maximum training epochs

are set to 5000, incorporating an early stopping criterion that halts training if no improvement in loss is observed for 20 consecutive epochs. All experiments were conducted on an Ubuntu 20.04 server equipped with an AMD Ryzen 3960X 24-Core processor at 3.8GHz, dual GeForce RTX 3090 GPUs, and 128 GB of RAM. The implementation uses Python 3.9.7 and PyTorch 1.11.0.

7.6.2 Experimental Results

Table 7.2: Overall performance (mean±std.) of causal discovery.

Model	Linear				Nonlinear				Lotka-Volterra				Lorenz 96			
	F1	AUC-PR	AUC-ROC	HD	F1	AUC-PR	AUC-ROC	HD	F1	AUC-PR	AUC-ROC	HD	F1	AUC-PR	AUC-ROC	HD
VAR	$0.969_{\pm 0.019}$	$0.998_{\pm 0.003}$	$0.999_{\pm 0.001}$	$0.011_{\pm 0.009}$	$0.473_{\pm 0.164}$	$0.529_{\pm 0.181}$	$0.676_{\pm 0.140}$	$0.258_{\pm 0.130}$	$0.533_{\pm 0.013}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{1.000_{\pm 0.000}}$	$0.044_{\pm 0.003}$	$0.404_{\pm 0.162}$	$0.562_{\pm 0.376}$	$0.764_{\pm 0.204}$	$0.360_{\pm 0.121}$
cMLP	$0.745_{\pm 0.029}$	$0.595_{\pm 0.038}$	$0.829_{\pm 0.0.25}$	$0.229_{\pm 0.033}$	$0.419_{\pm 0.134}$	$0.327_{\pm 0.079}$	$0.609_{\pm 0.089}$	$0.340_{\pm 0.217}$	$0.511_{\pm 0.011}$	$0.065_{\pm 0.014}$	$0.508_{\pm 0.007}$	$0.049_{\pm 0.001}$	$0.472_{\pm 0.058}$	$0.202_{\pm 0.027}$	$0.569_{\pm 0.038}$	$0.193_{\pm 0.031}$
cLSTM	$0.684_{\pm 0.042}$	$0.522_{\pm 0.048}$	$0.766_{\pm 0.047}$	$0.312_{\pm 0.062}$	$0.378_{\pm 0.000}$	$0.233_{\pm 0.000}$	$0.500_{\pm 0.000}$	$0.767_{\pm 0.000}$	$0.356_{\pm 0.176}$	$0.052_{\pm 0.001}$	$0.500_{\pm 0.000}$	$0.400_{\pm 0.428}$	$0.453_{\pm 0.048}$	$0.194_{\pm 0.021}$	$0.572_{\pm 0.031}$	$0.232_{\pm 0.035}$
TCDF	$0.943_{\pm 0.070}$	$0.933_{\pm 0.081}$	$0.950_{\pm 0.061}$	$0.033_{\pm 0.040}$	$0.473_{\pm 0.107}$	$0.343_{\pm 0.072}$	$0.655_{\pm 0.087}$	$0.307_{\pm 0.065}$	$0.853_{\pm 0.032}$	$0.749_{\pm 0.050}$	$0.890_{\pm 0.021}$	$0.019_{\pm 0.002}$	$0.429_{\pm 0.007}$	$0.290_{\pm 0.006}$	$0.645_{\pm 0.004}$	$0.260_{\pm 0.011}$
eSRU	$0.964_{\pm 0.070}$	$0.958_{\pm 0.082}$	$0.969_{\pm 0.061}$	$0.021_{\pm 0.041}$	$0.408_{\pm 0.152}$	$0.332_{\pm 0.071}$	$0.617_{\pm 0.092}$	$0.267_{\pm 0.069}$	$0.422_{\pm 0.039}$	$0.323_{\pm 0.030}$	$0.634_{\pm 0.016}$	$0.055_{\pm 0.002}$	$0.195_{\pm 0.024}$	$0.225_{\pm 0.009}$	$0.539_{\pm 0.009}$	$0.215_{\pm 0.006}$
PCMCI	$0.969_{\pm 0.031}$	$0.981_{\pm 0.040}$	$0.986_{\pm 0.042}$	$0.025_{\pm 0.038}$	$0.607_{\pm 0.094}$	$0.456_{\pm 0.172}$	$0.742_{\pm 0.147}$	$0.273_{\pm 0.175}$	$0.465_{\pm 0.025}$	$0.291_{\pm 0.019}$	$0.906_{\pm 0.017}$	$0.109_{\pm 0.008}$	$0.368_{\pm 0.004}$	$0.227_{\pm 0.007}$	$0.680_{\pm 0.013}$	$0.540_{\pm 0.021}$
GVAR	$0.862_{\pm 0.052}$	$0.981_{\pm 0.040}$	$0.986_{\pm 0.042}$	$0.131_{\pm 0.066}$	$0.421_{\pm 0.094}$	$0.562_{\pm 0.145}$	$0.683_{\pm 0.097}$	$0.487_{\pm 0.103}$	$0.787_{\pm 0.011}$	$0.988_{\pm 0.015}$	$0.999_{\pm 0.002}$	$0.027_{\pm 0.002}$	$0.568_{\pm 0.330}$	$0.582_{\pm 0.361}$	$0.776_{\pm 0.194}$	$0.142_{\pm 0.109}$
CUTS	$0.810_{\pm 0.076}$	$0.792_{\pm 0.066}$	$0.844_{\pm 0.050}$	$0.104_{\pm 0.034}$	$0.357_{\pm 0.040}$	$0.249_{\pm 0.014}$	$0.536_{\pm 0.032}$	$0.513_{\pm 0.124}$	$\mathbf{0.877_{0.031}}$	$0.791_{\pm 0.047}$	$0.892_{\pm 0.024}$	$\mathbf{0.011_{\pm 0.002}}$	$0.341_{\pm 0.003}$	$0.206_{\pm 0.002}$	$0.621_{\pm 0.004}$	$0.404_{\pm 0.012}$
AERCA	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{0.000_{\pm 0.000}}$	$\mathbf{0.826_{\pm 0.057}}$	$\mathbf{0.996_{\pm 0.013}}$	$\mathbf{0.998_{\pm 0.006}}$	$\mathbf{0.027_{\pm 0.014}}$	$0.857_{\pm 0.000}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{1.000_{\pm 0.000}}$	$0.026_{\pm 0.000}$	$\mathbf{0.800_{\pm 0.000}}$	$\mathbf{0.998_{\pm 0.002}}$	$\mathbf{0.999_{\pm 0.001}}$	$\mathbf{0.105_{\pm 0.000}}$

Table 7.3: Overall performance (mean±std.) of root cause analysis.

Dataset	Model	AC@1	AC@3	AC@5	AC@10	Avg@10	AC*@1	AC*@10	AC*@100	AC*@500	Avg*@500
Linear	ϵ-Diagnosis	$0.900_{\pm 0.300}$	$0.850_{\pm 0.189}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{1.000_{\pm 0.000}}$	$0.950_{\pm 0.043}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.250_{\pm 0.316}$	$0.086_{\pm 0.125}$
	RCD	$0.500_{\pm 0.500}$	$0.817_{\pm 0.189}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{1.000_{\pm 0.000}}$	$0.907_{\pm 0.076}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.150_{\pm 0.300}$	$0.064_{\pm 0.160}$
	CIRCA	$0.600_{\pm 0.490}$	$0.800_{\pm 0.306}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{1.000_{\pm 0.000}}$	$0.910_{\pm 0.106}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.025_{\pm 0.075}$	$0.233_{\pm 0.327}$	$0.088_{\pm 0.144}$
	AERCA	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{0.763_{\pm 0.137}}$	$\mathbf{0.990_{\pm 0.018}}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{0.998_{\pm 0.001}}$
Nonlinear	ϵ-Diagnosis	$0.400_{\pm 0.490}$	$0.667_{\pm 0.325}$	$0.880_{\pm 0.165}$	$\mathbf{1.000_{\pm 0.000}}$	$0.837_{\pm 0.139}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.092_{\pm 0.142}$	$0.056_{\pm 0.088}$
	RCD	$0.600_{\pm 0.490}$	$0.750_{\pm 0.344}$	$0.880_{\pm 0.165}$	$\mathbf{1.000_{\pm 0.000}}$	$0.878_{\pm 0.118}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.080_{\pm 0.240}$	$0.263_{\pm 0.405}$	$0.116_{\pm 0.215}$
	CIRCA	$0.700_{\pm 0.458}$	$0.717_{\pm 0.395}$	$0.835_{\pm 0.295}$	$\mathbf{1.000_{\pm 0.000}}$	$0.863_{\pm 0.160}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.017_{\pm 0.050}$	$0.160_{\pm 0.182}$	$0.064_{\pm 0.075}$
	AERCA	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{0.433_{\pm 0.132}}$	$\mathbf{0.830_{\pm 0.094}}$	$\mathbf{0.994_{\pm 0.010}}$	$\mathbf{0.995_{\pm 0.009}}$	$\mathbf{0.987_{\pm 0.094}}$
Lotka-Volterra	ϵ-Diagnosis	$0.100_{\pm 0.300}$	$0.133_{\pm 0.163}$	$0.138_{\pm 0.149}$	$0.247_{\pm 0.188}$	$0.158_{\pm 0.131}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$
	RCD	$0.100_{\pm 0.300}$	$0.133_{\pm 0.163}$	$0.138_{\pm 0.149}$	$0.247_{\pm 0.188}$	$0.158_{\pm 0.131}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$
	CIRCA	$0.120_{\pm 0.325}$	$0.107_{\pm 0.169}$	$0.120_{\pm 0.150}$	$0.225_{\pm 0.230}$	$0.146_{\pm 0.163}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$
	AERCA	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{0.997_{\pm 0.005}}$	$\mathbf{0.998_{\pm 0.004}}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{1.000_{\pm 0.000}}$
Lorenz96	ϵ-Diagnosis	$0.100_{\pm 0.300}$	$0.200_{\pm 0.221}$	$0.280_{\pm 0.312}$	$0.450_{\pm 0.330}$	$0.314_{\pm 0.225}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$
	RCD	$0.200_{\pm 0.400}$	$0.333_{\pm 0.333}$	$0.400_{\pm 0.358}$	$0.556_{\pm 0.337}$	$0.421_{\pm 0.278}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$
	CIRCA	$0.360_{\pm 0.480}$	$0.330_{\pm 0.244}$	$0.346_{\pm 0.249}$	$0.539_{\pm 0.263}$	$0.408_{\pm 0.220}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$
	AERCA	$\mathbf{0.996_{\pm 0.009}}$	$\mathbf{0.996_{\pm 0.009}}$	$\mathbf{0.997_{\pm 0.008}}$	$\mathbf{0.996_{\pm 0.008}}$	$\mathbf{0.990_{\pm 0.011}}$	$\mathbf{0.842_{\pm 0.016}}$	$\mathbf{0.970_{\pm 0.013}}$	$\mathbf{0.996_{\pm 0.009}}$	$\mathbf{0.996_{\pm 0.009}}$	$\mathbf{0.987_{\pm 0.010}}$
SWaT	ϵ-Diagnosis	$0.075_{\pm 0.179}$	$0.125_{\pm 0.217}$	$0.125_{\pm 0.217}$	$0.375_{\pm 0.383}$	$0.180_{\pm 0.194}$	$\mathbf{0.150_{\pm 0.357}}$	$0.125_{\pm 0.217}$	$0.125_{\pm 0.217}$	$\mathbf{1.000_{\pm 0.000}}$	$0.633_{\pm 0.128}$
	RCD	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.300_{\pm 0.458}$	$0.100_{\pm 0.161}$	$0.000_{\pm 0.000}$	$0.025_{\pm 0.043}$	$0.214_{\pm 0.066}$	$0.799_{\pm 0.070}$	$0.416_{\pm 0.028}$
	CIRCA	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.300_{\pm 0.458}$	$0.100_{\pm 0.161}$	$0.000_{\pm 0.000}$	$0.025_{\pm 0.043}$	$0.214_{\pm 0.066}$	$0.799_{\pm 0.070}$	$0.416_{\pm 0.028}$
	AERCA	$\mathbf{0.220_{\pm 0.111}}$	$\mathbf{0.290_{\pm 0.088}}$	$\mathbf{0.330_{\pm 0.048}}$	$\mathbf{0.455_{\pm 0.044}}$	$\mathbf{0.342_{\pm 0.052}}$	$0.020_{\pm 0.026}$	$\mathbf{0.320_{\pm 0.026}}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{0.950_{\pm 0.002}}$
MSDS	ϵ-Diagnosis	$0.004_{\pm 0.004}$	$0.266_{\pm 0.002}$	$0.452_{\pm 0.009}$	$\mathbf{1.000_{\pm 0.000}}$	$0.492_{\pm 0.001}$	$0.000_{\pm 0.000}$	$0.389_{\pm 0.410}$	$0.706_{\pm 0.310}$	$\mathbf{1.000_{\pm 0.000}}$	$0.880_{\pm 0.113}$
	RCD	$0.412_{\pm 0.048}$	$0.573_{\pm 0.010}$	$\mathbf{0.984_{\pm 0.001}}$	$\mathbf{1.000_{\pm 0.000}}$	$0.821_{\pm 0.012}$	$0.025_{\pm 0.026}$	$0.216_{\pm 0.806}$	$0.806_{\pm 0.205}$	$\mathbf{1.000_{\pm 0.000}}$	$0.908_{\pm 0.078}$
	CIRCA	$\mathbf{0.454_{\pm 0.238}}$	$0.860_{\pm 0.140}$	$0.917_{\pm 0.084}$	$\mathbf{1.000_{\pm 0.000}}$	$0.809_{\pm 0.035}$	$0.000_{\pm 0.000}$	$0.102_{\pm 0.108}$	$0.741_{\pm 0.273}$	$\mathbf{1.000_{\pm 0.000}}$	$0.884_{\pm 0.083}$
	AERCA	$0.381_{\pm 0.408}$	$\mathbf{0.908_{\pm 0.062}}$	$0.974_{\pm 0.027}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{0.896_{\pm 0.037}}$	$\mathbf{0.230_{\pm 0.004}}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{1.000_{\pm 0.000}}$	$\mathbf{0.997_{\pm 0.000}}$

Performance of Causal Discovery. Table 7.2 shows the results of AERCA and baselines for causal discovery. For baselines, different approaches can achieve good performance on different datasets. For example, VAR can achieve high F1, AUC-PR, AUC-ROC, and low HD on the Linear dataset, but the performance of VAR on other nonlinear datasets is poor,

which is expected as VAR is a linear model. Some other advanced approaches, such as TCDF, GVAR, and CUTS, can achieve good performance on the Lotka-Volterra dataset. However, none of the baseline approaches can achieve satisfactory performance on both Nonlinear and Lorenz96 datasets. In contrast, AERCA achieves the perfect performance on the Linear dataset with 1 F1, AUC-PR, AUC-ROC scores, and 0 Hamming distance, indicating AERCA can learn the causal graph without any error. For the more challenging three nonlinear datasets, AERCA can achieve high F1, AUC-PR, AUC-ROC scores and very low Hamining distance, showing the capability of AERCA for discovering the nonlinear causal relationships.

Performance of Root Cause Identification. Table 7.3 presents the results of AERCA and baseline models for root cause identification across synthetic and real-world datasets, utilizing the AC@K metric with consideration for the dimensionality of multivariate time series within each dataset. Although baselines show promising results in terms of AC@5 and AC@10 on both Linear and Nonlinear datasets, it's important to note that these datasets are of low dimensionality. For instance, in the Linear dataset with four dimensions, AC@5 essentially treats all time series as potential root causes, explaining why all approaches achieve a score of 1 for AC@5. In contrast, AERCA shows exceptional performance across all datasets, excluding the MSDS dataset, even at the AC@1 metric, indicating its capability to accurately identify the time series with the highest root cause score. On the MSDS dataset, AERCA can still identify the majority of root cause time series, as evidenced by the highest Avg@10 score. For the more challenging metric AC*@K, AERCA can achieve high AC*@1 scores on most datasets except the SWaT dataset, meaning that AERCA can successfully detect the root cause at specific time steps. Furthermore, AERCA achieves near-perfect AC*@10 scores on most datasets. Considering that there are thousands or even tens of thousands of candidates ($T * d$) on each dataset when trying to highlight the root cause at specific time steps, the performance of AERCA is promising.

Case Study. Figure 7.3 shows a short snippet of multivariate time series on the Nonlinear dataset, where we conduct four exogenous interventions on four different time series at

different time steps. We highlight the predicted root cause with the top 5 highest root cause scores (AC*@5) via purple bars. We can notice that AERCA correctly detects the root cause time series receiving the exogenous interventions at specific time steps. Meanwhile, as shown at the bottom of each time series, the distribution of root cause score (z-score) matches the exogenous variables, especially when the time series receives the exogenous interventions.

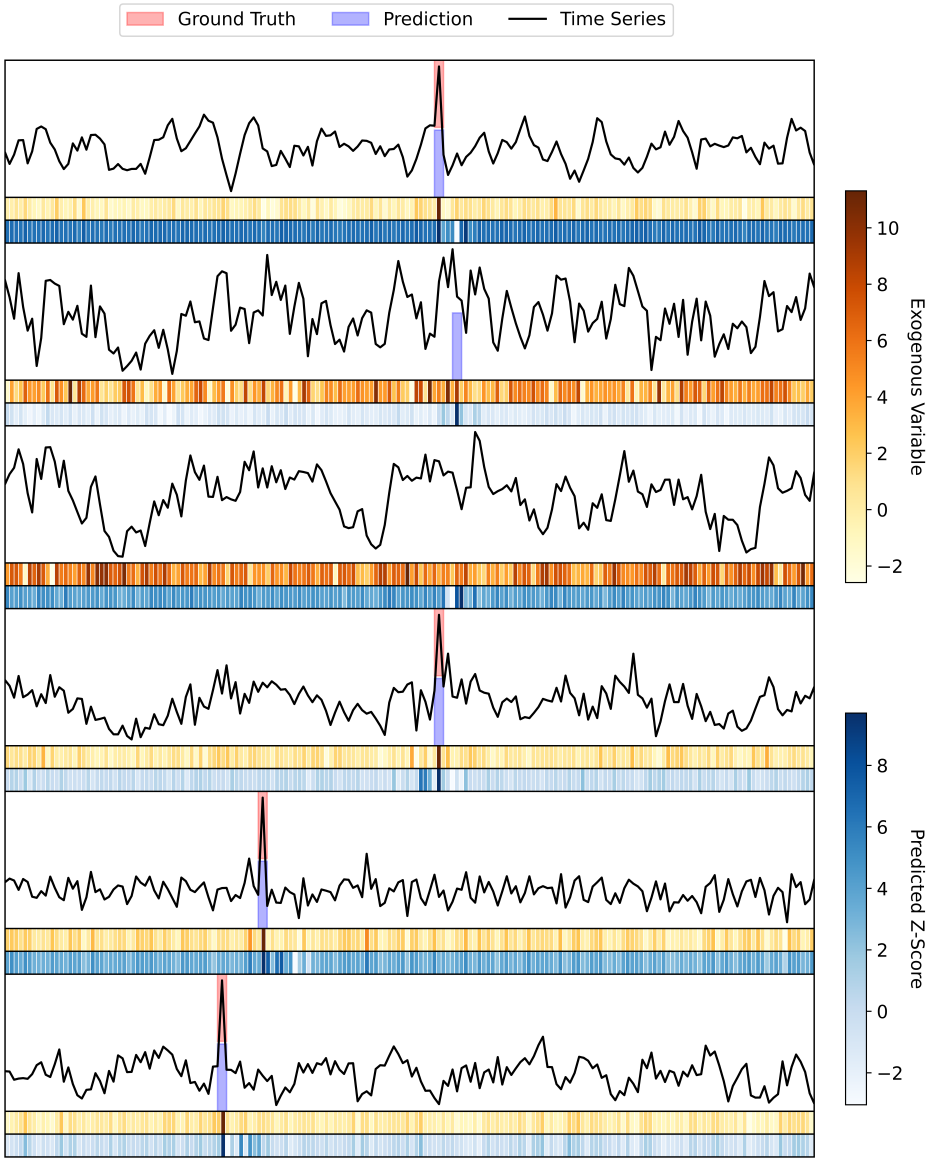
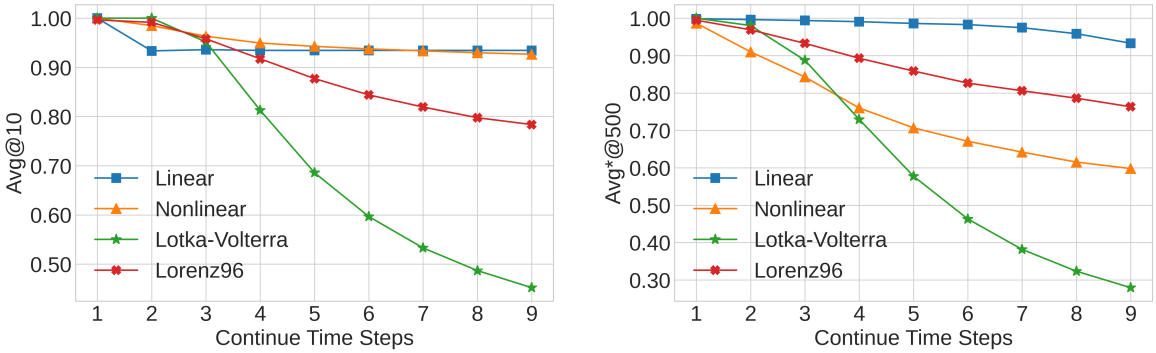


Fig. 7.3: Visualization of multivariate time series, exogenous variables, and predicted root cause scores on the Nonlinear dataset (6 dimensions) with the ground truth and predicted root cause.

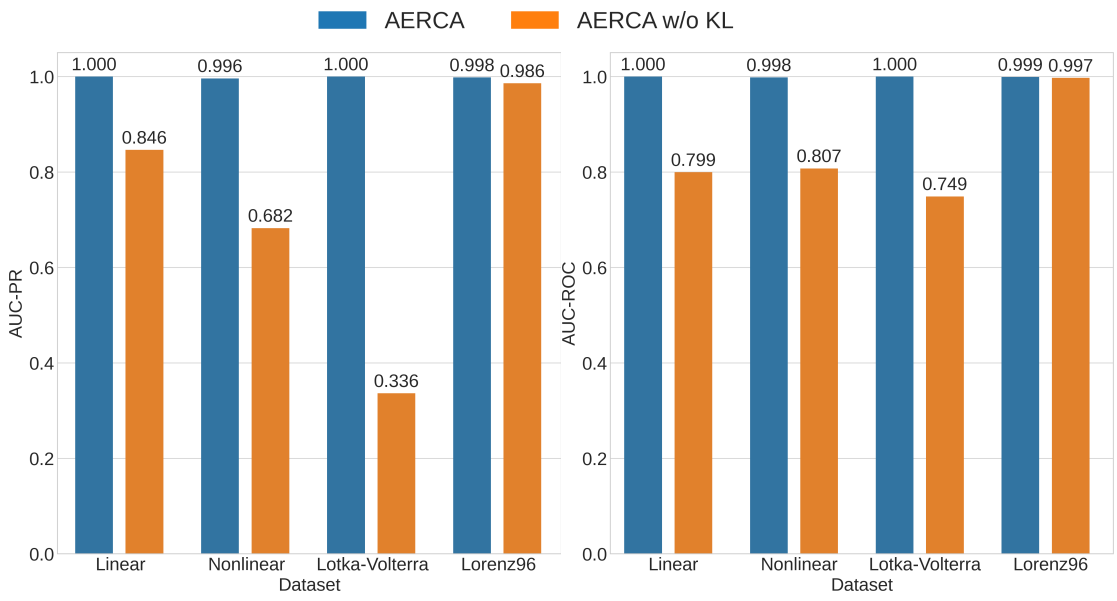


(a) Avg@10 (b) Avg*@500

Fig. 7.4: Performance of root cause identification with various numbers of continuous exogenous interventions.

Sensitivity Analysis. We evaluate the performance of AERCA on root cause identification when the anomalies are caused by continuous exogenous interventions. We tune the number of time steps having the exogenous interventions and check the performance change. Recall that to make the task more challenging, at each time step, the exogenous intervention is conducted on different time series. Figure 7.4 shows the evaluation results. We can observe that in terms of Avg@10, the performance of AERCA remains stable on both Linear and Nonlinear datasets and slightly decreases on the Lotka-Volterra dataset when increasing the time steps receiving the exogenous interventions. It shows that AERCA can identify the root cause correctly with continuous interventions on different time series. Meanwhile, in terms of Avg*@500, AERCA can still achieve reasonable performance on Linear, Nonlinear, and Lorenz96 datasets, indicating that in most cases, AERCA can identify the time series receiving exogenous interventions at specific time steps with high root cause scores. The main reason that the performance of AERCA on Lotka-Volterra significantly decreases is because Lotka-Volterra has more variables, which makes the candidates of root cause significantly larger, especially for computing the metric AC*@K. In summary, AERCA can achieve promising performance on root cause identification for continuous interventions when the number of dimensions in multivariate time series is moderate.

Ablation Study. To properly learn the exogenous variables, it is critical to ensure the exogenous variables of different time series are independent of each other. Therefore, we



(a) AUC-PR (b) AUC-ROC

Fig. 7.5: Impact of the independent constraint on exogenous variables (defined in Eq. 7.7) for causal discovery.

have an independent constraint on exogenous variables defined in Equation 7.7. To show the importance of the independent constraint for causal discovery, we conduct the ablation study to compare the performance of causal discovery when AERCA is trained with and without the independent constraint in the objective function. As shown in Figure 7.5, on Linear, Nonlinear, and Lotka-Volterra datasets, without the independent constraint on exogenous variables, the performance of AERCA for causal discovery is much worse. On Lorenz96, the impact is minor, but without the independent constraint, the performance of AERCA is still lower. The experimental results demonstrate the importance of the independent constraint on exogenous variables for causal discovery.

7.7 Summary

In this work, we have developed AERCA for the root cause analysis of anomalies in multivariate time series through Granger causal discovery. AERCA considers the anomalies caused by external interventions on exogenous variables in SCM. To achieve root cause analysis, AERCA explicitly considers the exogenous variables when simulating the data

generation process guided by SCM. After training on the normal time series data, AERCA can learn the causal relationships among time series as well as derive the exogenous variables. At deployment, the exogenous variables deviated from the normal values will have high root cause scores. Experimental results on multiple datasets show that AERCA can achieve state-of-the-art performance on both causal discovery and root cause identification.

CHAPTER 8
ON ROOT CAUSE LOCALIZATION AND ANOMALY MITIGATION THROUGH CAUSAL INFERENCE

In this chapter, we propose RootCLAM, which aims to achieve Root Cause Localization and Anomaly Mitigation from a causal perspective. Especially, we formulate anomalies caused by external interventions on the normal causal mechanism and aim to locate the abnormal features with external interventions as root causes. After that, we further propose an anomaly mitigation approach that aims to recommend mitigation actions on abnormal features to revert the abnormal outcomes such that the counterfactuals guided by the causal mechanism are normal. Experiments on three datasets show that our approach can locate the root causes and further flip the abnormal labels.

8.1 Introduction

Deep anomaly detection models have been used to automatically detect a variety of anomalies, such as bank fraud detection. As many anomaly detection tasks are high-stakes decision-making tasks, there is a growing demand for the transparency of the detection results, especially, for the outcomes as anomalies [202]. For example, if a credit card transaction is declined by an automated decision-making algorithm due to the potential fraudulent features of this transaction, the user would like to know which features lead to the transaction decline and how to avoid such a situation in the future.

To answer the question of which features lead to abnormal outcomes, several interpretable anomaly detection approaches are proposed based on the idea of feature attributions [141, 203, 204]. Although feature attribution-based approaches can highlight the abnormal features, they ignore the dependencies between different features, whereas some abnormal features may be caused by other upstream abnormal features. For example, if a loan application is declined, a feature attribution-based approach may highlight the low

income and low savings as abnormal features. However, the actual situation may be that low savings are caused by low income, and low income is the root cause of the loan application decline. Identifying the root cause of the anomaly can provide insights into the anomaly as well as efficient actions to fix the anomaly.

In this work, we study the problem of anomaly mitigation facilitated by the root cause localization. We propose a framework named Root Cause Localization and Anomaly Mitigation (RootCLAM). The framework consists of two phases. In the first phase, we attempt to identify and localize the features that are the root cause of the anomaly for each abnormal instance. Then, in the second phase, we answer the question of how to fix the abnormal outcome by finding the algorithmic recourse [160] on the abnormal outcome. Traditional algorithmic recourse may perform actions on any feature in order to improve or flip the outcome. However, in the context of anomaly mitigation, it is more natural to perform recourse actions on the root cause features as not all features are equally important for mitigation. Thus, our framework aims to find the algorithmic recourse by only using root cause features.

Developing RootCLAM faces several challenges. First, despite several root cause analysis approaches proposed for anomalies in time series data [175, 205–207], the research on the root cause analysis of the tabular data is still limited, especially in the context of anomaly detection. Second, to perform appropriate recourse actions on root cause features to change the outcome, one needs to quantitatively analyze the causal connection between these actions and the outcome [175, 205, 208, 209]. Last but not least, algorithmic recourse is known as providing a counterfactual interpretation of the outcome. However, existing counterfactual inference techniques [210–213] usually assume that the causal connections between features can be described by linear equations, which may not be realistic in practical situations.

To address these challenges, we first assume that the data generation is governed by a Structural Causal Model (SCM) [150], and treat the root cause as external interventions on specific features. As a result, the root cause localization is to identify features that are impacted by the external intervention. Then, we formulate the algorithmic recourse for anomaly mitigation as soft interventions [214] in order to represent the causal effect of

recourse actions on the outcome as a differentiable expression. Based on that, we develop a continuous optimization-based iterative algorithm that follows the causal graph topological order to compute the actions such that the outcome will be flipped to normal by performing the actions. In addition, we leverage the causal graph autoencoder to conduct counterfactual inference. In particular, we adopt the Variational Causal Graph Autoencoder (VACA) [215] which can deal with non-linear SCMs by leveraging graph neural networks. Finally, anomaly mitigation is achieved as the outcome of the algorithmic recourse based on root cause features.

For empirical evaluation, we conduct experiments on several semi-synthetic and real-world datasets. The results show that our method can produce the largest flipping ratio regarding the anomaly detection outcomes while requiring the minimum perturbation compared with the baseline methods.

8.2 Preliminary

8.2.1 Structural Causal Model (SCM)

We adopt Pearl's Structural Causal Model (SCM) [150] as the prime methodology for computing counterfactuals. Throughout this work, we use the upper/lower case alphabet to represent features/values.

Inferring causal effects in the SCM is facilitated by the intervention. The hard intervention forces some variable $X \in V$ to take a certain value x. For an SCM $\mathcal{M}$, intervention $do(X = x')$ is equivalent to replacing original function in F with $X = x'$. The soft intervention, on the other hand, forces some variables to take a certain functional relationship in responding to some other variables [214]. The soft intervention substitutes equation $x = f(x_{\mathsf{PA}}, u)$ with a new equation. After the intervention, the distributions of all features that are the descendants of X may be changed, called the interventional distributions.

8.2.2 Counterfactuals

Counterfactuals are about answering questions such as for two features $X, Y \in V$,

whether Y would be y had X been x' given that X is equal to x in the factual instance. Symbolically we denote this counterfactual instance as $x_{do(X=x')}|x$. The counterfactual question involves two worlds, the factual world and the counterfactual world, and cannot be answered directly by the do-operator. When the complete knowledge of the SCM is known, the counterfactual can be computed by the Abduction-Action-Prediction process [150]:

1) Abduction: Beliefs about the world are updated by taking into account all evidence given in the context. Formally, update the probability $P(u)$ to $P(u|e)$.

2) Action: Perform do intervention, $do(X = x')$, to reflect the counterfactual assumption, and a new causal model is created by interventions $\mathcal{M}' = \mathcal{M}_{do(X=x')}$.

3) Prediction: Counterfactual reasoning occurs over the new model $\mathcal{M}'$ using updated knowledge $P(u|e)$.

8.2.3 Causal Graph Autoencoder

A causal graph autoencoder is a type of deep learning model that aims to learn a latent representation of the data that captures the underlying causal relationships among variables given a causal graph. In this work, we adopt the Variational Causal Graph Autoencoder (VACA) [215] which can accurately approximate the interventional and counterfactual distributions on diverse SCMs and can deal with non-linear causal relationships. The VACA consists of an adjacency matrix A of the causal graph, a decoder $p_\zeta(\mathbf{x}|\mathbf{z}, A)$ which is a graph neural network (GNN) that takes as input a set of latent variables $\mathbf{z}$ and the matrix A and outputs the likelihood of $\mathbf{x}$, and an encoder $q_\xi(\mathbf{z}|\mathbf{x}, A)$ which is another GNN that takes $\mathbf{x}$ and A as input and outputs the latent variables of $\mathbf{z}$. The VACA is trained to fit the observational distribution.

To compute the counterfactual instance of a factual instance $\mathbf{x}$ under the hard intervention $do(X_i = x')$, the VACA first computes the distribution of $\mathbf{z}$ by feeding the factual instance $\mathbf{x}$ and A into encoder $q_\xi(\mathbf{z}|\mathbf{x}, A)$. Then, the VACA constructs the intervened instance $\bar{\mathbf{x}}$ by replacing the value of x_i in the factual instance $\mathbf{x}$ with the intervened value x', as well as the intervened matrix $\bar{A}$ by removing all incoming edges of node X_i in the causal

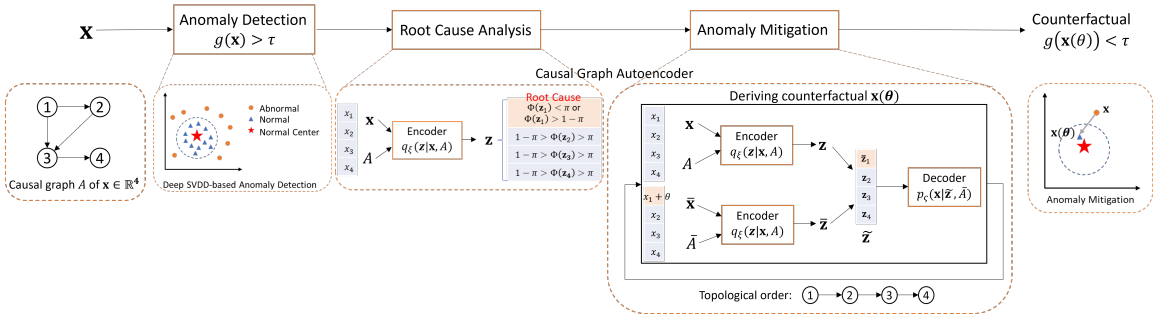


Fig. 8.1: The pipeline to achieve root cause identification and anomaly mitigation.

graph. The VACA feeds $\bar{\mathbf{x}}$ and $\bar{A}$ into encoder $q_\xi(\mathbf{z}|\mathbf{x}, A)$ to compute the intervened distribution of the latent variables, denoted by $\bar{\mathbf{z}}$. Next, the VACA removes the latent variable in $\mathbf{z}$ that corresponds to x_i, i.e., z_i, and replaces it with $\bar{z}_i$ in $\bar{\mathbf{z}}$ to obtain a new vector $\tilde{\mathbf{z}}$. This step is to perform the intervention in the hidden space that is equivalent to performing the intervention in the original feature space. Finally, $\tilde{\mathbf{z}}$ and $\bar{A}$ are fed into the decoder $p_\zeta(\mathbf{x}|\mathbf{z}, A)$ to compute the counterfactual instance.

8.3 RootCLAM

In this section, we introduce RootCLAM, which is a two-phase framework that recommends anomaly mitigation actions to flip abnormal outcomes to normal ones. When an anomaly is detected, root cause localization is first to identify the abnormal features leading to the abnormal outcome. Then, anomaly mitigation is to further find actions on an anomaly to flip the prediction from a fixed anomaly detection model with the consideration of the root cause of the anomaly. Figure 8.1 illustrates our framework for root cause analysis and anomaly mitigation.

8.3.1 Problem Formulation

We start with formulating the problem for root cause localization and anomaly mitigation. Consider an unlabeled dataset $\mathcal{X} = \{\mathbf{x}^{(n)}\}_{n=1}^N$ consisting of both normal and abnormal samples, where $\mathbf{x} = [x_1, ..., x_i, ...x_d] \in \mathbb{R}^d$ is a sample with d features. We adopt a score-based anomaly detection model $g(\cdot) : \mathcal{X} \to \mathbb{R}$, which labels abnormal samples if $g(\mathbf{x}) > \tau$,

where τ indicates the threshold. By applying $g(\cdot)$ on $\mathcal{X}$, we can obtain a set of detected abnormal samples $\hat{\mathcal{X}}^-$. Our goal is to find the root causes of the anomalies as well as the actions to fix them.

Root Cause. First, we need to define the root cause. Assume that the normal data are generated from a Structural Causal Model (SCM) given as follows:

$$\forall x_i \in \mathcal{X}, \quad x_i \sim P(x_i | \{x_j, \forall j \in X_{\mathsf{PA}_i}\}, u_i).$$

We consider that any anomaly is caused by certain external interventions on some features in the SCM. Thus, the root causes of anomalies are defined as follows.

Definition 3 *Given any anomaly* $\mathbf{x} \in \hat{\mathcal{X}}^-$*, the root causes of* $\mathbf{x}$ *is a set of features* $\mathcal{I}$ *that receives external interventions.*

We do not assume the type of the SCM, but we do assume that the external intervention on a feature x_i can be represented as an intervention on the exogenous variable u_i. It is straightforward to show that this assumption holds for some common types of SCM, such as the additive noise model where the structural function is a linear combination of X_{PA_i} and u_i. Based on this assumption, we treat the root cause as the feature where the intervention leads to a significant change in its distribution.

Definition 4 *(Root cause). Given an anomaly* $\mathbf{x} \in \hat{\mathcal{X}}^-$*, the root cause of* $\mathbf{x}$ *is a set of features* $\mathcal{I}$ *that receives an external intervention leading to a significant change in the marginal distributions of exogenous variables* $P(u_{\mathcal{I}})$*.*

It is worth noting that the features that are not the root cause may still exhibit abnormal behaviors. For example, suppose that a feature x_i receives an external intervention, meaning that the probability distribution $P(u_i)$ is changed to a different distribution $P'(u_i)$. Meanwhile, the change in x_i may propagate through the SCM, influencing another downstream feature x_j, where x_j is a child of x_i defined by SCM. As a result, the value of x_j may also become abnormal due to the propagation from the external intervention on x_i through the SCM, despite being a non-root cause.

Anomaly Mitigation. Once the anomaly is detected, one can perform recourse actions to modify the values of certain features to change the abnormal sample to a normal one. As it is natural to modify root cause features only, we consider the problem of anomaly mitigation that asks to find a minimum perturbation on the root cause features $i \in \mathcal{I}$ of a sample to flip the label made by $g(\cdot)$. From the causal perspective, the recourse actions can be modeled as soft interventions. Specifically, define the anomaly mitigation action as a parameter vector $\theta = [\theta_1, ..., \theta_i, ...\theta_d]$ ($\theta_j = 0$ if $j \notin \mathcal{I}$). For each root cause feature x_i, we formulate the action that changes x_i to $x_i + \theta_i$ as a soft intervention. Then, the consequence of the action on a sample $\mathbf{x}$ is the counterfactual instance of $\mathbf{x}$ under the soft intervention. We denote this counterfactual instance as $\mathbf{x}(\theta)$ which depends on the value of θ as well as the underlying SCM.

With the above notations, the problem of anomaly mitigation becomes to find the parameter vector θ that minimizes the cost of the changes made by the mitigation actions, subject to making the counterfactual instance $\mathbf{x}(\theta)$ a normal sample for each original abnormal sample $\mathbf{x}$. It is formulated as that the anomaly detection model should have the anomaly score less than the threshold τ by taking counterfactual sample $\mathbf{x}(\theta)$ as input, i.e., $g(\mathbf{x}(\theta)) \leq \tau$. By using the weighted L2 norm of the action values θ as the quantitative cost measure, given by $\|\mathbf{c} \cdot \theta\|_2$ where $\mathbf{c}$ is a cost vector for describing costs of revising all root cause features ($c_j = 1$ if $j \notin \mathcal{I}$), the problem is finally formulated as

$$\arg\min_{\theta} \|\mathbf{c} \cdot \theta\|_2 \quad \text{s.t. } \forall \mathbf{x} \in \hat{\mathcal{X}}^{-}, g(\mathbf{x}(\theta)) \leq \tau \tag{8.1}$$

Solving the optimization problem in Eq. (8.1) is not trivial. When an action is performed to change x_i to $x_i + \theta_i$, the downstream features that are causally related will also be affected by this action. For example, changing an annual salary usually has an impact on the account balance. Thus, the counterfactual instance $\mathbf{x}(\theta)$ is not simply equal to $\mathbf{x} + \theta$. Ignoring causal relationships will lead to incorrect action recommendations, and counterfactual inference is needed to derive the accurate consequence of actions. Next, we address this challenge by leveraging the Variational Causal Graph Autoencoder (VACA), a state-of-the-art causal

graph autoencoder.

8.3.2 Root Cause Localization

Based on the Definition 4, the idea of localizing the root cause features is to examine the exogenous variables of all features. If an exogenous variable u_i does not follow the regular distribution $P(u_i)$ learned from the normal data, the exogenous variable should be the root cause of an anomaly that receives the external intervention. In this way, even if a feature is abnormal, as long as its exogenous variable follows a similar distribution as the normal data, we treat it as a non-root cause feature and attribute the abnormal behavior to be propagated from its parents.

To this end, we leverage VACA to learn the distribution of the exogenous variable. As mentioned earlier, VACA contains an encoder that maps the features to a hidden exogenous representation, i.e., $\mathbf{z} \sim q_\xi(\mathbf{z}|\mathbf{x}, A)$, as well as a decoder that maps the hidden exogenous representation back to the feature space, i.e., $\mathbf{x} \sim p_\zeta(\mathbf{x}|\mathbf{z}, A)$. The decoder and encoder are implemented as graph neural networks, and all computations follow the structural equation specified by the SCM. For each feature $x_i \in \mathbf{x}$, the purpose of $z_i \in \mathbf{z}$ is to capture the information of x_i that cannot be explained by its parents. Thus, z_i plays a similar role to u_i, which implies that we can examine the distribution of $\mathbf{z}$ to localize the root causes.

Specifically, after training the VACA on normal data, for each sample $\mathbf{x} \in \hat{\mathcal{X}}^-$, we first derive the hidden variable $\mathbf{z}$ based on the encoder of VACA and further calculate the cumulative probability $\Phi(z_i)$ for each exogenous variable based on the distribution fitted from normal data. To identify the root cause features with significant changes in exogenous variables, we set a threshold π for the percentage of the values (in our experiments we use $\pi = 0.125$). If $\Phi(z_i)$ is smaller than π or larger than $1 - \pi$, we consider the feature x_i as a potential root cause. As there can be multiple root cause features in a particular sample, we examine the exogenous variables of all features and get a set of root cause features $\mathcal{I}$.

8.3.3 Causal Graph Autoencoder-based Anomaly Mitigation

For each sample in $\hat{\mathcal{X}}^{-}$, after getting the root causes, we further want to flip the abnormal outcome with minimum actions on root cause features $\mathcal{I}$. The challenge in solving Eq. (8.1) is how to compute counterfactual instance $\mathbf{x}(\theta)$ and solve θ as a continuous optimization problem. We propose to perform the Abduction-Action-Prediction process to conduct the counterfactual inference based on the VACA. Since we perform actions on all features, we consider an iterative Abduction-Action-Prediction process as follows:

$$
\begin{aligned}
& x_1(\theta) = \underbrace{x_1 + \theta_1}_{\text{Action}}, \\
& \text{for } i = 2 \cdots d, \quad \tilde{x}_i \sim \underbrace{\int P(x_i | \{x_j(\theta), \forall j \in \mathsf{PA}_i\}, u_i) \, \underbrace{P(u_i|\mathbf{x})}_{\text{Abduction}} \, \mathrm{d}\tilde{u}_i}_{\text{Prediction}}, \\
& \qquad\qquad x_i(\theta) = \underbrace{\tilde{x}_i + \theta_i}_{\text{Action}},
\end{aligned}
\tag{8.2}
$$

where the features are sorted in topological order. More specifically, to compute $\mathbf{x}(\theta)$, we: (1) infer the updated probability $P(u_i|\mathbf{x})$ (Abduction); (2) perform the action on each feature x_i (Action); and (3) infer the counterfactual values of the downstream features. Steps (2) and (3) are repeated until all features are modified.

There are two challenges in directly applying the VACA to our context. First, the VACA is designed to perform hard intervention where the connections from the parents to the intervened node are cut off. However, in our context, we conduct interventions on all actionable features. By using hard intervention, the parent-child relations of multiple features would be cut-off and cannot pass to downstream nodes, which totally changes the underlying SCM making the generated counterfactual instances infidelity. Therefore, we perform soft interventions on all features where the parent-child relations are preserved, which cannot be achieved by directly using the VACA to perform hard interventions on all features. Second, the hidden exogenous representation $\mathbf{z}$ produced by the encoder may not be in the same space as the features, but we want to compute the recourse on the original feature space. These two challenges mean that the action values cannot be directly added

on $\mathbf{z}$ when we adopt the VACA as the causal graph autoencoder.

We address the above challenges by proposing an iterative algorithm, where each iteration performs a hard intervention on one feature following a topological order. The idea is to pass the influence of each hard intervention to the downstream nodes before performing the hard intervention on the next node in the topological order, in order to simulate how the soft intervention works. Specifically, at the ith iteration, to take the generated action on feature X_i, we perform a hard intervention on X_i as $do(X_i = x_i + \theta_i)$ to obtain the intervened instance $\bar{\mathbf{x}}$. Then, we use the VACA to compute the interventional influence on all descendants of X_i similarly to the above discussion. In this process, $\bar{\mathbf{x}}$ is first transformed to the hidden representation $\bar{\mathbf{z}}$ by the encoder. Meanwhile, the sample $\mathbf{x}$ before the intervention is also transformed to the hidden representation $\mathbf{z}$ by the encoder. Then, $\bar{z}_i$ in $\bar{\mathbf{z}}$ replaces z_i in $\mathbf{z}$ to perform the intervention in the hidden space that is equivalent to performing the intervention in the original feature space. Finally, the interventional influences of this action are transmitted to all descendants of X_i by the decoder which produces the counterfactual instance of the sample under the intervention. It is worth noting that, at the beginning of the ith iteration, the value of x_i has already been updated by taking into account the interventional influences of actions taken on ancestors of X_i. As a result, after we perform the hard intervention on all features, we obtain the counterfactual instance under the recourse.

Finally, for the sake of generalization, instead of computing θ for each instance separately, we define a function $\theta = h_\phi(\mathbf{x})$ for generating the action given $\mathbf{x}$. By integrating the score-based anomaly detection model and VACA for computing the counterfactual instance into Eq. (8.1) and adding the constraint to the objective as regularization, we obtain the final objective function as follows:

$$\mathcal{L}(\phi) = \sum_{\mathbf{x}^{(n)} \in \hat{\mathcal{X}}^-} \max\left\{ g\left(\mathbf{x}^{(n)}(\theta^{(n)})\right) - \alpha\tau, 0 \right\} + \lambda \|\mathbf{c} \cdot \theta^{(n)}\|_2, \tag{8.3}$$

where $\theta^{(n)} = h_\phi(\mathbf{x}^{(n)})$ indicates the action values for the sample $\mathbf{x}^{(n)}$; λ is a hyperparameter balancing the actions on the anomalies and the flipping of abnormal outcomes; α is another hyperparameter controlling how close the anomaly score of counterfactual sample should

Algorithm 6: Training Procedure of RootCLAM for Mitigation Action Prediction

foreach $\mathbf{x} \in \hat{\mathcal{X}}^-$ **do**
  Compute root cause features $\mathcal{I}$ for $\mathbf{x}$
  $\tilde{\mathbf{x}} \leftarrow \mathbf{x}$
  foreach $i \in \mathcal{I}$ **do**
    Compute $\theta_i = h_{\phi_i}(\mathbf{x})$
    Draw $\tilde{\mathbf{z}} \sim q_\xi(\mathbf{z}|\tilde{\mathbf{x}}, A)$ `// Abduction`
    Compute $\bar{x}_i(\theta) = \tilde{x}_i + \theta_i$ `// Action`
    Replace $\tilde{x}_i$ in $\mathbf{x}$ with $\bar{x}_i(\theta)$ and get $\bar{\mathbf{x}}$
    Draw $\bar{\mathbf{z}} \sim q_\xi(\mathbf{z}|\bar{\mathbf{x}}, \bar{A})$
    Replace $\tilde{z}_i$ in $\tilde{\mathbf{z}}$ with $\bar{z}_i$ in $\bar{\mathbf{z}}$ and get $\mathbf{z}(\theta)$
    Draw $\mathbf{x}(\theta) \sim p_\zeta(\mathbf{x}|\mathbf{z}(\theta), \bar{A})$ `// Prediction`
    $\tilde{\mathbf{x}} \leftarrow \mathbf{x}(\theta)$
  Compute $\mathcal{L}(\phi)$ according to Eq. (8.3)
  Compute $\frac{\partial \mathcal{L}(\phi)}{\partial \phi}$
  Update $\phi = \phi - \eta \frac{\partial \mathcal{L}(\phi)}{\partial \phi}$
return h_ϕ

be to the threshold τ. Note that the only trainable parameters in this objective function are the parameters ϕ of $h_\phi(\mathbf{x})$ for generating the action values. Eq. (8.3) can be minimized using off-the-shelf gradient-based optimization algorithms. The training procedure is shown in Algorithm 6.

Practical Considerations. RootCLAM assumes the availability of a causal graph about the data. In practice, the causal graphs may not be available. In this case, we can leverage the causal discovery algorithms to identify the causal relations of observational data [216].

8.4 Experiments

8.4.1 Experimental Setup

Datasets. We conduct experiments on two semi-synthetic datasets and one real-world dataset. For the real-world dataset, as we do not have the ground-truth SCM, we only use it for a case study.

- **Loan** [152] is a *semi-synthetic dataset* about a loan approval scenario derived from the German Credit dataset [217], which consists of 7 endogenous features including loan

amount (L), loan duration (D), income (I), savings (S), education level (E), age (A), and gender (G). The label Y indicates the probability of loan approval. We treat the samples with high approval probabilities as normal and the samples with low approval probabilities as anomalous. The structural equations for data generation are be found in [152]. Due to the space limit, we do not include the equations in this work.

• **Adult** [215] is another *semi-synthetic dataset* about the annual income of a person derived from the real-world Adult dataset [217], which consists of 10 endogenous features of a person including age (A), education level (E), hours worked per week (H), race (R), native country (N), sex (S), work status (W), marital status (M), occupation sector (O), and relationship status (L). We use the SCM designed in the paper [215]. We follow the common settings of the adult dataset to treat samples with income less than \$50k as normal and samples with income more than \$50k as abnormal. We use the structural equations for data generation defined in [215].

Anomaly Injection. To quantify the performance of RootCLAM for root cause localization, we generate abnormal samples by revising exogenous variables of some features. Especially, to generate anomalies, we first randomly select one to four features and then change the distribution of the corresponding exogenous variables. For example, on the Loan dataset, we change the exogenous variable U_S of savings (S) from $\mathcal{N}(0, 25)$ to $\mathcal{N}(-25, 25)$. In this way, we have the ground truth of the root causes for each abnormal sample.

• **Donors** [1] is a *real-world dataset* that aims to predict whether a project on DonorsChoose.org is exciting to the business. The dataset consists of 10 endogenous features of a project, including "at least one teacher-referred donor", "fully funded", "at least one green donation", "great chat", "three or more non teacher-referred donors", "one non teacher-referred donor giving 100 plus", "donation from thoughtful donor", "great messages proportion", "teacher-referred count", "non teacher-referred count". A project must meet all of the following five criteria to be exciting: 1) was fully funded; 2) had at least one teacher-referred donor; 3) has a higher than average percentage of donors leaving an original message; 4) has at least

[1] https://www.kaggle.com/c/kdd-cup-2014-predicting-excitement-at-donors-choose

Table 8.1: Statistics of three datasets.

Dataset	# of Features	Normal Dataset	Unlabeled Dataset	
			Normal	Anomalous
Loan	7	10,000	10,000	1,000
Adult	10	10,000	10,000	1,000
Donors	10	10,000	26,710	2,671

one "green" donation; 5) has one or more of: 5.1) donations from three or more non teacher-referred donors, 5.2) one non teacher-referred donor gave more than $100, 5.3) the project received a donation from a "thoughtful donor".

We consider exciting projects as normal and non-exciting projects as abnormal, while anomaly mitigation is to provide guidance to make the project exciting. As a real-world dataset, we do not have the ground-truth SCM, so we only use it for a case study. The causal graph used in RootCLAM is approximated by the PC algorithm [218] with some minor edits to incorporate the domain knowledge. Figure 8.2 shows the causal graph on Donors.

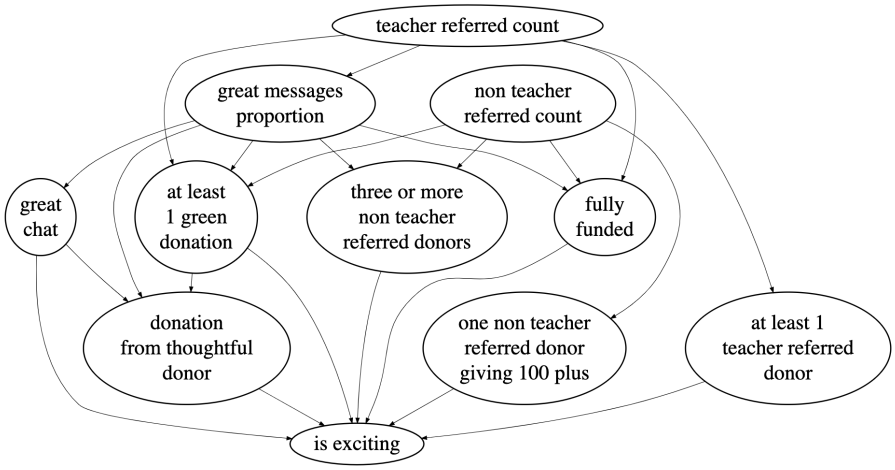


Fig. 8.2: Learned causal graph on Donors.

Table 8.1 shows the statistics of three datasets. To simulate the anomaly detection scenario, we set the ratio of abnormal samples to normal samples as 1:10 in the unlabeled dataset for testing.

Anomaly Detection Models. We adopt Deep Support Vector Data Description (Deep SVDD) [80] and autoencoder-based model (AE) [167] as anomaly detection models $g(\cdot)$.

• **Deep SVDD** derives the anomaly scores of the test sample based on its distance to the center $\boldsymbol{\mu}$ of a hypersphere constructed by normal samples, i.e., $g(\mathbf{x}) = \|r(\mathbf{x}) - \boldsymbol{\mu}\|_2$, where $r(\mathbf{x})$ indicates the hidden representation of a sample $\mathbf{x}$ derived from $r(\cdot)$. Then, the objective function (Eq. (8.3)) for the recourse recommendation can be rewritten as:

$$\mathcal{L}_S(\phi) = \sum_{\mathbf{x}^{(n)} \in \hat{\mathcal{X}}^-} \max\{\|\mathbf{x}^{(n)}(\theta^{(n)}) - \boldsymbol{\mu}\|_2 - \alpha\tau, 0\} + \lambda\|\mathbf{c} \cdot \theta^{(n)}\|_2.$$

• **AE**-based anomaly detection model derives the anomaly scores of samples based on the reconstruction errors of an autoencoder that is trained by normal samples, i.e., $g(\mathbf{x}) = \|\mathbf{x} - \hat{\mathbf{x}}\|_2$, where $\hat{\mathbf{x}}$ indicates the reconstructed sample from autoencoder. Then, to provide recourse for the AE-based anomaly detection model, the objective function (Eq. (8.3)) can be rewritten as:

$$\mathcal{L}_{AE}(\phi) = \sum_{\mathbf{x}^{(n)} \in \hat{\mathcal{X}}^-} \max\{\|\mathbf{x}^{(n)}(\theta^{(n)}) - \widehat{\mathbf{x}^{(n)}(\theta^{(n)})}\|_2 - \alpha\tau, 0\} + \lambda\|\mathbf{c} \cdot \theta^{(n)}\|_2.$$

In our experiments, we first train Deep SVDD and AE on the normal dataset, respectively, and then apply the models on the unlabeled dataset $\mathcal{X}$ and get the corresponding $\hat{\mathcal{X}}^-$ from each model.

Baseline for Root Cause Localization. We compare RootCLAM with CausalRCA [219], a state-of-the-art approach for root cause analysis. We use the implementation in the DoWhy package [220].

Baselines for Anomaly Mitigation. To our best knowledge, there is no causal anomaly mitigation approach. We compare RootCLAM with two baselines, C-CHVAE and NaiveAM.

• **C-CHVAE** [221] can find feasible counterfactual flipping the output of classifiers, but does not consider the underlying causal relationships when generating counterfactuals. We adapt C-CHVAE by replacing classifiers with anomaly detection models.

• **NaiveAM** directly predicts the action values on all feasible features without considering the underlying causal structure. Specifically, given a set of abnormal sample $\hat{\mathcal{X}}^-$, we still train a neural network $\hat{h}_\phi(\cdot)$ to predict the action value, $\hat{\theta} = \hat{h}_\phi(\mathbf{x})$, where $\mathbf{x} \in \hat{\mathcal{X}}^-$. However,

instead of generating the counterfactual samples guided by SCM, NaiveAM generates the revised samples by simply adding the action value on the original sample, i.e.,

$$\hat{\mathbf{x}}(\theta) = \mathbf{x} + \hat{\theta}. \tag{8.4}$$

NaiveAM is also trained on the objective function in Eq. (8.3) by replacing θ and $\mathbf{x}(\theta)$ with $\hat{\theta}$ and $\hat{\mathbf{x}}(\theta)$, respectively. After training, in order to evaluate whether the predicted actions can really flip the labels in the counterfactual world, on Adult and Loan datasets, we also generate the counterfactual samples based on the structural equations given $\hat{\theta}$, denoted as $\hat{\mathbf{x}}(\theta)$ (SCM).

Implementation Details. For a fair comparison, the hyperparameters of neural networks for action prediction in NaiveAM and RootCLAM are the same. We set the hyperparameters for VACA by following [215]. By default, the threshold for anomaly detection is set to 0.995 quantiles of the training samples' distances to the center (Deep SVDD) or the reconstruction errors (AE). For the intervention value prediction, we utilize a feed-forward network with structure m-2048-2048-n, where m is the input dimension and n is the number of actionable features. The costs $\mathbf{c}$ in Eq. (8.3) are user-specified functions for each root cause feature to represent preferences or feasibility of features changing. The cost functions can be changed according to the requirements or prior knowledge. To be fair, we use the standard deviation of each root cause feature as the cost for NaiveAM and RootCLAM. Our code is available online [2].

8.4.2 Experimental Results

The performance of anomaly detection. We evaluate the performance of anomaly detection in terms of the F1 score, the area under the receiver operating characteristic (AUROC), and the area under the precision-recall curve (AUPRC). Table 8.2 shows the anomaly detection evaluation results. In short, both AE and Deep SVDD can achieve good performance for anomaly detection, meaning that the predicted abnormal samples $\tilde{\mathcal{X}}^-$ have

[2] https://github.com/hanxiao0607/RootCLAM

high accuracy. It lays a solid foundation for action prediction.

After getting the abnormal set $\tilde{\mathcal{X}}^-$ of each dataset, we then train and test the root cause localization and anomaly mitigation with the train/test split ratio of 80/20.

Table 8.2: Anomaly detection on the unlabeled datasets.

Dataset	AE			Deep SVDD		
	F1	AUROC	AUPRC	F1	AUROC	AUPRC
Loan	0.923	0.998	0.982	0.888	0.993	0.944
Adult	0.893	0.984	0.899	0.837	0.923	0.823
Donors	0.967	0.998	0.979	0.988	0.999	0.998

Table 8.3: Root cause localization on the unlabeled datasets.

		AE				Deep SVDD			
		Accu.	Pre.	Rec.	F1	Accu.	Pre.	Rec.	F1
Loan	CausalRCA	0.707	0.522	0.680	0.591	0.704	0.508	0.561	0.533
	RootCLAM	0.728	0.545	0.765	0.636	0.727	0.523	0.776	0.631
Adult	CausalRCA	0.853	0.554	0.615	0.583	0.850	0.546	0.593	0.569
	RootCLAM	0.866	0.567	0.849	0.680	0.855	0.544	0.794	0.646

Table 8.4: The performance of anomaly mitigation in terms of the flipping ratio and norm of action values.

	Metric		Loan			Adult		
			C-CHVAE	NaiveAM	RootCLAM	C-CHVAE	NaiveAM	RootCLAM
AE	Flipping Ratio	$\hat{\mathbf{Y}}$	1.000	1.000	0.891	0.114	0.885	0.960
		$\mathbf{Y}$	0.499	0.337	0.839	0.065	0.598	1.000
	Action Value	$\|\mathbf{c} \cdot \theta\|_2$	22.383	6.382	5.185	115.862	34.389	14.504
Deep SVDD	Flipping Ratio	$\hat{\mathbf{Y}}$	1.000	1.000	0.988	0.671	1.000	1.000
		$\mathbf{Y}$	0.496	0.847	0.963	0.586	0.595	1.000
	Action Value	$\|\mathbf{c} \cdot \theta\|_2$	17.832	13.474	5.862	63.124	69.169	29.274

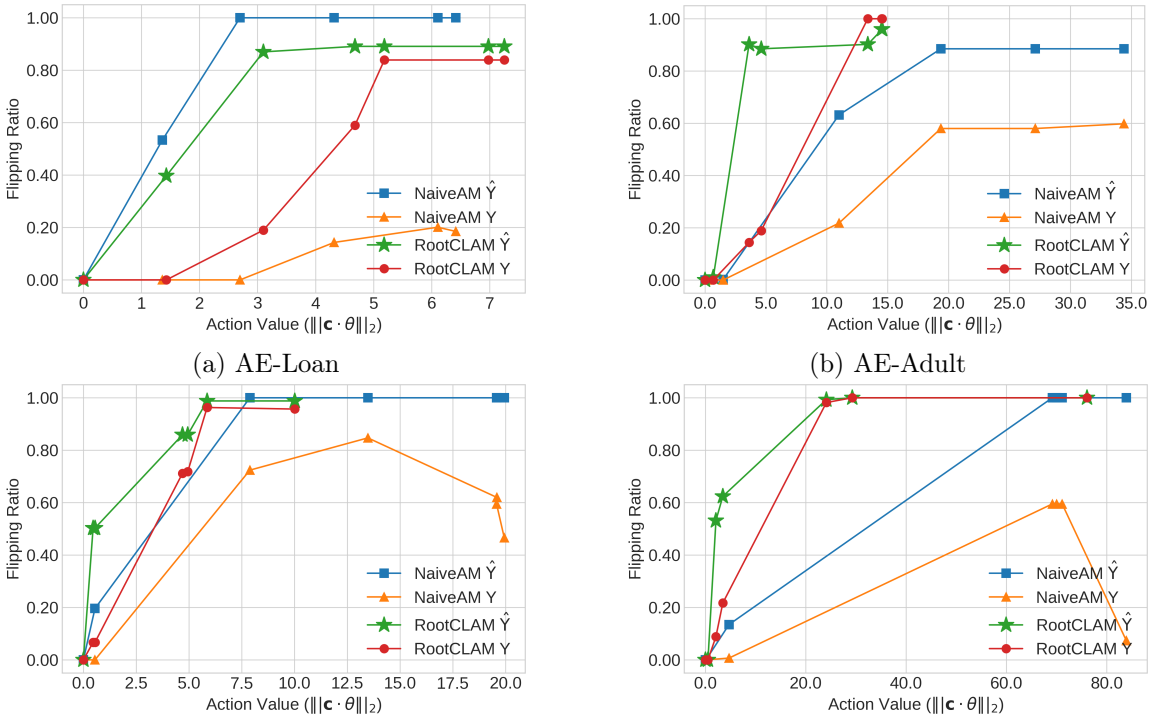
The performance of RootCLAM on root cause localization. After detecting the anomalies, the next step is to identify the root causes. We further evaluate the performance of RootCLAM on root cause localization in terms of accuracy, precision, recall, and F1. As shown in Table 8.3, RootCLAM outperforms CausalRCA in terms of accuracy and F1

score on both datasets. Especially, RootCLAM achieves much higher recall compared with CausalRCA, which means RootCLAM can identify more root cause features.

The performance of RootCLAM on counterfactual sample generation. Generating high-fidelity counterfactual samples is a fundamental requirement for predicting high-quality actions to flip the labels. We evaluate the quality of estimated counterfactual samples in terms of the mean squared error (MSE) as well as the standard deviation of the squared error (SSE) between the true and the estimated counterfactual samples on the Loan and Adult datasets that have the ground truth structural equations for data generation. On Loan, the MSE and SSE are 3.976 and 2.266, respectively, while on Adult, the MSE and SSE are 3.334 and 0.900, respectively. It means RootCLAM can get good counterfactual samples.

The performance of anomaly mitigation in terms of flipping ratio. We evaluate the performance of anomaly mitigation by examining the flipping ratio that anomalies are transferred to normal through the interventions predicted by $h_\phi(\cdot)$. The flipping ratio is calculated as the fraction of the number of flipped samples over all detected anomalies. Because we would like to check whether the predicted actions can really flip the labels in the counterfactual world, given the predicted action values from RootCLAM and baselines, we also use the ground-truth structural equations to generate the counterfactual samples. We calculate the flipping ratio by considering two scenarios: 1) whether the anomaly detection model would detect the counterfactual samples as normal, denoted as $\hat{\mathbf{Y}}$; 2) whether the ground truth Y is flipping from abnormal to normal based on the ground-truth structural equations, denoted as $\mathbf{Y}$.

As shown in Table 8.4, on Loan and Adult datasets, both RootCLAM and NaiveAM can successfully flip almost all abnormal samples detected. However, C-CHVAE cannot get good performance on the Adult dataset. For the flipping ratio on the ground truth label $\mathbf{Y}$, RootCLAM can successfully flip most of the abnormal samples on both datasets. It means the actions predicted by RootCLAM can reverse the majority of abnormal samples to normal in the counterfactual world. However, NaiveAM and C-CHVAE cannot get good



(a) AE-Loan (b) AE-Adult

(c) DeepSVDD-Loan (d) DeepSVDD-Adult

Fig. 8.3: Trade-off between flipping ratio and action value.

performance on flipping the ground truth label $\mathbf{Y}$. This is because both NaiveAM and C-CHVAE do not consider the underlying causal structure in the data, showing that simply revising the root cause features is not sufficient to flip the ground-truth labels.

The performance of anomaly mitigation in terms of the norm of action values. One requirement for anomaly mitigation is to conduct minimal interventions on the original samples. We further calculate the norm of action values, i.e., $\|\mathbf{c}\cdot\theta\|_2$, on the samples with successfully flipping labels. As shown in the last row of Table 8.4, RootCLAM makes much smaller changes on the original samples and still has higher flipping ratios on the ground truth label $\mathbf{Y}$.

The trade-off between the flipping ratio and the norm of action values. In the objective function (Eq. (8.3)), λ as a hyper-parameter controls the trade-off between the norm of action values and the flipping ratio in the training phase. A large λ value indicates that the model will be trained with an emphasis on minimizing the action values. Given the pre-

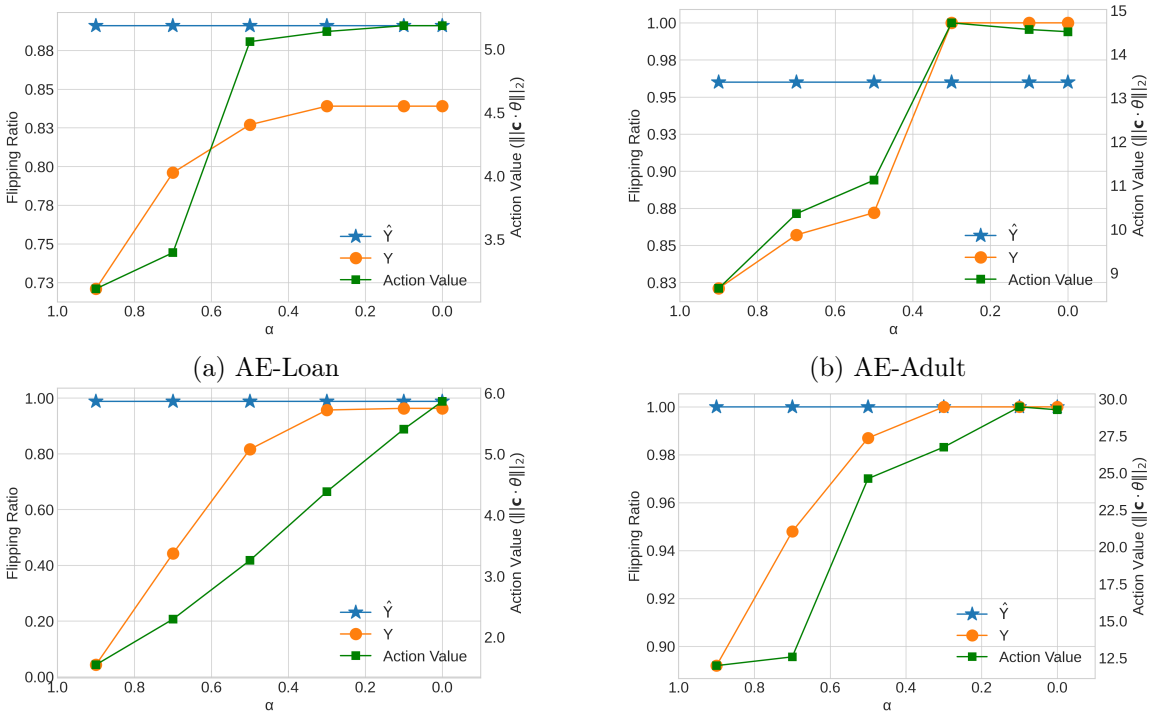
dicted action values, we adopt ground-truth structural equations to generate counterfactual samples and then check the flipping ratios based on anomaly detection models ($\hat{\mathbf{Y}}$) and the ground truth label $\mathbf{Y}$. Figure 8.3 shows the results. Each point on the line from left to right indicates the result from one λ value in the set $[1, 10^{-1}, 10^{-2}, 10^{-3}, 5 \times 10^{-4}, 10^{-4}, 10^{-5}]$. Because good mitigation action predictions should be able to flip the label with minimum changes, closing to the top-left corner indicates good performance.

First, on both datasets, we can notice that in most cases, increasing the norm of action values can improve the flipping ratio. It means most of the abnormal samples can be flipped as normal ones with sufficient changes. Therefore, the key is to conduct minimum interventions on the original samples. The exception is that when having a large norm of action values on NaiveAM to flip the ground-truth label Y, we can notice the flipping ratio either does not changes or drops, which shows the importance to consider the causal relationships when applying the mitigation actions.

As shown in Figure 8.3a, on the Loan dataset, both NaiveAM and RootCLAM can achieve a high flipping ratio evaluated by AE with very small action values ($\|\mathbf{c} \cdot \theta\|_2 < 3$). On the other hand, in terms of flipping the ground truth label $\mathbf{Y}$, RootCLAM can achieve a much higher flipping ratio compared with NaiveAM. On the Adult dataset, as shown in Figure 8.3b, RootCLAM can still achieve a near 100% flipping ratio on the detected label $\hat{\mathbf{Y}}$ as well as the ground truth label $\mathbf{Y}$, while the performance of NaiveAM is poor.

As shown in Figure 8.3c, on the Loan dataset, both NaiveAM and RootCLAM can achieve a near 100% flipping ratio evaluated by Deep SVDD with very small action values ($\|\mathbf{c} \cdot \theta\|_2 < 7.5$). On the other hand, in terms of flipping the ground truth label $\mathbf{Y}$, RootCLAM can achieve a higher flipping ratio with a lower norm of action values compared with NaiveAM. On the Adult dataset, as shown in Figure 8.3d, RootCLAM can still achieve better performance over NaiveAM by setting various λ values for flipping both the ground truth label $\mathbf{Y}$ and detected label $\hat{\mathbf{Y}}$.

Sensitivity analysis by setting various α in the objective function (Eq. (8.3)) for anomaly mitigation. The hyperparameter α in Eq. (8.3) controls how close the



(a) AE-Loan

(b) AE-Adult

(c) DeepSVDD-Loan

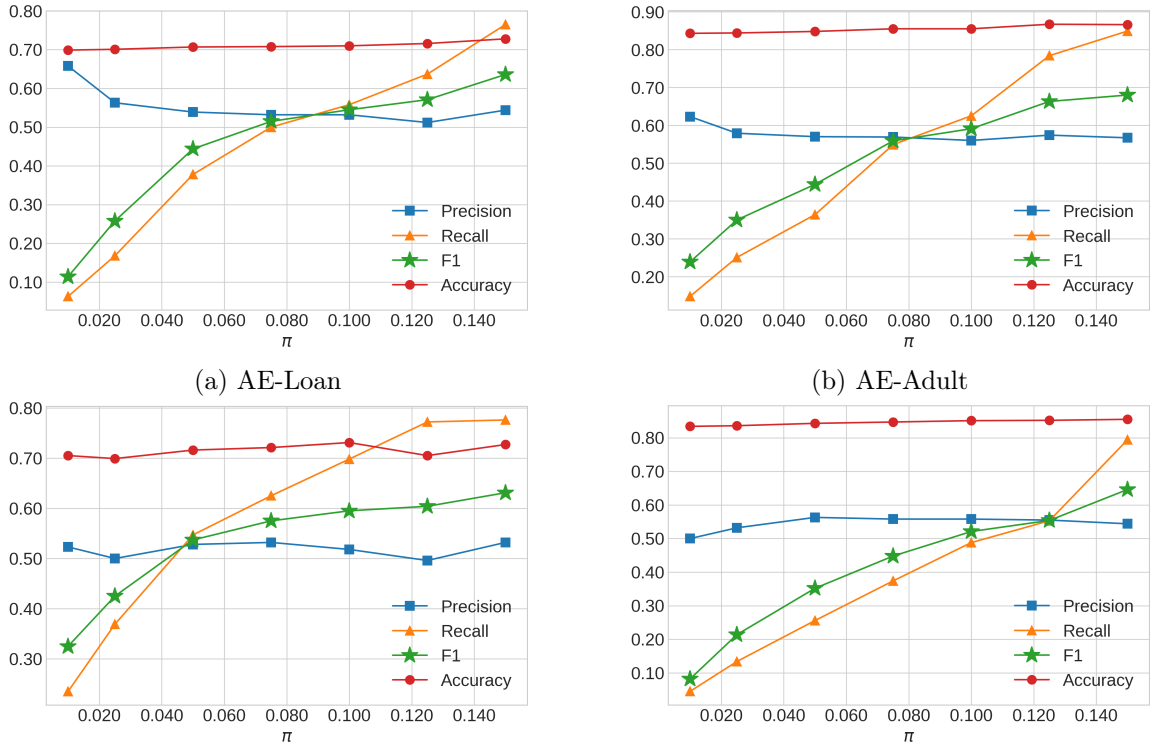
(d) DeepSVDD-Adult

Fig. 8.4: Sensitivity analysis by setting various α.

anomaly scores of counterfactual samples should be to the threshold. We evaluate the flipping ratios by tuning the hyperparameter α. A smaller α indicates that the counterfactual samples should be closer to the center of normal samples (DeepSVDD) or have a smaller reconstruction error (AE).

Figures 8.4a to 8.4d have similar observations. First, in all settings, the flipping ratios in terms of detected label $\hat{\mathbf{Y}}$ are high and keep stable, which shows that a small intervention on abnormal samples can flip the detecting results. Meanwhile, by reducing the α value, we can observe the increase of the flipping ratio in terms of ground-truth label $\mathbf{Y}$ as well as the norm of action value, which means flipping the ground-truth label requires more interventions.

Sensitivity analysis by setting various π for root cause localization. Because the root cause features are identified with a small or large cumulative probability controlled by π, we evaluate the performance of root cause localization by tuning the threshold π. As shown



(a) AE-Loan (b) AE-Adult

(c) DeepSVDD-Loan (d) DeepSVDD-Adult

Fig. 8.5: Sensitivity analysis by setting various π.

in Figure 8.5, on both datasets, increasing the threshold π can increase the recall of root cause localization with a minor negative impact on the precision. The overall performance in terms of accuracy and F1 keeps improving with a large π value.

Case study. We conduct case studies to show that RootCLAM can identify root causes and recommend mitigation actions.

Loan Dataset. Table 8.5 shows the case study on the Loan dataset with the root cause features $\mathcal{I}$={"loan amount", "loan duration"}. For the semi-synthetic Loan dataset, the positive values of features usually indicate above the average, while negative values indicate below the average. The rows $\hat{\mathbf{x}}(\theta)$ (SCM) and $\mathbf{x}(\theta)$ (SCM) indicate counterfactual samples generated based on the structural equations given the predicted action values from NaiveAM and RootCLAM, respectively, while $\mathbf{x}(\theta)$ (Eq. 8.2) indicates the counterfactual samples generated based on our approach.

Given an abnormal sample $\mathbf{x}$, RootCLAM successfully identifies the two root cause

Table 8.5: Case study on the Loan dataset, where "loan amount" (L) and "loan duration" (D) are root cause features.

			G	A	E	L	D	I	S	Y
		$\mathbf{x}$	0	-1.878	-0.095	2.423	5.634	-2.064	0.697	0.003
AE	NaiveAM	$\hat{\theta}$	/	/	/	-2.441	-8.159	0.217	-6.342	/
		$\hat{\mathbf{x}}(\theta)$ (SCM)	0	-1.878	-0.095	-0.017	-2.525	-1.847	-5.646	0.838
	RootCLAM	θ	/	/	/	-5.958	-11.336	/	/	/
		$\mathbf{x}(\theta)$ (Eq. 8.2)	0	-2.133	-0.089	-3.125	-9.154	-1.982	0.162	0.954
		$\mathbf{x}(\theta)$ (SCM)	0	-1.878	-0.095	-3.534	-11.659	-2.064	0.697	0.976
Deep SVDD	NaiveAM	$\hat{\theta}$	/	/	/	-1.655	-3.911	-1.869	-0.161	/
		$\hat{\mathbf{x}}(\theta)$ (SCM)	0	-1.878	-0.095	0.769	1.723	-3.932	0.536	0.083
	RootCLAM	θ	/	/	/	-2.157	-12.324	/	/	/
		$\mathbf{x}(\theta)$ (Eq. 8.2)	0	-2.134	-0.089	0.453	-7.600	-1.982	0.162	0.818
		$\mathbf{x}(\theta)$ (SCM)	0	-1.878	-0.095	0.267	-8.847	-2.064	0.697	0.850

G – 'gender', A – 'age', E – 'education level', L – 'loan amount', D – 'loan duration', I – 'income', S – 'savings'

features. Meanwhile, the mitigation actions predicted by RootCLAM indicate that reducing the loan amount (L) and the loan duration (D) can significantly improve the loan approval rate. On the other hand, although NaiveAM predicts more actions for anomaly mitigation, the odds of loan approval based on NaiveAM are still lower than the result from RootCLAM.

Table 8.6: Case study on the Adult dataset, where "hours worked per week" (H) is the root cause feature

			R	A	N	S	E	H	W	M	O	L	I
		$\mathbf{x}$	2	36.401	1	1	5.264	52.520	1	1	2	1	60,816
AE	NaiveAM	$\hat{\theta}$	/	9.219	/	/	0.266	-4.148	/	/	/	/	/
		$\hat{\mathbf{x}}(\theta)$ (SCM)	2	45.620	1	1	5.529	48.372	1	1	2	1	60,816
	RootCLAM	θ	/	/	/	/	/	-9.672	/	/	/	/	/
		$\mathbf{x}(\theta)$ (Eq. 8.2)	2	38.293	1	1	5.370	44.791	1	1	2	1	45,816
		$\mathbf{x}(\theta)$ (SCM)	2	36.401	1	1	5.264	42.848	1	1	2	1	45,816
Deep SVDD	NaiveAM	$\hat{\theta}$	/	20.734	/	/	0.549	-8.573	/	/	/	/	/
		$\hat{\mathbf{x}}(\theta)$ (SCM)	2	57.135	1	1	5.813	43.947	1	1	2	1	45,816
	RootCLAM	θ	/	/	/	/	/	-12.672	/	/	/	/	/
		$\mathbf{x}(\theta)$ (Eq. 8.2)	2	38.293	1	1	5.370	40.217	1	1	2	1	45,816
		$\mathbf{x}(\theta)$ (SCM)	2	36.401	1	1	5.264	39.848	1	1	2	1	45,816

R – 'race', A – 'age', N – 'native country', S – 'sex', E – 'education level', H – 'hours worked per week', W – 'work status', M – 'marital status', O – 'occupation sector', L – 'relationship status', I – 'income'

Adult Dataset. Table 8.6 shows the case study on the Adult dataset with the root cause features $\mathcal{I}$={"hours worked per week"}. In this case, the action values predicted by RootCLAM on the hours worked per week is negative, which indicates that reducing hours worked per week can make the sample normal (Income less than 50k). As we consider an income higher than 50k as abnormal, our predicted action value can indicate why an

Table 8.7: Case study on the Donors dataset

			F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	Y
		$\mathbf{x}$	0	1	0	1	0	1	0	66	0	3	1
AE	NaiveAM	$\hat{\theta}$	/	/	/	/	/	/	/	34	5	-5	/
		$\hat{\mathbf{x}}(\theta)$ (Eq. 8.4)	0	1	0	1	0	1	0	100	5	-3	1
	RootCLAM	θ	/	/	/	/	/	/	/	26	2	5	/
		$\mathbf{x}(\theta)$ (Eq. 8.2)	1	1	1	1	1	1	0	100	2	6	0
Deep SVDD	NaiveAM	$\hat{\theta}$	/	/	/	/	/	/	/	34	5	-4	/
		$\hat{\mathbf{x}}(\theta)$ (Eq. 8.4)	0	1	0	1	0	1	0	100	5	-2	1
	RootCLAM	θ	/	/	/	/	/	/	/	26	2	5	/
		$\mathbf{x}(\theta)$ (Eq. 8.2)	1	1	1	1	1	1	0	100	2	6	0

F1 – 'at least 1 teacher-referred donor', F2 – 'fully funded', F3 – 'at least 1 green donation', F4 – 'great chat', F5 – 'three or more non teacher-referred donors', F6 – 'one non teacher-referred donor giving 100 plus', F7 – 'donation from thoughtful donor', F8 – 'great messages proportion', F9 – 'teacher-referred count', F10 – 'non teacher-referred count'.

individual can have a high income, i.e., having a large number of hours worked per week. On the other hand, NaiveAM cannot ensure the success of anomaly mitigation. For the AE-based model, the income value is not changed based on the action values predicted from NaiveAM. For the DeepSVDD-based model, although the action values predicted by NaiveAM successfully reduce the income, NaiveAM predicts larger action values compared to RootCLAM.

Donors Dataset. We consider a project that is not exciting as an anomaly and aim to flip the label. Based on the definition of an exciting project, the original sample $\mathbf{x}$ in Table 8.7 is not exciting because this project fails to meet the requirements of at least one teacher-referred donor (F1) and at least one "green" donation (F3). In this case study, RootCLAM identifies "great messages proportion" (F8), "teacher-referred count" (F9), and "non teacher-referred count" (F10) as the root cause features. All root cause features are ancestors of exciting requirements shown in Figure 8.2. After getting the action values from $h_\phi(\cdot)$, we round to the nearest integer. Because we do not have the ground truth structural equations for Donors, Table 8.7 only shows the predicted counterfactual samples from the models.

For the purpose of anomaly mitigation, in order to make the project exciting, as shown in Table 8.7, the project host should try to have more 'great messages', increase the 'teacher-referred count' as well as 'non-teacher-referred count'. After doing such changes, as shown in

the last row, some key features, such as F1, F3, and F5, are flipped to 1. Then, we can notice that the counterfactual sample will be exciting. On the other hand, because NaiveAM does not consider the causal relationships among features, NaiveAM cannot derive the impact on other features after changing the root cause features. As a result, NaiveAM cannot flip the label.

8.5 Summary

In this work, we developed RootCLAM, a framework for root cause analysis and anomaly mitigation through causal inference. RootCLAM first learns a Variational Causal Graph Autoencoder from the normal data. Then, given an abnormal sample, RootCLAM identifies root cause features with the exogenous variables significantly deviated from the regular data. Then, RootCLAM computes mitigation actions as soft interventions on root cause features that can flip the anomalies to normal. Experiments show that RootCLAM achieves state-of-the-art performance on root cause localization and can further successfully fix most of the anomalies. The early version of this work is published at CIKM 2023 [222].

CHAPTER 9

ACHIEVING COUNTERFACTUAL FAIRNESS FOR ANOMALY DETECTION

In this chapter, we propose a Counterfactually Fair Anomaly Detection (CFAD) framework to address the challenges of achieving counterfactual fairness for anomaly detection. One major challenge is the unobservability of counterfactual data, as any intervention on one feature will subsequently affect the values of other features due to the underlying causal mechanism. To overcome this challenge, we develop an approach to generate counterfactual data based on a graph autoencoder. The second challenge is achieving counterfactual fairness while preserving high anomaly detection performance. To achieve this, we use an autoencoder as the base anomaly detection model and apply adversarial training to ensure that the hidden representations of factual and counterfactual data derived from the encoder cannot be distinguished by a discriminator. This leads to similar detection results for both factual and counterfactual data, ensuring counterfactual fairness. We do not require knowledge of the causal graph and structural equations but only assume that the data generation follows a generalized linear SCM. Experimental results show that CFAD enables counterfactual fairness for anomaly detection.

9.1 Introduction

Anomaly detection, which aims to detect samples that are deviated from the normal ones, has a wide spectrum of applications. Recently, deep anomaly detection models, powered by complex deep neural nets, have made promising progress in effectively detecting anomalies. Besides effectiveness, researchers recently notice the importance of taking the societal impact of anomaly detection into consideration as many anomaly detection tasks involve human individuals. Fairness as one fundamental component to build trustworthy AI has received much attention. Recent studies have shown that anomaly detection models can incur discrimination against certain groups. For example, a deep anomaly detection model

could overly flag black males as anomalies [223]. In the scenarios of credit risk analysis, anomaly detection models predict more females as anomalies [224].

Several fair anomaly detection models have been proposed, which ensure no discrimination against a particular group based on the sensitive feature [223–228]. However, these approaches mainly focus on achieving association-based fairness notions like demographic parity. Recent studies have demonstrated the importance of treating fairness as causation-based notions that concern the causal effect of the sensitive feature on the model outcomes [229–231]. Counterfactual fairness is one important causation-based fairness notion [232, 233]. It considers that a model is fair if, for a particular individual, the model outcome in the factual world is the same as that in the counterfactual world where the individual had belonged to a different group. To the best of our knowledge, no studies have been conducted to ensure counterfactual fairness in anomaly detection.

In this work, we focus on counterfactual fairness for anomaly detection with the goal to ensure that the detection outcomes remain consistent in both the factual and counterfactual worlds. Achieving counterfactual fairness for anomaly detection is challenging. First, we can only observe the factual data. The counterfactual data are unobservable and cannot be obtained by simply changing the sensitive feature of the factual data. This is because the data generation is governed by an underlying causal mechanism where any intervention on one feature will subsequently affect the values of other features. Second, in anomaly detection, we can only observe factual normal data. Building a detection model which ensures the detection results be unchanged for individuals across the factual and counterfactual worlds while also preserving high anomaly detection performance imposes additional challenges.

To tackle the above challenges, we propose a Counterfactually Fair Anomaly Detection (CFAD) framework. We do not require the knowledge of the causal graph and structural equations but only assume that the data generation follows a generalized linear Structural Causal Model (SCM). We use an autoencoder as the base anomaly detection model where the anomaly score of a sample is derived based on the reconstruction error of the autoencoder. Then, we propose a two-phase approach. In the first phase, motivated by [211] which

leverages the graph autoencoder for causal structure learning from observed data, we develop an approach to generate counterfactual data based on a graph autoencoder. In the second phase, we apply adversarial training [234, 235] on a vanilla autoencoder to achieve counterfactual fairness for anomaly detection. The idea is to ensure that the hidden representations of factual and counterfactual data derived from the encoder cannot be distinguished by a discriminator. As a result, the reconstruction error, i.e., anomaly score, will not differ much between the factual and counterfactual data, leading to similar detection results for both factual and counterfactual data.

9.2 Preliminary

Structural Causal Model (SCM). Our work adopts Pearl's Structural Causal Model (SCM) [150] as the prime methodology for defining and measuring counterfactual fairness. Throughout this work, we use the upper/lower case alphabet to represent variables/values.

Definition 5 *An SCM is a triple* $\mathcal{M} = \{U, V, F\}$ *where*

1) U *is a set of exogenous variables that are determined by factors outside the model. A joint probability distribution* $P(u)$ *is defined over the variables in* U*.*

2) V *is a set of endogenous variables that are determined by variables in* $U \cup V$*.*

3) F *is a set of deterministic functions* $\{f_1, \ldots, f_n\}$*; for each* $X_i \in V$*, a corresponding function* f_i *is a mapping from* $U \cup (V \setminus \{X_i\})$ *to* X_i*, i.e.,* $X_i = f_i(X_{pa(i)}, U_i)$*, where* $X_{pa(i)} \subseteq V \setminus \{X_i\}$ *called the parents of* X_i*, and* $U_i \subseteq U$*.*

An SCM is often illustrated by a causal graph $\mathcal{G}$ where each observed variable is represented by a node, and the causal relationships are represented by directed edges $\rightarrow$. In this graphical representation, the definition of parents is consistent with that in the SCM.

Inferring causal effects in the SCM is facilitated by the do-operator which simulates the physical interventions that force some variable $X \in V$ to take a certain value x. For an SCM $\mathcal{M}$, intervention $\text{do}(X = x)$ is equivalent to replacing original function in F with $X = x$. After the replacement, the distributions of all variables that are the descendants of

X may be changed. We call the SCM after the intervention the submodel, denoted by $\mathcal{M}[x]$. For any variable $Y \in V$ which is affected by the intervention, its interventional variant in submodel $M[x]$ is denoted by $Y[x]$.

Counterfactuals. Counterfactuals are about answering questions such as for two variables $X, Y \in V$, whether Y would be y had X been x in unit (or situation) $U = u$. Such question involves two worlds, the factual world represented by $\mathcal{M}$ and the counterfactual world represented by $\mathcal{M}[x]$, and hence cannot be answered directly by the do-operator. When the complete knowledge of the SCM is known, the counterfactual quantity can be computed by the three-step process:

1) Abduction: Update $P(u)$ by evidence e to obtain $P(u|e)$.

2) Action: Modify $\mathcal{M}$ by performing intervention do(x) to obtain the submodel $\mathcal{M}[x]$.

3) Prediction: Use modified submodel $\mathcal{M}[x]$ with updated probability $P(u|e)$ to compute the probability of $Y = y$.

9.3 CFAD

9.3.1 Counterfactual Fairness

We start by defining counterfactual fairness in the context of anomaly detection. Following the typical anomaly detection setting, we assume a training set $\mathcal{D} = \{d^{(n)}\}_{n=1}^{N}$ which consists of N normal samples/individuals and a test set that consists of both normal samples and anomalies. Each sample is given by $d^{(n)} = \{s^{(n)}, x^{(n)}\}$ where S denotes a binary sensitive variable and $X = \{X_i \mid i = 1 : m\}$ denotes all other variables (i.e., profile attributes). We then use Y to denote the anomaly label. For representation, we use $S = \{s^+, s^-\}$ to denote advantage and disadvantage groups respectively, and use $Y = \{0, 1\}$ to denote normal samples and anomalies respectively. The goal is to learn a detection model for computing an anomaly score $g(x^{(n)})$ based on the profile attributes for each individual n, which can be used to judge whether it is an anomaly.

To define counterfactual fairness, similar to [232], for each individual $d^{(n)}$ we consider its instance in the counterfactual world $\mathcal{M}_s$ by flipping the value of its sensitive variable to the opposite s (i.e., s^+ becomes s^- and vice versa), denoted by $d_{\text{cf}}^{(n)} = \{s, x_{\text{cf}}^{(n)}\}$ where $x_{\text{cf}}^{(n)}$ represents the profile attributes in the counterfactual world. Note that $x_{\text{cf}}^{(n)}$ may not be the same as $x^{(n)}$ due to the causal relation between S and X in the underlying data generation mechanism. Then, counterfactual fairness is defined as:

Definition 6 *An anomaly detection model is counterfactually fair if for each individual n we have $g(x^{(n)}) = g(x_{cf}^{(n)})$.*

9.3.2 Overview of Counterfactually Fair Anomaly Detection (CFAD)

The goal of CFAD is to train an anomaly detection model on $\mathcal{D}$ that can: (1) effectively detect anomalies, and (2) ensure counterfactual fairness. To achieve this goal, CFAD consists of two phases, counterfactual data generation and fair anomaly detection. Counterfactual data generation is to generate a counterfactual dataset $\mathcal{D}_{\text{cf}} = \{d_{\text{cf}}^{(n)}\}_{n=1}^{N}$ of $\mathcal{D}$ in which each counterfactual sample is generated by the submodel which flips the value of the sensitive variable to its counterpart. To this end, we assume a generalized linear SCM and develop a novel graph autoencoder for data generation. In the second phase, we make use of a standard autoencoder for anomaly detection where the anomaly score is derived based on the reconstruction error. To achieve fairness, we develop an adversarial training framework to train the autoencoder by taking the factual and counterfactual data as inputs. The idea is to make the hidden representations of the autoencoder not encode the information of the sensitive variable so that intervening the sensitive variable would not change the detection outcome. Figure 9.1 shows the framework of CFAD.

9.3.3 Phase One: Counterfactual Data Generation

We assume that the data generation follows a generalized linear SCM, which is a common assumption in gradient-based causal discovery. To ease representation, we also assume that S has no parents in the SCM. Our method can easily extend to cases where S has

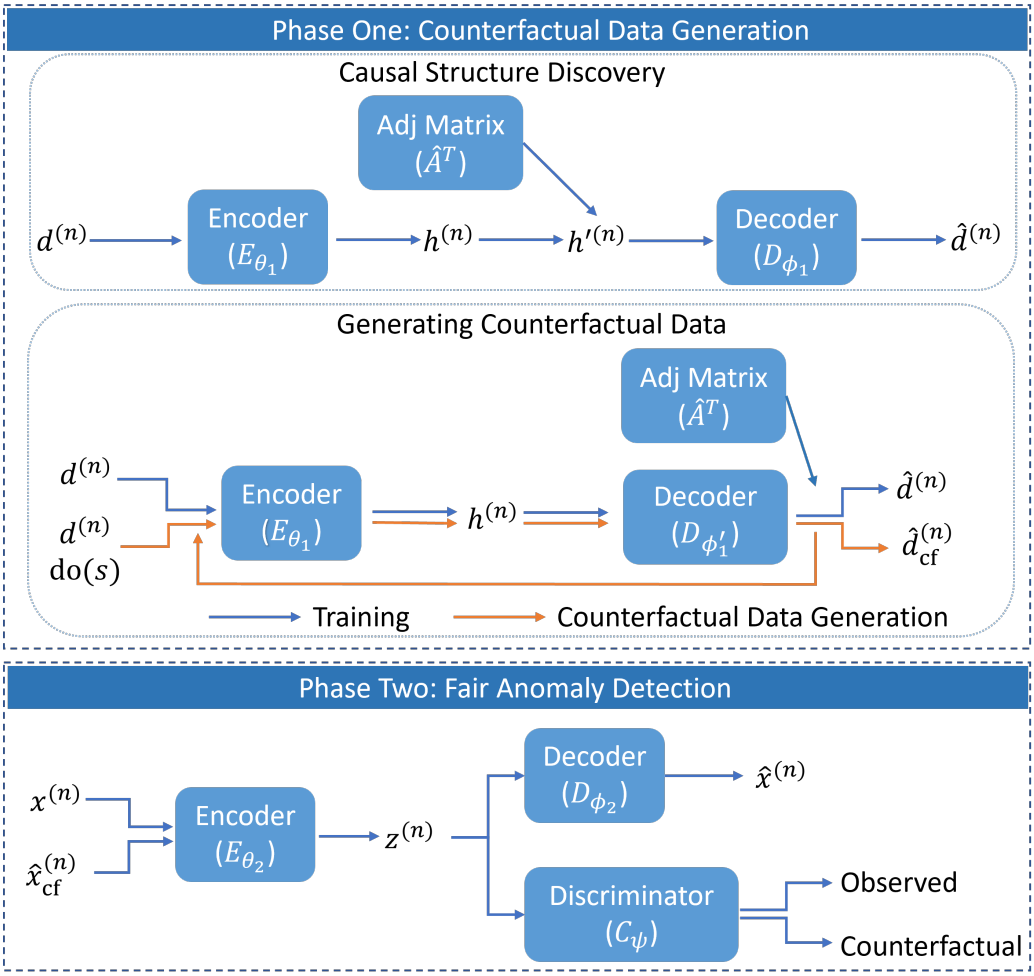


Fig. 9.1: Framework of CFAD

parents by keeping the values of S's parents unchanged in the counterfactual world since the intervention on S has no influence on its parents. Thus, W.L.O.G. the structural equation of each variable X_i in X can be written as follows.

$$X_i = A_{1,i} \cdot f(S) + \sum_{X_j \in X_{pa(i)} \setminus \{S\}} A_{j,i} \cdot f(X_j) + U_i, \tag{9.1}$$

where $f(\cdot)$ can be any linear/nonlinear function and $A_{j,i}$ is an element in the adjacency matrix $A \in \mathbb{R}^{(m+1)\times(m+1)}$ which indicates the weights of the generalized linear SCM. Each

sample $d^{(n)} = \{s^{(n)}, \{x_i^{(n)} \mid i = 1 : m\}\}$ satisfies Eq. (9.1). Following the Abduction-Action-Prediction process, from Eq. (9.1), we have

$$u_i^{(n)} = x_i^{(n)} - A_{1,i} \cdot f(s^{(n)}) - \sum_{X_j \in X_{pa(i)} \setminus \{S\}} A_{j,i} \cdot f(x_j^{(n)}).$$

Meanwhile, by performing intervention to flip $s^{(n)}$ to its counterpart s, the structural equation of counterfactual variable $X_i[s]$ in the submodel $\mathcal{M}[s]$ of Eq. (9.1) is given by

$$X_i[s] = A_{1,i} \cdot f(s) + \sum_{X_j \in X_{pa(i)} \setminus \{S\}} A_{j,i} \cdot f(X_j[s]) + U_i. \tag{9.2}$$

Note that S is fixed to s by the intervention and U_i is not affected by the intervention. Denoting the counterfactual of $d^{(n)}$ by $d_{\text{cf}}^{(n)} = \{s, \{x_i^{(n)}[s] \mid i = 1 : m\}\}$, it should satisfy Eq. (9.2). Thus, we have

$$x_i^{(n)}[s] = A_{1,i} \cdot f(s) + \sum_{X_j \in X_{pa(i)} \setminus \{S\}} A_{j,i} \cdot f(x_j^{(n)}[s]) + u_i^{(n)},$$

which leads to

$$x_i^{(n)}[s] = A_{1,i} \cdot f(s) + \sum_{X_j \in X_{pa(i)} \setminus \{S\}} A_{j,i} \cdot f(x_j^{(n)}[s]) + x_i^{(n)} - A_{1,i} \cdot f(s^{(n)}) - \sum_{X_j \in X_{pa(i)} \setminus \{S\}} A_{j,i} \cdot f(x_j^{(n)}). \tag{9.3}$$

Finally, we compute the value of $x_i^{(n)}[s]$ according to Eq. (9.3) following the topological order and derive $d_{\text{cf}}^{(n)}$ from the observational data.

The challenge in the above derivation is how to estimate function $f(\cdot)$ and adjacency matrix A of the SCM. Next, we develop a causal structure discovery approach based on the graph autoencoder as proposed in [211].

Causal Structure Discovery

We estimate the adjacency matrix of the SCM defined in Eq. (9.1) by a graph autoencoder model with parameters $\{\theta_1, \phi_1, \hat{A}\}$. Specifically, an encoder is first adopted to derive

the hidden representation of a sample $d^{(n)}$, i.e., $h^{(n)} = E_{\theta_1}(d^{(n)})$, where $E_{\theta_1}(\cdot)$ is parameterized by a multilayer neural network. Then, the message passing operation is applied on the hidden representation, i.e., $h'^{(n)} = \hat{A}^T h^{(n)}$, where $\hat{A}$ is a parameter matrix. Finally, a decoder is used to reconstruct the original input from $h'^{(n)}$, i.e.,

$$\hat{d}^{(n)} = D_{\phi_1}(h'^{(n)}) = D_{\phi_1}(\hat{A}^T E_{\theta_1}(d^{(n)})),$$

where $D_{\phi_1}(\cdot)$ is parameterized by a different multilayer neural network. Note that both the encoder $E_{\theta_1}(\cdot)$ and the decoder $D_{\phi_1}(\cdot)$ work in a variable-wise manner in order to preserve the order of the message passing in the SCM. To train the graph autoencoder model, the objective function is defined as:

$$\mathcal{L}_{\text{GAE}}(A, \theta_1, \phi_1) = \frac{1}{2N} \sum_{n=1}^{N} \|d^{(n)} - \hat{d}^{(n)}\|_2^2 + \lambda \|\hat{A}\|_1 \text{ s.t. } tr(e^{\hat{A} \odot \hat{A}}) - m - 1 = 0,$$

where the constraint $tr(e^{\hat{A} \odot \hat{A}}) - m - 1 = 0$ is to ensure acyclicity in the graph. After training, matrix $\hat{A}$ will be a good estimation of the adjacency matrix A.

One challenge in applying the graph autoencoder to our work is that, although the graph autoencoder can accurately estimate the adjacency matrix $\hat{A}$, it does not produce a good reconstruction of the input sample, which implies that it does not accurately estimate the function $f(\cdot)$ in the SCM. In order to generate the counterfactual data, the reconstructed sample with high fidelity is critical. Hence, we improve the graph autoencoder by adding another decoder that focuses on data reconstruction, where the trained matrix $\hat{A}$ and the encoder $E_{\theta_1}(\cdot)$ are reused in this step.

In particular, we similarly feed each sample $d^{(n)}$ to trained encoder $E_{\theta_1}(\cdot)$ to obtain the corresponding hidden representation. Then, in order to be consistent with the structural equations Eq. (9.1), different from [211] where the message passing operation is applied in the representation space, we first use a new variable-wise decoder $D_{\phi_1'}$ to transform the hidden representation back to the original data space, and then aggregate the message from the neighbors based on matrix $\hat{A}$. As a result, the reconstruction process of each sample is

given by the following equation.

$$\hat{d}^{(n)} = \hat{A}^T D_{\phi_1'}(E_{\theta_1}(d^{(n)})).$$

The objective function is to reconstruct the input with $\hat{A}$ and θ_1 fixed:

$$\mathcal{L}_{\mathrm{D}}(\phi_1') = \frac{1}{2N}\sum_{n=1}^{N}\sum_{i=1}^{d}\|d_i^{(n)} - \hat{d}_i^{(n)}\|_2^2.$$

After training, we obtain the approximated mapping function $\hat{f} = D_{\phi_1'} \circ E_{\theta_1}$.

Generating Counterfactual Data

Given estimated adjacency matrix $\hat{A}$ and function $\hat{f}$, for each sample $d^{(n)}$, we generate its counterfactual $d_{\mathrm{cf}}^{(n)}$ following the Abduction-Action-Prediction process. We first intervene $s^{(n)}$ to its counterpart s and compute $\hat{f}(s)$. Then, we sort all variables in X in a topological order and compute $\hat{x}_i^{(n)}[s]$ iteratively according to Eq. (9.3) where A and f are replaced by their estimators $\hat{A}$ and $\hat{f}$. Finally, we obtain $\hat{D}_{\mathrm{cf}} = \{\hat{d}_{\mathrm{cf}}^{(n)}\}_{n=1}^{N}$, where $\hat{d}_{\mathrm{cf}}^{(n)} = \{s, \{\hat{x}_i^{(n)}[s] \mid i = 1:m\}\}$.

9.3.4 Phase Two: Fair Anomaly Detection

We use the autoencoder as the base model for anomaly detection, which is trained to minimize the reconstruction errors of normal samples. It is worth noting that a fully-connected autoencoder model is used here which is different from the variable-wise autoencoder used in the previous section for counterfactual data generation. Meanwhile, to achieve counterfactual fairness, we leverage the idea of adversarial training to make the hidden representations derived by the autoencoder not encode the information of the sensitive variable. To this end, we develop a pre-training and fine-tuning framework to ensure the effectiveness of anomaly detection as well as counterfactual fairness. The reason for adopting the pre-training and fine-tuning training approach instead of the end-to-end training is that some counterfactual samples in $\hat{\mathcal{D}}$ could be anomalies. If we include all samples in $\hat{\mathcal{D}}$ to train

the autoencoder model, the performance of anomaly detection can be damaged. Hence, we use samples in $\mathcal{D}$ to pre-train the autoencoder model. Then, during fine-tuning, we slightly update the autoencoder so that the effectiveness of anomaly detection and counterfactual fairness can be balanced. Finally, we do not use the sensitive variable and only use the non-sensitive variables X to train the anomaly detection model.

To be more specific, in the pre-training phase, given the training set with normal samples $\mathcal{D}$, an encoder first maps each sample $x^{(n)}$ to a hidden representation $z^{(n)} = E_{\theta_2}(x^{(n)})$, and then a decoder aims to reconstruct the original input from the hidden representation $\hat{x}^{(n)} = D_{\phi_2}(z^{(n)})$. The objective function is to minimize the reconstruction error of normal samples:

$$\mathcal{L}_{\mathrm{AE}}(\theta_2, \phi_2) = \frac{1}{2N} \sum_{n=1}^{N} \|d^{(n)} - D_{\phi_2} \circ E_{\theta_2}(x^{(n)})\|_2^2.$$

After pre-training the autoencoder model, in order to achieve counterfactual fairness, we further incorporate the adversarial training strategy to further fine-tune the autoencoder model so that the hidden representation $z^{(n)}$ derived by the encoder is free of the information of the sensitive variable. To this end, for each sample $d^{(n)} = \{s^{(n)}, x^{(n)}\}$ and its counterfactual sample $\hat{d}_{\mathrm{cf}}^{(n)} = \{s, \hat{x}_{\mathrm{cf}}^{(n)}\}$, we first derive the hidden representations, $z^{(n)}$ and $z_{\mathrm{cf}}^{(n)}$, respectively, by feeding them to the encoder E_{θ_2}. Then, a discriminator C_ψ is applied on $z^{(n)}$ and $z_{\mathrm{cf}}^{(n)}$ to predict whether the hidden representations are from observed or counterfactual samples, which is a binary classification task. We parameterize the discriminator C_ψ by a multilayer neural network with the sigmoid function as the output layer and use the negative of the standard cross-entropy loss for binary classification tasks as the objective function to train the discriminator:

$$\mathcal{L}_{\mathrm{C}}(\theta_2, \psi) = \frac{1}{N} \sum_{n=1}^{N} [\log(C_\psi(z^{(n)})) + \log(1 - C_\psi(z_{\mathrm{cf}}^{(n)}))].$$

The discriminator is trained to accurately separate the hidden representations of observed and counterfactual samples. Meanwhile, to make the hidden representation derived from the encoder invariant to the change of sensitive attribute, the adversarial game is to train

the encoder E_{θ_2} to fool the discriminator C_ψ but still be good for reconstructing the original input. As a result, the objective function can be defined as a minimax problem:

$$\min_{\theta_2,\phi_2} \max_{\psi} \mathcal{L}_{\text{AE}}(\theta_2, \phi_2) + \lambda \mathcal{L}_{\text{C}}(\theta_2, \psi), \tag{9.4}$$

where λ is a hyper-parameter to balance the reconstruction error and adversarial loss. Besides minimizing the reconstruction error $\mathcal{L}_{\text{AE}}$, the encoder also tries to maximize the cross-entropy loss for the discriminator $\mathcal{L}_{\text{C}}(\theta_2, \psi)$. Once the discriminator is unable to distinguish the hidden representations from factual or counterfactual data, we expect that both factual and counterfactual samples have similar reconstruction errors.

After training, the anomaly score for a new sample $d = \{s, x\}$ is computed based on the reconstruction error:

$$g(x) = \|x - D_{\phi_2} \circ E_{\theta_2}(x)\|_2^2.$$

If the anomaly score $g(x) > \tau$, where τ is a hyperparameter of the model, we label the sample as anomalous, i.e., $\hat{y} = 1$.

9.4 Experiments

9.4.1 Experimental Setup

Datasets. We conduct experiments on a synthetic dataset and two real-world datasets, Adult and COMPAS. Table 9.1 summarizes the statistics of three datasets.

Table 9.1: Statistics of datasets.

	Synthetic		Adult		COMPAS	
	Training	Test	Training	Test	Training	Test
Normal (Y=0)	12000	4000	12000	4000	2000	1283
Abnormal (Y=1)	N/A	400	N/A	800	N/A	384

Synthetic Dataset. We first build a synthetic dataset with 21 variables where we can obtain the ground truth of counterfactuals. We first randomly generate the adjacency

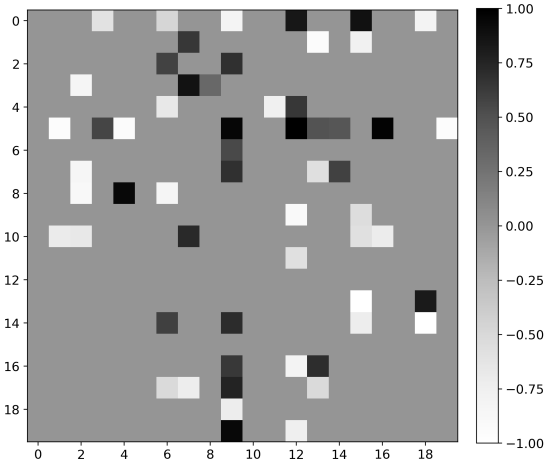


Fig. 9.2: Adjacency matrix A

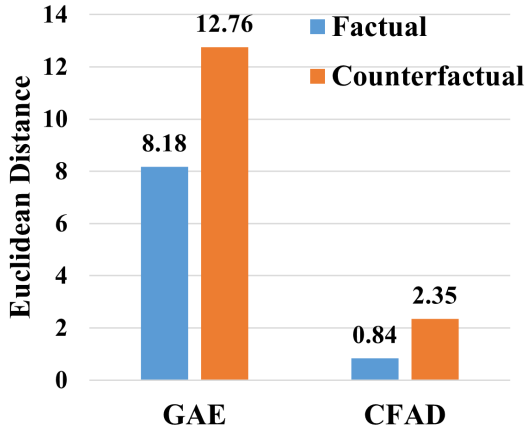
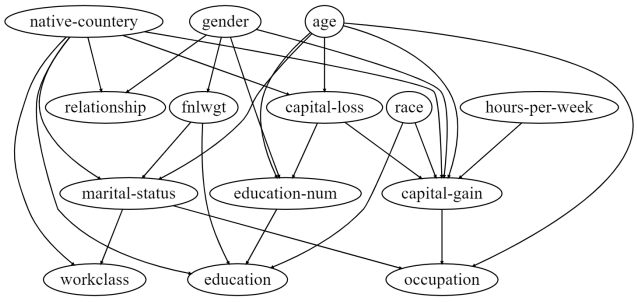
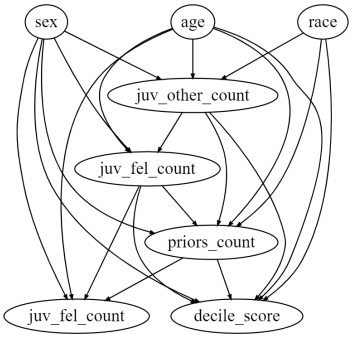


Fig. 9.3: Results on data generation.



(a) Adult



(b) COMPAS

Fig. 9.4: Learned causal graphs.

matrix A of a causal graph using the Erdős-Rényi model [236] where one node is defined as a root node for representing the sensitive variable S. Figure 9.2 shows the generated adjacency matrix A. The value of S is randomly generated with binarized value {-1, 1} to indicate sensitive and non-sensitive groups. Then, similar to [211], the rest 20 variables are generated based on the following data generating procedure: $X = 3A^T \cos(X + 1) + U$, where U is a standard Gaussian noise. Finally, one leaf node is selected as the decision attribute Y for determining anomalies. Specifically, for each sample, if the value of Y is greater than 0.85 quantile or smaller than 0.01 quantile, we label this sample as an anomaly, i.e., $Y = 1$. If the value of Y is between 0.3 and 0.7 quantiles, we label the sample as normal, i.e., $Y = 0$. Meanwhile, for both training and test sets, for 50% of the samples, their corresponding counterfactuals have labels that are different from the factual ones.

Adult Dataset. Adult is a real-world dataset with 14 features [217]. We treat "*gender*" as the sensitive attribute and samples with "*income* $>$ *50k*" as anomalies. We normalize all continuous features and binarize all categorical features. Figure 9.4a shows the causal graph on Adult learned in Phase One of our approach. Meanwhile, as we do not know the ground truth of counterfactuals, we use the generated counterfactual samples for measuring counterfactual fairness.

COMPAS Dataset. COMPAS is another real-world dataset [237], which consists of 8 features. We consider "race" as the sensitive attribute, where "African-American" and "Caucasian" are the disadvantage and advantage groups, respectively, and treat "recidivists" as anomalies. Similar to Adult, we normalize all continuous features and binarize all categorical features. Figure 9.4b shows the learned causal graph.

Baselines. We compare CFAD with the following baselines: 1) Principal Component Analysis (**PCA**), which is a dimensional reduction based anomaly detection approach; 2) One-class SVM (**OCSVM**), which is a one-class classification model that can detect outliers based on the observed normal samples; 3) Isolation Forest (**iForest**), which is a widely used tree-based anomaly detection model; 4) Autoencoder (**AE**), which is trained on normal data and widely-used for anomaly detection based on the deep autoencoder structure; 5) Deep Clustering based Fair Outlier Detection (**DCFOD**) [224], which adopts the adversarial training to achieve the group fairness in anomaly detection; 6) Fairness-aware Outlier Detection (**FairOD**) [226], which is also an autoencoder-based anomaly detection approach with fairness regularizers.

Evaluation Metrics. We evaluate the performance of anomaly detection based on Area Under Precision-Recall Curve **(AUC-PR)**, Area Under Receiver Operating Characteristic Curve **(AUC-ROC)**, and **Macro-F1**. We evaluate counterfactual fairness by computing the **changing ratio** of the samples whose detection outcomes are different from those for their corresponding counterfactuals, i.e., $changing\_ratio = \frac{\sum_{n=1}^{N} \mathbb{1}[\hat{y}^{(n)} \neq \hat{y}_{\text{cf}}^{(n)}]}{N}$, where $\mathbb{1}[\cdot]$ is the indicator function.

Implementation Details. Regarding baselines, we use Loglizer [70] to evaluate PCA, OC-SVM, and iForest. We implement FairOD and DCFOD based on public source code [224]. By default, the threshold τ for anomaly detection is set based on the 0.95 quantile of reconstruction errors (AE, FairOD, and CFAD) or distance to the normal center (DCFOD) in the training set. Our code on CFAD is available online[1].

9.4.2 Experimental Results

Counterfactual Data Generation. We first evaluate the performance of counterfactual data generation in the synthetic dataset by comparing CFAD with GAE [211] in terms of Euclidean distance between the generated and ground-truth samples. As shown in Figure 9.3, on the factual data, CFAD achieves a much lower reconstruction error compared with GAE. More importantly, for counterfactual data generation, CFAD is much better compared with GAE. It indicates that by incorporating a variable-wise decoder $D_{\phi_1'}$ for data generation, CFAD can generate counterfactual samples with high fidelity.

Table 9.2: Anomaly detection on synthetic and real datasets with threshold $\tau = 0.95$. For AUC-PR, AUC-ROC, and Macro-F1, the higher the value the better the effectiveness; for Changing Ratio, the lower the value the better the fairness.

Method	Synthetic Dataset				Adult Dataset				COMPAS Dataset			
	AUC-PR	AUC-ROC	Macro-F1	Changing Ratio	AUC-PR	AUC-ROC	Macro-F1	Changing Ratio	AUC-PR	AUC-ROC	Macro-F1	Changing Ratio
PCA	0.992	0.999	0.908	0.478	0.238	0.582	0.476	0.261	0.365	0.642	0.595	0.268
OC-SVM	0.776	0.953	0.477	0.399	0.282	0.638	0.482	0.285	0.337	0.593	0.488	0.376
iForest	0.190	0.693	0.570	0.271	0.312	0.658	0.570	0.279	0.311	0.567	0.564	0.415
AE	0.957	0.996	0.883	0.461	0.349	0.640	0.608	0.590	0.344	0.616	0.581	0.407
DCFOD	0.383	0.832	0.721	0.212	0.249	0.623	0.533	0.071	0.260	0.569	0.466	0.067
FairOD	0.580	0.873	0.689	0.261	0.222	0.621	0.531	0.131	0.265	0.548	0.493	0.068
CFAD	0.947	0.996	0.930	0.199	0.319	0.589	0.576	0.057	0.314	0.596	0.539	0.049

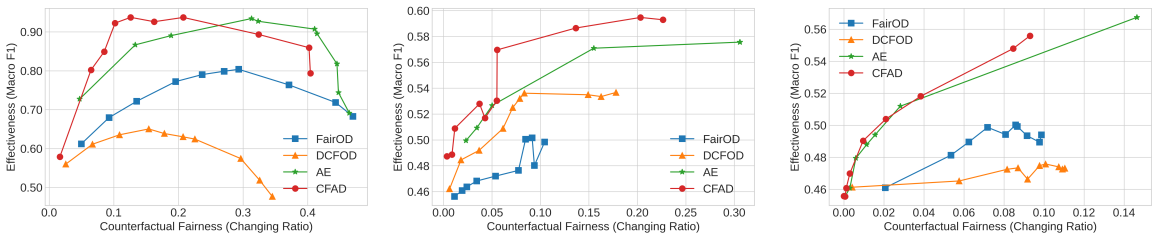
Anomaly Detection. We further evaluate the performance of anomaly detection in terms of effectiveness as well as fairness. Table 9.2 shows the evaluation results. We report the mean value after five runs.

Synthetic Dataset. CFAD can well balance the effectiveness and fairness in anomaly detection with high AUC-PR, AUC-ROC, and Macro-F1 and a low changing ratio. AE can achieve good performance on anomaly detection, but its changing ratio is high. DCFOD

[1] https://github.com/hanxiao0607/CFAD

and FairOD, which achieve group fairness in anomaly detection, both have relatively low changing ratios, but their effectiveness in anomaly detection is not satisfactory.

Real Datasets. We have similar observations on the Adult and COMPAS datasets. CFAD achieves good performance in both effectiveness and fairness. For baselines that have no fairness component, their performance is good in terms of the effectiveness in anomaly detection, but they all have high changing ratios. Similarly, although DCFOD and FairOD have relatively low changing ratios, their effectiveness is much worse than other approaches.



(a) Synthetic (b) Adult (c) COMPAS

Fig. 9.5: Trade-off between effectiveness and fairness.

Trade-off Between Effectiveness and Fairness. We further investigate the trade-off between effectiveness and fairness by varying the threshold as different quantiles of reconstruction errors or distances in the training set. We plot the effectiveness and fairness of each threshold setting of four approaches CFAD, AE, DCFOD, and FairOD in Figure 9.5, where the x-axis is the changing ratio (counterfactual fairness), the y-axis indicates the Macro-F1 score (effectiveness), and each dot in the line indicates the result from one threshold. The dots from right to left indicate the performance based on quantiles including $\{0.8, 0.85, 0.9, 0.95, 0.97, 0.98, 0.99, 0.995, 0.999\}$. Ideally, we expect an anomaly detection model can achieve a high Marco-F1 score with a low changing ratio, which is the top left corner of the figure.

As shown in Figure 9.5, CFAD performs best when the effectiveness trades off with fairness, as CFAD is closest to the top left corner of the figure. Specifically, on the Synthetic dataset, CFAD achieves much higher Macro-F1 values (effectiveness) with similar changing rates (fairness) compared with DCFOD and FairOD. Meanwhile, for most of the thresholds

chosen based on quantiles, CFAD has higher Macro-F1 and lower changing ratios compared with AE. On the Adult and COMPAS datasets, CFAD can have higher Macro-F1 values and lower changing ratios compared with DCFOD and FairOD.

9.5 Summary

In this work, we have developed a counterfactually fair anomaly detection (CFAD) framework, which is able to effectively detect anomalies and also ensure counterfactual fairness. The core idea of CFAD is to generate counterfactual data governed by a learned causal structure based on the proposed graph autoencoder model. Then, by using a vanilla autoencoder as the anomaly detection model, an adversarial training strategy is adopted to ensure the representations derived by the autoencoder without the information of sensitive attributes. After that, counterfactual fairness is achieved by having similar reconstruction errors for both factual and counterfactual samples. The experimental results show that CFAD can achieve counterfactually fair anomaly detection while well-balancing the trade-off between effectiveness and fairness. The early version of this work is published at PAKDD 2023 [210].

CHAPTER 10

CONCLUSIONS AND FUTURE WORK

In this chapter, we summarize our works. We further propose several potential research directions associated with our previous chapters on anomaly detection.

10.1 Conclusions

This dissertation addresses the multifaceted challenges of responsible anomaly detection, focusing on enhancing performance in diverse conditions, improving explainability, and ensuring fairness. In specific, we tackle key issues in the area of anomaly detection, focusing on:

- Improving Detection Performance: How to enhance accuracy in detecting anomalies across different scenarios, especially with limited data and in changing environments. This includes developing advanced models that can adapt and maintain high performance over time.

- Making Models Understandable: How to make anomaly detection models more transparent, allowing users to see why anomalies are flagged. This involves creating techniques that explain the model's decisions in a way that users can easily understand.

- Fairness in Detection: How to ensure that our anomaly detection methods are fair and do not discriminate against any group. We look into ways to design systems that treat all users equitably and avoid bias.

In Chapter 2, we delve into enhancing log anomaly detection with the integration of Generative Pre-trained Transformers (GPT), proposing the novel LogGPT framework. Traditional methods, while effective, often struggle with the dynamic and complex nature of log data, limiting their predictive accuracy. LogGPT, through its reinforcement learning and novel reward mechanism, significantly improves anomaly detection performance. By

pre-training on normal log sequences and fine-tuning with a focus on anomaly detection, LogGPT shows superior results over state-of-the-art methods across multiple datasets, confirming its robustness and adaptability in identifying log anomalies.

In Chapter 3, we explore the implementation of LogTAD, a novel framework for unsupervised cross-system log anomaly detection via domain adaptation. This approach leverages the adversarial domain adaptation technique to align log data distributions across different systems, facilitating effective anomaly detection in newly deployed or less familiar systems with minimal data requirements. By utilizing a small subset of logs from the target system, LogTAD demonstrates high accuracy in cross-system anomaly detection, significantly mitigating the challenge of limited anomalous samples and showcasing its adaptability and efficiency in diverse system environments.

In Chapter 4, we developed FADS, a framework tailored for few-shot anomaly detection, leveraging reinforced data selection for enhanced performance. Simultaneously, we introduced FADScr, an extension of FADS, incorporating a combinatorial reward mechanism. This innovative approach selectively augments the training set with high-quality, weakly-labeled samples from a large, unlabeled dataset, substantially improving anomaly detection and classification accuracy. Through rigorous experimentation, FADS and FADScr demonstrated significant advancements in detecting and classifying anomalies with only a limited number of labeled samples, outperforming existing methodologies.

In Chapter 5, we introduce InterpretableSAD, an innovative framework focused on the interpretation of anomaly detection within sequential log data. This framework uniquely combines the power of negative sampling and Integrated Gradients (IG) for both anomaly sequence identification and the pinpointing of fine-grained anomalous events. By generating anomalous sequences through negative sampling, we effectively enrich the training dataset, enabling the model to learn the distinguishing features of anomalies. Furthermore, leveraging IG provides clear insights into which specific events within a sequence contribute to its classification as anomalous, enhancing the interpretability of the detection process. This dual approach not only improves anomaly detection accuracy but also aids system administrators

in locating system vulnerabilities or issues, thereby significantly contributing to system reliability and security.

In Chapter 6, we focus on RecAD, a novel approach for providing algorithmic recourse in the context of anomaly detection within multivariate time series data. RecAD introduces a mechanism to recommend corrective actions for identified anomalies, aiming to reverse the detected abnormal behavior to normal with minimal intervention costs. By considering the causal relationships and downstream impacts of interventions, RecAD enhances the interpretability and effectiveness of anomaly mitigation strategies. This approach is validated through extensive experiments, demonstrating its capability to recommend actionable insights for anomaly correction in complex time series data.

In Chapter 7, we present AERCA, an innovative autoencoder-based framework for root cause analysis in multivariate time series. AERCA leverages Granger causality to capture and model the causal dependencies and exogenous variables' distributions in normal states, facilitating the identification of anomalies' root causes. Through a comprehensive encoder-decoder architecture, it effectively highlights significant deviations in exogenous variables, offering a novel approach to understanding and diagnosing anomalies. Tested across various datasets, AERCA proves highly effective in accurately determining causal relationships and pinpointing anomalies' origins.

In Chapter 8, we introduce RootCLAM, a framework designed to address root cause localization and anomaly mitigation using causal inference techniques. RootCLAM differentiates itself by focusing on the identification of root causes for anomalies through causal analysis and then proposing mitigation actions to correct these anomalies. This approach not only identifies the symptoms of anomalies but also addresses their underlying causes, offering a more comprehensive solution for anomaly detection and resolution. RootCLAM's effectiveness is demonstrated through its application on several datasets, showcasing its ability to accurately identify root causes and suggest effective mitigation strategies.

In Chapter 9, we introduce CFAD, a framework designed to ensure counterfactual fairness in anomaly detection. By generating counterfactual data and applying fair anomaly

detection methods, CFAD aims to maintain detection outcomes consistent across factual and counterfactual scenarios, irrespective of sensitive attributes. This novel approach addresses the challenge of counterfactual fairness in anomaly detection, demonstrating its effectiveness and counterfactual fairness across various datasets through experimentation. CFAD represents a significant step forward in creating equitable and accurate anomaly detection models.

10.2 Future Work

Privacy Preservation. As anomaly detection often involves sensitive data, preserving privacy while maintaining high detection performance is paramount. Future research could explore privacy-preserving techniques such as differential privacy, secure multi-party computation, or homomorphic encryption within anomaly detection frameworks. Developing models that can operate effectively on encrypted or anonymized data, ensuring that individual privacy is safeguarded while still providing accurate anomaly detection, will be crucial for the widespread adoption of these technologies in sensitive domains.

Robustness to Adversarial Attacks. Another future research should focus on enhancing the robustness of anomaly detection models against adversarial attacks, a critical challenge in maintaining system integrity and reliability. Developing sophisticated defense mechanisms that can identify and neutralize adversarial attempts without compromising detection accuracy is essential. Exploring adversarial training, anomaly detection in adversarial settings, and leveraging insights from cybersecurity can provide robust frameworks capable of withstanding sophisticated threats.

Fairness without Losing Accuracy. Last but not least, ensuring fairness in anomaly detection models without sacrificing accuracy. This involves creating algorithms that can detect anomalies across diverse datasets without bias towards any group or individual. Future work could include the development of fairness-aware learning algorithms, exploration of bias mitigation techniques in the data preprocessing and model training phases, and the establishment of benchmarks and metrics for evaluating both fairness and accuracy concurrently.

REFERENCES

[1] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[2] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in neural information processing systems*, 2000, pp. 1008–1014.

[3] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *ArXiv*, vol. abs/1707.06347, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:28695052

[4] R. Ramamurthy, P. Ammanabrolu, K. Brantley, J. Hessel, R. Sifa, C. Bauckhage, H. Hajishirzi, and Y. Choi, "Is reinforcement learning (not) for natural language processing?: Benchmarks, baselines, and building blocks for natural language policy optimization," *arXiv preprint arXiv:2210.01241*, 2022.

[5] P. J. Rousseeuw and M. Hubert, "Anomaly detection by robust statistics," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 2, p. e1236, 2018.

[6] J. Frontera-Pons, M. A. Veganzones, F. Pascal, and J.-P. Ovarlez, "Hyperspectral anomaly detectors using robust estimators," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 9, no. 2, pp. 720–731, 2015.

[7] I. Khemakhem, D. Kingma, R. Monti, and A. Hyvarinen, "Variational autoencoders and nonlinear ica: A unifying framework," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2207–2217.

[8] S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland, "Data mining for credit card fraud: A comparative study," *Decision support systems*, vol. 50, no. 3, pp. 602–613, 2011.

[9] A. Abdallah, M. A. Maarof, and A. Zainal, "Fraud detection system: A survey," *Journal of Network and Computer Applications*, vol. 68, pp. 90–113, 2016.

[10] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.

[11] M. Toledano, I. Cohen, Y. Ben-Simhon, and I. Tadeski, "Real-time anomaly detection system for time series at scale," in *KDD 2017 Workshop on Anomaly Detection in Finance*. PMLR, 2018, pp. 56–65.

[12] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[13] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Computing*, vol. 22, pp. 949–961, 2019.

[14] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, M. Gao, H. Hou, and C. Wang, "Machine learning and deep learning methods for cybersecurity," *Ieee access*, vol. 6, pp. 35 365–35 381, 2018.

[15] G. Androulidakis, V. Chatzigiannakis, and S. Papavassiliou, "Network anomaly detection and classification via opportunistic sampling," *IEEE network*, vol. 23, no. 1, pp. 6–12, 2009.

[16] O. Salem, A. Guerassimov, A. Mehaoua, A. Marcus, and B. Furht, "Sensor fault and patient anomaly detection and classification in medical wireless sensor networks," in *2013 IEEE international conference on communications (ICC)*. IEEE, 2013, pp. 4373–4378.

[17] L. Martí, N. Sanchez-Pi, J. M. Molina, and A. C. B. Garcia, "Anomaly detection based on sensor data in petroleum industry applications," *Sensors*, vol. 15, no. 2, pp. 2774–2797, 2015.

[18] J. Marzat, H. Piet-Lahanier, F. Damongeot, and E. Walter, "Model-based fault diagnosis for aerospace systems: a survey," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of aerospace engineering*, vol. 226, no. 10, pp. 1329–1360, 2012.

[19] F. Lopez, M. Saez, Y. Shao, E. C. Balta, J. Moyne, Z. M. Mao, K. Barton, and D. Tilbury, "Categorization of anomalies in smart manufacturing systems to support the selection of detection mechanisms," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 1885–1892, 2017.

[20] C. Kim, J. Lee, R. Kim, Y. Park, and J. Kang, "Deepnap: Deep neural anomaly pre-detection in a semiconductor fab," *Information Sciences*, vol. 457, pp. 1–11, 2018.

[21] J. Goh, S. Adepu, K. N. Junejo, and A. Mathur, "A dataset to support research in the design of secure water treatment systems," in *International conference on critical information infrastructures security*. Springer, 2017, pp. 88–99.

[22] P. Seeböck, J. I. Orlando, T. Schlegl, S. M. Waldstein, H. Bogunović, S. Klimscha, G. Langs, and U. Schmidt-Erfurth, "Exploiting epistemic uncertainty of anatomy segmentation for anomaly detection in retinal oct," *IEEE transactions on medical imaging*, vol. 39, no. 1, pp. 87–98, 2019.

[23] N. Shvetsova, B. Bakker, I. Fedulova, H. Schulz, and D. V. Dylov, "Anomaly detection in medical imaging with deep perceptual autoencoders," *IEEE Access*, vol. 9, pp. 118 571–118 583, 2021.

[24] T. Schlegl, P. Seeböck, S. M. Waldstein, G. Langs, and U. Schmidt-Erfurth, "f-anogan: Fast unsupervised anomaly detection with generative adversarial networks," *Medical image analysis*, vol. 54, pp. 30–44, 2019.

[25] L. Cheng, K. R. Varshney, and H. Liu, "Socially responsible ai algorithms: Issues, purposes, and challenges," *Journal of Artificial Intelligence Research*, vol. 71, pp. 1137–1181, 2021.

[26] H. Liu, Y. Wang, W. Fan, X. Liu, Y. Li, S. Jain, Y. Liu, A. Jain, and J. Tang, "Trustworthy ai: A computational perspective," *ACM Transactions on Intelligent Systems and Technology*, vol. 14, no. 1, pp. 1–59, 2022.

[27] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," in *TKDD*. ACM, 2012.

[28] M. Braei and S. Wagner, "Anomaly detection in univariate time-series: A survey on the state-of-the-art," *arXiv preprint arXiv:2004.00433*, 2020.

[29] P. Malhotra, L. Vig, G. Shroff, P. Agarwal *et al.*, "Long short term memory networks for anomaly detection in time series." in *ESANN*, vol. 2015, 2015, p. 89.

[30] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2828–2837.

[31] M. A. Bashar and R. Nayak, "Tanogan: Time series anomaly detection with generative adversarial networks," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2020, pp. 1778–1785.

[32] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020.

[33] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM computing surveys (CSUR)*, vol. 54, no. 2, pp. 1–38, 2021.

[34] V. Jyothsna, R. Prasad, and K. M. Prasad, "A review of anomaly based intrusion detection systems," *International Journal of Computer Applications*, vol. 28, no. 7, pp. 26–35, 2011.

[35] J. Wang, Y. Chen, W. Feng, H. Yu, M. Huang, and Q. Yang, "Transfer learning with dynamic distribution adaptation," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 11, no. 1, pp. 1–25, 2020.

[36] S. Niu, Y. Hu, J. Wang, Y. Liu, and H. Song, "Feature-based distant domain transfer learning," in *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2020, pp. 5164–5171.

[37] H. Ott, J. Bogatinovski, A. Acker, S. Nedelkoski, and O. Kao, "Robust and transferable anomaly detection in log data using pre-trained language models," in *2021 IEEE/ACM international workshop on cloud intelligence (CloudIntelligence)*. IEEE, 2021, pp. 19–24.

[38] J. Soldani and A. Brogi, "Anomaly detection and failure root cause analysis in (micro) service-based cloud applications: A survey," *ACM Computing Surveys (CSUR)*, vol. 55, no. 3, pp. 1–39, 2022.

[39] Z. Li, Y. Zhu, and M. van Leeuwen, "A survey on explainable anomaly detection," *arXiv preprint arXiv:2210.06959*, 2022.

[40] G. Pang and C. Aggarwal, "Toward explainable deep anomaly detection," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 4056–4057.

[41] Y. Ming, P. Xu, H. Qu, and L. Ren, "Interpretable and steerable sequence learning via prototypes," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 903–913.

[42] S. Venkataramanan, K.-C. Peng, R. V. Singh, and A. Mahalanobis, "Attention guided anomaly localization in images," in *European Conference on Computer Vision*. Springer, 2020, pp. 485–503.

[43] E. Delaney, D. Greene, and M. T. Keane, "Instance-based counterfactual explanations for time series classification," in *ICCBR*, 2021, pp. 32–47.

[44] D. Sulem, M. Donini, M. B. Zafar, F.-X. Aubet, J. Gasthaus, T. Januschowski, S. Das, K. Kenthapadi, and C. Archambeau, "Diverse counterfactual explanations for anomaly detection in time series," *arXiv preprint arXiv:2203.11103*, 2022.

[45] E. Ates, B. Aksar, V. J. Leung, and A. K. Coskun, "Counterfactual explanations for multivariate time series," in *ICAPAI*, 2021, pp. 1–8.

[46] M. Salehi, H. Mirzaei, D. Hendrycks, Y. Li, M. H. Rohban, and M. Sabokrou, "A unified survey on anomaly, novelty, open-set, and out-of-distribution detection: Solutions and future challenges," *arXiv preprint arXiv:2110.14051*, 2021.

[47] S. Yuan and X. Wu, "Trustworthy anomaly detection: a survey," *arXiv preprint arXiv:2202.07787*, 2022.

[48] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017.

[49] H. Guo, S. Yuan, and X. Wu, "Logbert: Log anomaly detection via bert," in *2021 international joint conference on neural networks (IJCNN)*. IEEE, 2021, pp. 1–8.

[50] V.-H. Le and H. 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.

[51] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019.

[52] M. Landauer, S. Onder, F. Skopik, and M. Wurzenberger, "Deep learning for anomaly detection in log data: A survey," *Machine Learning with Applications*, vol. 12, p. 100470, 2023.

[53] W. Xu, L. Huang, A. Fox, D. Patterson, and M. 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.

[54] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 eighth ieee international conference on data mining*. IEEE, 2008, pp. 413–422.

[55] Y. Wang, J. Wong, and A. Miner, "Anomaly intrusion detection using one class svm," in *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004.* IEEE, 2004, pp. 358–364.

[56] W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, S. Tao, P. Sun *et al.*, "Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs." in *IJCAI*, vol. 7, 2019, pp. 4739–4745.

[57] Z. Wang, Z. Chen, J. Ni, H. Liu, H. Chen, and J. Tang, "Multi-scale one-class recurrent neural networks for discrete event sequence anomaly detection," in *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 2021, pp. 3726–3734.

[58] W. J. Murdoch, P. J. Liu, and B. Yu, "Beyond word importance: Contextual decomposition to extract interactions from lstms," *arXiv preprint arXiv:1801.05453*, 2018.

[59] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[60] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[61] S. Zhang, Y. Liu, X. Zhang, W. Cheng, H. Chen, and H. Xiong, "Cat: Beyond efficient transformer for content-aware anomaly detection in event sequences," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 4541–4550.

[62] P. He, J. Zhu, Z. Zheng, and M. 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.

[63] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[64] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[65] A. Oliner and J. 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.

[66] W. Xu, L. Huang, A. Fox, D. Patterson, and M. Jordan, "Largescale system problem detection by mining console logs," *Proceedings of SOSP'09*, 2009.

[67] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[68] K.-L. Li, H.-K. Huang, S.-F. Tian, and W. Xu, "Improving one-class svm for anomaly detection," in *Proceedings of the 2003 international conference on machine learning and cybernetics (IEEE Cat. No. 03EX693)*, vol. 5. IEEE, 2003, pp. 3077–3081.

[69] Q. Lin, H. Zhang, J.-G. Lou, Y. Zhang, and X. Chen, "Log clustering based problem identification for online service systems," in *Proceedings of the 38th International Conference on Software Engineering Companion*, 2016, pp. 102–111.

[70] S. He, J. Zhu, P. He, and M. R. Lyu, "Experience report: System log analysis for anomaly detection," in *2016 IEEE 27th international symposium on software reliability engineering (ISSRE)*. IEEE, 2016, pp. 207–218.

[71] Z. Chen, J. Liu, W. Gu, Y. Su, and M. R. Lyu, "Experience report: Deep learning-based system log analysis for anomaly detection," *arXiv preprint arXiv:2107.05908*, 2021.

[72] X. Han, S. Yuan, and M. Trabelsi, "Loggpt: Log anomaly detection via gpt," in *2023 IEEE International Conference on Big Data (BigData)*. IEEE, 2023, pp. 1117–1122.

[73] T.-F. Yen, A. Oprea, K. Onarlioglu, T. Leetham, W. Robertson, A. Juels, and E. Kirda, "Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks," in *Proceedings of the 29th annual computer security applications conference*, 2013, pp. 199–208.

[74] R. Vaarandi and M. Pihelgas, "Logcluster-a data clustering and pattern mining algorithm for event logs," in *2015 11th International conference on network and service management (CNSM)*. IEEE, 2015, pp. 1–7.

[75] Y. Liang, Y. Zhang, H. Xiong, and R. Sahoo, "Failure prediction in ibm bluegene/l event logs," in *Seventh IEEE International Conference on Data Mining (ICDM 2007)*. IEEE, 2007, pp. 583–588.

[76] P. He, J. Zhu, S. He, J. Li, and M. R. Lyu, "Towards automated log parsing for large-scale log data analysis," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 6, pp. 931–944, 2017.

[77] S. Lu, X. Wei, Y. Li, and L. 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/CyberSciTech). IEEE, 2018, pp. 151–158.

[78] P. He, J. Zhu, S. He, J. Li, and M. R. Lyu, "An evaluation study on log parsing and its use in log mining," in *2016 46th annual IEEE/IFIP international conference on dependable systems and networks (DSN)*. IEEE, 2016, pp. 654–661.

[79] R. Chen, S. Zhang, D. Li, Y. Zhang, F. Guo, W. Meng, D. Pei, Y. Zhang, X. Chen, and Y. Liu, "Logtransfer: Cross-system log anomaly detection for software systems with transfer learning," in *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, 2020.

[80] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *International conference on machine learning*. PMLR, 2018, pp. 4393–4402.

[81] X. Han and S. Yuan, "Unsupervised cross-system log anomaly detection via domain adaptation," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 3068–3072.

[82] G. Pang, C. Shen, L. Cao, and A. v. d. Hengel, "Deep learning for anomaly detection: A review," *arXiv preprint arXiv:2007.02500*, 2020.

[83] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[84] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM computing surveys (csur)*, vol. 53, no. 3, pp. 1–34, 2020.

[85] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," *arXiv preprint arXiv:1703.05175*, 2017.

[86] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, "Matching networks for one shot learning," *Advances in neural information processing systems*, vol. 29, pp. 3630–3638, 2016.

[87] A. Ayyad, Y. Li, R. Muaz, S. Albarqouni, and M. Elhoseiny, "Semi-supervised few-shot learning with prototypical random walks," in *AAAI Workshop on Meta-Learning and MetaDL Challenge*. PMLR, 2021, pp. 45–57.

[88] K. Ding, Q. Zhou, H. Tong, and H. Liu, "Few-shot network anomaly detection via cross-network meta-learning," in *Proceedings of the Web Conference 2021*, 2021, pp. 2448–2456.

[89] S. Yuan, P. Zheng, X. Wu, and H. Tong, "Few-shot insider threat detection," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 2289–2292.

[90] G. Pang, C. Ding, C. Shen, and A. v. d. Hengel, "Explainable deep few-shot anomaly detection with deviation networks," *arXiv preprint arXiv:2108.00462*, 2021.

[91] Y. Lu, F. Yu, M. K. K. Reddy, and Y. Wang, "Few-shot scene-adaptive anomaly detection," in *European Conference on Computer Vision*. Springer, 2020, pp. 125–141.

[92] X. Zhang, Y. Xu, Q. Lin, B. Qiao, H. Zhang, Y. Dang, C. Xie, X. Yang, Q. Cheng, Z. 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.

[93] M.-h. Oh and G. Iyengar, "Sequential anomaly detection using inverse reinforcement learning," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & data mining*, 2019, pp. 1480–1490.

[94] L. Ruff, R. A. Vandermeulen, N. Görnitz, A. Binder, E. Müller, K.-R. Müller, and M. Kloft, "Deep semi-supervised anomaly detection," in *International Conference on Learning Representations*, 2019.

[95] D. Hendrycks, M. Mazeika, and T. Dietterich, "Deep anomaly detection with outlier exposure," in *International Conference on Learning Representations*, 2018.

[96] D. Zha, K.-H. Lai, M. Wan, and X. Hu, "Meta-aad: Active anomaly detection with deep reinforcement learning," in *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2020, pp. 771–780.

[97] M. E. Villa-Pérez, M. Á. Álvarez-Carmona, O. Loyola-González, M. A. Medina-Pérez, J. C. Velazco-Rossell, and K.-K. R. Choo, "Semi-supervised anomaly detection algorithms: A comparative summary and future research directions," *Knowledge-Based Systems*, vol. 218, p. 106878, 2021.

[98] S. Akcay, A. Atapour-Abarghouei, and T. P. Breckon, "Ganomaly: Semi-supervised anomaly detection via adversarial training," in *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part III 14*. Springer, 2019, pp. 622–637.

[99] C. Zhong, M. C. Gursoy, and S. Velipasalar, "Deep actor-critic reinforcement learning for anomaly detection," in *2019 IEEE global communications conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[100] M. N. Kurt, O. Ogundijo, C. Li, and X. Wang, "Online cyber-attack detection in smart grid: A reinforcement learning approach," *IEEE Transactions on Smart Grid*, vol. 10, no. 5, pp. 5174–5185, 2018.

[101] C.-I. Chang and S.-S. Chiang, "Anomaly detection and classification for hyperspectral imagery," *IEEE transactions on geoscience and remote sensing*, vol. 40, no. 6, pp. 1314–1325, 2002.

[102] Z. Yu, L. Chen, Z. Cheng, and J. Luo, "Transmatch: A transfer-learning scheme for semi-supervised few-shot learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 12 856–12 864.

[103] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *International conference on machine learning*. PMLR, 2016, pp. 478–487.

[104] T. Lucas, P. Weinzaepfel, and G. Rogez, "Barely-supervised learning: Semi-supervised learning with very few labeled images," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 2, 2022, pp. 1881–1889.

[105] Y. Koizumi, S. Murata, N. Harada, S. Saito, and H. Uematsu, "Sniper: Few-shot learning for anomaly detection to minimize false-negative rate with ensured true-positive rate," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 915–919.

[106] X. Zhou, W. Liang, S. Shimizu, J. Ma, and Q. Jin, "Siamese neural network based few-shot learning for anomaly detection in industrial cyber-physical systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5790–5798, 2020.

[107] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.

[108] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 military communications and information systems conference (MilCIS)*. IEEE, 2015, pp. 1–6.

[109] A. collaborative project between the Communications Security Establishment (CSE) & the Canadian Institute for Cybersecurity (CIC), "A realistic cyber defense dataset (cse-cic-ids2018)."

[110] J. Glasser and B. Lindauer, "Bridging the gap: A pragmatic approach to generating insider threat data," in *2013 IEEE Security and Privacy Workshops*. IEEE, 2013, pp. 98–104.

[111] B. Schölkopf, R. C. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt, "Support vector method for novelty detection," *Advances in neural information processing systems*, vol. 12, 1999.

[112] C. Elkan and K. Noto, "Learning classifiers from only positive and unlabeled data," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008, pp. 213–220.

[113] M. Kato, T. Teshima, and J. Honda, "Learning from positive and unlabeled data with a selection bias," in *International conference on learning representations*, 2018.

[114] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, V. Vapnik *et al.*, "Support vector regression machines," *Advances in neural information processing systems*, vol. 9, pp. 155–161, 1997.

[115] D. E. Rumelhart, G. E. Hinton, R. J. Williams *et al.*, "Learning internal representations by error propagation," 1985.

[116] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[117] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[118] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[119] X. Han, D. Xu, S. Yuan, and X. Wu, "Few-shot anomaly detection and classification through reinforced data selection," in *2022 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2022, pp. 963–968.

[120] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 3319–3328.

[121] K. Zhang, J. Xu, M. R. Min, G. Jiang, K. Pelechrinis, and H. Zhang, "Automated it system failure prediction: A deep learning approach," in *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 2016, pp. 1291–1300.

[122] F. Liu, Y. Wen, D. Zhang, X. Jiang, X. Xing, and D. Meng, "Log2vec: a heterogeneous graph embedding based approach for detecting cyber threats within enterprise," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1777–1794.

[123] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[124] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[125] J. Wei and K. Zou, "Eda: Easy data augmentation techniques for boosting performance on text classification tasks," *arXiv preprint arXiv:1901.11196*, 2019.

[126] S. Kobayashi, "Contextual augmentation: Data augmentation by words with paradigmatic relations," *arXiv preprint arXiv:1805.06201*, 2018.

[127] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[128] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Neural Information Processing Systems (NIPS)*, 2013, pp. 1–9.

[129] J. Ding, Y. Quan, X. He, Y. Li, and D. Jin, "Reinforced negative sampling for recommendation with exposure data." in *IJCAI*, 2019, pp. 2230–2236.

[130] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[131] C. Molnar, *Interpretable machine learning*. Lulu. com, 2020.

[132] M. Du, N. Liu, and X. Hu, "Techniques for interpretable machine learning," *Communications of the ACM*, vol. 63, no. 1, pp. 68–77, 2019.

[133] M. T. Ribeiro, S. Singh, and C. Guestrin, "Anchors: High-precision model-agnostic explanations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[134] ——, ""why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[135] S. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *arXiv preprint arXiv:1705.07874*, 2017.

[136] Y.-H. H. Tsai, M. Ma, M. Yang, R. Salakhutdinov, and L.-P. Morency, "Multimodal routing: Improving local and global interpretability of multimodal language analysis," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[137] A. Brown, A. Tuor, B. Hutchinson, and N. Nichols, "Recurrent neural network attention mechanisms for interpretable system log anomaly detection," in *Proceedings of the First Workshop on Machine Learning for Computing Systems*, 2018, pp. 1–8.

[138] Q. P. Nguyen, K. W. Lim, D. M. Divakaran, K. H. Low, and M. C. Chan, "Gee: A gradient-based explainable variational autoencoder for network anomaly detection," in *2019 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2019, pp. 91–99.

[139] N. Liu, X. Huang, and X. Hu, "Accelerated local anomaly detection via resolving attributed networks." in *IJCAI*, 2017, pp. 2337–2343.

[140] P. Sturmfels, S. Lundberg, and S.-I. Lee, "Visualizing the impact of feature attribution baselines," *Distill*, vol. 5, no. 1, p. e22, 2020.

[141] J. Sipple, "Interpretable, multidimensional, multimodal anomaly detection with negative sampling for detection of device failure," in *International Conference on Machine Learning*. PMLR, 2020, pp. 9016–9025.

[142] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[143] X. Han, H. Cheng, D. Xu, and S. Yuan, "Interpretablesad: Interpretable anomaly detection in sequential log data," in *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021, pp. 1183–1192.

[144] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano, "A review on outlier/anomaly detection in time series data," in *ACM CSUR*, 2021, pp. 1–33.

[145] H. Ren, B. Xu, Y. Wang, C. Yi, C. Huang, X. Kou, T. Xing, M. Yang, J. Tong, and Q. Zhang, "Time-series anomaly detection service at microsoft," in *KDD*, 2019, pp. 3009–3017.

[146] D. T. Shipmon, J. M. Gurevitch, P. M. Piselli, and S. T. Edwards, "Time series anomaly detection; detection of anomalous drops with limited features and sparse examples in noisy highly periodic data," *arXiv preprint arXiv:1708.03665*, 2017.

[147] A.-H. Karimi, G. Barthe, B. Schölkopf, and I. Valera, "A survey of algorithmic recourse: contrastive explanations and consequential recommendations," in *ACM CSUR*, 2022, pp. 1–29.

[148] S. Nedelkoski, J. Bogatinovski, A. K. Mandapati, S. Becker, J. Cardoso, and O. Kao, "Multi-source distributed system data for ai-powered analytics," in *ESOCC*, 2020, pp. 161–176.

[149] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, "Usad: Unsupervised anomaly detection on multivariate time series," in *KDD*, 2020.

[150] J. Pearl, *Causality*. Cambridge university press, 2009.

[151] S. Schmidl, P. Wenig, and T. Papenbrock, "Anomaly detection in time series: a comprehensive evaluation," in *VLDB*, 2022.

[152] A.-H. Karimi, J. Von Kügelgen, B. Schölkopf, and I. Valera, "Algorithmic recourse under imperfect causal knowledge: a probabilistic approach," *Advances in Neural Information Processing Systems*, 2020.

[153] A.-H. Karimi, B. Schölkopf, and I. Valera, "Algorithmic recourse: from counterfactual explanations to interventions," in *ACM FAccT*, 2021.

[154] J. von Kügelgen, A.-H. Karimi, U. Bhatt, I. Valera, A. Weller, and B. Schölkopf, "On the fairness of causal algorithmic recourse," in *AAAI*, 2022.

[155] R. Dominguez-Olmedo, A. H. Karimi, and B. Schölkopf, "On the adversarial robustness of causal algorithmic recourse," in *ICML*, 2022, pp. 5324–5342.

[156] B. Ustun, A. Spangher, and Y. Liu, "Actionable recourse in linear classification," in *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 2019.

[157] S. Joshi, O. Koyejo, W. Vijitbenjaronk, B. Kim, and J. Ghosh, "Towards realistic individual recourse and actionable explanations in black-box decision making systems," *arXiv preprint arXiv:1907.09615*, 2019.

[158] R. Poyiadzi, K. Sokol, R. Santos-Rodriguez, T. De Bie, and P. Flach, "Face: feasible and actionable counterfactual explanations," in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 2020, pp. 344–350.

[159] M. Pawelczyk, S. Bielawski, J. v. d. Heuvel, T. Richter, and G. Kasneci, "Carla: a python library to benchmark algorithmic recourse and counterfactual explanation algorithms," *arXiv preprint arXiv:2108.00783*, 2021.

[160] D. Datta, F. Chen, and N. Ramakrishnan, "Framing algorithmic recourse for anomaly detection," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 283–293.

[161] C. W. Granger, "Investigating causal relations by econometric models and cross-spectral methods," *Econometrica: journal of the Econometric Society*, pp. 424–438, 1969.

[162] R. Dahlhaus and M. Eichler, "Causality and graphical models in time series analysis," *Oxford Statistical Science Series*, pp. 115–137, 2003.

[163] M. Nauta, D. Bucur, and C. Seifert, "Causal discovery with attention-based convolutional neural networks," *Machine Learning and Knowledge Extraction*, vol. 1, no. 1, p. 19, 2019.

[164] A. Tank, I. Covert, N. Foti, A. Shojaie, and E. B. Fox, "Neural granger causality," in *IEEE PAMI*, 2021, pp. 4267–4279.

[165] R. Marcinkevičs and J. E. Vogt, "Interpretable models for granger causality using self-explaining neural networks," *arXiv preprint arXiv:2101.07600*, 2021.

[166] H. Lütkepohl, *New introduction to multiple time series analysis*. Springer Science & Business Media, 2005.

[167] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Müller, "A unifying review of deep and shallow anomaly detection," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 756–795, 2021.

[168] A. A. Cook, G. Mısırlı, and Z. Fan, "Anomaly detection for iot time-series data: A survey," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6481–6494, 2019.

[169] S. Tuli, G. Casale, and N. R. Jennings, "Tranad: Deep transformer networks for anomaly detection in multivariate time series data," *arXiv preprint arXiv:2201.07284*, 2022.

[170] N. Bacaër, *A short history of mathematical population dynamics*. Springer, 2011, vol. 618.

[171] X. Han, L. Zhang, Y. Wu, and S. Yuan, "Algorithmic recourse for anomaly detection in multivariate time series," *arXiv preprint arXiv:2309.16896*, 2023.

[172] H. Jayathilaka, C. Krintz, and R. Wolski, "Performance monitoring and root cause analysis for cloud-hosted web applications," in *Proceedings of the 26th International Conference on World Wide Web*, 2017, pp. 469–478.

[173] V. Jeyakumar, O. Madani, A. Parandeh, A. Kulshreshtha, W. Zeng, and N. Yadav, "Explainit!–a declarative root-cause analysis engine for time series data," in *Proceedings of the 2019 International Conference on Management of Data*, 2019, pp. 333–348.

[174] G. Yu, P. Chen, H. Chen, Z. Guan, Z. Huang, L. Jing, T. Weng, X. Sun, and X. Li, "Microrank: End-to-end latency issue localization with extended spectrum analysis in microservice environments," in *Proceedings of the Web Conference 2021*, 2021, pp. 3087–3098.

[175] C. K. Assaad, I. Ez-Zejjari, and L. Zan, "Root cause identification for collective anomalies in time series given an acyclic summary causal graph with loops," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2023, pp. 8395–8404.

[176] M. Li, Z. Li, K. Yin, X. Nie, W. Zhang, K. Sui, and D. Pei, "Causal inference-based root cause analysis for online service systems with intervention recognition," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 3230–3240.

[177] Y. Zhang, Z. Guan, H. Qian, L. Xu, H. Liu, Q. Wen, L. Sun, J. Jiang, L. Fan, and M. Ke, "Cloudrca: A root cause analysis framework for cloud computing platforms," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 4373–4382.

[178] A. Ikram, S. Chakraborty, S. Mitra, S. Saini, S. Bagchi, and M. Kocaoglu, "Root cause analysis of failures in microservices through causal discovery," *Advances in Neural Information Processing Systems*, vol. 35, pp. 31 158–31 170, 2022.

[179] D. Wang, Z. Chen, J. Ni, L. Tong, Z. Wang, Y. Fu, and H. Chen, "Interdependent causal networks for root cause localization," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 5051–5060.

[180] H. Shan, Y. Chen, H. Liu, Y. Zhang, X. Xiao, X. He, M. Li, and W. Ding, "?-diagnosis: Unsupervised and real-time diagnosis of small-window long-tail latency in large-scale microservice platforms," in *The World Wide Web Conference*, 2019, pp. 3215–3222.

[181] M. Kim, R. Sumbaly, and S. Shah, "Root cause detection in a service-oriented architecture," *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, no. 1, pp. 93–104, 2013.

[182] J. Weng, J. H. Wang, J. Yang, and Y. Yang, "Root cause analysis of anomalies of multitier services in public clouds," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1646–1659, 2018.

[183] P. Wang, J. Xu, M. Ma, W. Lin, D. Pan, Y. Wang, and P. Chen, "Cloudranger: Root cause identification for cloud native systems," in *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE, 2018, pp. 492–502.

[184] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, and N. V. Chawla, "A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 1409–1416.

[185] H. Zhao, Y. Wang, J. Duan, C. Huang, D. Cao, Y. Tong, B. Xu, J. Bai, J. Tong, and Q. Zhang, "Multivariate time-series anomaly detection via graph attention network," in *ICDM*, 2020, pp. 841–850.

[186] P. Spirtes, C. N. Glymour, R. Scheines, and D. Heckerman, *Causation, Prediction, and Search*. MIT press, 2000.

[187] D. Wang, Z. Chen, Y. Fu, Y. Liu, and H. Chen, "Incremental causal graph learning for online root cause analysis," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 2269–2278.

[188] C. K. Assaad, E. Devijver, and E. Gaussier, "Survey and evaluation of causal discovery methods for time series," *Journal of Artificial Intelligence Research*, vol. 73, pp. 767–819, 2022.

[189] A. Siffer, P.-A. Fouque, A. Termier, and C. Largouet, "Anomaly detection in streams with extreme value theory," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 1067–1075.

[190] S. Absar, Y. Wu, and L. Zhang, "Neural time-invariant causal discovery from time series data," in *2023 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2023, pp. 1–8.

[191] M. E. Newman, "Estimating network structure from unreliable measurements," *Physical Review E*, vol. 98, no. 6, p. 062321, 2018.

[192] A. P. Mathur and N. O. Tippenhauer, "Swat: A water treatment testbed for research and training on ics security," in *2016 international workshop on cyber-physical systems for smart water networks (CySWater)*. IEEE, 2016, pp. 31–36.

[193] R. Moraffah, P. Sheth, M. Karami, A. Bhattacharya, Q. Wang, A. Tahir, A. Raglin, and H. Liu, "Causal inference for time series analysis: Problems, methods and evaluation," *Knowledge and Information Systems*, vol. 63, pp. 3041–3085, 2021.

[194] U. Hasan, E. Hossain, and M. O. Gani, "A survey on causal discovery methods for iid and time series data," *Transactions on Machine Learning Research*, 2023.

[195] X. Sun, O. Schulte, G. Liu, and P. Poupart, "Nts-notears: Learning nonparametric dbns with prior knowledge," *arXiv preprint arXiv:2109.04286*, 2021.

[196] R. Pamfil, N. Sriwattanaworachai, S. Desai, P. Pilgerstorfer, K. Georgatzis, P. Beaumont, and B. Aragam, "Dynotears: Structure learning from time-series data," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 1595–1605.

[197] M. Ma, J. Xu, Y. Wang, P. Chen, Z. Zhang, and P. Wang, "Automap: Diagnose your microservice-based web applications automatically," in *Proceedings of The Web Conference 2020*, 2020, pp. 246–258.

[198] S. Khanna and V. Y. Tan, "Economy statistical recurrent units for inferring nonlinear granger causality," *arXiv preprint arXiv:1911.09879*, 2019.

[199] J. Runge, P. Nowack, M. Kretschmer, S. Flaxman, and D. Sejdinovic, "Detecting and quantifying causal associations in large nonlinear time series datasets," *Science advances*, vol. 5, no. 11, p. eaau4996, 2019.

[200] Y. Cheng, R. Yang, T. Xiao, Z. Li, J. Suo, K. He, and Q. Dai, "Cuts: Neural causal discovery from irregular time-series data," *arXiv preprint arXiv:2302.07458*, 2023.

[201] C. Liu, W. Yang, H. Mittal, M. Singh, D. Sahoo, and S. C. Hoi, "Pyrca: A library for metric-based root cause analysis," *arXiv preprint arXiv:2306.11417*, 2023.

[202] E. Panjei, L. Gruenwald, E. Leal, C. Nguyen, and S. Silvia, "A survey on outlier explanations," *The VLDB Journal*, pp. 1–32, 2022.

[203] J. Kauffmann, K.-R. Müller, and G. Montavon, "Towards explaining anomalies: a deep taylor decomposition of one-class models," *Pattern Recognition*, vol. 101, p. 107198, 2020.

[204] P. Liznerski, L. Ruff, R. A. Vandermeulen, B. J. Franks, M. Kloft, and K.-R. Müller, "Explainable deep one-class classification," *arXiv preprint arXiv:2007.01760*, 2020.

[205] W. Yang, K. Zhang, and S. Hoi, "A causal approach to detecting multivariate time-series anomalies and root causes," 2023. [Online]. Available: https://openreview.net/forum?id=f25VGPzATcn

[206] Y. Meng, S. Zhang, Y. Sun, R. Zhang, Z. Hu, Y. Zhang, C. Jia, Z. Wang, and D. Pei, "Localizing failure root causes in a microservice through causality inference," in *2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*. IEEE, 2020, pp. 1–10.

[207] K. Budhathoki, L. Minorics, P. Blöbaum, and D. Janzing, "Causal structure-based root cause analysis of outliers," in *International Conference on Machine Learning*. PMLR, 2022, pp. 2357–2369.

[208] C. Louizos, U. Shalit, J. M. Mooij, D. Sontag, R. Zemel, and M. Welling, "Causal effect inference with deep latent-variable models," *Advances in neural information processing systems*, vol. 30, 2017.

[209] Y. Yu, J. Chen, T. Gao, and M. Yu, "Dag-gnn: Dag structure learning with graph neural networks," in *ICML*, 2019.

[210] X. Han, L. Zhang, Y. Wu, and S. Yuan, "Achieving counterfactual fairness for anomaly detection," in *Advances in Knowledge Discovery and Data Mining: 27th Pacific-Asia Conference, PAKDD 2023*. Springer, 2023.

[211] I. Ng, S. Zhu, Z. Chen, and Z. Fang, "A graph autoencoder approach to causal structure learning," *arXiv preprint arXiv:1911.07420*, 2019.

[212] K. Kiritoshi, T. Izumitani, K. Koyama, T. Okawachi, K. Asahara, and S. Shimizu, "Estimating individual-level optimal causal interventions combining causal models and machine learning models," in *The KDD'21 Workshop on Causal Discovery*. PMLR, 2021, pp. 55–77.

[213] M. J. Vowels, N. C. Camgoz, and R. Bowden, "D'ya like dags? a survey on structure learning and causal discovery," *ACM CSUR*, 2021.

[214] J. Correa and E. Bareinboim, "A calculus for stochastic interventions: Causal effect identification and surrogate experiments," in *AAAI*, 2020.

[215] P. Sánchez-Martin, M. Rateike, and I. Valera, "Vaca: Designing variational graph autoencoders for causal queries," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 7, 2022, pp. 8159–8168.

[216] C. Glymour, K. Zhang, and P. Spirtes, "Review of causal discovery methods based on graphical models," *Frontiers in genetics*, vol. 10, p. 524, 2019.

[217] A. Asuncion and D. Newman, "Uci machine learning repository," 2007.

[218] M. Kalisch and P. Bühlman, "Estimating high-dimensional directed acyclic graphs with the pc-algorithm." *Journal of Machine Learning Research*, vol. 8, no. 3, 2007.

[219] D. Janzing, K. Budhathoki, L. Minorics, and P. Blöbaum, "Causal structure based root cause analysis of outliers," *arXiv preprint arXiv:1912.02724*, 2019.

[220] A. Sharma and E. Kiciman, "Dowhy: An end-to-end library for causal inference," *arXiv preprint arXiv:2011.04216*, 2020.

[221] M. Pawelczyk, K. Broelemann, and G. Kasneci, "Learning model-agnostic counterfactual explanations for tabular data," in *Proceedings of The Web Conference 2020*, 2020, pp. 3126–3132.

[222] X. Han, L. Zhang, Y. Wu, and S. Yuan, "On root cause localization and anomaly mitigation through causal inference," in *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2023, pp. 699–708.

[223] H. Zhang and I. Davidson, "Towards fair deep anomaly detection," in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. ACM, 2021.

[224] H. Song, P. Li, and H. Liu, "Deep clustering based fair outlier detection," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 1481–1489.

[225] P. Deepak and S. S. Abraham, "Fair outlier detection," in *21th International Conference on Web Information Systems Engineering: WISE 2020*, 2020, pp. 447–462.

[226] S. Shekhar, N. Shah, and L. Akoglu, "Fairod: Fairness-aware outlier detection," in *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, 2021, pp. 210–220.

[227] M. Almanza, A. Epasto, A. Panconesi, and G. Re, "k-clustering with fair outliers," in *WSDM*. ACM, 2022.

[228] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," *ACM computing surveys (CSUR)*, vol. 54, no. 6, pp. 1–35, 2021.

[229] N. Kilbertus, M. Rojas-Carulla, G. Parascandolo, M. Hardt, D. Janzing, and B. Schölkopf, "Avoiding discrimination through causal reasoning," in *NIPS*, 2017.

[230] R. Nabi and I. Shpitser, "Fair inference on outcomes," in *AAAI*, 2018.

[231] B. van Breugel, T. Kyono, J. Berrevoets, and M. van der Schaar, "Decaf: Generating fair synthetic data using causally-aware generative networks," in *Advances in Neural Information Processing Systems*, 2021.

[232] M. J. Kusner, J. Loftus, C. Russell, and R. Silva, "Counterfactual fairness," *Advances in neural information processing systems*, vol. 30, 2017.

[233] S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the gdpr," *Harv. JL & Tech.*, vol. 31, p. 841, 2017.

[234] H. Edwards and A. Storkey, "Censoring representations with an adversary," *arXiv preprint arXiv:1511.05897*, 2015.

[235] D. Madras, E. Creager, T. Pitassi, and R. S. Zemel, "Learning adversarially fair and transferable representations. corr abs/1802.06309 (2018)," *arXiv preprint arXiv:1802.06309*, 2018.

[236] X. Zheng, B. Aragam, P. K. Ravikumar, and E. P. Xing, "Dags with no tears: Continuous optimization for structure learning," *Advances in Neural Information Processing Systems*, 2018.

[237] J. Dressel and H. Farid, "The accuracy, fairness, and limits of predicting recidivism," *Science advances*, vol. 4, no. 1, p. eaao5580, 2018.

CURRICULUM VITAE

Xiao Han

Education

- Ph.D. in Computer Science. Utah State University, Logan, UT. August 2020 - May 2024 (Expected).
- M.S. in Data Analytics. George Washington University, Washington, DC. August 2018 - May 2020.
- M.Eng. in Computer Science. Oregon State University, Corvallis, OR. September 2014 - December 2017.
- B.Econ. in Finance. Shandong University, Jinan, Shandong, China. September 2008 - May 2012.
- B.Eng. in Computer Science and Technology. Shandong University, Jinan, Shandong, China. September 2008 - May 2012.

Research Interests

Data mining, machine learning, and artificial intelligence, with a particular focus on anomaly detection, fairness-aware machine learning, root cause analysis, and reinforcement learning.

Published Conference Papers

- **Xiao Han**, Shuhan Yuan, and Mohamed Trabelsi. LogGPT: Log Anomaly Detection via GPT. In 2023 IEEE International Conference on Big Data (**Big Data**). 2023.

- **Xiao Han**, Lu Zhang, Yongkai Wu, and Shuhan Yuan. On Root Cause Localization and Anomaly Mitigation through Causal Inference. In Proceedings of the 32nd ACM International Conference on Information & Knowledge Management. (**CIKM**). 2023.

- **Xiao Han**, Lu Zhang, Yongkai Wu, and Shuhan Yuan. Achieving Counterfactual Fairness for Anomaly Detection. In Pacific-Asia Conference on Knowledge Discovery and Data Mining. (**PAKDD**). 2023.

- **Xiao Han**, Depeng Xu, Shuhan Yuan, and Xintao Wu. Few-shot Anomaly Detection and Classification Through Reinforced Data Selection. In 2022 IEEE International Conference on Data Mining (**ICDM**). 2022.

- **Xiao Han**, He Cheng, Depeng Xu, and Shuhan Yuan. InterpretableSAD: Interpretable Anomaly Detection in Sequential Log Data. In 2021 IEEE International Conference on Big Data (**Big Data**). 2021.

- **Xiao Han** and Shuhan Yuan. Unsupervised cross-system log anomaly detection via domain adaptation. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management. (**CIKM**). 2021.

Preprints

- **Xiao Han**, Saima Absar, Lu Zhang, and Shuhan Yuan. Root Cause Analysis of Anomalies in Multivariate Time Series through Granger Causal Discovery. under review. 2024.

- **Xiao Han**, Lu Zhang, Yongkai Wu, and Shuhan Yuan. On Interpretable Anomaly Detection Using Causal Algorithmic Recourse. arXiv preprint. 2022.