Towards Effective Performance Diagnosis for Distributed Applications

Ruyue Xin

Towards Effective Performance Diagnosis for Distributed Applications

Ruyue Xin

Towards Effective Performance Diagnosis for Distributed Applications

Ruyue Xin

This research was funded by the EU Horizon 2020 and Horizon Europe research and innovation program under grant agreements 825134 (ARTICONF project), 862409 (BlueCloud project), 824068 (ENVRIFAIR project) and 101094227 (BlueCloud 2026).

Cover design by Ruyue Xin and Midjourney
Printed and bound by Ipskamp Printing.

ISBN: 978-94-6473-267-2

Towards Effective Performance Diagnosis for Distributed Applications

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. P.P.C.C. Verbeek
ten overstaan van een door het College voor Promoties ingestelde commissie,
in het openbaar te verdedigen in de Agnietenkapel
op woensdag 1 november 2023, te 16.00 uur

door Ruyue Xin
geboren te Henan

Promotiecommissie

Promotores:	prof. dr. ir. C.T.A.M. de Laat	Universiteit van Amsterdam
	dr. P. Grosso	Universiteit van Amsterdam
Copromotores:	dr. Z. Zhao	Universiteit van Amsterdam
Overige leden:	prof. dr. R.A. Prodan	University of Klagenfurt
	prof. dr. A.D. Pimentel	Universiteit van Amsterdam
	prof. dr. R.V. van Nieuwpoort	Universiteit van Amsterdam
	dr. C. Papagianni	Universiteit van Amsterdam
	dr. Z.A. Mann	Universiteit van Amsterdam

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

Contents

Chapter 1

Introduction

Cloud computing provides elastic and on-demand resources for customizing data storage, processing, and communication, transforming how software applications are developed, deployed, and managed [286]. By providing on-demand access to a wide range of computing resources, cloud environments have made it possible to build and run distributed applications across multiple machines and data centers, enabling better scalability and performance than traditional applications. According to a report by Marketsandmarkets, the cloud computing market produced up to $545.8 billion in 2022, and it will increase up to $1240.9 billion in 2027[1]. As a leading cloud provider, Microsoft Azure[2] has deployed tens of millions of physical servers in thousands of data centers on five continents, running applications and services for more than 700 million users[3].

Distributed cloud applications often contain multiple components or microservices that communicate with each other over a network to perform complex tasks. With the rise of microservices architecture, applications are being broken down into more minor, specialized services that can be developed, deployed, and scaled independently. Many companies have delivered their core business through microservice technologies. However, this distribution introduces new challenges in assuring the performance of applications. The performance of distributed applications can be unstable due to the complex dependencies between services and the inherent dynamism of cloud environments, and performance anomalies such as degraded response time caused by resource saturation may severely affect the quality of the user experience [250]. Therefore, it is crucial to perform diagnosis and mitigate performance issues in distributed applications.

Performance diagnosis, defined as detecting abnormal performance phenomena, e.g., degradation, predicting anomalies to forestall future incidents, and localizing root causes of performance anomalies, is vital for distributed applications

[1] https://www.marketsandmarkets.com/Market-Reports/cloud-computing-market-234.html
[2] https://azure.microsoft.com/en-us/
[3] https://www.usesignhouse.com/blog/microsoft-azure-stats

[108]. Large-scale performance data can be collected by continuously monitoring the running status of applications and infrastructures. Based on performance data, anomaly detection can be developed to identify abnormal behaviour, such as sudden spikes in resource usage or slow response times and alert operators to potential issues for preventing future incidents [107]. Additionally, performance diagnosis should provide insight into the root cause of performance anomalies, allowing operators to take prompt actions such as migration of services or resource scaling to mitigate the impact of these issues [45]. Therefore, effective performance diagnosis is critical to maintaining the reliability, availability, and responsiveness of distributed applications and ensuring that they deliver the level of service that users expect.

Performance diagnosis systems for distributed applications have been studied in recent years. Ibidunmoye et al. [108] reviewed performance anomaly detection and bottleneck identification, in which they formulated fundamental research problems, categorized detection methods, and proposed research trends and open challenges. In recent years, Artificial Intelligence (AI)-based performance diagnosis systems have been popular because AI methods outperform traditional statistical methods and adapt to large-scale data and complex scenarios [209]. AI-based performance diagnosis systems can be summarized as comprising four main components: data collection, data preprocessing, anomaly detection, and root cause localization [209, 250]. Data collection and preprocessing are important for subsequent processing, while research is mainly about data quality. Research on performance anomaly detection mainly focuses on developing models that can accurately and efficiently detect anomalies based on collected performance data; statistical and Machine Learning (ML) methods are developed [108]. On the other hand, research about root cause localization is focused on developing techniques that can identify the underlying causes of performance anomalies, such as resource-related metrics in faulty services [209]. Despite promising advancements for performance diagnosis systems, AI-based methods face potential hazards and may lose public trust due to poor robustness and inexplainability [266]. As a result, research in this area aims to develop performance diagnosis systems that can proactively detect and address performance issues in distributed applications, ensuring user satisfaction and optimal diagnosis performance, such as good accuracy and robustness.

Despite the progress in developing AI-based performance diagnosis systems for distributed applications, several challenges still need to be solved. From a data perspective, there are two primary challenges. The first is the need for more data labels, as most monitoring data does not contain labels that can be immediately used to train ML-based models [218]. Labelling time-series data is often manual and time-consuming, which can limit the scalability of performance diagnosis systems [104]. The second challenge is data noise, which can significantly influence the performance of anomaly detection methods and increase the false-positive detection rate [211]. Monitoring data collected from a distributed

network often contain noise, making it challenging to detect performance anomalies accurately. Regarding ML methods used in AI-based performance diagnosis systems, trustworthiness requirements to prevent potential hazards of AI models, such as low robustness, and meet user satisfaction is necessary [128]. Specifically, several challenges must be addressed for AI-based performance diagnosis systems. Firstly, services in distributed applications are often heterogeneous and have different characteristics, which can result in diverse anomaly symptoms for the same issue. Secondly, there is a trade-off between detection accuracy and robustness, as some detection methods may be highly accurate but not robust enough to handle different performance issues [293]. Thirdly, the complex dependencies between services make it difficult to accurately model the anomaly propagation resulting from faulty services. Finally, with the many anomalous metrics introduced in a system, it can be challenging to identify the root cause of a performance anomaly, leading to delays in addressing performance issues and optimizing performance. Addressing these challenges is critical to developing robust and effective performance diagnosis systems for distributed applications.

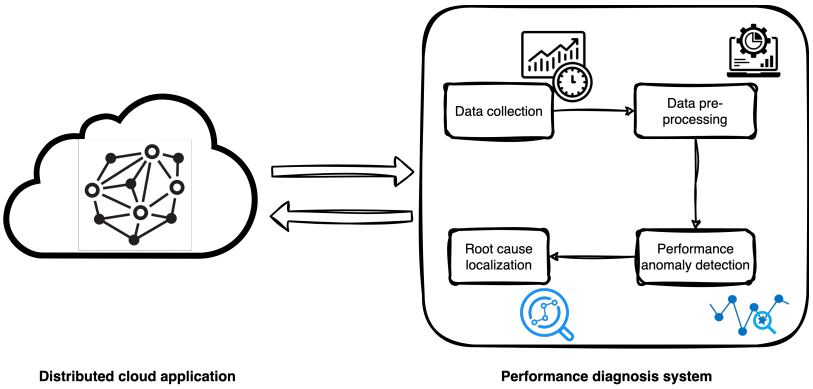


Figure 1.1: An illustration of the performance diagnosis system

To overcome the challenges of performance diagnosis in distributed applications, this thesis aims to develop a comprehensive performance diagnosis system that can accurately detect anomalies and localize their root causes to provide actionable insights to operators. The proposed system will first provide real-time monitoring and visualization tools to collect performance data, such as CPU usage, memory usage, and network traffic. Then based on performance data, the diagnosis system leverages ML techniques to detect anomalies and use causal inference and dependency analysis to localize the root causes of performance anomalies. Additionally, the proposed system will address the challenges of missing data labels and data noise by leveraging unsupervised learning techniques and data cleansing algorithms to preprocess monitoring data. The system will be evaluated using real-world data collected from distributed applications to demonstrate its effectiveness and efficiency in detecting and localizing performance anomalies.

As a result, the proposed performance diagnosis system aims to enhance the reliability, availability, and responsiveness of distributed applications and improve user satisfaction. An illustration of the system is shown in Figure 1.1.

1.1 Research questions

To tackle the challenges of building a performance diagnosis system, we formulate our main research question as:

RQ: How to effectively diagnose the performance of distributed applications in cloud environments at runtime?

To answer this question, we further define the following sub-questions:

RQ1: What are the state-of-the-art technologies for achieving trustworthy performance diagnosis systems?

Due to its convenience, recent research has concentrated on utilizing AI technologies to develop performance diagnosis systems. However, these AI-based methods face potential challenges and a loss of public trust due to their poor robustness and lack of explainability. To provide developers and operators actionable guidance for distributed cloud applications, a general performance diagnosis system should be summarized and its trustworthiness should be achieved with different requirements, for example, accuracy and robustness for anomaly detection, explanation for detected anomalies. While there is a rare survey about trustworthy performance diagnosis systems, we consider it is critical to review the current state-of-the-art technologies for creating a performance diagnosis system and establish requirements for developing an effective trustworthy performance diagnosis framework.

RQ2: How can a performance diagnosis framework be developed to identify performance issues and determine root causes effectively?

Firstly, an effective performance diagnosis framework of distributed cloud applications requires continuous monitoring and data collection. Through monitoring, a large amount of performance data can be collected and valuable insights into the performance of applications can be gained. However, addressing noise in the collected performance data is crucial, as it can affect diagnosis results. In addition, an effective performance diagnosis framework should detect anomalies in the application's behaviour, such as slow response times or resource bottlenecks, and localize the root cause of these anomalies, such as CPU hog, via monitoring data. Therefore, it is crucial to develop a performance diagnosis framework that can keep monitoring applications and diagnose performance issues effectively at runtime.

RQ3: How to improve the accuracy and robustness of detecting performance anomalies?

For performance anomaly detection, addressing the challenge of missing data labels and data noise is crucial to improving accuracy. An effective anomaly de-

tection method should meet two main challenging requirements: high accuracy for identifying anomalies and good robustness when application patterns change. Research about statistical and ML methods has been developed recently. Developing anomaly detection methods based on existing methods can improve the trade-off between accuracy and robustness. Furthermore, advanced deep learning-based methods to extract multiple dimensions of information from data can be used to improve detection performance.

RQ4: How to localize root causes of detected performance anomalies at a fine-grained level?

To effectively address the challenges of performance diagnosis systems, fine-grained root cause localization is necessary to pinpoint the faulty service and identify which metrics are causing performance anomalies. Root cause localization provides explanation of performance anomalies and enables engineers to quickly take corrective actions, such as scaling resources or adjusting configuration parameters. Fine-grained root cause localization requires advanced techniques, such as causal inference and dependency analysis, to accurately identify the root cause of the anomaly. Improving the fine-grained root cause localization of performance anomalies is crucial for ensuring the reliability and availability of distributed cloud applications.

1.2 Key contributions

This thesis contributes a literature review, framework, models, and evaluations for a performance diagnosis framework of distributed cloud applications. Specifically, the main contributions of this thesis are:

Trustworthy Performance Diagnosis for Distributed Applications: Requirements, Methods, and Challenges

In recent years, AI-based performance diagnosis systems have been gaining the attention of both academia and industry. However, there is a lack of comprehensive surveys exploring state-of-the-art trustworthy performance diagnosis systems. While existing surveys mainly focus on general requirements of trustworthy AI [119], specific aspects of trustworthy AI technologies [162], or technical details of performance diagnosis frameworks [275], a survey specifically dedicated requirements and state-of-the-art technologies in building trustworthy performance diagnosis systems remains rare. To bridge the gap, we provide a survey of the state-of-the-art trustworthy performance diagnosis systems, extracting technical trustworthiness requirements and exploring state-of-the-art technologies to meet these requirements in each component of a performance diagnosis system. Our main contributions are as follows:

- We introduce research about trustworthiness requirements and extract essential technical trustworthiness requirements for AI-based performance di-

agnosis systems, including data privacy, fairness, robustness, explainability, and human intervention.

- From data to AI models, we propose a systematic performance diagnosis framework and unify technical trustworthiness requirements into the framework. We then provide a comprehensive taxonomy and approaches to enhance the trustworthiness of AI-based performance diagnosis systems.

- We discuss the future research directions and challenges of trustworthy AI-based performance diagnosis systems. Several key issues must be addressed, such as a more profound and fundamental understanding of robustness and explainability of detection models.

Effective Performance Diagnosis Framework for Distributed Applications

To ensure the desired service quality of a distributed application, operators must continuously monitor its run-time status, detect performance anomalies, and diagnose the root causes. The initial crucial step for an effective performance diagnosis framework is establishing reliable data monitoring and collection processes. However, challenges arise when dealing with collected performance data, such as the lack of data labels and the presence of data noise, which requires preprocessing methods. In addition, an effective performance diagnosis framework has the challenge of accurate performance anomaly detection and root cause localization. Therefore, it is imperative to look deep into existing methods to gain a comprehensive understanding and evaluation of them. In brief, our main contributions are as follows:

- We design an integrated framework to implement performance diagnosis effectively by collecting monitoring data, utilizing metrics selection to filter useless monitoring metrics, diverse performance anomaly detection methods, and fine-grained root cause localization.

- We implement a Decentralized Application (DApp) through automate deployment. We monitor its run-time status, inject anomalies to simulate real scenarios and collect real-time performance data for subsequent analysis.

- We evaluate the framework on the performance data collected from the DApp and two public datasets. Results present the effectiveness and performance of our diagnosis system.

Performance Anomaly Detection Methods with Enhanced Accuracy and Robustness

Effectively detecting run-time performance anomalies is crucial to identify abnormal performance behaviour and to prevent potential incidents of distributed cloud applications. Considering fewer labels in real scenarios, we focus on weakly

supervised and unsupervised detection methods. We identify high accuracy and good robustness as the main challenging requirements for performance anomaly detection and provide two detection models. We develop an Ensemble Learning-Based Detection (ELBD) framework that incorporates classic detection methods rather than enhances a single model. In addition, we develop a deep learning-based method for unsupervised detection methods and evaluate its detection performance on accuracy and robustness. The contributions mainly include:

- Based on base detection methods, we propose the ELBD framework, including three classic linear ensemble methods (maximum, average, and weighted average) and a deep ensemble method.

- We propose a metric to evaluate the detection performance of the ELBD framework in terms of accuracy, robustness, and multi-step prediction and conduct experiments on different datasets for performance evaluation.

- We develop a deep learning-based unsupervised detection method that outperforms the base detection methods, as demonstrated in our comparison experiments.

Root Cause Localization Framework with Gradient-based Causal Inference

Root cause localization aims to accurately identify the underlying causes of performance anomalies, such as resource-related metrics in faulty services. In this research, we propose the CausalRCA framework, which includes a gradient-based causal structure learning model and a root cause inference model to build anomaly propagation paths and perform root cause analysis effectively. We evaluate the localization performance of CausalRCA on the Sock Shop microservice benchmark. Our experimental results show that CausalRCA has improved localization accuracy. Our contributions can be summarized below:

- We propose an automated, fine-grained root cause localization framework named CausalRCA, which analyzes monitoring data and localizes faulty services and system resources in real time.

- We provide a gradient-based causal structure learning method in CausalRCA, which can automatically capture linear and non-linear causal relations between monitoring metrics.

- We conduct coarse- and fine-grained experiments to evaluate the localization performance of CausalRCA and demonstrate that the proposed framework has the best localization accuracy compared with baseline methods.

1.3 Sources of chapters

We provide a quick overview of papers on which each chapter is based.

- Chapter 2 is based on the following paper:
  - **Ruyue Xin**, Jingye Wang, Peng Chen, and Zhiming Zhao. "Trustworthy AI-based Performance Diagnosis Systems for Cloud Applications: A Review." ACM computing survey (under revision).
- Chapter 3 is based on the following paper:
  - **Ruyue Xin**, Jardenna Mohazzab, Zeshun Shi, and Zhiming Zhao. "CBProf: Customisable Blockchain-as-a-Service Performance Profiler in Cloud Environments." In Blockchain–ICBC 2021: 4th International Conference, Held as Part of the Services Conference Federation, SCF 2021, Virtual Event, December 10–14, 2021, Proceedings, pp. 131-139. Cham: Springer International Publishing, 2022.
  - **Ruyue Xin**, Hongyun Liu, Peng Chen, Paola Grosso, and Zhiming Zhao. "FIRED: a fine-grained robust performance diagnosis framework for cloud applications." arXiv preprint arXiv:2209.01970 (2022).
- Chapter 4 is based on the following paper:
  - **Ruyue Xin**, Hongyun Liu, Peng Chen, and Zhiming Zhao. "Robust and accurate performance anomaly detection and prediction for cloud applications: a novel ensemble learning-based framework." Journal of Cloud Computing 12, no. 1 (2023): 1-16.
  - Yujia Song, **Ruyue Xin**, Peng Chen, Rui Zhang, Juan Chen, and Zhiming Zhao. "Identifying performance anomalies in fluctuating cloud environments: a robust correlative-GNN-based explainable approach." Future Generation Computer Systems (2023). (as co-first author)
- Chapter 5 is based on the following paper:
  - **Ruyue Xin**, Peng Chen, and Zhiming Zhao. "Causalrca: Causal inference based precise fine-grained root cause localization for microservice applications." Journal of Systems and Software 203 (2023): 111724.

Code repositories

The source codes and data are accessible through these repositories.

- ELBD framework: https://github.com/AXinx/ELBD.git

- CGNN-MHSA-AR method: https://github.com/AXinx/CGNN-MHSA-AR.git
- CausalRCA framework: https://github.com/AXinx/CausalRCA_code.git

1.4 Thesis overview

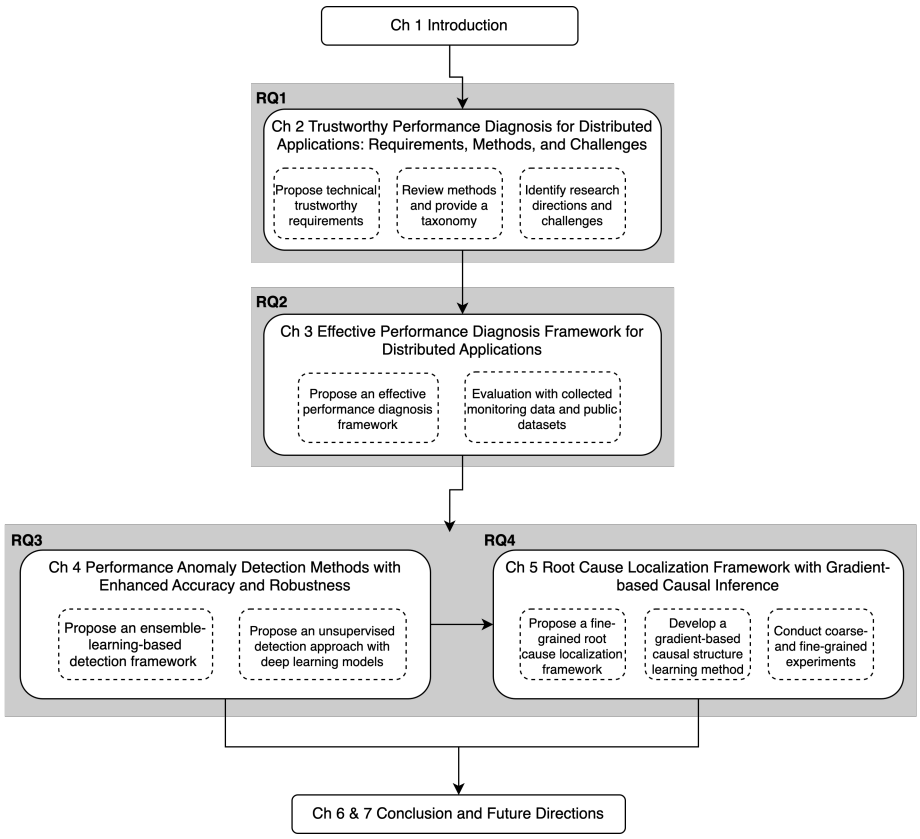


Figure 1.2: Thesis overview (inluding chapters, research questions, and contributions)

To provide an overview of our thesis, we present Figure 1.2, which depicts the relationships between chapters, research questions, and the key contributions of each chapter. The thesis comprises seven chapters in total. Chapter 1 introduces the thesis's background, research questions, contributions, and structure. Chapter 2 reviews technical requirements, state-of-the-art research and approaches for each component, and research directions and challenges that need further exploration to achieve building trustworthy performance diagnosis systems in RQ1. Based on the review of requirements and technologies for performance diagnosis systems, we propose a performance diagnosis framework in Chapter 3 in response to effectively pre-process data, identify performance anomalies, and determine

root causes in RQ2. The framework works with a monitoring tool, data preprocessing technologies, performance anomaly detection and root cause localization methods. We then focus on improving the performance of anomaly detection and root cause localization in the framework in Chapters 4 and 5. In Chapter 4, we concentrate on the improvement of accuracy and robustness for anomaly detection methods related to RQ3. We provide an ensemble-based method, a weakly-supervised approach that integrates existing detection methods based on ensemble learning and achieves a trade-off between improving detection accuracy and robustness. Additionally, we develop a deep learning-based unsupervised method to improve detection accuracy. Chapter 5 focuses on fine-grained root cause localization for detected performance anomalies in response to RQ4. We provide a root cause localization framework, including gradient-based causal structure learning and root cause inference methods. We evaluate the framework using performance data collected from a benchmark microservice application, and the results show that the framework outperforms existing methods. After conducting these research works, Chapter 6 summarizes our findings and presents the conclusion that an effective performance diagnosis framework can be realized through appropriate data preprocessing, accurate and robust anomaly detection, and fine-grained root cause localization for detected performance anomalies. Finally, Chapter 7 provides an overview of future research directions.

Chapter 2

Trustworthy Performance Diagnosis for Distributed Applications: Requirements, Technologies, and Challenges

Distributed cloud applications are developing rapidly because cloud environments provide elastic and on-demand resources for customizing data storage, processing, and communication. The performance of distributed applications can be unstable due to the inherent dynamism of clouds. Performance anomalies, such as degraded response time caused by resource saturation, may severely affect the quality of the user experience [250]. An effective performance diagnosis system is often developed based on AI approaches to detect performance anomalies and identify potential root causes. At the same time, AI methods have potential hazards that could degrade the user experience and trust. For example, issues with data privacy may compromise the security of AI models, and low robustness can be hard to apply in complex cloud environments. Therefore, defining the requirements for building a trustworthy AI-based performance diagnosis system has become essential. This chapter systematically overviews trustworthiness requirements in AI-based performance diagnosis systems. We first introduce trustworthiness requirements and extract five essential aspects from a technical perspective, including data privacy, fairness, robustness, explainability, and human intervention. We then unify these requirements into a general performance diagnosis framework, ranging from data collection to model development. Next, we comprehensively offer related work for each component and concrete actions to improve trustworthiness in the framework. Finally, we identify possible research directions and challenges for the future development of trustworthy AI-based performance diagnosis systems.

This chapter is based on:

- **Ruyue Xin**, Jingye Wang, Peng Chen, and Zhiming Zhao. "Trustworthy AI-based Performance Diagnosis Systems for Cloud Applications: A Review." ACM computing survey (under revision).

2.1 Introduction

Performance diagnosis, defined as detecting abnormal performance phenomena, e.g., degradation, predicting anomalies to forestall future incidents, and localizing causes of performance anomalies, is vital for distributed cloud applications. In recent years, research about AI-based performance diagnosis systems has been popular because AI models can outperform traditional diagnosis methods and adapt to large-scale data and complex cloud environments [108]. However, it is worth noting that AI-based performance diagnosis systems need to be trustworthy because some potential hazards of AI models could degrade the user experience and undermine trustworthiness. For example, performance diagnosis systems with low robustness are difficult to apply to large-scale distributed applications with complex data patterns.

AI technologies bring great convenience to human society, but have many potential hazards and are losing the public's trust, such as low data privacy, poor robustness, and inexplicability. The Equifax [17] data breach made the personal data of millions of customers accessible, severely compromising user privacy. Guidelines for trustworthy AI have been proposed recently to deal with the above problems. The European Union (EU) published three general ethics guidelines and seven requirements for trustworthy AI [51]. In addition, the EU released and applied the General Data Protection Regulation (GDPR) [183] to protect data privacy. The International Organization for Standardization (ISO) [1] also presented approaches and requirements to establish trust in AI systems. Research on trustworthy AI provides approaches to meet these requirements. For instance, Marques et al. [162] mainly focus on explainability and summarize current developments of the formal Explainable AI (XAI). Existing explorations of trustworthy AI demonstrate the current necessity to develop AI-based systems with trustworthiness requirements.

For a performance diagnosis system, we can consider improving its trustworthiness with different requirements (e.g., fairness) across multiple components in the system, e.g., data collection, anomaly detection, and root cause localization. In addition, the trustworthiness of the entire system, for example, human intervention to enhance diagnosis performance, should be assured. Some research about trustworthy diagnosis systems has been developed. Yu et al. [259] propose a trustworthy fault diagnosis system that integrates artificial neural network and rule-based reasoning to meet three trustworthiness requirements: explanation, accountability, and fairness. Zhang et al. [275] introduce a federated transfer learning method to ensure data privacy and increase model robustness. Nguyen et al. [170] propose a gradient-based explainable Variational Autoencoder (VAE) for anomaly detection, and it achieves explainability by analyzing gradients computed of each feature. The above research shows that existing trustworthy diagnosis systems studies focus on different requirements and aspects of a system, which inspires us to summarize them and provide guidelines for building a trustworthy

AI-based performance diagnosis system.

2.1.1 Related survey

Several surveys have discussed trustworthiness requirements for AI systems from different aspects. Kaur et al.[119] provide a complete overview of trustworthy AI, including analyzing all seven trustworthiness requirements, reviewing different approaches that can help mitigate AI risks to increase the trust of systems, and discussing existing strategies for validating and verifying systems and the current standardization efforts for trustworthy AI. Li et al.[128] provide a comprehensive guide for building trustworthy AI systems, ranging from data acquisition to model development, system development, deployment, and continuous monitoring and governance. Liu et al.[147] present a comprehensive appraisal of trustworthy AI from a computational perspective. They focus on six crucial dimensions, review related technologies according to a taxonomy, and summarize their applications in real-world systems. Moreover, Cho et al.[47] propose a system-level trustworthiness metric framework called STRAM that accommodates four submatrices (security, trust, resilience, and agility) to measure the quality of computer-based systems.

Besides surveys about the trustworthiness requirements of AI systems, some researchers provide overviews of specific requirements. For example, Mehrabi et al.[164] create a taxonomy for fairness definitions that ML researchers have defined to avoid bias in AI systems. This paper also reviews state-of-the-art methods that researchers have tried to address unfair outcomes of ML methods. Harley et al.[92] discuss integrity, the cornerstone of information security, to promote information trustworthiness through formal information flow models, the data modification view, and the relationship to data quality.

There are survey works related to performance diagnosis. Duarte et al.[65] provide a survey about comparison-based diagnosis in diverse, complex computer systems, and clarify and uncover the potential of this technology. A comparison-based diagnosis is a realistic approach to detecting faulty units based on the outputs of tasks executed by system units. Ibidunmoye et al.[108] provide an overview of performance anomaly detection and bottleneck identification research in computing systems. They categorize existing solutions based on multiple factors, such as the detection goals, nature of applications and systems, system observability, and detection methods. Soldani et al.[209] provide a structured overview and qualitative analysis of currently available techniques for anomaly detection and root cause analysis in modern multi-service applications.

Existing surveys focus on general requirements of trustworthy AI, specific aspects of trustworthy AI technologies, or technical details of performance diagnosis frameworks. However, there is rarely a survey about trustworthy performance diagnosis systems. To provide cloud application developers and operators with actionable guidance, extracting trustworthiness requirements for performance di-

agnosis systems from a systems perspective is necessary to build reliable performance diagnosis systems.

2.1.2 Research questions

To fill the gap of existing survey works, we define our main research questions as: *what are the state-of-the-art technologies for achieving trustworthy performance diagnosis systems?* To answer it, three sub-questions are proposed:

- What are trustworthiness requirements for different components of an AI-based performance diagnosis system?
- What are the state-of-the-art technologies to achieve these trustworthiness requirements?
- What are the future research directions and challenges of trustworthy AI-based performance diagnosis systems?

2.1.3 Methodology

Our objective is to provide technological solutions that meet the trustworthiness requirements of performance diagnosis systems. To achieve this, we conduct a comprehensive survey from 2012 and later based on the main scientific databases and publishers: Google Scholar, ACM Digital Library, Ieee Xlore, Elsevier, Springer, and arXiv. We use different keywords and filter papers mainly according to relevance. For performance diagnosis systems, we first search with the keywords "performance diagnosis systems" or "performance diagnosis + distributed applications" to get an overview of performance diagnosis systems. We summarize that a general performance diagnosis system includes four components. Then we organize surveyed research in each component based on data types. We identify three main data types in data collection and review corresponding data preprocessing methods, anomaly detection methods, and root cause localization methods tailored to each data type. Specifically, in the case of anomaly detection methods, because the feature matrix can be obtained after data preprocessing, we categorize the detection techniques based on the availability of data labels.

To identify relevant technologies that meet trustworthiness requirements for each component, we begin by conducting searches using specific keywords. For instance, we use "robustness technology + AI systems," to gather research on robustness technologies in AI systems. We can classify the common technologies highlighted in these studies into their respective components by analyzing and summarising them. For example, techniques like data augmentation, which

enhances robustness, are typically found in the data preprocessing stage. In contrast, ensemble learning methods, known for their robustness, are predominantly employed in the design of detection models.

The chapter is summarized as follows. Section 2.2 presents an overview of trustworthy AI and performance diagnosis systems. Section 2.3 presents a systematic survey of trustworthiness requirements and approaches in each component of performance diagnosis systems. Section 2.4 points future research directions and existing challenges. Finally, Section 2.5 concludes this chapter.

2.2 Background

2.2.1 Trustworthy AI

For trustworthy AI, many organizations have identified several factors and summarized the principles of AI trustworthiness. The EU published ethics guidelines and general requirements [51], and released the GDPR [183] to protect data privacy. The ISO has presented different approaches to establish trust in AI systems using the properties of fairness, transparency, accountability, and controllability [1]. The China Academy of Information and Communications Technology and JD Explore Academy published a trustworthy AI white paper. They proposed that current trustworthy AI principles should be converged on five aspects: transparency, security, fairness, accountability, and privacy protection [28]. The US Government Accountability Office provided a framework for the accountability of AI [173]. The Defense Advanced Research Project Agency launched the XAI program, whose motive is to make these AI systems explainable [86].

Research about trustworthy AI has been developed in the academy in recent years. Kaur et al. [119] discuss the need, proposed methods, and technical challenges of five trustworthy AI requirements: fairness, explainability, accountability, privacy, and acceptance. Li et al. [128] represent the technical challenges of trustworthy AI in five aspects: robustness, explainability, transparency, reproducibility, and generalization, and ethical requirements in terms of fairness, privacy, and accountability. Liu et al. [147] propose that trustworthy AI is expected to show the properties of accuracy, robustness, and explainability from a technical perspective. Kumar et al. [125] mainly focus on the ethics of algorithms with prevention of harm, respect for human autonomy, fairness, and explainability. The ethics of data includes five principles: human-centered, individual data control, transparency, accountability, and equality. Marques et al. [162] focus on explainability and summarize the recent developments and existing challenges in the formal XAI. Toreini et al. [226] propose classifying trustworthy technologies in fairness, explainability, auditability, safety, and discussing if and how these support the required qualities.

Research about trustworthy AI usually focuses on different requirements,

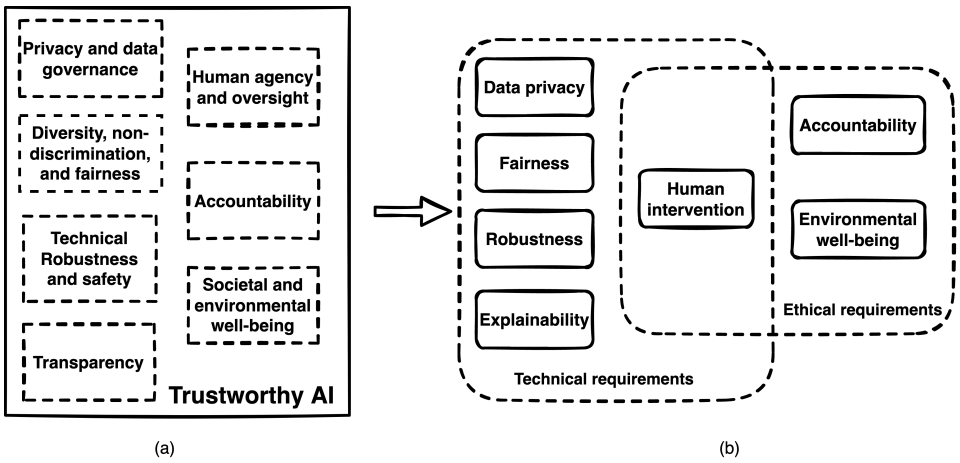


Figure 2.1: Trustworthiness requirements for AI systems

while all of them are included in EU requirements. We present the seven key requirements published by EU [51] in Figure 2.1(a). According to above research [1, 28, 86, 119, 125, 128, 147, 162, 173, 226], we extract keywords of each requirement and classify them into technical requirements and ethical requirements, as shown in Figure 2.1(b). Technical requirements describe that trustworthy can be ensured with technical methods in the design, development, and use phase of an AI system [51]. Ethical requirements are non-technical and raise the demand for appropriate management strategies of AI systems considering that efforts of various roles and steps need to be aligned [128]. We summarize that the technical requirements for an AI system encompass data privacy, fairness, robustness, and explainability. Ethical requirements involve aspects such as human intervention, accountability, and environmental well-being. While human intervention is typically considered as an ethical requirement, some technologies in AI systems can perform better with human intervention. Therefore, this chapter will mainly focus on those technical requirements for improving the trustworthiness of AI-based performance diagnosis systems.

Technical requirements overview

A general overview of technical requirements is represented below.

Data privacy. Generally, data privacy protection refers to prevent the unauthorized use of data that can identify a person directly or indirectly [119]. AI-based performance diagnosis systems work based on a vast amount of collected data, which require data privacy protects data from unauthorized access. In addition, private organizations, governments, or hackers could misuse data, leading to harmful consequences. For example, the leak of monitoring data can give information on application status to other companies. As a result, it is vital to protect the privacy of the data to both avoid harmful consequences and promise

the development of applications. To address growing concerns about data privacy, existing protection techniques should cover the entire lifecycle of AI systems, for example, protect data information and develop privacy-preserving models [128].

Fairness. AI systems frequently manifest bias and discrimination through unfair treatment of different groups of people based on their protected information (e.g., gender, race, and ethnicity). For example, [8] points out that a risk assessment tool used by the judicial system to predict future criminals was biased against black people. These examples show that bias can mislead black-box AI systems and cause harm or unfairness. Therefore, in AI systems, fairness refers to avoiding or mitigating the effects of bias and discrimination. When developing and applying AI-based performance diagnosis systems, biases can take different forms, such as data and model bias. Data bias mainly exists because of imbalanced data and insufficient labeled data. For example, the imbalanced data distribution makes an anomaly classifier biased toward the majority class (i.e., normal data) [219]. Unfair models exist due to the potentially high correlation between minority groups and outliers, which will produce injustice outcomes by overly flagging the samples from the minority groups [266].

Robustness. The robustness of an algorithm or system refers to its ability to handle execution errors, erroneous inputs, or unknown data [128]. A lack of robustness might cause a system to behave in an unintended or harmful manner, affecting its trustworthiness. For performance diagnosis systems, robustness can be categorized at the levels of data and models, respectively. Distributed applications are deployed in complicated environments, which means that various data distributions and features exist in collected performance data. Robustness against distribution shifts has been a common problem in various applications [34]. Furthermore, AI models are widely acknowledged to be vulnerable to malicious attacks. Among the various forms of attacks, the adversarial attack and defenses against it have raised concerns in academia and industry. For performance diagnosis systems, model robustness requires less sensitivity to small changes in the underlying data distribution and can adapt to different scenarios [73]. In addition, defense approaches for adversarial attacks can be considered to gurantee model robustness [222, 267].

Explainability. For AI models, explainability is a fundamental factor in determining whether they can be trusted since it addresses how AI models make decisions. The motivation for the explainability of AI comes from various aspects. From the perspective of scientific research, it is essential to understand all the intrinsic mechanisms of the data, parameters, procedures, and outcomes in an AI system. The mechanisms also fundamentally determine AI trustworthiness. Explainable AI can be realized through data and models. From a data perspective, selecting useful features is helpful for subsequent model interpretation. Form a model perspective, designing a series of fully or partially explainable models can be considered, such as K-Nearest Neighbor (KNN) [180]. In addition, analyzing model inputs, intermediate results, and outputs to find important features is

helpful for model explainability. For example, for deep learning models like the Convolutional Neural Network (CNN), the inspection of intermediate features is a widely used means of explaining model behavior [115].

Human intervention. In AI systems, human intervention refers to human participation in the decision-making of AI models to improve their performance. For performance diagnosis systems, data labeling can effectively improve diagnosis performance, and manual data labeling is the most primitive and accurate method. In addition, integrating human feedback to improve model results has been studied. For example, an analyst can provide direct feedback to an unsupervised anomaly detector to improve detection accuracy [206].

2.2.2 Performance diagnosis system

The performance of distributed cloud applications, which refers to their ability to successfully complete tasks, is of great importance [63, 224]. Performance management involves monitoring and measuring relevant performance data of applications and infrastructures. It also encompasses the analysis of the data, as well as the detection of performance anomalies and localizing root causes. Table 2.1 provides surveyed papers of performance diagnosis systems for distributed cloud applications. Based on these papers, we summarize that a general performance diagnosis system includes four components: data collection, data preprocessing, anomaly detection, and root cause localization. In data collection, there are different types of performance data. Free-text log data is collected chiefly. Time-series data, especially monitoring metrics, is also commonly used. In addition, trace data is being explored for performance diagnosis in distributed applications. For data preprocessing, different methods are developed based on data types. For example, a log parser has been developed to extract information from log data, and feature extraction and feature selection methods are mostly used. Existing performance diagnosis research mainly focuses on anomaly detection and root cause localization. Anomaly detection aims to find abnormal patterns that do not meet the estimated behavior during the operation of the system. Root cause localization is to identify the root causes of an anomaly after it occurs.

Table 2.1: AI-based performance diagnosis systems

Ref	Data collection	Preprocessing	Diagnosis module	
			Anomaly detection	Root cause localization
[63]	Text (log)	log parser	Deep learning (LSTM) and classic mining (density clustering) approaches for detection	
[268]	Text (log)	Log embedding	KNN, Naive Bayes, Neural Networks, and Random Forests classification	
[44]	Text (log)	Log parse and embedding	Graph Attention Network (unsupervised)	Rank based on distance
[132]	Text (log)	Log parser	Unified attention based BiLSTM model for anomaly detection	
[75]	Sequence (trace)	Feature selection	Multi-layer BiLSTM to identify task and job failures in the cloud	
[22]	Sequence (trace)	Log parser	Masked span prediction based on encoder-decoder structure	
[135]	Sequence (trace)	Feature selection	Calculate anomaly severity of traces	Calculate suspicious scores of services
[39]	Time series (metrics)	Data normalization and weight multiplication	Multi-classification: SNN and SVM	
[218]	Time series (metrics)	Data standardization and partition	OmniAnomaly (GRU, VAE and Planar NF)	
[159]	Time series (metrics)	Metric selection		Graph construction with PC and random walk
[41]	Time series (metrics)	Metric selection	CUSUM to detect change points	Build causality graph and causal inference

2.3 Requirements and technologies for trustworthy diagnosis systems

Based on our review of trustworthy AI and performance diagnosis systems, we unify the five technical trustworthiness requirements into performance diagnosis systems, as shown in Figure 2.2. Fairness is important for data collection, considering the imbalance issue in performance data. Data preprocessing can promise the robustness and explainability of diagnosis systems. For anomaly detection, fairness, robustness, and explainability must be satisfied when developing detection models. Research about robustness and explainability in root cause localization has been developed. Data runs through the entire system, meaning data privacy needs to be guaranteed throughout the system. Moreover, human intervention to improve diagnosis outcomes can be considered. We will introduce the diagnosis system and these trustworthiness requirements in detail next.

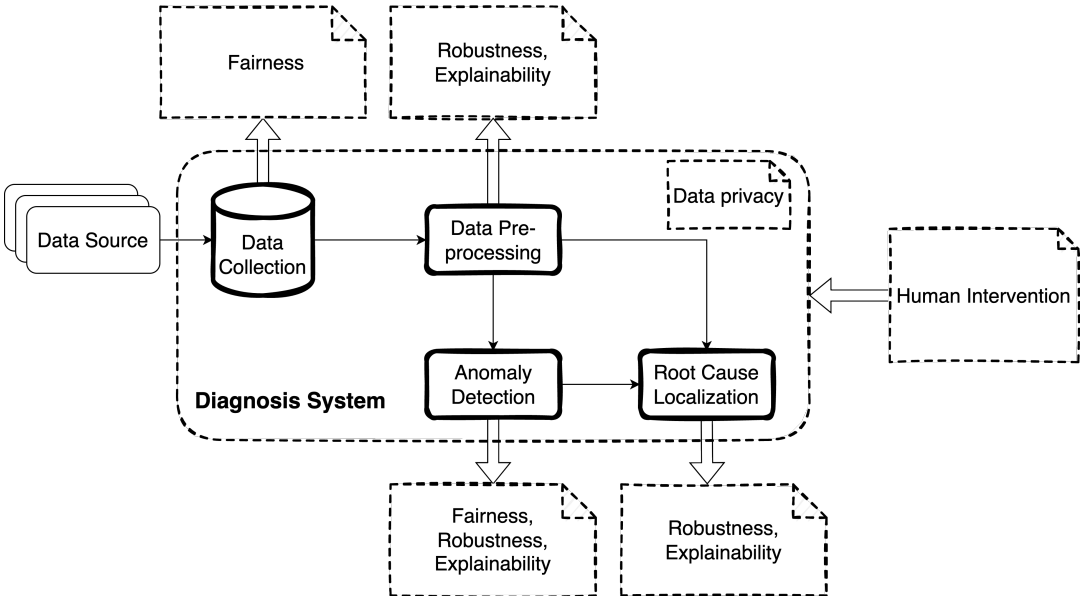


Figure 2.2: trustworthiness requirements for a diagnosis system

2.3.1 Data collection

In performance diagnosis systems, collecting data for training models is the primary step in the diagnosis pipeline. We will first introduce different types of collected performance data. Then we will provide research on fairness for data collection.

Performance data

Performance data can be collected with monitoring tools for a running application deployed in cloud environments. Performance data has multiple forms and types and includes information that can reflect the application's health status. Performance data can be classified into three categories based on data types, log data, trace data, and monitoring metrics [216].

Log data is usually unstructured and free-text. Figure 2.3(a) shows an example of log data from the HDFS logs on the Amazon EC2 platform [253]. Each line printed on the system console is a log message. Each log message consists of timestamps, a constant part (log event), and a variable part (log parameter). The log event comprises fixed text strings and is a template for a log message. The log parameter records system attributes, e.g., URL, file name, or IP address. A log sequence consists of a sequence of log events that records the execution flow of a specific task. Log events from the same log sequence share the same task ID, which can be used to link the events chronologically. Log events can be acquired from log messages through log parsing, a preprocessing method for log data.

Trace data is graph-like abstractions, as shown in Figure 2.3(b), containing information for understanding the execution workflow and performance of distributed applications. It consists of spans and traces, where each span represents a specific operation within a service, and a trace represents the execution process of an external request. Spans record details such as start time, end time, service name, HTTP path, and function in remote procedure calls [22], while traces provide a higher-level view of the interactions and dependencies between services [87]. Analyzing trace data enables operators to gain insights into service interactions, pinpoint where failures occur and root causes. To facilitate analysis and reduce the complexity of trace data, preprocessing steps, same with log parsing methods, can be employed to transform raw spans into a structured format [22].

Monitoring metrics are time-series data, as shown in Figure 2.3(c), and can be classified into service- and resource-level data [108]. Service-level data includes Key Performance Indicator (KPI) like latency and throughput. Latency represents the time taken for an operation to complete, while throughput measures the rate of work performed, such as the number of user requests completed in a given time interval [108]. Resource-level data captures the current capacity and utilization of infrastructures, including CPU, memory, disk, and network [84]. Resource capacity refers to the storage size or processing strength, while resource utilization measures the amount of capacity used compared to the available capacity. For example, CPU usage reflects the time the CPU is busy executing instructions, memory utilization tracks the storage consumed by a process or application, and network utilization quantifies the ratio of transmitted packets to the full transmission capacity of a network link in a given time interval.

In conclusion, performance data includes log data, trace data, and monitoring metrics, as shown in Figure 2.3. Log data records which actions are executed by

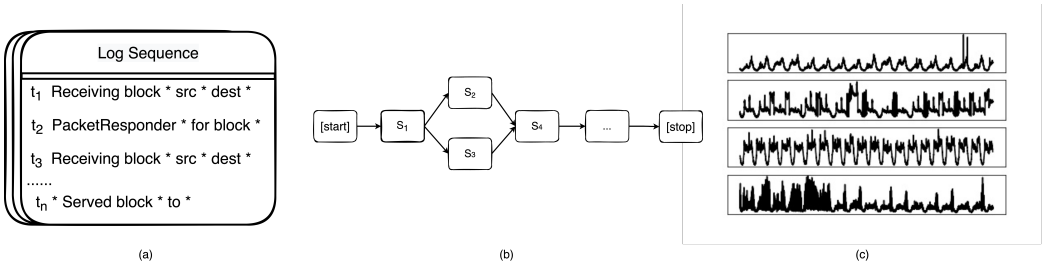


Figure 2.3: Performance data: (a) log data, (b) trace data, (c) monitoring metrics.

the application at runtime. Trace data is a graph-like abstraction built on logs that encode information for the multiple services serving a particular user request [22]. Finally, monitoring metrics are time-series data representing the utilization of the available resources and the status of the infrastructure, typically regarding CPU, memory, disk, network, throughput, and service latency. Generally, monitoring metrics and log data focus on a service or resource level. Trace data relates to the interactions between the different components within a distributed cloud application.

trustworthiness requirements and methods

Performance data collected from distributed applications is important for training models of anomaly detection and root cause localization. However, collected data is usually recognized as a common source of bias for AI systems, and it may affect the results of AI models in a wrong way, e.g., missing data labels can cause poor accuracy of anomaly detection models. Therefore, when using AI models to make critical performance diagnosis, fairness related to data quality is an essential requirement for trustworthy data collection.

Fairness. In performance diagnosis systems, the bias of data collection mainly comes from two aspects: imbalanced data and missing labels. Imbalanced data means that the number of anomalies is minimal compared to normal records in the training dataset of detection models. It is a common situation when collecting performance data because distributed applications run normally most of the time. However, in diagnosis systems, imbalanced data significantly impacts the outputs of anomaly detection methods because most of the data being learned by detection methods is normal. This situation will lead to the suboptimal performance of these methods when detecting anomalies [217]. Furthermore, collected performance data usually does not contain labels that can be immediately used for training AI models, and labeling data is often cumbersome [218]. Therefore, unsupervised learning methods are desirable in performance diagnosis systems, and more research is focusing on this area. However, because supervised learning methods have their advantages in diagnosis performance, it is still meaningful

to manually annotate labels or explore methods for automated labeling. At the same time, fair data annotation is necessary to ensure label quality and avoid mislabeling and missing labels.

The corresponding bias mitigation techniques for the two biases in data collection include data sampling and data annotation. *Data sampling* methods [54] alleviate the class imbalance by adjusting the dataset, and they can be further categorized into two types: undersampling and oversampling [256]. Undersampling removes the majority class in samples from the data, so the degree of imbalance can be adjusted. The mainstream undersampling methods are random undersampling and informed undersampling. Random undersampling removes a set of majority samples from the original data in a random manner [94]. Informed undersampling is to adjust imbalanced data distribution with some mechanisms or assemble training classifiers [149]. In addition, there are other methods for undersampling, for example, cluster-based methods to replace the original majority with new samples [277], and the Tomek Link method pairs samples that are close to each other but have different classes [225]. Oversampling duplicates or generates minority samples to compensate for the lack of minority samples. The mainstream oversampling methods include random oversampling and synthetic oversampling. Random oversampling randomly replicates a set of minority samples from the original data [163]. Synthetic oversampling generates synthetic minority samples using the existing samples. SMOTE [38] and MWMOTE [14] are two representative synthetic oversampling methods.

Data annotation is especially crucial for anomaly detection because anomaly data is the minority of collected performance data. Data annotation has three mainstream methods: manual, crowdsourcing, and active learning [130]. In order to avoid human bias creeping into annotated data, experts must be carefully chosen for manual labelling [128]. Crowdsourcing is an innovative way for data annotation, and it aims to harness workers' knowledge to process machine-hard tasks [33]. In crowdsourcing, requesters split complex tasks into micro-tasks and publish them on crowdsourcing platforms like Amazon Mechanical Turk[1]. The employed workers answer the questions while getting monetary rewards. Task assignment [262, 284], and truth inference [12, 283] are crucial problems for labeling anomalies and root causes in diagnosis systems. In recent years, Active Learning (AL) has been developed to solve artificial intelligence tasks using both machine and human labor [269]. An AL method works by manually labeling a small subset of objects in a dataset and using the labeled data to train a model first. Then it uses a trained model to classify unlabeled data and selects samples with high uncertainties to be labeled manually again. The AL method will iterate these processes until high-quality data labels are obtained. Statistical AL methods and their variants are the most popular [20, 50]. At the same time, deep AL methods have been developing rapidly in recent years, and a comprehensive survey can be

[1]https://www.mturk.com/

found in [185].

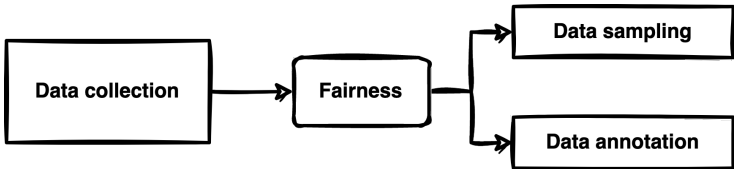


Figure 2.4: Trustworthiness requirements in data collection component

In conclusion, fairness emerges as the primary trustworthiness requirement for the data collection component. To address this requirement, it is crucial to tackle two sources of bias: imbalanced data and missing labels. Upon reviewing literature, we summarize two key solutions: data sampling and data annotation, as shown in Figure 2.4. These approaches aim to mitigate the impact of imbalanced data by adjusting class distributions and address the issue of missing labels through appropriate annotation techniques. By implementing these approaches, we can enhance the fairness of the data collection process and promote a more equitable foundation for subsequent analysis and modeling.

2.3.2 Data preprocessing

Data preprocessing aims to enhance data quality and extract relevant information to improve diagnosis performance, including detection and localization precision. Data preprocessing in diagnosis systems includes several operations, which will be introduced in this section. In addition, in detail, trustworthiness requirements for data preprocessing and related research will be provided.

Data preprocessing methods

In performance diagnosis systems, data preprocessing methods should be appropriate for the collected performance data. We provide several useful data preprocessing methods in performance diagnosis systems based on data types, which will be introduced in detail.

Log parsing. As mentioned in 2.3.1, raw log messages and trace data contain much specific information. For example, log messages have the IP address, file name, etc., and trace data include service names, HTTP paths, etc. At the same time, they are all unstructured data and require log parsing methods [22]. Log parsing to parse each raw log message to extract its log event and parameters has been developed. For example, in Figure 2.3(a), the raw log message is parsed into the log event "Receiving block * src: * dest: *", and there are three different log parameters represented by * in this message. Some research of log parsers only focuses on extracting log events [153, 253]. Yu et al. [263] simplify logs by substituting timestamps with numbers at the beginning of logs, removing

logging levels, replacing identifiers such as IPs, URLs, etc., with simple text, and shortening text in logs. Zhang et al. [273] provide a general log parser approach by learning regular patterns from heterogeneous logs. Log parsing is one of the early steps in log data analysis. Traditional log parsing methods can not extract information accurately [95, 291]. As a result, log parser methods extracting more information for analysis are developed. Du et al. [63] maintains a vector of parameter values, which is used by subsequent training along with the log entry, that contains the elapsed time between a log entry and its predecessor. In addition, ML methods to extract text information from logs are applied. Zhang et al. [276] extract the semantic information of log events and transform each log event into a fixed-dimension vector. Huang et al. [102] design a log sequence encoder and a parameter value encoder to obtain their representations correspondingly.

Time-series data processing. Collected monitoring metrics can be seen as multivariate time-series data, providing valuable insights into the health status of the application [106]. To prepare this data for subsequent modeling, various data preprocessing methods can be employed from temporal and feature dimensions. In the temporal dimension, preprocessing methods to transform the time-series data into meaningful sequence data fragments with classification labels are developed [48]. For instance, specific time windows can be identified as abnormal data, with features such as average, maximum, and minimum values retained. In the feature dimension, techniques like feature selection and extraction are applied to reduce noise and feature dimensions. Feature selection methods (e.g., filter [193], wrapper [67], embedded [155]) aim to choose a subset of relevant features [37]. Feature extraction techniques, such as Principal Components Analysis (PCA) [254] and linear discriminant analysis [239] generate new feature subsets from existing combinations.

Other methods. Except for the above processing for collected performance data, data preprocessing includes other operations such as data cleaning, smoothing, normalization, transformation, and partitioning, as shown in Figure **??**. Data cleaning is to enhance data quality. For example, missing value imputation methods can be applied [49]. In addition, in performance diagnosis systems, prediction-based and reconstruction-based models are sensitive to abnormal samples in training data, which requires outlier detection methods, such as spectral residual [184] to detect outliers and replace them [278]. Data smoothing is to smooth out the random transient noise in collected data. Data denoising methods such as mean and median filters have been developed for many years [88]. In addition, Fast Fourier Transform (FFT) is studied mostly for time-series data. For example, [181] highlighted the areas with high-frequency change by FFT and reconfirmed them with the Z-value test. Data normalization usually normalizes original data to the range of (0,1) with min-max or z-score normalization [78]. Data transformation is to ensure data compatibility with models. One-hot encoding can be applied to extracted log events [230]. The data partition is to divide data into subsets. For example, we need to divide data into training and testing

Table 2.2: Data preprocessing methods

Method	Data type	Description
Log parsing	Log and trace data	Extracting log event and information for further analysis
Time-series data processing	Monitoring metrics	Convert data into sequence data fragments
Data Cleaning	Any	Enhance data quality
Data Smoothing	Any	Smooth out the random transient noise in collected data
Data Normalization	Any	Normalise original data to the range
Data Transformation	Any	Ensure data compatibility with models
Data Partition	Any	Divide data into subsets

sets for supervised learning [36]. In addition, we can define sliding windows to divide time-series data into sequences fragments [218].

As a result, we summarize data preprocessing methods in Table 2.2. For log and trace data, log parsing methods to transform raw data into vector representations are developed. For monitoring metrics, we identify technologies to process multivariate time-series data, such as feature selection and extraction. In addition, we introduce general data preprocessing methods, such as data cleaning and smoothing.

Trustworthiness requirements and methods

Data preprocessing converts original data into representations that can be used for subsequent model training through a series of operations. In addition, data preprocessing is vital for improving diagnosis performance, such as accuracy, efficiency, and robustness. To build trustworthy diagnosis systems, robustness and explainability are two crucial requirements.

Robustness. Distributed applications are often deployed in complex environments, resulting in the presence of multiple data distributions and diverse features within the collected performance data [211]. Consequently, applying pre-trained diagnosis models to different situations becomes challenging, leading to a lack of robustness in their performance. To enhance robustness, one possible approach is to consider enriching the data during the data preprocessing stage to encompass a wider range of situations. This can help improve the overall performance of the system and ensure its ability to handle diverse scenarios effectively.

In the data preprocessing component, two technologies can be considered to improve the robustness of diagnosis models: data augmentation and adversarial attack. Data augmentation is a series of strategies for enlarging and enhancing data size while maintaining labels. *Data augmentation* can minimize overfitting and improve model robustness and generalization. Data augmentation is a common practice in image recognition with neural networks [203]. Data augmentation techniques for text and time-series data (corresponding to log data, trace data, and monitoring metrics) generally use random transformations to augment the training data, same as for images [109, 129]. For example, flipping or jittering to generate more data [182]. In addition, for time-series data, random warping

in the time dimension [231], and frequency warping [110] can also be used. Because random transformation is not applicable to diverse amounts of data with each feature having different properties, techniques to synthesize new data with information inherent to the data have been developed, such as pattern mixing, generative models, and pattern decomposition methods [109]. Pattern mixing is mixing existing patterns of time-series data and creating new samples with features from both patterns. Generative models use the distributions of features in data to generate new samples. Statistical models such as Gaussian trees [31] and hidden Markov models [257] have been proposed. In addition, generative models with neural networks for time series generation have recently become popular, such as generative adversarial networks [81]. Data decomposition methods extract features from the dataset, such as trend components in time-series data [18], to generate new patterns. The advantage of these data augmentation methods is that they preserve the distribution of time series in the dataset. In addition, for log data, Han et al. [89] propose the robust online evolving anomaly detection framework, which adopts natural language processing to remove the effects of noise and uses online learning theory to update parameters dynamically.

The concept of *adversarial attack* was first proposed by [221] for image recognition. The main idea is to generate adversarial examples by adding small, subtle perturbations to input data, causing ML models to give a false prediction with high confidence. Based on this idea, many researchers have developed algorithms for constructing such adversarial examples, relying on the architecture and parameters of the deep learning model [82, 160]. There have been some studies about the adversarial attack on time-series data in recent years. The sensitivity of time-series data to adversarial perturbations has not been considered, unlike images. Oregi et al. [174] adopt a soft KNN coupled with dynamic time warping to generate adversarial examples. Karim et al. [118] propose using an adversarial transformation network to attack various prediction models. Harford et al. [91] propose transforming the existing adversarial transformation network onto a distilled model to attack various multivariate time-series prediction models. There is limited adversarial attack research for deep time-series prediction models, such as long- and short-term time-series network and recurrent neural network [127]. Fawaz et al. [69] utilize the fast gradient sign method and basic iterative method attacks to attack residual network classifiers for univariate time-series data.

Explainability. Explainable AI plays a crucial role in performance diagnosis systems as it fosters a deeper understanding of the diagnostic results and can be helpful to enhance public trust. Furthermore, explainability provides valuable insights into improving the performance of diagnosis systems. As data forms the foundation of dignosis system, gaining a comprehensive understanding of the original data is essential. In data preprocessing component, various operations can be applied to original data to enhance the explanation of performance data.

Although different data types exist in performance data, numerous studies have proposed approaches that can be applied to enhance data explainability.

These approaches can be broadly categorized into three aspects: data visualization, feature analysis, and feature importance. *Data visualization* is to better understand the data before using it [208]. Data visualization systems have been developed for different data types. For example, ELK stack[2] (ElasticSearch, Logstash, and Kibana) is convenient for storing and visualizing logs, traces, and metrics. In addition, Prometheus[3] and Grafana[4] are more suitable for collection and visualization of time-series metrics.

Except for visualization of collected performance data, *feature analysis* before feeding them into training models can provide more understanding of data distributions. For logs and traces, many recent studies, e.g., [63, 96, 143], as well as industrial solutions, e.g., Splunk [5], ELK, Logentries[6], have evolved to provide powerful text search and analytics capabilities. For time-series data, analysis for single variate mainly includes determining stationarity, seasonality, and autoregressive character of data [204]. At the same time, it is possible to perform feature selection or extraction for multivariate time-series data to understand training models better.

Feature importance [243] technologies for explainable AI can be divided into preprocessing and in-processing. Preprocessing feature importance is based on feature selection methods [37]. Filter methods score each feature, use scores as weights to represent the importance of features, and then sort them according to weights [193]. Wrapper methods regard selecting features as a search optimization problem, generating different combinations, evaluating these combinations, and comparing them [67]. Embedded methods use some ML algorithms and models for training first [155]. After obtaining the weight coefficients of each feature, features are ordered according to their coefficients.

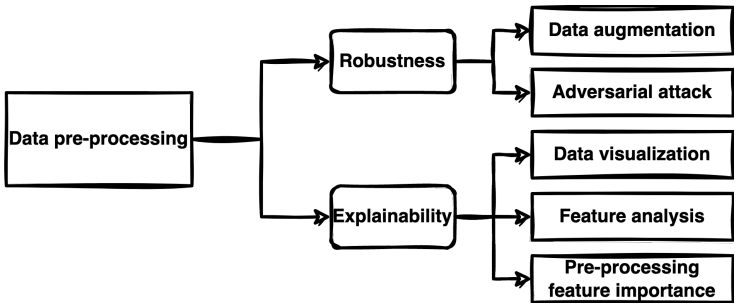


Figure 2.5: Trustworthiness requirements in data preprocessing component

As a result, robustness and explainability are essential trustworthiness requirements for the data preprocessing component, as illustrated in Figure 2.5. To

[2]https://www.elastic.co/elastic-stack/
[3]https://prometheus.io/
[4]https://grafana.com/
[5]http://www.splunk.com/
[6]https://logentries.com/

enhance robustness, data augmentation techniques and adversarial attacks can be employed as part of the data preprocessing stage to enrich the original data. On the other hand, to improve explainability, techniques such as data visualization, feature analysis, and assessing the importance of pre-processed features can be utilized to gain a deeper understanding of the data.

2.3.3 Performance anomaly detection

Performance diagnosis is detecting abnormal performance phenomena, e.g., degradation, predicting anomalies to forestall future incidents, and localizing the causes of performance anomalies. Research for anomaly detection has been developed for many years. This section will introduce anomaly detection methods and discuss related research on trustworthiness requirements for anomaly detection.

Anomaly detection methods

In general, performance anomalies are associated with anomalous indicators that point to service or infrastructure issues. For example, anomalous indicators of resource consumption can suggest a burst of service workloads. After preprocessing, collected performance data can be represented as a feature matrix, and many different detection methods have been developed, and ML methods are widely applied [26, 98, 180]. In this section, we will mainly introduce ML-based anomaly detection methods in supervised, semi-supervised, and unsupervised learning.

Supervised learning methods. Supervised learning uses labeled data to train models and has been developed for several years. For example, Hu et al. [98] defined six meta-features to statistically describe the local dynamics of multivariate time-series sequences and used the Support Vector Machine (SVM) to detect the anomalies. Ren et al. [184] propose an algorithm based on spectral residual and CNN for a time-series anomaly detection service that helps customers monitor the application continuously and alert for potential incidents on time. The advantage of supervised learning is that the accuracy of detection is relatively higher than that of unsupervised learning in most cases. However, labeling performance anomalies requires human experts and is time-consuming in reality.

Unsupervised learning methods. Unsupervised learning methods are developed, considering there are usually no labels in reality. We provide a classification of unsupervised performance anomaly detection methods, as shown in Table 2.3. The density-based such as Local Outlier Factor (LOF) [26], distance-based such as (K-Nearest Neighbor) KNN [180], kernel-based such as One-Class Support Vector Machine (OCSVM) [197], and ensemble-based methods such as Isolation Forest (IForest) [146] are most used and usually focus on different features in data. The detection performance of these methods varies greatly for different datasets because they focus on different characteristics in data [250]. In addition, many unsupervised deep detection methods are developed based on neural

Table 2.3: Machine learning-based anomaly detection methods

Model Classification	Required data	Model
Supervised	With labels	SR+CNN [184] etc.
Unsupervised	No labels	Density-based [26] Distance-based [180] Kernal-based [197] Ensemble-based [146] Neural network [122, 192]
Semi-supervised	Few labels	SLA-VAE [104] etc.

network [122, 192]. Su et al. [218] provided a stochastic recurrent neural network named OmniAnomaly for various devices' multivariate time series anomaly detection. The method utilizes VAE to reconstruct input data and uses the reconstruction probabilities to determine anomalies. Audibert et al. [11] proposed an unsupervised anomaly detection method called USAD for multivariate time series IT system monitoring data. USAD is designed based on an adversely trained autoencoder, which allows it to isolate anomalies while providing fast training compared to other deep learning methods. Deep detection methods usually have good accuracy, but the model training is time-consuming.

Semi-supervised learning methods. Semi-supervised learning can train a model using data when only a few labels exist. Camacho [30] et al. presented a semi-supervised approach for anomaly detection. The method extends the unsupervised multivariate statistical network monitoring approach based on Principal Components Analysis (PCA) [254] by introducing a supervised optimization technique to learn the optimum scaling in the input data. Huang et al. [104] present SLA-VAE, a semi-supervised learning anomaly detection framework using Variational Autoencoder (VAE) [7]. The model uses semi-supervised VAE to identify anomalies in multivariate time series and employs active learning to update the online model via a small number of uncertain samples. Semi-supervised methods are more practical in real distributed applications and perform better than unsupervised detection methods.

In summary, we classify anomaly detection methods into supervised, unsupervised, and semi-supervised based on the requirement for data labels, as shown in Table 2.3. Supervised models train models with labels, which can achieve the best detection accuracy, but labeling data manually is time-consuming and impractical. Unsupervised learning methods work without labels, which can be used broadly, and research is more focused on improving detection accuracy and robustness. While semi-supervised learning methods can train models with fewer labels and achieve good detection performance, which are the most practical.

Trustworthiness requirements and methods

Anomaly detection is a crucial component in performance diagnosis systems. Many studies on performance anomaly detection methods have been carried out,

mainly aiming to improve the diagnosis performance, such as accuracy and robustness. In addition, fairness and explainability are necessary trustworthiness requirements for anomaly detection methods.

Fairness. The fairness requirement for anomaly detection mainly focuses on the imbalanced data issue. Besides data sampling methods in Section 2.3.1, cost-sensitive has also been developed for processing imbalanced data. *Cost-sensitive* methods improve the detectors by applying different costs for misclassifying samples [124]. For cost-sensitive learning-based methods, determining an optimal cost representation is essential. Nikolaou et al. [171] incorporate the shifted decision threshold and calibrated probability estimation for cost-sensitive learning in imbalanced data classification. Wu et al. [114] use cost-sensitive multi-set feature learning to learn discriminant features. Yin et al. [256] use the cost for both positive and negative output calculations and obtain a weighting mechanism to maximize the cost.

Robustness. In performance diagnosis systems, achieving model robustness is crucial as it entails reduced sensitivity to minor changes in the underlying data distributions and the ability to adapt to diverse scenarios. To ensure model robustness, it is imperative to design algorithms that can effectively identify and accommodate different data distributions. Furthermore, numerous defense approaches have been proposed to mitigate the impact of adversarial attacks on the system, enhancing models resilience and reliability. Robust models enable diagnosis systems to maintain consistent performance across varying conditions and enhances its reliability in diagnosing performance issues.

Existing research to enhance model robustness can be categorized into: robust representation, ensemble learning, and adversarial defense. Some research focuses on learning *robust representation* of input data for robust models designed. For example, Su et al. [218] propose OmniAnomaly for multivariate time-series anomaly detection. The model maintains robustness by learning multivariate data representations and using reconstruction probability to identify abnormalities. Zhao et al. [278] also provide a robust model for multivariate time-series data by learning a distribution of stochastic variables, which is more robust to perturbations and noise. Zhang et al. [276] provide a log-based anomaly detection method using an attention-based bidirectional Long Short Term Memory (LSTM) model. The model uses the semantic vector transformed by log data and can handle similar but unstable log events. Zhao et al. [281] propose a robust anomaly detector containing two layers: identifying the log data quality and classification for anomaly detection. The detector is robust to unreliable datasets with label noise.

Ensemble learning combines several base models and produce an optimal predictive one [93, 290]. The idea is that errors of a single model will be compensated by other models. It can be used to reduce variance and improve the accuracy and robustness of anomaly detection models [73]. Ensemble learning methods can be classified into supervised, semi-supervised and unsupervised for anomaly

detection [61]. As for supervised ensemble learning, Tyralis et al. [229] propose an ensemble learning method by combining ten ML algorithms. The weights are estimated through a k-fold cross-validation procedure in the training set, and a properly selected loss function is minimized. Tama et al. [223] proposed a stacked ensemble for anomaly-based intrusion detection systems in a web application. They use three classifiers (random forest, gradient boosting machine, and XGBoost) and provide a generalized linear model as a combiner. Semi-supervised ensemble learning mainly focuses on expanding the training set and utilizing expanded training sets. Yu et al. [265] proposed a multi-objective subspace selection process to generate the optimal combination of feature subspaces and an auxiliary training set based on the sample confidence to improve the performance of the classifier ensemble. Sjoerd et al. [55] presented a reliable semi-supervised ensemble learning method to exploit unlabeled data to generate diverse classifiers through self-training and combine these classifiers into an ensemble for prediction. For unsupervised ensemble learning, research mainly focuses on consensus clustering. Ensemble clustering can be classified into three categories [101], pair-wise co-occurrence based methods [71], graph partitioning based methods [99] and median partition based methods [100]. Clustering ensemble algorithms have their advantages in improving clustering accuracy and robustness. The disadvantage of these unsupervised ensemble learning methods is that they are unsuitable for large-scale applications due to the efficiency bottleneck.

For model robustness, we discussed the adversarial attack in Section 2.3.2, which adds small perturbations to the data. The key to improving model robustness lies in *adversarial defense* mechanisms. Adversarial training, as proposed by [82], incorporates data perturbed with an adversarial attack into the training set alongside their correct labels. For unsupervised learning, Goodge et al. [83] propose the approximate projection autoencoder, which incorporates two defenses, approximate projection, and feature weighting, into a general autoencoder to improve robustness under adversarial attacks. Lo et al. [152] develop the principal latent space method by updating latent embeddings, and the model purifies the latent embeddings of the autoencoder based on PCA. Tuli et al. [228] provide a deep transformer network-based anomaly detection method, and it includes a two-phase adversarial training for improving model generalizability and robustness to diverse input.

Explainability. In AI-based performance diagnosis systems, models are often perceived as "black boxes," which show difficulties in understanding their decision-making processes and reasoning. Interpreting these models is crucial as it allows us to gain insights into their inner workings and design more effective models. However, achieving model explainability is a complex task because of numerous parameters and intricate interactions within the models. Nevertheless, ongoing research is focusing on addressing these challenges to enhance our understanding of these models and improve their overall performance.

Different studies have been developed to achieve explainable AI, and we

can classify them into self-explainable models and feature importance. *Self-explainable* models are human-understandable because interpretability is built into architectures. Self-explainable models are widely used for anomaly detection. For example, the auto-regressive integrated moving average model [166] and GAM [60] can be used for time-series anomaly detection. The tree-based model IForest [146] has good detection performance, and its outputs are easy to understand. In addition, the bayesian model [213] can use probability to represent all uncertainty within the model, which provides a clear interpretation. Some models, except those with built-in explainable architecture, are designed to provide interpretation by incorporating explainable components into original models. For example, Nguyen et al. [170] propose a gradient-based explainable VAE for anomaly detection. Its explainability is achieved by analyzing the gradients contributed by each feature of the data point. Aguilar et al. [5] propose a decision tree-based autoencoder for detecting anomalies. It can explain its results by finding the correlations between all attributes.

In-processing feature importance is to explain model results by analyzing features. Lundberg et al. [156] propose SHAP, which explains predictions by calculating feature importance and uses Shapley values from game theory to ensure consistency of the explanations. DeepLIFT [205], a method for decomposing the output prediction of a neural network on a specific input by back-propagating the contributions of all neurons in the network to every feature of the input. LIME [187] is a model-agnostic method. It calculates the feature importance of samples with local self-explainable models. SAGE [52] provides global interpretation, and it is also a model-agnostic method that quantifies model-based and universal predictive power while accounting for feature interactions. Huang et al. [105] provide DeepExplainer, which outputs the significance of features based on feature importance and uses the significance to explain detection results.

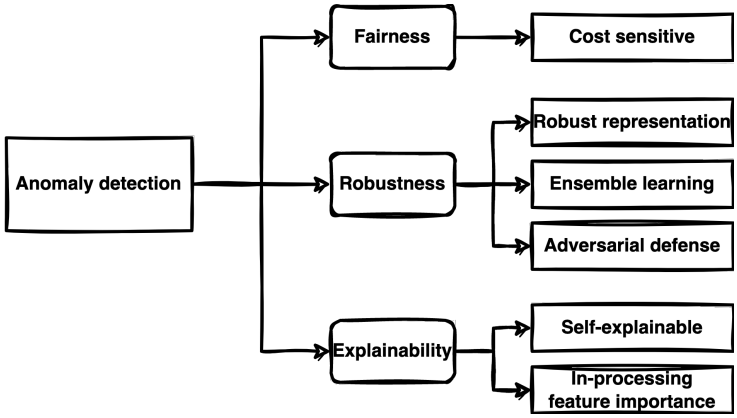


Figure 2.6: Trustworthiness requirements in anomaly detection component

In conclusion, trustworthiness requirements for anomaly detection encompass fairness, robustness, and explainability, as shown in Figure 2.6. To address the is-

sue of imbalanced data and promote fairness, cost-sensitive techniques have been developed. Robustness is crucial in the presence of diverse data distributions within performance data. Approaches such as robust representation learning using deep neural networks, ensemble learning methods to reduce model variance, and adversarial defense mechanisms against adversarial attacks have been developed. To enhance explainability, self-explainable models like tree-based models and in-processing feature importance techniques have been introduced. These advancements contribute to the overall trustworthiness of anomaly detection systems.

2.3.4 Root cause localization

For distributed applications with detected performance anomalies, root cause localization to determine the causes of anomalies can help enable rapid recovery and loss mitigation. In this section, we first introduce research about root cause localization. Then, we will discuss trustworthiness requirements and methods for root cause localization.

Root cause localization method

In recent years, research has been developed for root cause localization in clouds [240] [247]. Based on data types, we can categorize these studies into three groups: log-based, trace-based, and metric-based [209].

Log-based research. Log-based root cause localization is mainly based on log data with multi-dimensional attributes after log parsing. The root cause can be one or more combinations of attribute values in multiple dimensions, which means the major challenge for root cause localization is the huge search space for potential root causes. Various techniques to reduce the search space for multi-dimensional root cause localization have been proposed. Bhagwan et al. [21] propose Adtributor assuming that the root cause only relies on one attribute. The Adtributor relies on forecasting for attribute combinations in all one-attribute cuboids to calculate explanation power and surprise. Lin et al. provide [142] iDice, which uses isolation power to measure the degree of separation between abnormal and normal values within an attribute combination. Sun et al. [220] use a novel potential score based on the ripple effect for anomaly propagation and adopt the Monte Carlo tree search algorithm and a hierarchical pruning strategy to determine the root cause in multi-dimensional attribute space. Li el at. [133] propose SwissLog, which determines the root cause with an ID relation graph built based on ID information in log data.

Trace-based research Trace-based root cause localization works with call graphs between services and mainly localizes root causes at the service level. Zhou et al. provide MEPFL [288], which trains a supervised ML model to predict the root-cause microservices with a training corpus built by fault injection. Liu

et al. propose TraceAnomaly [148], which focuses on detecting structural or latency anomalies in traces. Li et al. provide TraceRCA [135], which suggests an insight that a service with more abnormal traces and fewer normal traces passing through is more likely to be the root-cause service. They use a unified metric to measure the insight of each service. Yu et al. propose TraceRank [261], which uses spectrum analysis and the PageRank-based random walk methods to pinpoint abnormal services.

Metric-based research. Based on monitoring data, some researchers identify root causes of performance anomalies with statistical analysis. Want et al. [234] conduct correlation analysis based on mutual information to determine the root-cause metric for the anomalies they detect. However, given that correlation does not ensure causation [29], statistical analysis can not pinpoint root causes. In addition, some researchers have developed a topology graph-based analysis technique that reconstructs the topology graph of a running application. For example, Wu et al. [247] generate a topology graph based on deployment information and extract a weighted anomalous subgraph by parsing resource-level monitoring metrics. Brandón et al. [25] make snapshots of abnormal states of the application as graphs and then identify the root cause of a new anomaly by graph matching. This research uses a reconstructed application topology graph to determine root causes, which can only be used for coarse-grained root cause localization. Root cause localization can be explored at two granularities: coarse-grained and fine-grained. Coarse-grained means that only faulty services can be identified. Fine-grained is defined as identifying both the faulty service and the root cause metric of the service, which can help operators choose accurate actions to mitigate performance anomalies [248].

Existing CI-based root cause localization research works by constructing a causal graph based on monitoring data, i.e., including causal structure learning and root cause inference [209], as shown in Table 2.4. Coarse-grained root cause localization usually builds a causal graph based on service level objective (SLO) metrics, such as service latency, and focuses on determining faulty services. For example, Microscope [85, 141] collect information on service interactions and monitoring service latency and then processes them based on Peter-Clark (PC) and Breadth First Search (BFS) algorithms to determine possible faulty service of detected anomalies. In addition, CloudRanger [235], MS-Rank [157, 158], and AutoMAP [159] all exploit PC and random walk algorithms to build causal graphs and infer root causes. MS-Rank and AutoMAP use metrics not only service latency but also throughput, power, and resource consumption, whereas AutoMAP develops novel operations to refine the causal graph. Various coarse-grained root cause localization studies have been conducted, but they cannot assist operators in resolving application anomalies with accurate actions.

To address the drawback of coarse-grained root cause localization, some researchers focus on fine-grained root cause localization. Chen et al. first proposed CauseInfer [41, 42], which infers the faulty service and root cause metric by con-

Table 2.4: Classification of metric-based root cause localization research

Reference	Year	Causal structure learning	Root cause inference	Input	Root cause	Granularity
Micorscope[85, 141]	2018	Parallelized PC	BFS	Service latency	Faulty service	Coarse-grained
CloudRanger[235]	2018	PC	Random walk	Service latenct	Faulty service	Coarse-grained
MS-Rank[157, 158]	2019	PC	Random walk	Multi metrics	Faulty service	Coarse-grained
AutoMAP[159]	2020	PC	Random walk	Multi metrics	Faulty service	Coarse-grained
CauseInfer[41, 42]	2014, 2016	PC	DFS	Service latency	Faulty service	Fine-grained
MicroCause[165]	2021	PCTS	Random walk	Resource metrics	Root cause metric	Fine-grained
MicroDiag[246]	2021	DirectLiNGAM	PageRank	Service and resource metrics	Root cause metric	Fine-grained

structing a causality graph of monitoring metrics in each service with the PC algorithm and traversing the metric causality graph with a Depth First Search (DFS) method. After several years, Meng et al. provided MicroCause [165], which mainly focuses on the root cause metric localization in a faulty service. It provides a PC-based causal graph building method for time-series data and infers root causes with the random walk method. Afterward, Wu et al. proposed MicroDiag [246], which focuses on fine-grained root cause localization and applies a direct Linear Non-Gaussian Acyclic Model (LiNGAM) to build causal graphs and PageRank to infer root causes. Fine-grained root cause localization focuses mainly on SLO metrics and monitoring resource metrics of services and determining the root cause resource metric to help operators take actions like scaling resources [227].

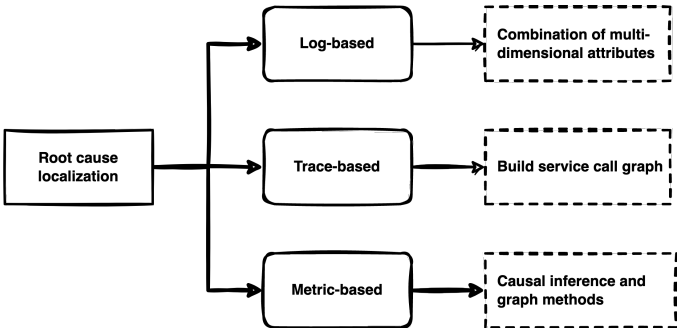


Figure 2.7: Root cause localization methods

In conclusion, we summarize root cause localization methods in Figure 2.7. Log-based research [4] mainly localizes root causes based on text log parsing.

Trace-based research [121, 240] gathers information through complete tracing of the execution paths and then identifies root causes along those paths. In addition, metrics-based research uses monitoring data collected from applications and underlying infrastructures to construct causal graphs and infer root causes.

Trustworthiness requirements and methods.

Root cause localization research is mainly based on causal inference and graph methods. Several studies about robust root cause localization for log data are proposed for trustworthiness requirements. Sun et al. [220] achieve robustness by using a novel potential score based on the ripple effect for anomaly propagation. Li et al. [136] propose a generic and robust method called Squeeze, which combines bottom-up and top-down strategies with a generalized ripple effect method. Furthermore, Li et al. [134] propose a novel probabilistic clustering method called PSqueeze to reduce the influence of noise and improve model robustness. These studies mainly focus on developing novel robust approaches based on the *ripple effect*. On the other hand, for *causal inference* and graph-based root cause localization on monitoring metrics, there is little research about model robustness. For explainability, root cause localization can be seen as the explanation of detected methods. In addition, root cause localization based on causal inference is inherently interpretable and easy to understand. In conclusion, trustworthiness requirements for root cause localization component is currently being developed with a focus on ensuring robustness through the incorporation of ripple effects, as well as enhancing explainability through the use of causal inference techniques, as shown in Figure 2.8.

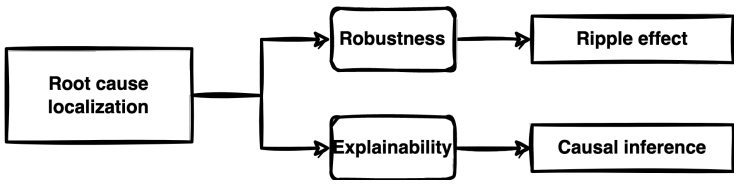


Figure 2.8: Trustworthiness requirements in root cause localization component

2.3.5 System trustworthiness requirements

In addition to the trustworthiness requirements for each component within the diagnosis system, there are additional considerations that apply to the system as a whole. One such requirement is data privacy, which is a crucial aspect throughout the entire process of performance diagnosis and can be viewed as an "in-the-loop" requirement. Furthermore, human intervention, referring to the involvement of external humans, can occur at various components and can be

considered an "over-the-loop" requirement. This section will introduce the two requirements and related research in detail.

Data privacy

Having massive amounts of data available to develop diagnosis algorithms is a great benefit. However, it also provides an avenue for compromising the security of AI models built from the data by providing an additional vector for hackers to attack. Data governance is about allocating authority and control over data and the exercise of such authority through decision-making in data-related matters [24]. Data privacy is vital in data governance and should focus on the diagnosis system through which data is collected, managed, and used. We mainly focus on DP and FL for privacy-preserving methods in diagnosis systems.

Blockchain-based data storing. Blockchain provides an easy-to-use platform for distributed data storage and protection [186]. Using blockchain, a group of users, also known as miners, create blocks used to validate and record transactions. Transactions in performance diagnosis systems can be collected through logs and monitoring metrics. With blockchain technologies, storing and protecting a large volume of data has been explored. Li et al. [131] present a clear definition of the transactions in a non-cryptocurrency system and illustrate how the transactions are processed in Internet of Things (IoT) data. Zhou et al. [287] design a blockchain-based decentralized IoT system in which anonymity and the amount of confidentiality of blockchain provide strong privacy for users and devices to hide sensitive data. Liang et al. [140] provide secure data storage based on blockchain technologies. A recovery scheme in the blockchain-based network is provided by improving the decentration, tampering-proof, real-time monitoring, and management of storage systems. Liu et al. [145] combine blockchain into edge computing to provide more secure data storage and transmission, supporting tamper resistance and traceability for the IoT.

Differential privacy. DP can be used to hide certain input data from the output [62]. In other words, when looking at the statistical results calculated from input data, one cannot determine whether the input data contain a certain record. DP is achieved by adding random noise to the input data or data analysis procedure so that the input difference can be hidden by the noise [144]. The performance data in diagnosis systems has little sensitive user information, so we can focus on using DP framework to obfuscate the input data or apply it to models. Dwork et al. [66] define the typical DP as ϵ-DP, and it measures how well a randomized statistical function on a dataset reflects whether an element has been removed. For obfuscating input data, Zhang et al. [274] propose an obfuscate function and apply it to the training data before feeding them to the model training task. There are studies about ML with DP. Rubinstein et al. [190] suggest privacy-preserving mechanisms for SVM learning, which add noise to the output classifier and yield close approximations to non-private SVMs. Song et

al. [210] propose private stochastic gradient descent for general convex objectives and validate the approach's effectiveness using logistic regression for classification. Abadi et al. [2] introduce a simpler DP stochastic gradient descent algorithm, which adds DP noise to the gradients, and the whole training process involves multiple iterations. In addition, Wei et al. [238] propose a framework combining the concept of DP into FL to prevent information leakage effectively.

Federated learning. FL is popular in academia and industry as a solution to collaborative model training tasks using data from multiple parties [128]. It is designed to address data privacy issues that prevent ML algorithms from properly using multiple data types. Existing FL algorithms can be categorized into horizontal FL, vertical FL, and federated transfer learning algorithms [255]. Horizontal FL describes the situation in which each party has different samples, but the samples share the same feature space. In horizontal FL, it is common for all parties to calculate and upload local gradients so that the central server can aggregate them into a global model [270]. Methods like homomorphic encryption [9] and differential privacy [66] are used to ensure the security of switching gradients in horizontal FL. Many ML models, such as the logical regression model [76], tree structure model [154], and neural network model [189] based on horizontal FL, have been gradually developed. Federated transfer learning describes the condition in which none of the parties overlaps in either the sample or the feature space [270]. With federated transfer learning, knowledge can be shared without compromising user privacy, and complementary knowledge can be transferred between domains in a data federation, enabling a party in a target domain to leverage rich labels from a source domain to build flexible and effective models [150].

Several studies have examined federal learning to preserve privacy in performance diagnosis systems. Cui et al. [53] introduce a FL framework based on the blockchain for anomaly detection. This framework is based on FL and blockchain, using these technologies' traits to ensure the system's privacy and robustness. Liu et al. [151] introduce an FL framework for time series IoT data. Aside from federal learning, some researchers propose mechanisms for processing encrypted or securing data. Pei et al. [178] propose a personalized federal anomaly detection framework FedPAD for cellular traffic data that aggregates data from different organizations while protecting privacy and security.

Human intervention

Human intervention, referred to as human-in-the-loop or interactive ML [249, 269], such as labeling data or participating in decisions, has been applied to various AI systems to compensate for limited performance. Unsupervised deep learning methods are being developed rapidly, considering fewer labels exist in reality. However, supervised learning needs less data and has better performance than unsupervised learning. **Data annotation** and active learning have been

discussed in 2.3.1. Human work for data preprocessing, such as parsing the source files and generating the logs, is necessary. With limited labels, Zhou et al. [289] propose a deep, weakly supervised anomaly detection method that leverages the autoencoder to fit the normal data and then extracts a feature representation.

In addition to data annotation, humans are necessary for **hyper-parameter tuning** for deep learning methods. For deep learning methods in diagnosis systems, hyper-parameters, such as network layer, epoch, and learning rate, are determined and tested by human [198]. Automated ML [236] is developing to reduce the demand for human experts, but it still has a long way to go. Furthermore, **human intervention** provides feedback for trained AI models is vital for compensating for decision-making. Xu et al. [252] update evaluation results based on human judgment, which makes users feel satisfied and confident about their detection results. Ding et al. [59] propose a collaborative, contextual bandit algorithm named GraphUCB for attributed networks. They consider improving detection performance by integrating feedback from human experts into the model to update its selection strategy in the next round. Siddiqui et al. [206] develop a human-in-the-loop anomaly detection system where an analyst can provide direct feedback to the unsupervised anomaly detector. Duan et al. [64] use the Q-learning algorithm to build the core part of the anomaly detection model and provide a feedback mechanism to update the detection model and the abnormal level of abnormal logs. In addition, there is less research on human intervention in root cause localization. However, revising causal graphs built by algorithms based on human knowledge can be considered to improve localization performance.

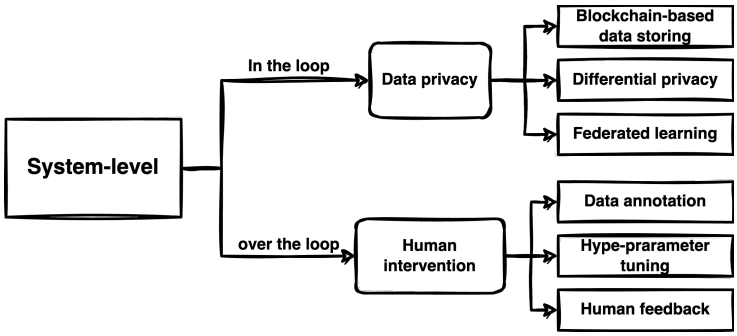


Figure 2.9: Trustworthiness requirements in system-level

In conclusion, we identify data privacy and human intervention as two trustworthiness requirements that need to be ensured for the entire diagnosis system. For data privacy, technologies such as blockchain-based data storing, differential privacy, and federated learning to protect data privacy from data collection to model development have been developed. For human intervention, data annotation for original data, hyper-parameter tuning during model training, and human feedback to improve diagnosis performance can be considered. These technolo-

Table 2.5: Representative papers of methods for building trustworthy performance diagnosis systems

System component	Trustworthiness requirement	Method	Reference
Data collection	Fairness	Data sampling	[14, 38, 54, 94, 149, 163, 225, 256, 277]
		Data annotation	[20, 33, 50, 128, 130, 185, 269]
Data preprocessing	Robustness	Data augmentation	[18, 31, 81, 89, 109, 110, 182, 231, 257]
		Adversarial attack	[69, 82, 91, 118, 160, 174, 221]
	Explainability	Data visualization	[208]
		Feature analysis	[63, 96, 143, 204]
		Preprocessing feature importance	[37, 67, 193, 243]
Anomaly detection	Fairness	Cost sensitive	[114, 124, 171, 256]
	Robustness	Robust representation	[218, 276, 278, 281]
		Ensemble learning	[55, 61, 101, 223, 229, 265]
		Adversarial defense	[82, 83, 152, 228]
	Explainability	Self-explainable	[5, 60, 146, 166, 170, 180, 213]
		In-processing feature importance	[52, 105, 156, 187, 205]
Root cause localization	Robustness	Ripple effect	[134, 136, 220]
	Explainability	Causal inference	[46, 201, 202, 214, 264, 282]
In-the-loop	Data privacy	Blockchain-based data storing	[131, 140, 145, 186, 287]
		Differential privacy	[2, 62, 66, 144, 190, 210, 238, 274]
		Federated learning	[53, 128, 151, 178, 255, 270]
Over-the-loop	Human intervention	Data annotation	[20, 33, 50, 128, 130, 185, 269, 289]
		Hype-prarameter tuning	[198]
		Human feedback	[59, 64, 206, 252]

gies contribute to the overall trustworthiness of the diagnosis system by preserving privacy and leveraging human expertise throughout the diagnosis process.

In summary, we provide the Table 2.5 to represent methods and related papers for building a trustworthy performance diagnosis systems. Different data types exist in the data collection component, which requires fairness because there are problems with imbalanced data and fewer labels. For data preprocessing, data augmentation and adversarial attacks to guarantee the robustness of diagnosis models are crucial. In addition, feature processing methods will be helpful for model interpretation. Anomaly detection models need to focus on fairness, robustness, and explainability. Meanwhile, root cause localization explains detected anomalies, and localization model robustness is also explored. Data privacy should be assured through data to model in diagnosis systems. Besides, human intervention is necessary for diagnosis systems and can play a role in data annotation, hyper-parameter tuning, and feedback. This section will provide a detailed introduction to each component in diagnosis systems, combining related research and trustworthiness requirements.

2.4 Future research directions and challenges

Lots of progress has been made in trustworthy performance diagnosis systems. However, some issues and challenges in this field still need to be addressed. Therefore, the following research directions have been identified from existing literature.

Enhance performance data quality. High-quality labelled data can im-

prove detection and localization accuracy in performance diagnosis systems. However, in real scenarios, obtaining a large amount of labelled data can be challenging, and data fluctuations are caused by unstable cloud environments, resulting in irregular patterns. To address these issues, weakly-supervised learning with ML methods which leverage limited labelled data and provide promising performance, has drawn more attention in academia and industry [113]. In addition, effective data preprocessing and cleaning are essential to ensure data quality and remove noise or irregularities that could undermine the performance of diagnosis systems.

Robust and accurate anomaly detection. Anomaly detection has been researched for several years, and many detection methods have been developed. Existing research mainly focuses on improving detection accuracy for performance anomaly detection of distributed applications. However, practical anomaly detection should also consider model robustness. Dynamic cloud environments create diverse anomalies and data patterns, making robust anomaly detection methods crucial for consistent detection performance. Several studies about model robustness have been developed. However, these methods may compromise detection accuracy [272]. Therefore, more studies to improve model robustness and accuracy need to be addressed in the future.

Precise and fine-grained root cause localization. Root cause localization is essential to explain detected anomalies. Research on root cause localization for distributed applications has recently attracted attention. Accurate localization is essential to help operators take action. In addition, most metric-based methods focus on service-level localization, while fine-grained localization identifying both the faulty service and metric can be more helpful for fast recovery of cloud application [159]. However, metric-level localization faces challenges such as multiple metrics and complex dependencies. As a result, we believe that root-cause localization methods can be explored further in terms of precise and fine-grained pinpointing in the future.

Secure data governance for data and models. A vast amount of performance data can be collected in distributed applications, containing lots of application and infrastructure-related information. When applying this data to a performance diagnosis system, it is necessary to consider implementing secure data governance from data collection to model training. Collected performance data, primarily stored in centralized or distributed environments, is highly likely to be attacked or stolen [200]. Blockchain-based data storage to encrypt data and avoid data tampering can be considered to solve this problem. However, the inefficiency of blockchain-based storage is a big issue because data synchronization is time-consuming and requires further research.

For model training, FL [255] has been developed in recent years. FL can overcome data transmission delays while maintaining data privacy, considering that performance data collected from distributed applications is distributed across different regions and infrastructures. Therefore, FL is suitable for secure data

governance in performance diagnosis systems. However, it has many challenges to be addressed in the future. For example, FL models have low efficiency, especially when ML models have many parameters. In addition, it is not easy to coordinate multiple devices and heterogeneous collected data in FL models.

Automatic operations after diagnosis. To ensure the running of distributed applications, recovery from performance anomalies is essential. Diagnosis results can be used for recovery solutions such as scaling virtual machines or migrating services. However, selecting the best operation is challenging because different operations exist, and multiple constraints must be considered. For example, it is vital to recovering from anomalies in real-time before they are discovered by users [6]. In addition, an automatic operation system that combines monitoring, diagnosis, and recovery for intelligent performance management of distributed applications is also worth exploring.

Meet ethical trustworthiness requirements. The EU provides seven trustworthiness requirements, while this chapter mainly focuses on five technical trustworthiness requirements for performance diagnosis systems, data privacy, fairness, robustness, explainability, and human intervention. To build comprehensively trustworthy AI-based performance diagnosis systems, it is essential to develop research to meet ethical trustworthiness requirements, for example, meeting environmental well-being through energy research [70, 120], and meeting accountability requirements through law research [241].

2.5 Conclusion

This chapter provides a systematic overview of studies on trustworthiness requirements in AI-based performance diagnosis systems. We combine the five essential technical trustworthiness requirements for AI models, namely data privacy, fairness, robustness, explainability, and human intervention, with the general performance diagnosis framework covering data collection, preprocessing, anomaly detection, and root cause localization. From this integration, we summarize ten requirements, such as fairness in data collection and robustness in anomaly detection, and provide a comprehensive review of methods for each requirement. We believe the presented survey will offer practical guidance for researchers to develop advanced performance diagnosis systems. Finally, we identify several research directions and challenges. This thesis mainly focuses on the quality improvement of data, robust and accurate anomaly detection, and precise and fine-grained root cause localization. In the future, we will focus on the studies of other challenges, such as secure data governance, automatic operation, and ethical trustworthiness requirements.

Chapter 3

Effective Performance Diagnosis Framework for Distributed Applications

To run a distributed application with the required service quality, operators have to continuously monitor the run-time status, detect potential performance anomalies, and diagnose the root causes. However, the existing monitoring tools lack automated deployment and a customized interface. In addition, an effective performance diagnosis framework is required for operators to detect and maintain performance issues. In addition, collected performance data lacks high-quality labels and usually contains noise which will affect diagnosis performance. Existing performance anomaly detection methods usually focus on different characteristics in data and have varying detection performances. Moreover, the current root cause localization models make locating system-level root causes of application performance anomalies difficult for effective adaptation decisions. We propose a FIne-grained pErformance Diagnosis (FIED) framework to tackle monitoring challenges and employ real-time, fine-grained methods to detect performance anomalies and locate root cause metrics of anomalies.

This chapter is based on:

- **Ruyue Xin**, Jardenna Mohazzab, Zeshun Shi, and Zhiming Zhao. "CBProf: Customisable Blockchain-as-a-Service Performance Profiler in Cloud Environments." In Blockchain–ICBC 2021: 4th International Conference, Held as Part of the Services Conference Federation, SCF 2021, Virtual Event, December 10–14, 2021, Proceedings, pp. 131-139. Cham: Springer International Publishing, 2022.
- **Ruyue Xin**, Hongyun Liu, Peng Chen, Paola Grosso, and Zhiming Zhao. "FIRED: a fine-grained robust performance diagnosis framework for cloud applications." arXiv preprint arXiv:2209.01970 (2022).

3.1 Introduction

Cloud environments provide elastic and on-demand resources for developing applications [286]. However, because of the inherent dynamism of clouds, performance anomalies of distributed applications, such as degraded response time caused by resource saturation, may severely affect the quality of the user experience. In addition, considering complex dependencies and multiple components in distributed applications, it's difficult for operators to detect performance anomalies and identify root causes. Traditionally, operators perform diagnoses for distributed applications manually, which is complicated and time-consuming. Data of different monitoring metrics, e.g., CPU and memory usage, can be continuously collected, reflecting the run-time status of distributed applications [278]. Therefore, we could consider a performance diagnosis solution that leverages monitoring data and supports rapid recovery and loss mitigation for distributed applications.

Performance diagnosis involves detecting abnormal performance phenomena, predicting anomalies, and localizing their causes based on performance data [108]. A general performance diagnosis framework consists of four components: data collection, data preprocessing, anomaly detection, and root cause localization. Each component has been extensively researched and various methods have been developed. Data collection relies on monitoring tools to capture accurate performance data reflecting the application's running status. It is important to consider automatic deployment and a customized user-friendly interface to accommodate the complexities of the cloud environment. Data preprocessing mainly addresses the challenge that noise exists in the collected data. Overcoming these challenges is crucial to ensure accurate results in anomaly detection and root cause localization.

In recent years, research about methods for performance diagnosis have been developed and mainly focus on performance anomaly detection and root cause localization. For performance anomaly detection, numerous existing methods [26, 98] focus on improving detection accuracy. However, because the scaling of cloud infrastructures will change the distribution of monitoring data, it is important to observe the detection performance of different detection methods. As for root cause localization, approaches are still developing [74][25] and most of them are focusing on service-level or container-level faults [247][260], which can not provide accurate maintain instructions for operators. To fill these gaps, we are motivated to develop a a FIne-grained pErformance Diagnosis (FIED) framework that can collect real-time performance data, pre-process data properly, detect performance anomalies, and identify the root causes in metric-level.

The rest of this chapter is organized as follows. In Section 3.2, we first analyze requirements for a performance diagnosis framework, then provide the overview of the FIED framework and introduce each component in detail. In Section 3.3, we introduce collected performance data with the monitoring tool and conduct experiments to evaluate data preprocessing, anomaly detection, and root cause localization components. Finally, we draw our conclusions in Section 3.4.

3.2 The performance diagnosis framework

In this section, we first analyze the requirements and challenges of the performance diagnosis framework for distributed applications. Then, we describe our FIED framework and introduce each component in detail.

3.2.1 Requirement analysis

To ensure the development of an effective performance diagnosis system, several design requirements need to be considered for each component. We identify them from several use cases in the EU project SWITCH[1] and ARTICONF[2].

- Continuously monitoring run-time applications. While many monitoring tools exist, there are specific requirements that need to be considered. One such requirement is the development of mechanisms for automatic deployment of applications and monitoring tools that enable easy testing and evaluation of different scenarios, such as different anomalies. Additionally, it is important to design a customized user-friendly interface that allows users to conveniently observe the performance of distributed applications.

- Data preprocessing to reduce noise. This requirement involves addressing the challenge of noise reduction. Methods need to be designed to extract or select the most relevant features that contribute to the accurate detection of performance anomalies.

- Anomaly detection for varying performance data. While multiple detection methods exist, they often focus on different characteristics in the data and have varying performance. In the diagnosis framework, it is necessary to incorporate several different anomaly detection methods to cater to the diverse nature of performance data. In addition, evaluating the performance of these methods on various datasets becomes essential to gain insights into the selection of existing detection methods and to guide the development of advanced detection methods.

- Root cause localization needs be accurate. Complex dependencies between services makes it challenging to model anomaly propagation path. In addition, it is difficult to identify the root cause metric while multiple metrics exist.

To meet these requirements and tackle challenges, we develop an effective performance diagnosis framework which includes a monitoring tool with automated deployment, data preprocessing technologies to address data challenges, different unsupervised detection methods, and fine-grained root cause localization.

[1] https://www.switchproject.eu/
[2] https://articonf.eu/

3.2.2 Framework overview

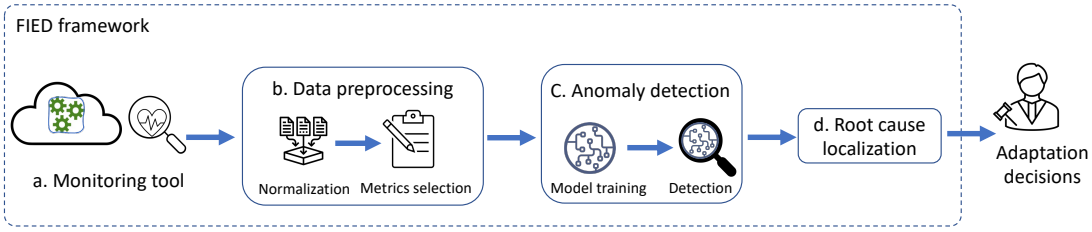


Figure 3.1: The FIED framework for performance diagnosis of distributed applications.

For effective performance diagnosis of distributed applications, we provide the FIne-grained pErformance Diagnosis (FIED) framework, which can be seen in Figure 4.1. The framework can effectively detect performance anomalies and localize root causes for cloud applications, and it works with several modules. At first, we collect multivariate time-series data with a monitoring tool continuously. For collected data, we focus on service- and resource-level data, such as service latency, CPU and memory usage, and input them into the FIED framework. In FIED framework, we first perform (a) a monitoring tool to monitor run-time status and collect performance data. We then provide (b) data preprocessing, including data normalization to scale data, and the metrics selection that involves the use of feature selection or extraction methods to filter multivariate data and reduce data dimensions. Subsequently, pre-processed data is used to (c) anomaly detection for model training and detection of performance anomalies. Once an anomaly occurs, we start the (d) root cause localization to discover the causes of the anomaly. The localization results can be used for rapid recovery of distributed applications. In this section, we will introduce these technologies in detail, and provide experiments for evaluation.

3.2.3 Monitoring tool

Automated deployment. First, we implement a monitoring tool to processes deployment requests and facilitate deployment automatically. We collect deployment requests from users, such as Virtual Machine (VM) types, numbers, and providers. After receiving configuration requirements, CloudsStorm [285], which is a framework for managing an application-defined infrastructure, is used for automatic deployment. When deploying a distributed application in the cloud, the nodes of a network are represented by VMs, and the nodes must form a network to ensure their communication with each other. Therefore, the deployment component includes functions for VM creation and communication. All services will be started automatically after deployment. The automated deployment tooling includes Prometheus and Grafana for monitoring and visualization.

Data collection. The monitoring tool allows the monitor to collect real-time performance data and store it in MongoDB[3], which is a time-series database providing persistent storage. Furthermore, we can visualize real-time performance data and build the graphical user interface with Vue.js[4]. We configure a dashboard to visualize the data collected by the database. With this component, we can check many performance metrics, like block numbers and committed transactions, in real-time. This component is also interactive, so users can customize the performance of a specific period. With the monitoring tool, the specification, retrieval, and storage of customized requests, such as execution time and send rate of workloads is possible. After sending requests, we provide an overview and comparison of application performance, such as transaction latency and throughput. The comparison can provide users with a clear perspective regarding a specific purpose or a certain performance measure.

3.2.4 Data preprocessing

Data normalization. For collected performance data, we apply z-score normalization [194] to ensure that all data have the same scale. The z-score method uses the mean and standard deviation of the original data for normalization so that the processed data follows the normal distribution. After normalization, we represent data with R_i^t ($i = [1, ..., N]$ is the index of resource metrics. N is the number of all resource metrics. $t \in \mathbb{N}^*$ is the index of timestamps) as input data. Next, we provide the metrics selection for the input data, including feature selection and extraction methods. After metrics selection, data D_j^t will be used to diagnose the running status of distributed applications where $j = [1, ..., n]$ is the index of data dimensions and n is data dimensions after reduction.

Metrics selection. Multivariate data usually contains noise, introducing unnecessary variance into a developed model. Therefore, metrics selection to identify relevant metrics and reduce data dimension is needed. We provide feature selection and extraction methods within the metrics selection module, taking into consideration the availability of service-level data. With service-level data, we can easily select relevant resource metrics from multiple metrics with filter methods, which extract a subset from all features [37]. Specifically, Pearson's correlation is generally used to measure the relevance between features, and it provides a fast estimation for feature selection. In addition, the feature extraction method creates a subset of new features from combination of existing features. For example, principal components analysis (PCA) [254] can be used to extract main features in data and reduce data dimensions without labels. With the metrics selection, we can extract related resource metrics automatically.

For feature selection, we provide a filter method of correlation analysis for all

[3]https://www.mongodb.com/
[4]https://vuejs.org/

metrics. For time-series data, we use K^t to represent service-level data, and R_i^t represent resource metrics. We calculate the Pearson's correlation of the labels with each resource metric.

$$r_i = \frac{cov(K^t, R_i^t)}{\sigma_{K^t}\sigma_{R_i^t}} \tag{3.1}$$

A significant test for the Pearson's correlation.

$$s_i = r_i\sqrt{\frac{d-2}{1-r_i^2}} \tag{3.2}$$

Here, d is the number of timestamps, which is also the sample number of each resource metric. In order to filter out low correlation metrics, we set the threshold $s_i < 0.05$ and $|r_i| > 0.5$ for all correlation results.

In addition, we can use the feature extraction method PCA to transform a dataset with lots of variables into a smaller one that still contains most of the information in the original dataset. The process steps of PCA are: 1) getting the covariance matrix of original features; 2) calculating eigenvectors and eigenvalues of the covariance matrix to identify principal components; 3) sorting eigenvalues and selecting eigenvectors with high eigenvalues as feature vectors; 4) recasting the original data based on feature vectors. In step 3, the number of selected eigenvectors determines the data dimensions after reduction. Therefore, we can see that based on these calculations, PCA can be used without labels and achieves principal feature selection and data dimension reduction. In practice, we set the reduction dimension based on a calculated percentage of variance [3]. We apply the correlation analysis and PCA to resource metrics R_i^t and compare their performance in section 3.3.3.

3.2.5 Anomaly detection

Different anomaly detection methods usually focus on different features in data, such as density-based, distance-based, and this results in diverse detection accuracy on different data. Therefore, to have a comprehensive understanding of monitoring data characteristics, we select four classic unsupervised methods (IForest, KNN, LOF, OCSVM) for performance anomaly detection.

IForest is based on the decision tree algorithm [72]. Many isolation trees make up an isolation forest to make anomaly detection results more credible. To build an isolation tree, we need to randomly select a feature in data and a value between the max and min values of that feature first, and then perform a binary partition to divide data into two sides of tree nodes. We can iterate the binary partition until the data has only one feature or reaches the limit height of the tree. The random partitioning of features will produce shorter paths in the tree for the anomalous data points, thus distinguishing them from the others.

KNN is a distance-based algorithm [180]. It calculates each point's distance (Euclidean, Manhattan) with k nearest neighbors and sets the distance as an anomaly score. Based on the assumption that similar things exist in close proximity, points with a high anomaly score mean they are far from others and can be distinguished as anomalies.

LOF is a density-based algorithm [26]. The density here is a local density, while the locality is given by k nearest neighbors, and their distance is used to estimate the density. Thus, by comparing the local density of a point to the local densities of its neighbors, we can identify regions of similar density, and points that have a substantially lower density than their neighbors will be considered anomalies.

OCSVM is based on support vector machine (SVM) [197]. A property of SVM is that it can create a non-linear decision boundary by projecting data through a non-linear function into a high-dimensional space. The non-linear function is known as the kernel function. After the projection, a hyper-plane can be found for separation, and points are separated into different classes. Because the kernel function calculation is time-consuming, it usually works slowly for large-scale data.

For each detection method, the input is preprocessed data. The processing of input data includes model initialization, fitting data, and output anomaly scores. Model initialization includes the setup of hyper-parameters, such as anomaly fractions, which can be set based on data characteristics. After fitting the data, an anomaly score vector will be outputed. We use the anomaly score vector of each detection method to identify anomalies and evaluate the performance of each detection method.

3.2.6 Root cause localization

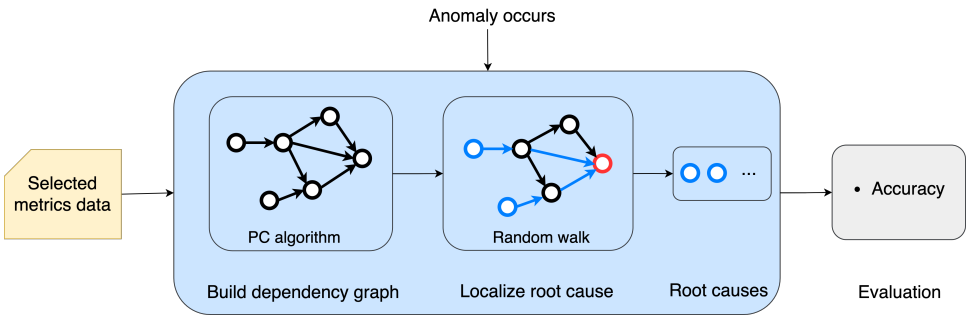


Figure 3.2: The root cause localization pipeline

Performance anomaly detection allows us to know the status of distributed applications. When an anomaly occurs, localizing the root causes of the anomaly can enable the application to recover effectively. We provide the pipeline of

root cause localization for performance anomalies in Figure 3.2. The input data consists of selected metrics and anomaly labels. Selected metrics are metrics after feature selection, which can be identified as CPU or memory related. We do not use metrics after feature extraction because there is no clear meaning of extracted features. For these selected time-series metrics, we extract their causal relations and build a dependency graph with the PC (named after its authors, Peter and Clark) algorithm. Based on the dependency graph, we use a random walk to find the propagation path and localize root causes. Finally, we evaluate the localization accuracy.

Build dependency graph

The causality between system resources and application performance is obvious, for example, low network bandwidth will cause high response latency. To extract the relation, a causal Directed Acyclic Graph (DAG) is commonly used in practical applications because of its intuitiveness. The most popular method for constructing a causal DAG from observational data is the PC algorithm [116].

We use the PC algorithm to discover the causal relationship between system resources and performance anomalies. There are four steps to build a dependency graph with the PC algorithm:

- Construct a fully connected graph of the m random variables (all nodes are connected).
- Perform a conditional independence test on each adjacent variable under the significance level α. If conditional independence exists, the edge between the two variables is removed. In this step, the size of the conditional variable set S increases step by step until no more variables can be added into S.
- Determine the direction of some edges based on v-structure[168].
- Determine the direction of the rest of the edges.

Based on the PC algorithm, we build a dependency graph for all selected metrics D_j^t and the anomaly labels K^t. We define the anomaly labels K^t as an anomaly indicator. In addition, other causal inference methods can be used to build the dependency graph, like Additive Noise Model (ANM) [97]. We also compare their localization performance in our experiments.

Localize root causes

In a dependency graph, there can be many paths that point to the anomaly indicator, which makes it hard to localize root causes. To solve this problem, we apply a Random Walk algorithm to the dependency graph, which performs well

Algorithm 1: Random walk for the causal DAG

Input : DAG G, path length l, start node N
Output: Path points to the start node

```
path = [N]
while len(path)<l do
    cur_node = path[-1];
    if len(list(G.predecessors(cur_node)))>0 then
        predecessor = random.sample(list(G.predecessors(cur_node, 1);
        path.extend(predecessor);
    else
        break;
return path
```

in capturing anomaly propagation. The random walk procedure in a dependency graph is presented in Algorithm 1.

In this algorithm, we set the anomaly indicator as the start node. Furthermore, we end up with a path pointing to the start node by randomly selecting the predecessors of the current node. We iterate the algorithm many times and get several paths. The last node of each path can be regarded as the root cause. By counting and ranking root cause nodes, we can finally get the root cause set.

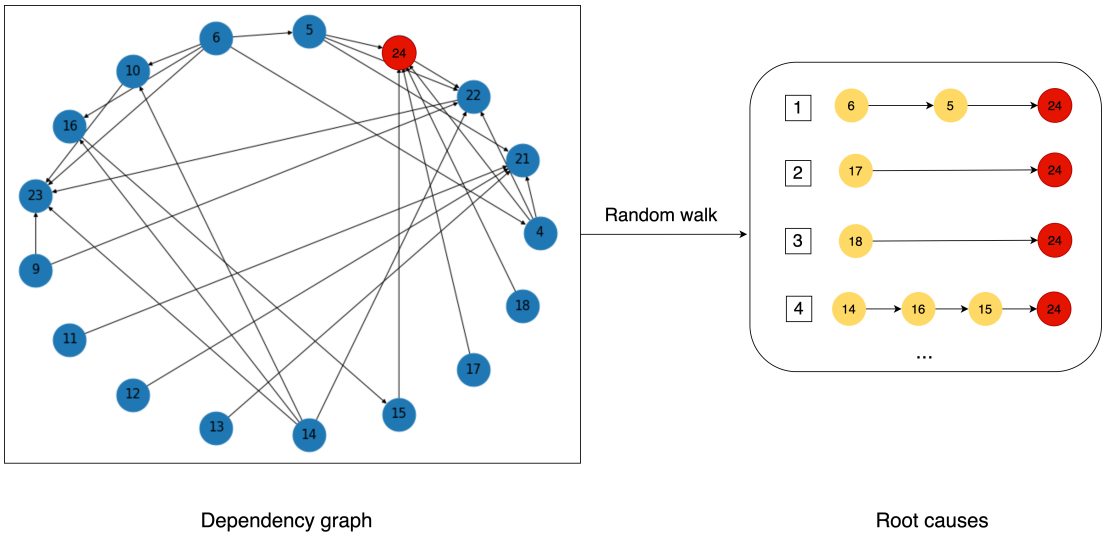


Figure 3.3: Visualization of root cause localization pipeline

We provide an example for the root cause localization pipeline in Figure3.3. Nodes 0-23 represent selected metrics, and node 24 represents the anomaly indicator. We first build a dependency graph for these metrics with the PC algorithm. Isolated nodes which have no causality relation with others are removed in the

dependency graph. Next, for the dependency graph, we use the random walk algorithm to get paths pointing to the anomaly indicator and rank all root cause nodes. We can see that there are paths like $6 \rightarrow 5 \rightarrow 24$, $17 \rightarrow 24$, $18 \rightarrow 24$, $14 \rightarrow 16 \rightarrow 15 \rightarrow 24$. After ranking, the root causes are localized as $\{6, 17, 18, 14\}$.

3.3 Experiments and results

Based on our framework, we deploy and collect performance data from a Decentrailized Application (DApp). Then we add another two public datasets, and provide different experiments to evaluate each component in the FIED framework.

- We implement automatic deployment and monitoring with the monitoring tool for a DApp. We present the interface of run-time status and collected performance data with inject anomalies for FIED evaluation.
- We evaluate metrics selection by comparing the detection performance of unsupervised detection methods with data processed by metrics selection methods.
- We conduct performance anomaly detection experiments for the four detection methods and compare their detection accuracy on multiple datasets.
- We check the feasibility of the root cause localization pipeline, compare different causal inference methods, and observe time spent.

In this section, we will introduce a collected dataset and two public datasets that are used in our experiments. For the evaluation of each component, we present the experimental settings and evaluation results in detail next.

3.3.1 DApp monitoring

In business scenarios where real-time transactions are required, e.g., energy trading or crowd journalisms[195], the performance quality of a DApp, such as transaction throughput, latency, and failure rates, are critical to the business value. To deliver such a quality-critical DApp in cloud environments, one needs to select cloud services carefully, customize their capacities, and monitor the run-time status of the application. We implement the deployment and monitoring of a DApp with our monitoring tool. Figure 3.4 shows the DApp example developed with Hyperledger Fabric[5]. For the DApp, different organizations, which contain many peer nodes, are deployed on VMs and monitored by Prometheus. After deployment, we use Hyperledger Caliper[6] to simulate workload generation and check the run-time status of the DApp through an interface.

[5]https://www.hyperledger.org/use/fabric
[6]https://github.com/hyperledger/caliper

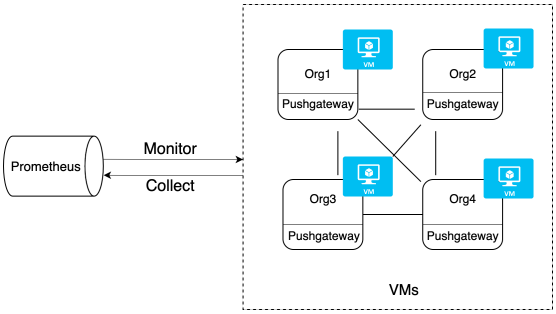


Figure 3.4: The monitor component and a DApp in cloud

Collected performance data. For the running DApp, we mainly collect system resource metrics, which can be seen in Table 3.1. When the DApp receives transaction requests stably, we add system pressures with *stress-ng*[7], such as disk pressure to inject anomalies manually. We increase disk pressure by 20 minutes every hour. We monitor the DApp for twelve hours and collect data at 15-second intervals. Ultimately, the DApp monitoring data contains 3237 samples and 229 resource-related metrics for our experiments. The general information can be seen in the Table 3.2.

Table 3.1: Description of system resource metrics

Resource Metrics	Description
CPU related	Per core and overall load, usage, idle time, I/O wait time, hard and soft interrupt counts, context switch count, etc.
Memory related	Free, cached, active, inactive, dirty memory, etc.
Disk related	Disk space used, IOps, I/O usage, read/write rate, etc.
Network related	Receive/transmit network traffic, etc.

3.3.2 Experimental settings

Public datasets. Except performance data collected from the DApp, we use another two public datasets in our experiments.

[7]https://kernel.ubuntu.com/ cking/tarballs/stress-ng/

Table 3.2: General information of three datasets

Dataset	Number of samples	Number of features	Number of extracted features	Anomaly fraction (%)
DApp monitoring data	3237	229	15	28.14
SMD data	28479	38	5	9.46
Vichalana data	45486	13	6	6.45

Server Machine Dataset (SMD) is a dataset collected and made publicly available by a large internet company[218]. It contains data collected from many different server machines and includes 38 metrics. In addition, domain experts have labeled anomalies in SMD based on incident reports.

Vichalana is a multivariate time-series dataset that can be used for performance anomaly detection in API Gateways [77]. It has different anomalies, such as high CPU and memory usage. Performance metrics in this dataset are collected when the system operates in normal and anomalous mode. The information of SMD and Vichalana data used in our experiments can be seen in Table 3.2.

Parameters. The DApp monitoring data is collected from a deployed DApp in a cloud environment. Here, we use Azure[8] as the cloud environment and deploy the monitor component and DApp separately. The monitor component is deployed on a VM with the following properties: Ubuntu 18.04 as operating system, 2CPU, 4G Memory, 32GiB Storage. The DApp is deployed on VMs which have properties: Ubuntu 18.04 as the operating system, 4CPU, 16G Memory, and 32GiB Storage.

For feature extraction, we need to determine the reduction dimensions of PCA. In general, PCA needs to retain as much variance information of original data as possible, such as 95%. Therefore, we set the reduction dimensions to 15 for DApp monitoring data based on a calculated percentage of variance [3].

As for each base detection method, their hyper-parameters are set as below. Anomaly fractions need to be determined first. For the DApp monitoring data, because we inject anomalies 20 minutes every hour, we set the anomaly fraction as 0.3. For SMD and Vichalana data, we use the default anomaly fraction, which is 0.1. Next, the hyper-parameters of each detection method need to be determined. We set the tree number for IForest to 100. The neighbor number in KNN is 5. In LOF, we set the neighbor number as 20. In OCSVM, we use the radial basis function kernel function.

Evaluation indicators. We evaluate the detection performance of these detection methods with F1 score to indicate accuracy, and time spent to indicate efficiency. F1 score is a function of both Precision and Recall. The Precision is about how much of the data detected as anomalies are true anomalies, while recall is about how much of the real anomaly data is detected as anomalies. So, we calculate F1 score as below:

$$F1\ score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3.3}$$

Therefore, we mainly focus on the F1 score for detection accuracy. Our experiment results also evaluate and present the time spent on each unsupervised detection method and test time for the deep ensemble method.

To evaluate the accuracy of root cause localization, we use two performance metrics: $AC@k$ and Avg. These two metrics are most commonly used to evaluate

[8]https://azure.microsoft.com/en-us/

the rank result of the root cause localization task [165][247]. $AC@k$ represents the probability that top k results localized by algorithms include the real root causes for a given anomaly. When the k is small, the higher $AC@k$ indicates the algorithm identifies the actual root cause more accurately. We calculate $AC@k$ as follows:

$$AC@k = \frac{\sum_{i<k} R[i] \in V_{rc}}{min(k, |V_{rc}|)} \tag{3.4}$$

where $R[i]$ is the result of rank of all metrics for the anomaly. V_{rc} is the root cause set of the anomaly. $Avg@k$ evaluates the overall performance of the localization algorithm by computing the average $AC@k$. The calculation is as follows:

$$Avg@k = \frac{1}{k} \sum_{1 \leq j \leq k} AC@j \tag{3.5}$$

We use $k = 1, 2, 3, 4$ in our experiments to give a comprehensive evaluation of localization accuracy.

3.3.3 Metrics selection evaluation

We execute experiments to validate the metrics selection component in the performance diagnosis framework. We only use the DApp monitoring data here because we know the detailed information of each metric, for example, CPU usage, system load. The metrics selected by the two methods – correlation analysis and PCA, and all metrics are used as the input to the four detection methods. Afterward, we check the effect of metrics selection methods by comparing the detection performances of the four detection methods.

For metrics selection, we apply the correlation analysis and PCA separately on the DApp monitoring data. Figure 3.5 shows the correlation analysis result based on r-values in descending order. We calculate the correlation between all monitoring metrics and fewer labels. The results show that metrics like the amount of unevictable memory and iowait have high r-values, which means there is a relationship between these metrics and the occurrence of the transaction failure anomaly. In addition, we reduce the data dimensions from 229 to 15 based on PCA method.

We then compare the effects of metrics selection methods based on the performance of the various detection methods. We use the data after correlation analysis, PCA, and without metrics selection as the input of detection methods, respectively. The F1 score and time spent of each detection method can be seen in Figure 3.6 and Figure 3.7. In Figure 3.6, we can see that after correlation analysis, three detection methods, IForest, KNN and OCSVM have the highest F1 score. In comparison, the F1 score of LOF is slightly lower. For PCA, the F1 scores of KNN and LOF are higher than without metrics selection. But the F1

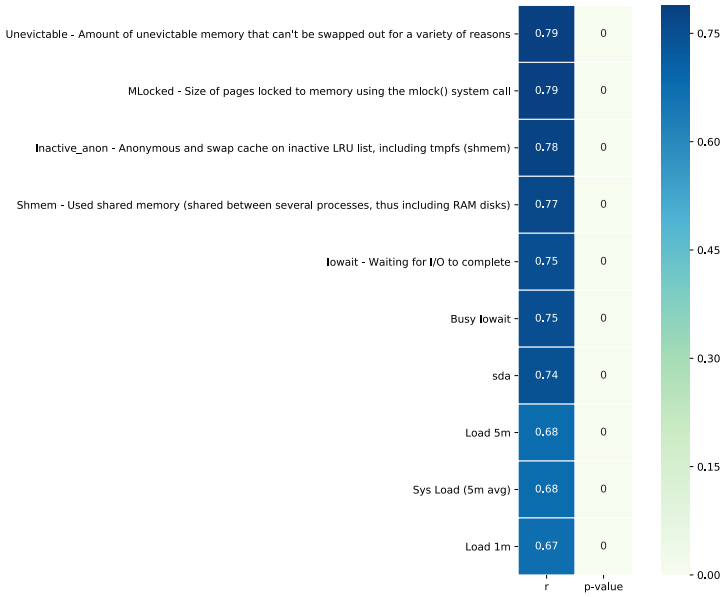


Figure 3.5: Top 10 resource metrics with high relevance to performance anomalies

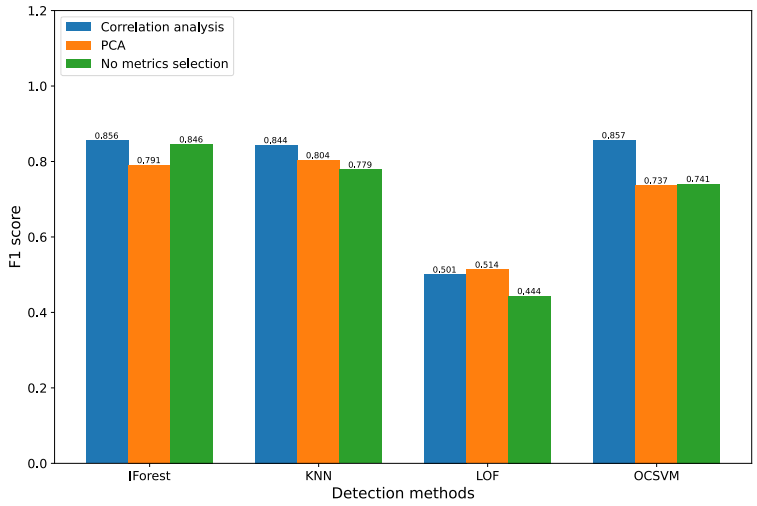


Figure 3.6: Detection accuracy by detection methods as function of the metrics selection methods: correlation analysis, PCA and no metrics selection.

scores of IForest and OCSVM are lower than without metrics selection. We can say that metrics selection based on correlation analysis improves the detection accuracy. In addition, in Figure 3.7, we can see that without metrics selection, the time spent is about 2 to 10 times for each detection method compared with using metrics selection.

In conclusion, we can see that with metrics selection, the detection accuracy is improved and the time spent is reduced compared with no metrics selection.

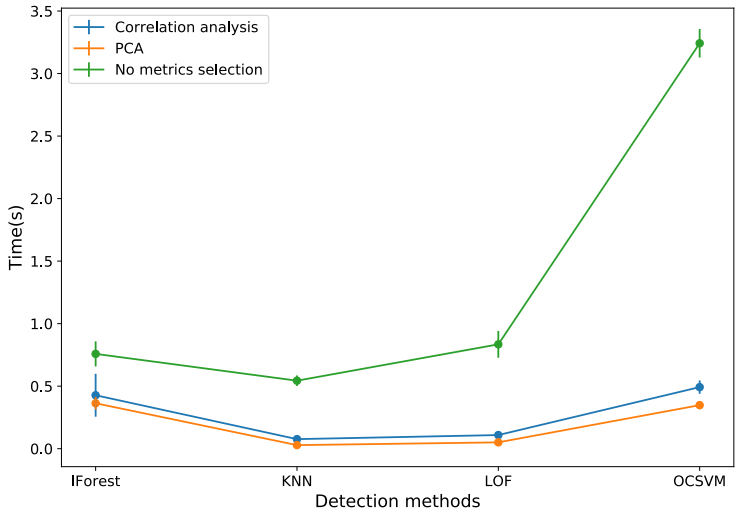


Figure 3.7: Time spent by detection methods as function of the metrics selection methods: correlation analysis, PCA and no metrics selection.

In addition, correlation analysis has better detection accuracy than PCA. Next, we will use data after metrics selection as the input of performance anomaly detection and root cause localization.

3.3.4 Performance of detection methods

We apply the four detection methods (IForest, KNN, LOF, and OCSVM) to the DApp monitoring data, SMD, and Vichalana data. The performance of their detection accuracy can be seen in Table 3.3.

Table 3.3: Performance of different detection methods. For each dataset, we show the F1 score of the best detection method in bold.

Detection methods	DApp monitoring data		SMD data		Vichalana data		Average F1 score
	F1 score	Time(s)	F1 score	Time(s)	F1 score	Time(s)	
IForest	0.791	0.318±0.012	**0.752**	1.278±0.020	0.658	1.981±0.070	**0.734**
KNN	**0.803**	0.025±0.002	0.571	0.311±0.005	0.552	0.776±0.069	0.642
LOF	0.514	0.044±0.002	0.547	0.538±0.011	0.513	1.468±0.123	0.525
OCSVM	0.737	0.305±0.008	0.605	33.923±0.892	**0.678**	95.118±0.019	0.673

For the DApp monitoring data, we can see that the KNN has the highest F1 score, 0.803, demonstrating that the data has clustering characteristics because KNN is good at identifying clusters in data. IForest takes into account different features in the data. IForest usually has good detection performance [32], as well as on the DApp monitoring data with an F1 score of 0.791. If the abnormal features are concentrated in a few dimensions, it will be hard to detect anomalies for LOF. Therefore, LOF has the lowest F1 score, 0.514, for the DApp monitoring

data. The F1 score of OCSVM is 0.737, which is not high enough because the projection through a kernel function cannot be divided into normal and abnormal data very well. For time spent, we can see that IForest and OCSVM spend about 0.3s, which is higher than other detection methods because the calculation of features takes some time, but the time spent is under 0.5s overall, which is not high actually. As a result, for the DApp monitoring data, the KNN is the best of the four detection methods.

For SMD data, we can see that IForest has the highest F1 score, 0.752, which shows the advantage of IForest for anomaly classification through multiple features. However, F1 scores are not high for other detection methods, showing too much noise in this dataset, and the overall distribution of normal and abnormal data is similar. Thus, we can say that anomalies may be mainly in a few features in the SMD data. In addition, the time spent on OCSVM is higher than on others because the kernel function calculation in OCSVM is time-consuming. On the other hand, IForest has the best detection accuracy and takes about 1.3s, which is the best detection method.

For Vichalana data, we can see that OCSVM has the highest F1 score, 0.678, showing that the non-linear projection can classify normal and abnormal data but is not very accurate. The F1 score of IForest is 0.658, slightly lower than OCSVM, which means that abnormal data distribution varies in different features, making it hard to detect. The F1 scores of KNN and LOF are pretty low, showing that the overall distribution of normal and abnormal data is also similar. It is worth noting that the time spent on OCSVM is relatively high because the dataset includes more than 40k samples, and it takes too much time for kernel function calculation in OCSVM. Here, IForest only takes about 2s, which is quite faster than OCSVM.

In conclusion, we can see that detection accuracy varies for these unsupervised detection methods on the three different datasets. For example, KNN performs the best on the DApp monitoring data but relatively poorly on the SMD and Vichalana data. Overall, IForest has the highest average F1 score 0.734, as its effectiveness in handling high-dimensional and large datasets. However, it is still critical to develop suitable performance anomaly detection methods for improving detection accuracy for distributed applications.

3.3.5 Root cause localization evaluation

We conduct experiments to validate the feasibility of root cause localization in the performance diagnosis framework. Our experiments are implemented based on DApps monitoring data, because we have clear description of each metric. We will identify which metrics in the DApps monitoring data cause performance anomalies. We apply root cause localization methods on the DApps monitoring data. In Table 5.1, we provide 24 selected metrics and 1 anomaly indicator of DApp monitoring data. We classify them into CPU/MEM/NET/Disk related

Table 3.4: Description of selected resource metrics

Index	**Type**	**Metric**	Ground truth
0	Memory related	Unevictable - Amount of unevictable memory that can't be swapped out for a variety of reasons	
1	Memory related	Size of pages locked to memory using the mlock() system call	
2	Memory related	Inactive_anon - Anonymous and swap cache on inactive LRU list, including tmpfs (shmem)	
3	Memory related	Shmem - Used shared memory (shared between several processes, thus including RAM disks)	
4	CPU related	Iowait - Waiting for I/O to complete	✓
5	CPU related	Busy Iowait	✓
6	Disk related	sda	✓
7	CPU related	Load 5m	
8	CPU related	Sys Load (5m avg)	
9	CPU related	Load 1m	
10	CPU related	CPU Busy	
11	Memory related	Pagesout - Page out operations	
12	Disk related	sda - Successfully written bytes	✓
13	Disk related	sda - Written bytes	✓
14	Disk related	sda - discard	✓
15	Network related	OutOctets - Sent octets	
16	Network related	trans eth0	
17	Disk related	Processes blocked waiting for I/O to complete	✓
18	Memory related	Dirty - Memory which is waiting to get written back to the disk	
19	CPU related	Sys Load (15m avg)	
20	CPU related	Load 15m	
21	Memory related	Writeback - Memory which is actively being written back to disk	
22	Disk related	sda - Writes completed	✓
23	CPU related	Idle	
24	Anomaly indicator	txn_fail_label	

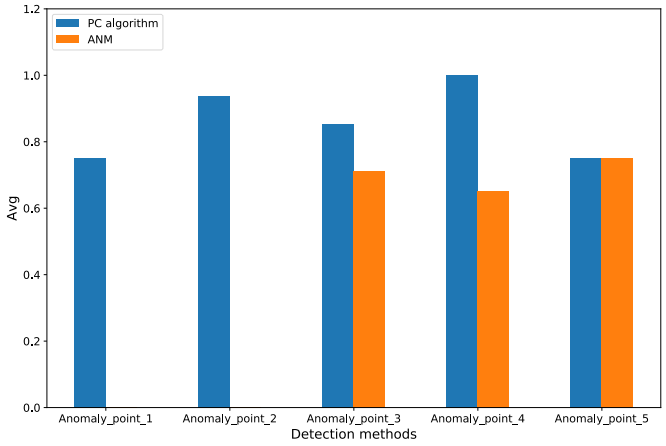


Figure 3.8: root cause localization accuracy based on different dependency graph building methods

metrics. For the DApp we are monitoring, we add I/O pressure to inject anomalies. The root causes are I/O related as shown in Table 5.1. As for methods in the localization pipeline, we set α in the PC algorithm as 0.05, and the iteration for the random walk is 500.

Table 3.5: root cause localization accuracy for anomalies with the PC algorithm

Metric	AC@1	AC@2	AC@3	AC@4	Avg	Time(s)
Anomaly_point_1	1.0	0.50	-	-	0.75	0.84
Anomaly_point_2	1.0	1.0	1.0	0.75	0.94	0.80
Anomaly_point_3	1.0	0.50	0.67	0.75	0.73	0.59
Anomaly_point_4	1.0	-	-	-	1.0	0.69
Anomaly_point_5	1.0	0.50	-	-	0.75	0.91

When an anomaly is detected, we start to localize its root causes. With anomaly injection, we get several anomalies periods, and each of them lasts for 20 minutes. We select 5 of them randomly and compare the localization accuracy based on two different dependency graph building methods: the PC and the ANM algorithms. The comparison results can be seen in Figure 3.8. To evaluate the localization accuracy, we use the *Avg* metric with different values of k due to the uncertain nature of the number of predicted root causes. We can see that, for different anomaly points, the PC algorithm has better performance than the ANM algorithm. In addition, for anomaly_point_1 and anomaly_point_2, we can see that the ANM algorithm does not discover real root causes because the dependency graph does not extract causality relations from data. Therefore, the PC algorithm has better localization accuracy and more stable performance for the DApp monitoring data. We also show detailed localization performance of

PC algorithm in Table 3.5. We calculate the detection accuracy of AC@1, AC@2, AC@3, AC@4, and Avg, and present the localization results. We can see that the real root cause can be localized directly for different anomaly points. As for anomaly_point_2 and anomaly_point_3, we can see that multiple root causes are discovered, including many real root causes, so the localization accuracy is high. Also, we provide the time spent of building dependency graph and localizing root causes, we can see that the localization for these anomalies can complete within 1s, which means that the localization can be done in real-time with given data.

In conclusion, our experiments demonstrate the feasibility of real-time root cause localization in the performance diagnosis framework. In addition, we build the dependency graph and localize root causes at fine-grained. Our experiments also show that the localization based on the PC algorithm is accurate for the DApp monitoring data.

3.4 Conclusion

In this chapter, to achieve effective performance diagnosis, we provide a performance diagnosis framework named FIED, which can collect real-time performance data, handle noise in monitoring metrics, effectively detect performance anomalies and localize root causes of distributed applications. The performance anomaly detection provide evaluation for different unsupervised detection methods. The proposed root cause localization method can identify root causes in a metric granularity with high accuracy. The monitoring tool in FIED includes automated deployment and customized interface for observing application status. The preprocessing component aims to reduce noise in data and improve diagnosis performance.

We provide experiments to evaluate the effect of metric selection, and results show that it can help improve detection accuracy and reduce time spent. For performance anomaly detection, our experiments show that IForest has the highest average F1 score 0.734, as its effectiveness in handling high-dimensional and large datasets. However, it is still critical to develop suitable performance anomaly detection methods for improving detection accuracy. We provide the root cause localization pipeline to fine-grained identify the root causes of performance anomalies accurately and in real-time. The pipeline includes building the dependency graph with the PC algorithm, localizing and ranking root causes with a random walk. We apply the localization pipeline to the DApp monitoring data. We compare the PC and ANM algorithms to build the dependency graph, and the results show that the PC algorithm has the average localization accuracy higher than 0.7. Our experiments also demonstrate the feasibility of real-time root cause localization based on the PC algorithm. More research into improving localization accuracy can be considered in the future.

Chapter 4

Performance Anomaly Detection Methods with Enhanced Accuracy and Robustness

In the previous chapter, we proposed the FIne-grained pErformance Diagnosis (FIED) framework and demonstrated the feasibility of each component. However, in the anomaly detection component, we have observed that existing detection methods have varying performance for different datasets because they focus on different features in data. In addition, effective anomaly detection methods should meet challenging requirements, including high accuracy in detecting anomalies and robustness to changing data patterns, while few studies have addressed both challenges simultaneously. To address these issues, we propose an ensemble learning-based detection (ELBD) framework that integrates well-selected existing methods, including three classic linear ensemble methods and a novel deep ensemble method. Our deep ensemble method, which is weakly supervised, achieves the highest accuracy and robustness for performance anomaly detection in distributed applications. Furthermore, we propose an unsupervised detection method called CGNN-MHSA-AR for multivariate time series anomaly detection. This method leverages temporal and feature information to achieve superior accuracy compared to baseline detection methods.

This chapter is based on:

- **Ruyue Xin**, Hongyun Liu, Peng Chen, and Zhiming Zhao. "Robust and accurate performance anomaly detection and prediction for cloud applications: a novel ensemble learning-based framework." Journal of Cloud Computing 12, no. 1 (2023): 1-16.
- Yujia Song, **Ruyue Xin**, Peng Chen, Rui Zhang, Juan Chen, and Zhiming Zhao. "Identifying performance anomalies in fluctuating cloud environments: a robust correlative-GNN-based explainable approach." Future Generation Computer Systems (2023). (as co-first author)

4.1 Introduction

Performance anomaly detection plays a vital role in operating cloud services, and applications [23] [292]. Cloud performance anomalies such as degraded response time, often caused by underlying system resource shortages, may severely affect the quality of an application's user experience and service. With monitoring tools, performance data such as resource usage of applications can be collected [10]. At the same time, anomaly detection to build a profile of performance data, and detect deviations from the profile for distributed applications can be developed [167]. Considering it is tedious and time-consuming to label data manually because various anomalies exist, weakly-supervised or unsupervised learning to pick up interesting structures in the data and learning features is popular [169]. As a result, effective anomaly detection to identify abnormal behaviors and predict anomalies to forestall future incidents is required in cloud computing systems.

Performance data of cloud computing systems, such as CPU and memory usage, are usually represented as multivariate time series. Multivariate time series reflects health status of cloud computing system and can be used to identify abnormal behavior or events in real-time [35]. In this context, we can highlight two challenging requirements for performance anomaly detection methods. At first, the main target of real-time anomaly detection models is to improve detetion accuracy, reduce False Position Rate (FPR), and achieve better performance [199]. However, the rapid increase of resources, dynamic cloud environments will cause irregular data fluctuations and increase the FPR of anomaly detection. For example, sudden changes in a certain feature, e.g., CPU usage, do not necessarily mean anomalies of the system. In addition, improving robustness of detection method to meet changes in data patterns and maintain performance consistency is essential because different data distributions exist in multiple monitoring data.

Existing anomaly detection methods have often been developed using statistics [207] or machine learning [26, 98] based methods. Most methods focus on improving detection accuracy. For example, Audibert et al.[11] developed the USAD based on an adversely trained AutoEncoder and achieved the best detection accuracy. Deep learning-based methods for multivariate time series anomaly detection are also popular recently. STGCN[258] is a novel Graph Neural Network (GNN)-based model tackling the time series prediction problem in the traffic domain and improving detection accuracy with multi-head self-attention. OmniAnomaly[218] utilizes a stochastic recurrent neural network to capture long-term temporal information and a planar normalizing flow to define and detect anomalies. However, existing deep learning-based detection methods mainly target improving detection accuracy for specific scenarios, which cannot meet the requirements of complex and dynamic cloud computing systems. Studies on improving the robustness of detection methods usually use adversarial training, which needs to make a trade-off between robustness and accuracy[272], rather than simultaneously improving accuracy and robustness.

We thus define our research question as *"how to effectively detect and predict performance anomalies with high accuracy and good robustness?"*. To address the challenges of fewer labels and data noise, we focus on unsupervised and weakly supervised detection methods and provide feature extraction to filter noise in the data. To answer our research question, we provide two solutions. Firstly, considering various detection models have been developed, we adopt integrate existing methods with ensemble learning [73] to capture different features in the data instead of solely improving individual models. Therefore, we develop an Ensemble Learning-Based Detection (ELBD) framework that incorporates classic detection methods. We apply linear ensemble methods such as maximum, average, and weighted average ensemble, which heavily rely on base detection methods, while all of them can hardly surpass the performance of the individual base detection methods. We then introduce a deep ensemble method that effectively extracts and incorporates the strengths of the base detection methods with a neural network to enhance detection accuracy and robustness.

Secondly, we recognize that deep ensemble method improve detection performance but has limited improvement due to the inherent limitations of classical detection methods in extracting information from data. To overcome this problem and enhance detection accuracy and robustness, we explore the development of advanced deep learning-based detection methods. Furthermore, to reduce the reliance on labeled data, we shift our focus to unsupervised learning methods. Specifically, we propose the Correlative-GNN with Multi-Head Self-Attention and Auto-Regression Ensemble Method (CGNN-MHSA-AR) for unsupervised multivariate time series anomaly detection in distributed applications. This method leverages the power of deep learning techniques, including GNN, multi-head self-attention, and auto-regression, to achieve accurate anomaly detection without the need of labeled data.

The rest of this chapter is organized as follows. In Section 4.2, we introduce the details of the ELBD framework and conduct experiments to evaluate detection accuracy and robustness. In Section 4.3, we provide detailed description of each module of the CGNN-MHSA-AR model and present experimental results and analysis. Finally, we draw our conclusion in Section 4.4.

4.2 An ensemble learning-based detection framework

Base detection methods focus on different features in data and have diverse performances. Therefore, it is reasonable to consider that the integration of base methods can extract more features from data and improve detection performance. Furthermore, ensemble learning is proposed with the assumption that by combining several base models, the errors of a single model will be compensated by others. Therefore, we consider integrating base methods with ensemble learning and pro-

pose an Ensemble Learning-Based Detection (ELBD) framework, including three classic linear ensemble methods (maximum, average, and weighted average) and a deep ensemble method.

4.2.1 Problem definition

Multivariate time-series data are timestamped data points sequences and can be represented as D. Then each data point will be D_i^t ($i = [1, ..., n]$ is the index of resource metrics. n is the number of resource metrics. $t \in \mathbb{N}^*$ is the index of timestamps). Multivariate time-series data anomaly detection is to learn the characteristics of data D and determine whether an observation D_{n+1} is anomalous or not. For multi-step anomaly prediction, we will use data D for training, and determine whether $D_{n+1}, D_{n+2}, ..., D_{n+p}$ is anomalous.

In this section, we first provide the performance of classic detection methods. Then we propose an ELBD framework, which is developed based on ensemble learning and aims to improve detection accuracy and robustness by non-linearly integrating information extracted by classic detection methods. In addition, we implement multi-step prediction ability in the deep ensemble method in ELBD framework.

4.2.2 Basic idea

The ELBD framework can be seen in Figure 4.1. First, input data is multivariate time-series monitoring data, including system and service level data, which can be collected and used as input. In this section, we mainly focus on system resource data. We can represent input data as D_i^t ($i = [1, ..., n]$ is the index of resource metrics. n is number of resource metrics. $t \in \mathbb{N}^*$ is the index of timestamps). Next, preprocessing needs to be done for the input data, including feature extraction and train/test split. Feature extraction has been introduced in 4.2.3. There is no need to do the train/test split for unsupervised learning. However, the train/test split is important to avoid over-fitting for weakly-supervised learning. Therefore, we do the train/test split for the deep ensemble method, as seen in the experimental settings. After preprocessing, data D_j^t ($j = [1, ..., d]$ is the index of data dimensions. d is data dimensions after reduction) will be the input of anomaly detection methods.

The base method selection provides unsupervised detection methods. In this section, we manually select four typical base methods, which have been introduced in detail in 3.2.5. The output of base methods can be assembled as an anomaly score matrix. For the matrix, we provide three linear ensemble methods without training and a deep ensemble method, which needs to be trained with a neural network. The output of anomaly detection methods can be represented as C_m^t (m is the index of all detection methods). We mainly focus on accuracy, robustness, and multi-step prediction ability to evaluate the multiple detection methods.

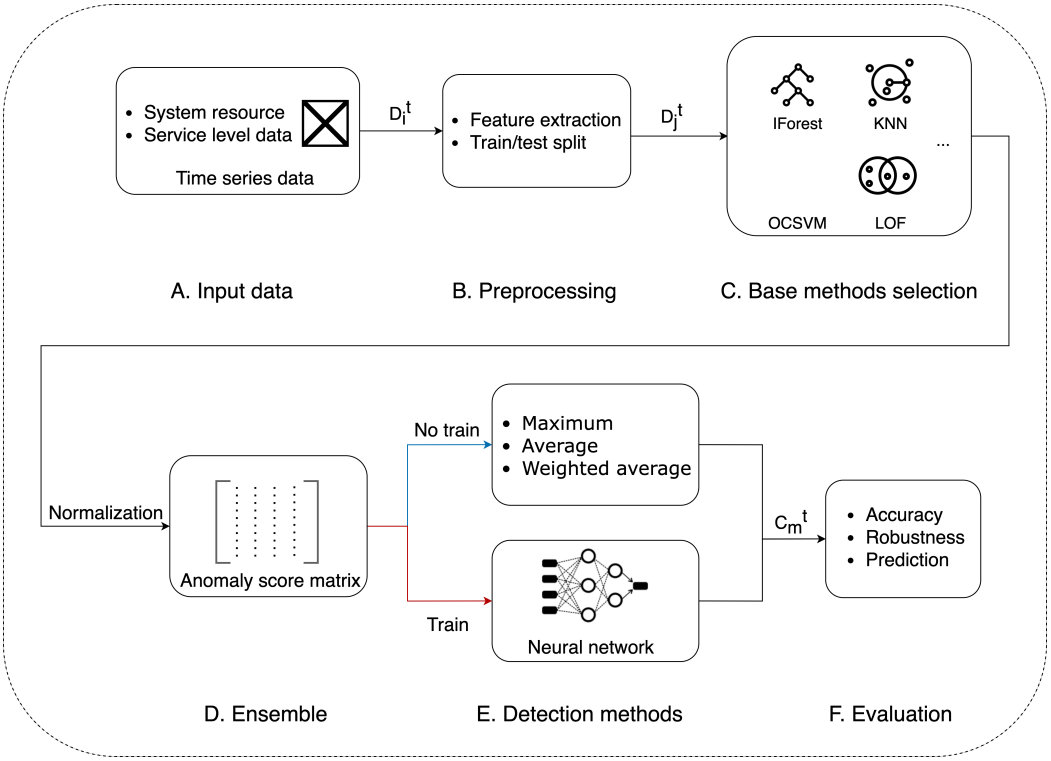


Figure 4.1: ELBD framework, including three classic ensemble methods without training (blue line) and a deep ensemble method which need to train a neural network (red line).

4.2.3 Feature extraction

Multivariate data usually contains noise, which can induce unnecessary variance in a model. Therefore, preprocessing data through feature extraction to remove redundant information and reduce data dimension is needed. For feature extraction, we apply PCA, which is an unsupervised method that uses eigenvalue decomposition to compress and denoise data [254]. Our experiment in Section 3.3.3 has approved that it can reduce data dimension and computation costs well.

4.2.4 Linear ensemble methods

The outputs of base methods have different meanings and scales. For example, the anomaly score of IForest is calculated based on path depth, and KNN is based on distance. Because all the features should be measured in the same units, we apply z-score normalization [194] to ensure that all outputs have the same scale. The z-score method uses the mean and standard deviation of the original data for normalization so that the processed data follows the normal distribution. After normalization, we can represent the anomaly score vector C_k^t (k represents base detection methods) of each base method as O_k^t. Here, k is the index of base

detection methods and $k \in [1, r]$, r is the number of base methods. Therefore, by taking each anomaly score vector as a column, we can get the anomaly scores matrix M:

$$M = \begin{bmatrix} O_1^1 & O_2^1 & O_3^1 & O_4^1 \\ O_1^2 & O_2^2 & O_3^2 & O_4^2 \\ \vdots & \vdots & \vdots & \vdots \\ O_1^t & O_2^t & O_3^t & O_4^t \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

The left side of table 4.1 can be seen as an example of the matrix. For matrix M, we provide linear ensemble methods first, including maximum ensemble, average ensemble, and weighted average ensemble.

The **maximum ensemble** is to select the max value of each row in matrix M and form a new anomaly score vector.

$$V_{max} = \max_k O_k^t, t \in \mathbb{N}^* \tag{4.1}$$

The **average ensemble** is to calculate the average of each row and form a new anomaly score vector.

$$V_{avg} = \frac{1}{r} \sum_{k=1}^{r} O_k^t, t \in \mathbb{N}^* \tag{4.2}$$

A limitation of the average ensemble is that each base detection method contributes equally to the final anomaly scores. However, some methods perform better or worse than others. Therefore, we can consider assigning different weights for these methods. For example, we assign more weights to better methods and fewer to worse ones. **Weighted average ensemble** is a method developed based on this idea.

Based on the assumption that if a mixed model can maximize the information provided by each model, the mixed model has the best weight distribution strategy. Mutual information can measure the difference between models, which can be used to calculate the weight of each base method [126]. To calculate the mutual information of two models, we first need to transfer anomaly scores into anomaly classes (0 or 1). We assume n samples in the two models, a and b. Next, we use N_0^a and N_1^a to represent the number of normal and abnormal data in model a, and N_0^b and N_1^b to represent the number of normal and abnormal data in model b. In addition, N_0^{ab} and N_1^{ab} represent the data that is detected as normal and abnormal by both models. Then we can calculate the MI of models a and b:

$$\begin{aligned} I(A,B) = N_0^{ab} \log \frac{n * N_0^{ab}}{N_0^a * N_0^b} + (N_0^a - N_0^{ab}) \log \frac{n * (N_0^a - N_0^{ab})}{N_0^a * N_1^b} \\ + (N_0^b - N_0^{ab}) \log \frac{n * (N_0^b - N_0^{ab})}{N_1^a * N_0^b} + N_1^{ab} \log \frac{n * N_1^{ab}}{N_1^a * N_1^b} \end{aligned} \tag{4.3}$$

To normalize it, we can calculate:

$$\phi(A,B) = \frac{I(A,B)}{\sqrt{(\sum_{i=0}^{1} N_i^a \log \frac{N_i^a}{n})(\sum_{i=0}^{1} N_i^b \log \frac{N_i^b}{n})}} \tag{4.4}$$

Therefore, the average mutual information of base method is:

$$\sigma_k = \frac{1}{r-1} \sum_{l=1, l\neq k}^{r} \phi(\lambda^{(k)}, \lambda^{(l)}), k \in [1, r] \tag{4.5}$$

Here, each base method is $\lambda^{(k)}$. σ_k is the standard value of the difference between models and $\sigma_k \in [0, 1]$. The smaller the value, the greater the difference between the two models. Based on the difference value of each model, we calculate the weights with $w_k = \sigma_k * Z$, Z is the normalization factor. The new anomaly score vector can be calculated as:

$$V_{w\_avg} = \frac{1}{r} \sum_{k=1}^{r} \sigma_k * O_k^t, t \in \mathbb{N}^* \tag{4.6}$$

in Table 4.1, we provide five samples as an example to show how maximum, average, and weighted average ensemble methods work. In the left part of the table, we show the anomaly scores of four detection methods. In the right part, we can easily get the maximum and average anomaly scores. As for the weighted average ensemble, we assign the weights as (0.39, 0.28, 0.04, 0.29) for base methods based on the calculation. These new anomaly score vectors will be used to identify anomalies and evaluate the performance of these ensemble methods.

Table 4.1: Linear ensemble methods example: on the left side is anomaly scores obtained by each base method; on the right side is anomaly scores obtained through ensemble methods.

Index	IForest	KNN	LOF	OCSVM	Max	Avg	Weighted Avg
1	-0.41	-0.23	0.14	-0.88	0.14	-0.35	-0.49
2	-0.18	-0.03	0.63	-0.86	0.63	-0.11	-0.33
3	2.29	5.14	1.07	0.62	5.14	2.28	2.76
4	2.36	4.56	0.86	0.11	4.56	1.97	2.42
5	1.99	1.5	-0.3	-0.19	1.99	0.75	1.14

4.2.5 The deep ensemble method

The ensemble methods above try to combine different anomaly scores linearly. However, the linear combination may not represent the information extracted by

each model well. Therefore, we provide a **deep ensemble** method in Figure 4.2, and it combines base methods in a nonlinear way by using an Multi-Layer Perceptron (MLP). An MLP is a supplement to a feed-forward neural network. It consists of three layers: the input layer, the output layer, and the hidden layer. An MLP is suitable for classification or regression problems where inputs are assigned a class or real-value label. Therefore, the deep ensemble method is weakly-supervised and needs to be trained with some labels. Considering that there are fewer labels in reality, we design to train the deep ensemble with fewer labels and then test the trained model.

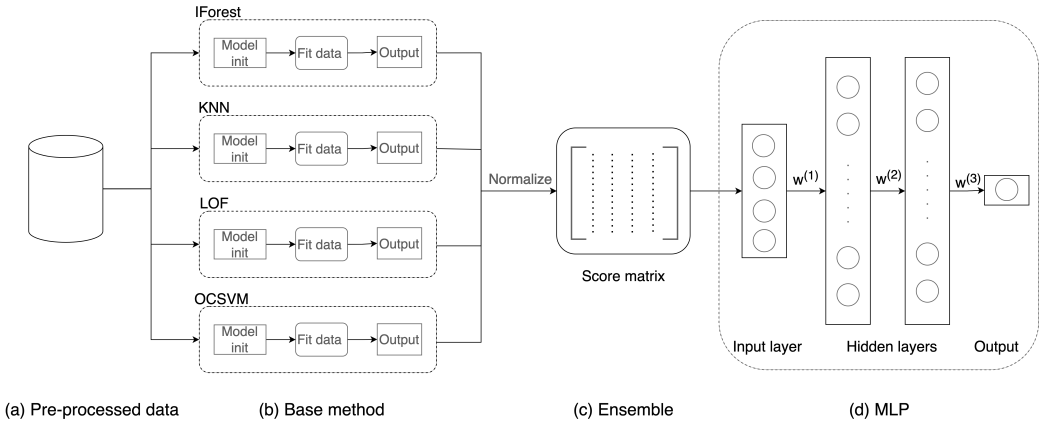


Figure 4.2: The architecture of deep ensemble method includes four steps: (a)preprocessing data is sent to four (b)base methods; then after normalization, the (c)ensemble of their outputs forms a score matrix; we finally input the score matrix into an (d)MLP for training.

We provide the MLP architecture in Figure 4.2. The input layer receives the anomaly score matrix M at first. We have two hidden layers consisting of an arbitrary number of neurons and use ReLU as an activation function. The output layer has one neuron and outputs the probability using the softmax activation function. We define $x = [O_1^t, O_2^t, O_3^t, O_4^t]$. $W^{(1)}$ and $b^{(1)}$ are weights and biases of the first layer. $W^{(2)}$, $b^{(2)}$ and $W^{(3)}$, $b^{(3)}$ are weights and bias of the two hidden layers. The output can be calculated based on the below functions.

$$
\begin{aligned}
z^{(1)} &= W^{(1)}x + b^{(1)}, \\
h^{(1)} &= ReLu(z^{(1)}), \\
z^{(2)} &= W^{(2)}h^{(1)} + b^{(2)}, \\
h^{(2)} &= ReLu(z^{(2)}), \\
z^{(3)} &= W^{(3)}h^{(2)} + b^{(3)}, \\
h^{(3)} &= softmax(z^{(3)})
\end{aligned}
\tag{4.7}
$$

For the output $h^{(3)}$, we can calculate the difference between the predicted and actual results y with the cross-entropy error function below. Here, y is the label at time t. The optimization goal is to minimize this equation by constantly adjusting parameters.

$$l = -y^T \log h^{(3)} \tag{4.8}$$

The deep ensemble method needs to be trained with fewer labels, and then the trained model can be applied to other data to detect anomalies. If we let y be the label of time $t + s$ (s is steps), we can train a model with prediction ability. We provide an ELBD framework for improving detection accuracy, robustness, and predicting anomalies. Experimental results can be seen next.

4.2.6 Experiments and results

Experimental settings

We design four experiments to evaluate the performance of the ELBD framework.

- E1: performance of methods in the ELBD framework. To evaluate the improvement in detection accuracy and algorithm robustness, we compare the performance of methods in the ELBD framework with the best-performing base detection method.
- E2: comparison with other deep detection methods. We design different deep detection methods, such as combining base detection methods with MLP, or replacing MLP with other deep learning methods, and compare their detection performance with ELBD.
- E3: impact of different amounts of labels. Considering that the deep ensemble is a weakly-supervised method, we conduct experiments to observe the detection performance based on different amount of labels.
- E4: multi-step prediction of the deep ensemble method. As for the deep ensemble method, we evaluate its multi-step prediction ability.

No hyper-parameter exists for maximum, average, and weighted average ensemble methods. We first do the train/test split for the deep ensemble method. Because there are fewer labels in real scenarios, we use only 10% of data with labels to train the model. Next, hyper-parameters in the MLP for the three datasets are the same. The input layer has 4 neurons because we have 4 base methods. In addition, we set 20 neurons in the two hidden layers and the output layer as 1. We train 100 epochs and set the batch size to 20. We use the Adam optimizer for stochastic gradient descent with an initial learning rate of 10^{-3} during model training. We train the deep ensemble method 10 times. We show the error bar in figures and take the average of evaluation metrics in tables, such as F1 score and time, as the final result.

Experimental results

E1: Performance of methods in ELBD framework. We provide different methods in the ELBD framework to improve detection performance. We apply these methods to the DApp monitoring, SMD, and Vichalana datasets to evaluate them. We compare these methods with the best-performing base method and evaluate the detection accuracy and robustness.

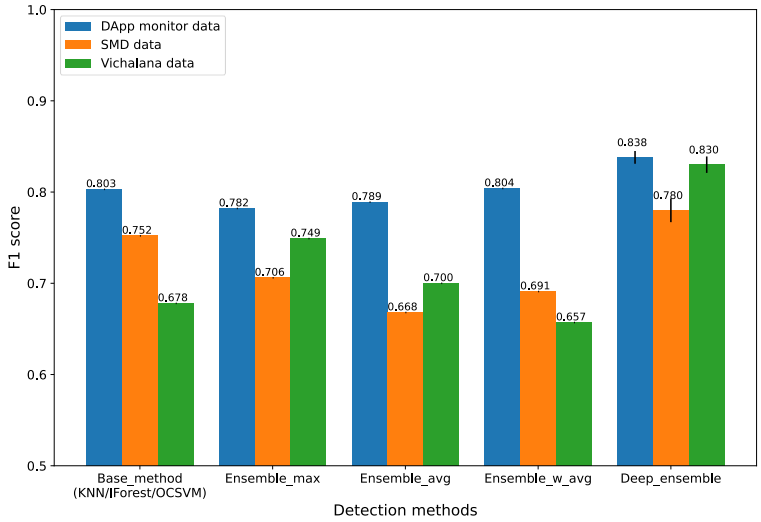


Figure 4.3: Detection accuracy of ensemble methods

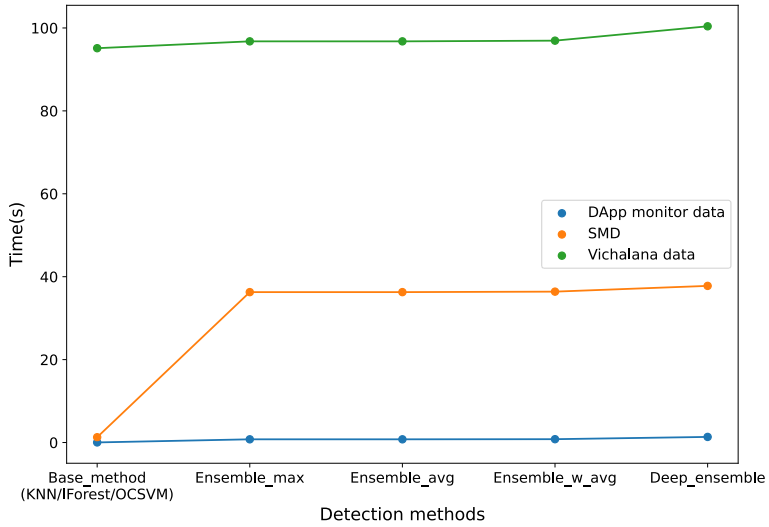


Figure 4.4: Time spent of ensemble methods

For the DApp monitoring data in Figure 4.3, we can see that the F1 score of the weighted average ensemble is higher than KNN, maximum, and average ensemble, which shows that ensemble methods can improve the detection accuracy by integrating extracted information of base methods. In addition, the weighted

average ensemble assigns weights to base methods to highlight their different contributions. The most noteworthy thing in Fig 4.3 is that the deep ensemble method has the highest F1 score, 0.838. We train the deep ensemble method with only 10% labels, but the improvement is significant. The result shows that the nonlinear combination of base methods can extract more information and help improve detection accuracy. As for time spent, in Figure 4.4, we can see that the deep ensemble method spends about 1.3s for data testing, and other ensemble methods spend about 1s. Time spent on each method for the DApp monitoring data not high overall.

For SMD data in Figure 4.3, we can see that the F1 score of the IForest is 0.752, which is higher than the maximum, average, and weighted average ensemble methods. Ensemble methods rely heavily on base methods, and other base methods (KNN, LOF, and OCSVM) perform poorly. The most important thing is that the deep ensemble has the best F1 score, 0.780, which is higher than other methods, showing its superior detection ability by integrating information non-linearly. Figure 4.4 presents the time spent of these methods. We can see that the maximum, average, and weighted average ensemble spend about 36s, and the deep ensemble spends about 37s. Still, ensemble methods rely on base methods, so their time spent is mainly because of the kernel function calculation in OCSVM and the computational cost of the neural network.

For Vichalana data in Figure 4.3, we can see that the F1 scores of the maximum and average ensembles are higher than OCSVM, which shows the detection performance improvement of ensemble-based methods. In contrast, the weighted average ensemble does not assign weights well. In addition, the deep ensemble has the best F1 score, 0.830, which greatly improves detection accuracy compared with other methods and shows the advantages of the non-linear combination of base methods. Figure 4.4 presents the time spent on these methods. We can see that the maximum, average, and weighted average ensemble spend about 100s, and the deep ensemble spends about 96s. The time spent is still mainly because the large-scale data makes the kernel function calculation in OCSVM time-consuming. In addition, the neural network's computational cost takes a little time.

Table 4.2: Rank results of algorithm robustness

Method	IForest	KNN	LOF	OCSVM	Emsemble _max	Ensemble _avg	Ensemble _w_avg	**Deep _ensemble**
DApp monitoring data	4	3	8	7	6	5	2	**1**
SMD	2	7	8	6	3	5	4	**1**
Vichalana data	5	7	8	4	2	3	6	**1**
Average rank	3.667	5.667	8	5.667	3.667	4.333	4	**1**
Robustness score	0.619	0.333	0	0.333	0.619	0.524	0.571	**1**

As for algorithm robustness, we provide rank results in Table 4.2. We rank the detection accuracy of all methods, including best base methods and methods

in the ELBD framework, and calculate their average rank and robustness score, respectively. In the Table, we can see that the deep ensemble method has the best detection accuracy on the three different datasets, showing that it has not only superior detection accuracy but outstanding robustness for different data distributions. Other ensemble methods rely heavily on base detection methods, showing poor robustness. In conclusion, we can say that the deep ensemble method in the ELBD framework improve detection performance in terms of detection accuracy and robustness.

E2: Comparison with other deep detection methods.

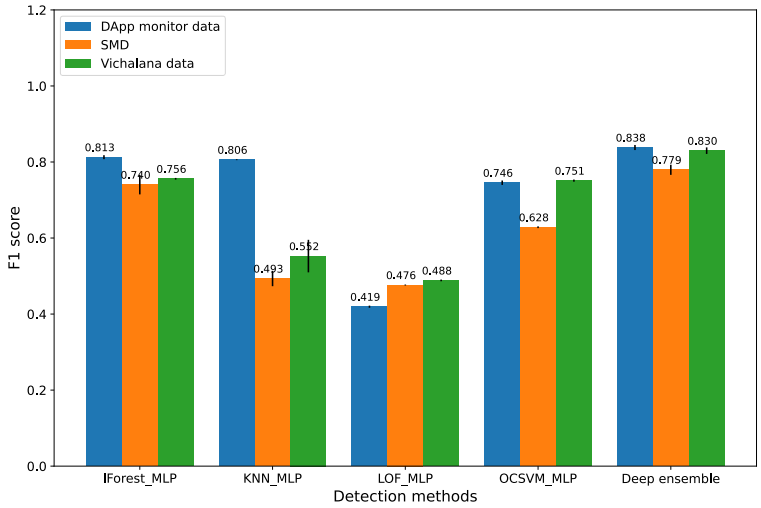


Figure 4.5: Detection accuracy of deep detection methods

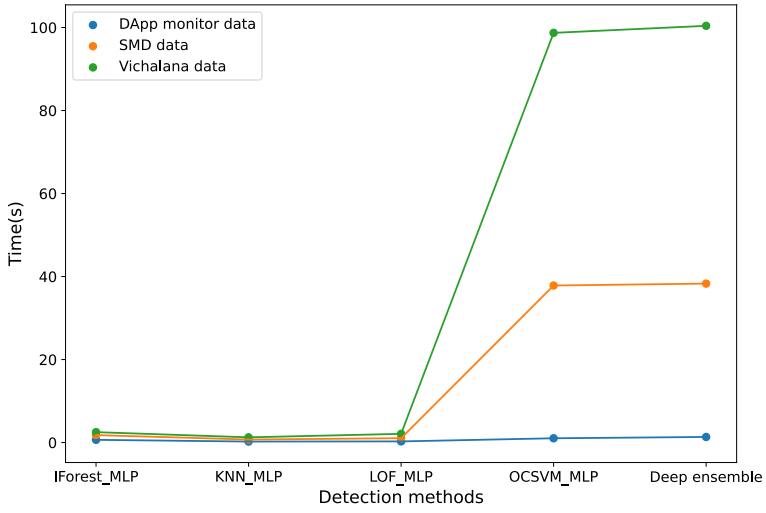


Figure 4.6: Time spent of deep detection methods

The deep ensemble method needs fewer labels to train, making it unfair to compare it with unsupervised methods. Therefore, we design experiments to

compare its performance with other weakly-supervised detection methods. We create deep detection models by combining each base learner with an MLP and comparing their performance with the deep ensemble method at first. In addition, we also extend the deep ensemble method by replacing MLP with CNN and LSTM, and we provide a comparison of their detection performance.

The comparison results of detection accuracy is shown in Figure 4.5. We can see that the deep ensemble method has the highest F1 score for the three datasets, showing that the deep ensemble method achieves the best detection accuracy, and it is a robust method that can be applied to different datasets. In Figure 4.6, we show the test time for each method. We can see that for the DApp monitoring data, the test time of all methods is similar and small, around 1s. For the SMD and Vichalana data, the test time for OCSVM-based and deep ensemble methods are similar and significant higher than other methods, because of time-consuming OCSVM kernal function calculation and computational cost of the neural network.

As for replacing MLP with CNN and LSTM, the comparison of their detection performance can be seen in Table 4.3. We can see that for both the DApp monitoring data and SMD data, the deep ensemble (MLP) has the highest F1 score, and it benefits from having more parameters. The deep ensemble (CNN) has the lowest F1 score because the pooling layer in CNN will compress information, and this does not suit time-series data very well. The deep ensemble (LSTM) performs similarly to the deep ensemble (MLP) because it can effectively capture long-term dependencies in time-series data, which is well-suited for this type of data. As for spent time, we can see that these deep learning methods take similar time for test data. For Vichalana data, we can see that the deep ensemble (LSTM) has higher detection accuracy and lower spent time, showing that it maybe more suitable for large-scale data. As a result, we can see that the deep ensemble method can be extended easily, and replacing the MLP with other deep learning methods may improve detection accuracy.

Table 4.3: Performance of different deep ensemble methods

Detection methods	DApp monitoring data		SMD data		Vichalana data	
	F1 score	Time(s)	F1 score	Time(s)	F1 score	Time(s)
Deep ensemble (MLP)	0.838±0.007	1.352±0.038	0.780±0.013	38.303±0.097	0.830±0.009	100.402±0.043
Deep ensemble (CNN)	0.832±0.003	1.402±0.073	0.764±0.019	40.747±0.036	0.812±0.009	96.262±0.100
Deep ensemble (LSTM)	0.827±0.006	1.846±0.142	0.765±0.014	42.139±0.219	0.833±0.002	94.782±0.047

E3: Impact of different amounts of labels. The deep ensemble needs to train with fewer labels. We use 10% labels for all the experiments above. Here, we design an experiment to test the impact of amounts of labels for detection ability of the deep ensemble method. We conduct experiments on the three datasets, and the results can be seen in Table 4.4.

For the DApp monitoring data, SMD data and Vichalana data, we set different amounts of labels (10%, 30%, 50%, 70%, 90%) to train the deep ensemble method.

Previous experiments show that with only 10% labels for training and testing for all data, the F1 score of the deep ensemble method is higher than all the base learners and linear ensemble methods. Furthermore, we can see that more labels are used for training, and the F1 score is higher, which is easy to explain given that more samples with labels provide more information to learn. Besides, the time spent for each dataset in the table is similar, and this shows that the amounts of labels for training the model have little effect on the test time. To conclude, the deep ensemble method can achieve superior performance with fewer labels, such as 10%, to train, and the trained models can be used on other data with high detection accuracy.

Table 4.4: Impact of amounts of labels on the DApp monitoring data and SMD data

Number of labels	DApp monitoring data		SMD data		Vichalana data	
	F1 score	Time(s)	F1 score	Time(s)	F1 score	Time(s)
10% labels	0.838±0.007	1.352±0.038	0.780±0.013	38.303±0.097	0.830±0.009	100.402±0.043
30% labels	0.866±0.009	1.298±0.042	0.792±0.015	37.890±0.038	0.837±0.008	103.123±0.161
50% labels	0.879±0.011	1.334±0.046	0.805±0.018	38.020±0.047	0.839±0.008	98.532±0.040
70% labels	0.887±0.012	1.293±0.046	0.807±0.012	39.104±0.041	0.841±0.010	99.071±0.121
90% labels	0.888±0.011	1.290±0.038	0.811±0.007	37.000±0.071	0.843±0.010	101.853±0.062

E4: Multi-step prediction of the deep ensemble method. With the deep ensemble method, we can predict multi-step performance anomalies. We mainly test its prediction ability on the DApp monitoring data. The time interval in the DApp monitoring data is 15s. Thus, we can use every 4 steps, which is 1 minute, as the prediction step. We use the first 1500 samples to train the model. Then, we predict whether the anomaly will happen or not after one or two or three minutes. To evaluate the prediction ability, we present the prediction accuracy with the F1 score in Figure 4.7.

In Figure 4.7, we can see that the longer the prediction time, the lower the detection accuracy, which means that it is difficult to predict long-term anomalies because dependency between data diminishes over time. In addition, we can see that all F1 scores are higher within four minutes than 0.8, which is good detection accuracy. Therefore, we can say that it is available for the deep ensemble to predict anomalies in the next four minutes with high accuracy. We also show the time spent testing the prediction ability in Figure 4.7. We can see that the testing time is around 1.1s, meaning that the deep ensemble method can predict anomalies quickly.

For all the detection methods, we provide a table 4.5 to compare their performance in terms of detection accuracy, algorithm robustness, and multi-step prediction. In the table, we can see that neither base detection nor linear ensemble methods have prediction ability. In addition, we can notice that IForest and weighted average ensemble methods have good detection accuracy and robustness. The most important thing is that the deep ensemble method perfectly

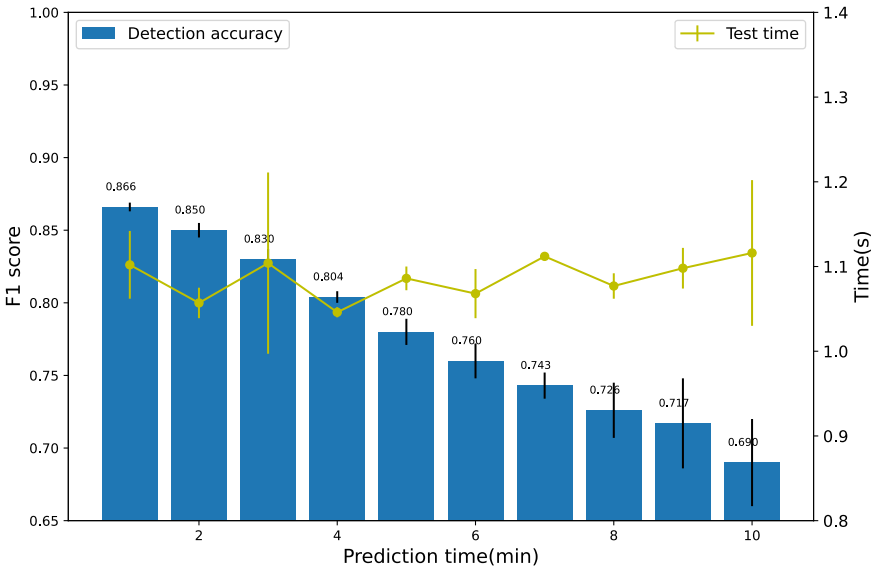


Figure 4.7: Prediction accuracy and time spent for different time steps of the deep ensemble method on the DApp monitoring data

addresses three challenges and has the highest *ARP_score* 5.166, which is much better than other methods.

Table 4.5: Comparison results of all detection methods

Challenge	Indicator	IForest	KNN	LOF	OCSVM	Emsemble _max	Ensemble _avg	Ensemble _w_avg	**Deep _ensemble**
Detection accuracy	F1 score	0.734	0.642	0.525	0.673	0.745	0.719	0.717	**0.816**
Algorithm robustness	Robustness score	0.619	0.333	0	0.333	0.619	0.524	0.571	**1**
Multi-step prediction	Prediction score	-	-	-	-	-	-	-	**3.350**
ARP_score		1.353	0.975	0.525	1.006	1.364	1.243	1.288	**5.166**

In conclusion, we provide the performance evaluation of ensemble methods in the ELBD framework. Our experiments show that these methods improve detection accuracy and robustness by integrating extracted information from base methods. Among those, the deep ensemble method has superior detection performance in terms of accuracy, robustness, and multi-step prediction. In addition, the deep ensemble method can predict anomalies in the next four minutes with high accuracy. While the deep ensemble method demonstrates the superiority of deep learning methods, its improvement is limited as it depends on the partial information extracted by base detection methods. For instance, algorithms like IForest, LOF, and OCSVM primarily emphasize the feature dimension rather than the time dimension in time-series data, which limit their detection accuracy [90]. To enhance the accuracy and robustness of anomaly detection, we propose an advanced deep learning-based method that leverages the strengths of deep

learning techniques.

4.3 A robust correlative-GNN-based Approach

Effective performance anomaly detection requires processing irregular data with accident fluctuations and avoiding false positive detections. Figure 4.8 shows the normal fluctuations of underlying resources. The red box part may be detected as an anomaly due to fluctuating CPU and memory usage, but during this period the system is actually healthy and both disk IOps and network traffic are fluctuating steadily. To avoid high FPR caused by this situation, it is vital to consider correlations between variables in multivariate time series to differentiate normal fluctuations from anomalies.

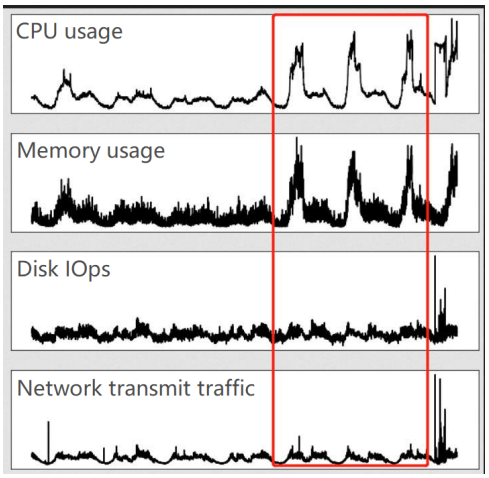


Figure 4.8: An example of multivariate time series. The red box represents normal fluctuations.

Deep learning makes it possible to extract information from unstructured data. The GNN is good at processing graph structure data and mining inter-dependencies between nodes. We can use GNN to mine inter-dependencies from both feature and time dimensions for multivariate time series. To improve detection robustness, we need to extract as much information as possible from multivariate time series. Multi-head self-attention has been developed to extract context information in sequence data, and Gate Recurrent Unit (GRU) is good at capturing long-term temporal dependencies. In addition, the Autoregressive (AR) model can be used to maintain linear relations in time series. Based on these ideas, we propose a Correlative-GNN with Multi-Head Self-Attention and Auto-Regression Ensemble Method (CGNN-MHSA-AR) for unsupervised multivariate time series anomaly detection in cloud computing systems and efficient abnormal explanation results are achieved on five public datasets. In this sec-

tion, we provide the problem statement of unsupervised multivariate time series anomaly detection and introduce the overall architecture of our model in detail.

4.3.1 Problem statement

We define multivariate time series in a cloud computing system as $X = \{x_1, x_2, \cdots x_n\}$, where n is the number of timestamps in a sliding window. We also define $x_t = \{v_t^1, v_t^2, \cdots v_t^m\}$ as a vector at time t, where m represents the number of features. For data $X \in R^{n \times m}$, the task of multivariate time series anomaly detection is to learn the characteristics of data X and determine whether an observation x_{n+1} is anomalous or not.

For multivariate time series anomaly explanation, our goal is to find the root cause of the anomaly. Having located the abnormal time point $x_t = \{v_t^1, v_t^2, \cdots v_t^m\}$ in the test set, we have to determine which feature at that time point is abnormal.

Typical unsupervised deep learning methods usually train and model normal data of multivariate time series and identify abnormal points through high reconstruction errors, such as the LSTM-VAE[177]. The LSTM-VAE replaces the feed-forward network in a VAE with LSTMs to extract temporal dependencies but ignores feature inter-dependencies and contextual information in multivariate time series. However, the LSTM-VAE only models normal data distributions and lacks information extracted from complex abnormal data, which may increase the FPR in anomaly detection.

To improve the detection accuracy and robustness of multivariate time series anomaly detection, we first provide two parallel GNNs to learn inter-dependencies in both feature and time dimensions. We then integrate multi-head self-attention to capture context information, GRU to extract long-term dependency, and AR model to maintain linear relations in multivariate time series. We will introduce the overall architecture and detailed modules next.

4.3.2 Overall architecture

The overall architecture of CGNN-MHSA-AR is shown in Figure 4.9, which is composed of the following modules in order:

- Data preprocessing: we perform data preprocessing with data normalization and denoising for original multivariate time series.
- 1D-CNN feature extraction: we use the one-dimensional convolutional layer to extract local patterns of each feature in preprocessed data.
- Correlation calculation based parallel GNNs: we provide parallel GNNs to learn inter-dependencies in multivariate time series from feature and time dimensions, and correlation calculation is used in this module.

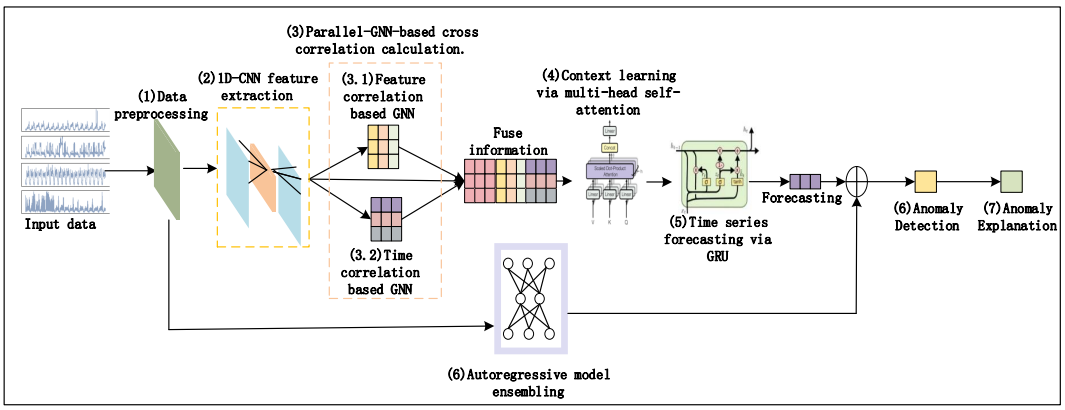


Figure 4.9: Overall architecture of CGNN-MHSA-AR.

- Context learning via multi-head self-attention: we use a multi-head self-attention to extract context information in multivariate time series.
- Time-series forecasting via GRU: we use a GRU to capture the dependencies between different time series.
- Autoregressive model ensembling: we utilize an AR model to maintain linear relations in original data.
- Anomaly Detection and Explanation: we utilize anomaly scoring functions for anomaly detection and anomaly explanation.

4.3.3 Modules of CGNN-MHSA-AR

Data preprocessing. For original data, we apply *min-max* data normalization to ensure that all data has the same scale. In addition,we adopt the Fourier transform to denoise data as shown in Figure 4.9.(1). We treat each row of time series as univariate time series and use Fast Fourier Transform (FFT) to denoise and replace detected noise with zero.

1D-CNN feature extraction. The one-dimensional convolution neural network (1D-CNN) is widely used in sequence processing because it is good at recognizing local patterns of sequences. For preprocessed multivariate time series, we use the 1D-CNN with kernel size 7 to extract information along the time dimension, as shown in Figure 4.9.(2).

Correlation calculation based parallel GNNs. Generally, given a graph, we can use GNN to get new representations of nodes by considering inter-dependencies between nodes. For multivariate time series, inter-dependencies in data can be exploited with correlation calculation in both feature and time dimensions. However, traditional correlation calculation methods, such as Pearson correlation, do

not work when a cloud computing platform is running stable, because multivariate time series will keep certain values unchanged[139]. Therefore, the correlation in time series is calculated by using the inner product of vectors, and we will consider both feature and time correlation calculations.

For feature correlation calculation, we define a feature as a vector $x_i = \{v_i{}^1, v_i^2, \cdots, v_i{}^n\}$, where n is the number of timestamps in a sliding window. We utilize the inner product between different vectors to get correlations of different features. We use $c_{ij}^{feature}$ to represent the correlation between feature i and j, and the formula is as follows:

$$c_{ij}^{feature} = \frac{\sum_{t=0}^{n} v_i^t v_j^t}{k} \tag{4.9}$$

where k represents the rescaling factor, and equal to the length of a sliding window in this work. We calculate feature correlations of all features in the multivariate time series and finally obtain the feature correlation matrix with a shape of $m \times m$. m represent the number of features in the multivariate time series.

For time correlation calculation, we define the vector at time i as $x_i = \{v_i{}^1, v_i^2, \cdots, v_i{}^m\}$, where m is still the number of features in the multivariate time series. We can use c_{ij}^{time} to represent the correlation between time i and j, and the formula is as follows:

$$c_{ij}^{time} = \frac{\sum_{k=0}^{m} v_i^k v_j^k}{\lambda} \tag{4.10}$$

where λ represents the rescaling factor, and equal to the length of a sliding window. We calculate time correlations between of all timestamps and finally obtain the time correlation matrix with a shape of $n \times n$. n represents the length of a sliding window.

As shown in Figure 4.9.(3.1), we use feature correlations between a vector i with others as weights of a GNN and calculate the output representation of the vector as follows:

$$h_i = \sigma(\sum_{j=1}^{m} c_{ij}^{feature} v_i^j) \tag{4.11}$$

where σ represents the sigmoid activation function. Similarly, for Figure 4.9.(3.2), we use time correlations between a vector i with others as weights of a GNN and calculate the output representation of the vector as follows:

$$h_i = \sigma(\sum_{j=1}^{n} c_{ij}^{time} v_i^j) \tag{4.12}$$

To fuse different information, we concatenate the output representations based on feature correlation and time correlation as well as the convoluted representation to a matrix with a shape of $n \times 3m$, where each row represents a $3m$ dimensional feature vector for each timestamp.

Context learning via multi-head self-attention. As shown in Figure 4.9.(4), to make the network's complexity scales with the input size, we set the multi-head self-attention mechanism with m heads and $3m$ embedding dimension to learn different contextual information from data. where m is the number of features in the multivariate time series.

Time series forecasting via GRU. After multi-head self-attention, we use a GRU to extract time dependence in time series, as shown in Figure 4.9.(5). We set the neurons of GRU are 150 and the number of layers is the default value of 1. Finally, we use all n data in a sliding window to predict the value at the next timestamp. We use $x_{n+1,i}^{forecast}$ to represent the predicted value of the i-th feature at the time n+1.

Autoregressive model ensembling. Due to the nonlinearity of convolutional, multi-head self-attention, and GRU modules, the output is not sensitive to original input[103]. To address this drawback, we apply a first-order AR model to preprocessed data and directly integrate its predicted value with the output after GRU. We use $x_{n+1,i}^{AR}$ to represent the predicted value of the i-th feature at the time n+1 after the AR model. The final prediction result can be formulated as follows:

$$\hat{x}_{n+1,i} = \alpha x_{n+1,i}^{forecast} + (1-\alpha) x_{n+1,i}^{AR} \tag{4.13}$$

where α is to adjust the nonlinear and AR prediction results. In this work, we set $\alpha = 0.5$.

Finally, we define the root mean square error as loss function:

$$Loss = \sqrt{\sum_{i=1}^{m} \left(x_{n+1,i} - \hat{x}_{n+1,i}\right)^2} \tag{4.14}$$

Anomaly detection. After training the model, we get the predicted value $\hat{x}_{n+1}$ at time n+1. We follow [176] to use the error between the actual value x_{n+1} and the predicted value as the anomaly score, and the formula is as follows:

$$S_{n+1} = \frac{1}{m}\sum_{i=1}^{m} s_{n+1}^{i} = \frac{1}{m}\sum_{i=1}^{m} \sqrt{\left(x_{n+1,i} - \hat{x}_{n+1,i}\right)^2} \tag{4.15}$$

We identify a timestamp as an anomaly if its anomaly score is larger than a threshold.

4.3.4 Experiments and analysis

In this section, we conduct experiments to evaluate the anomaly detection accuracy and performance of abnormal explanation of CGNN-MHSA-AR. We first compare the detection performance of CGNN-MHSA-AR with baseline methods

on seven public datasets. Then we provide ablation experiments to analyze the importance of modules in CGNN-MHSA-AR. Finally, we utilize five datasets to test the ability of abnormal explanation of CGNN-MHSA-AR.

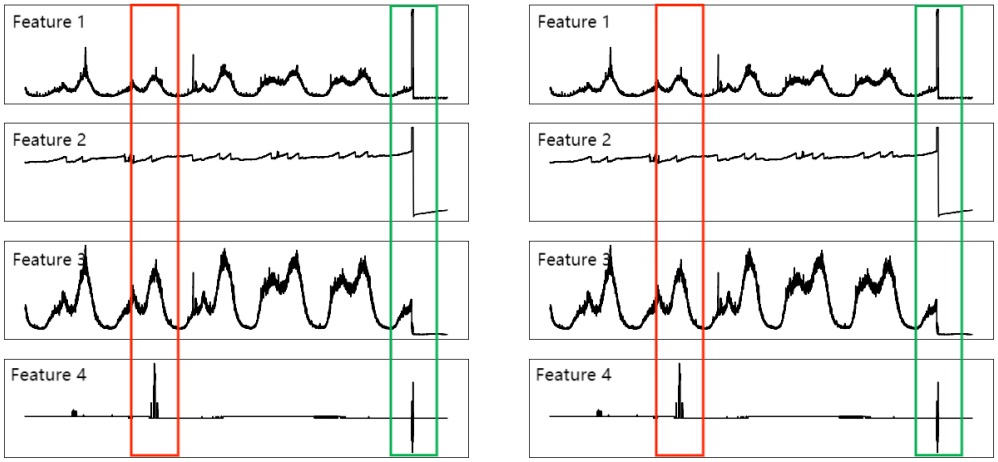
Datasets

We use seven public datasets: SMD and its four sub-datasets, Soil Moisture Activate Passive satellite (SMAP), and Mars Science Laboratory rover (MSL). SMD is a five-week-long real-time dataset of 28 cloud platform servers[218], which contain 708405 data points from the training set and 708420 data points from the testing set. The anomaly rate of SMD is 4.16%. In this dataset, we found that the four datasets, machine-1-3 (the training set has 23702 data points and the testing set has 23703 data points), machine-1-8 (23698 data points in the training and test sets each), machine-2-6 (28743 data points in the training and test sets each), and machine-3-5 (the training set has 23690 data points and the testing set has 23691 data points), did not perform well in many detection models because the irregular fluctuations in the data would lead to false positives in abnormal detection. We analyze that false positive anomaly detection is caused by irregular fluctuations in data, as shown in Figure 4.10. Figure 4.10(a) and Figure 4.10(b) illustrate four-feature segments for machines 1-3 and 1-8, respectively. It is possible to detect anomalies in the red box in Figure 4.10(a) because of fluctuations in Feature 4. While features 1, 2, and 3 fluctuate steadily, the system is actually operating in a healthy state. Each feature has apparent fluctuations when we look at the green box, which represents abnormal data segments. Figure 4.10(b) shows a similar scenario. Due to fluctuations in features 1 and 2, anomalies may be detected in the red box, which will lead to lower detection accuracy. Therefore, we use these four datasets to prove that our model can effectively resolve the false positive issue and improve detection accuracy.

The SMPA and MSL are spacecraft datasets provided by NASA [218]. The abnormal rate is 13.13% and 10.72%, respectively. For MSL, we choose 28317 data points from the training set and 20000 data points from the testing set. And a total of 20000 data points are selected from the training set and 20000 from the testing set in SMAP. We use SMAP and MSL to prove that CGNN-MHSA-AR has good anomaly detection ability in other fields.

Experimental settings

We use Python 3.7 and CPU-only PyTorch 1.11.0. We set the sliding window size as $n = 100$, the 1D-CNN with kernel size 7, the neurons of GRU are 150 and the number of layers is the default value of 1, the multi-head self-attention mechanism with m heads and $3m$ embedding dimension. To train CGNN-MHSA-AR, we set epochs as 10, batch size as 256, learning rate as 0.001, dropout as 0.4 and we use the Adam optimizer.



(a) A sample segment in SMD machine-1-3 dataset (b) A sample segment in SMD machine-1-8 dataset

Figure 4.10: Typical segment in datasets that has both normal fluctuations and anomalies. An anomalous data set is displayed in green box while a normal data set is displayed in red box.

Experimental results

E1: Performance of anomaly detection. We provide comparison results between CGNN-MHSA-AR with baseline methods on seven public datasets and an analysis of their detection performance.

Comparison results. For comparison, we select seven baseline methods, including five statistical methods (from PYOD[280]): LODA, IForest, CBLOF, HBOS and DeepSVDD; and two deep learning methods: MTAD-GAT[279] and GDN[58]. We calculate the best-f1 of each model and present the comparison results in Table 4.6.

Table 4.6: Comparison of anomaly detection performance for multiple detection methods on seven datasets.

Method	machine-1-3			machine-1-8			machine-2-6			machine-3-5			SMD			MSL			SMAP			F1 Average Rank
	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1	
LODA	0.348	0.581	0.435	0.476	0.460	0.468	0.996	0.651	0.787	0.725	0.694	0.709	0.555	0.729	0.631	0.931	0.647	0.764	0.509	0.999	0.674	4.857
CBLOF	0.974	0.925	**0.949**	0.663	0.424	0.517	0.997	0.959	**0.978**	0.838	0.938	0.885	0.600	0.761	0.671	0.854	0.925	0.888	0.435	0.999	0.607	2.571
HBOS	0.987	0.401	0.571	0.252	0.463	0.326	0.851	0.660	0.743	0.507	0.999	0.340	0.992	0.198	0.330	0.205	0.999	0.341	0.396	0.999	0.568	6.714
IForest	0.947	0.924	**0.935**	0.472	0.853	0.608	0.911	0.898	**0.904**	0.614	0.704	0.656	0.601	0.505	0.549	0.669	0.925	0.775	0.302	0.999	0.464	4.000
DeepSVDD	0.468	0.806	0.592	0.709	0.854	0.775	0.702	0.905	0.791	0.832	0.971	0.897	0.788	0.334	0.470	0.981	0.730	0.837	0.462	0.272	0.342	4.429
GDN	0.285	0.558	0.377	0.461	0.250	0.324	0.874	0.294	0.440	0.444	0.617	0.518	0.227	0.151	0.181	0.205	0.942	0.345	0.197	0.917	0.343	7.571
MTAD-GAT	0.449	0.873	0.593	0.599	0.433	0.503	0.885	0.620	0.703	0.725	0.584	0.647	0.642	0.773	0.702	0.974	0.944	**0.959**	0.427	0.999	0.598	4.143
CGNN-MHSA-AR	**0.837**	**0.870**	**0.853**	**0.712**	**0.956**	**0.816**	**0.777**	**0.999**	**0.875**	**0.918**	**0.955**	**0.936**	**0.839**	**0.867**	**0.853**	**0.906**	**0.944**	**0.925**	**0.727**	**0.999**	**0.842**	**1.714**

Table 4.6 shows that CGNN-MHSA-AR outperforms all other methods on

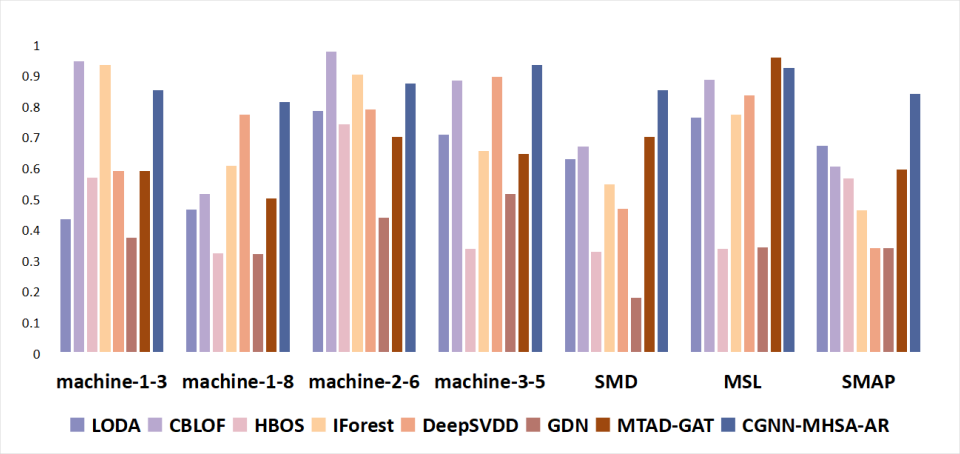


Figure 4.11: F1-score of *CGNN-MHSA-AR* and all baseline models.

machine-1-8, machine-3-5, SMD, and SMAP. The average F1 on these four datasets can reach 86.1%. The F1 of CGNN-MHSA-AR is slightly lower than the best baselines on machine-1-3, machine-2-6, and MSL. CGNN-MHSA-AR outperforms the state-of-the-art method (MTAD-CAT) on all six datasets except the MSL dataset and F1-scores are relatively increased by 26%, 31%, 17%, 28%, 15%, 24%. The robustness of CGNN-MHSA-AR is much better than all baselines because the precision of CGNN-MHSA-AR is above 0.7, and the recall is above 0.85 on all seven datasets, while no baseline can achieve this. In terms of the average ranking of F1-score, CGNN-MHSA-AR also performs best.

As shown in Figures 4.11, we can see that the CGNN-MHSA-AR performs well in the F1-score, and the fluctuations of CGNN-MHSA-AR on seven datasets are all small, which proves the excellent robustness of CGNN-MHSA-AR. Furthermore, on average, as shown in Figure 4.12, CGNN-MHSA-AR has the best ranking across the three evaluation metrics for all datasets.

Performance analysis. Baseline methods have different performances on these public datasets. LODA is a lightweight anomaly detector that is very practical in sensor failure. LODA consists of multiple one-dimensional histograms, which approximate the probability density of the input data and project it into a single vector. A low density indicates a larger outlier in the sample. However, LODA's insufficient dependency extraction between time series results in its poorer performance than CGNN-MHSA-AR.

Based on the characteristics of a sample, HBOS divides it into multiple intervals, and intervals with fewer samples are more likely to be outliers. HBOS performs well in global anomaly detection but cannot detect local outliers. GDN analyzes sensor relationships based on a graph and then identifies deviations from learned patterns. However, GDN ignores correlations in the time dimension, which makes it hard to predict various behaviors. CGNN-MHSA-AR extracts correlations between different times and features in parallel, making it perform

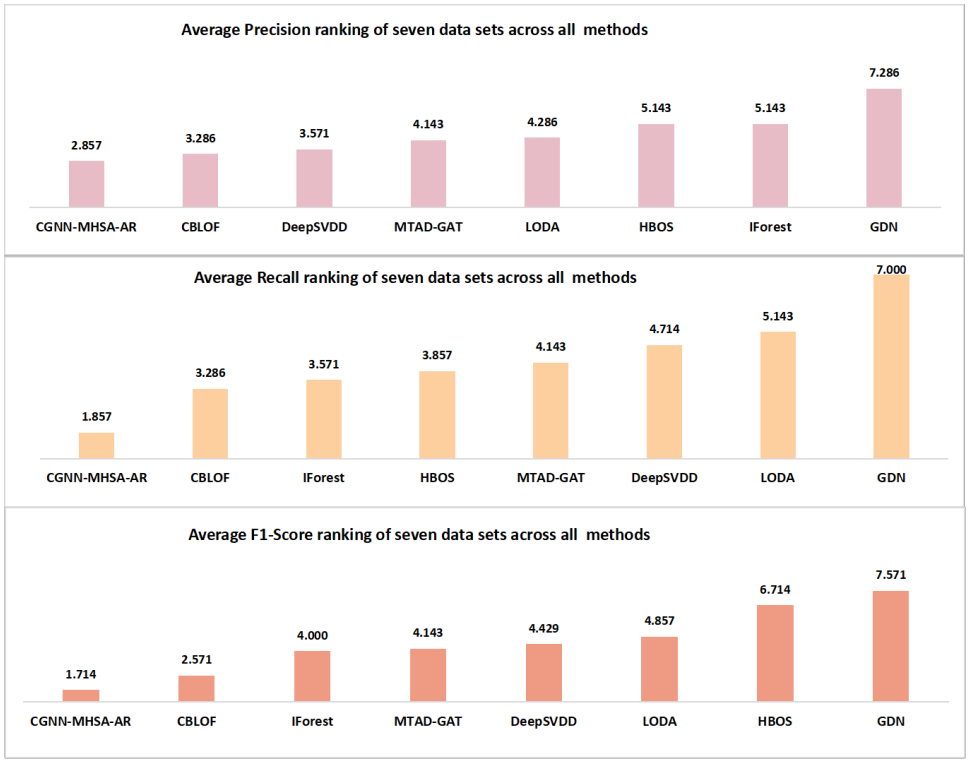


Figure 4.12: Average Precision, Recall, F1-score ranking of seven data sets across all methods.

better than GDN and HBOS.

CBLOF is a cluster-based local outlier detector that uses clusters to identify dense regions in data and then performs a density estimate for each cluster. When a data point deviates significantly from most data, it is considered abnormal. IForest defines anomalies as sparsely distributed points far away from groups with high density. Due to the small density of abnormal points, the tree model can easily detect abnormal points. Consequently, the abnormal data will be closer to the root of the isolation tree when it is established, while the normal data will be farther from it. The performance of CBLOF and IForest on machine-1-3 and machine-2-6 is better than CGNN-MHSA-AR because of the low density of outliers in these two datasets.

MTAD-GAT considers each univariate time series as a feature. It uses two graph attention layers simultaneously to learn both the temporal and feature dimension dependencies of multivariate time series. However, MTAD-GAT ignores maintenance of linear relationship for dynamic periodic data, resulting in a decrease in prediction accuracy. In contrast, the CGNN-MHSA-AR addresses this problem by adding the AR model's prediction results. For the MSL dataset, MTAD-GAT extracts correlations between data with a better weight matrix, leading to a slightly better performance than CGNN-MHSA-AR.

E2. Ablation experiments We conduct ablation experiments to analyze the influence of convolutional layers, correlation calculation of feature and time dimensions, GRU, and AR model in CGNN-MHSA-AR. Table 4.7 shows experimental results.

Table 4.7: Ablation Study:F1-Score for CGNN-MHSA-AR and its ablated versions

Method	machine-1-3	machine-1-8	machine-2-6	machine-3-5	SMD	MSL	SMAP
CGNN-MHSA-AR	**0.853**	**0.816**	**0.875**	**0.936**	**0.853**	**0.925**	**0.842**
w/o conv	0.851	0.794	0.842	0.929	0.849	0.883	0.811
w/o time-correlation	0.778	0.790	0.827	0.929	0.850	0.917	0.701
w/o feature-correlation	0.616	0.758	0.847	0.927	0.852	0.882	0.820
w/o GRU	0.763	0.789	0.801	0.911	0.851	0.923	0.815
w/o ar	0.473	0.487	0.770	0.649	0.681	0.793	0.636

The influence of convolutional layers. The impact of convolutional layers on the anomaly detection ability of our proposed model can be seen in Table 4.7 w/o conv, and the best-f1 is reduced by an average of 2%. For machine-2-6, the best-f1 is reduced most, 3.3%. Therefore, these results show that convolutional layers can help CGNN-MHSA-AR to better extract correlations between the temporal dimension and the feature dimension via convolution.

The influence of correlation calculation. We verify the effect of correlation calculation on CGNN-MHSA-AR by removing the time and feature correlations, denoting w/o time-correlation and w/o feature-correlation in Table 4.7. We can discover that in w/o time-correlation, the best-f1 value is reduced by 4.4% on average. For SMAP, the correlation calculation of the time dimension has the greatest impact, and the best-f1 is reduced by 14.1%. In w/o feature-correlation, the best-f1 is reduced by 5.6% on average. For machine-1-3, the correlation calculation for the feature dimension has the greatest impact, and the best-f1 is reduced by 23.7%. From these results, we can see that feature and time correlation calculations can extract inter-dependencies between non-adjacent vectors, which plays a crucial role in the final performance of our model.

The influence of GRU. We verify the effect of GRU in our model by removing GRU, denoting w/o GRU in Table 4.7. We can see that the average best-f1 value drops by 3.5% when we remove the GRU. The GRU has the most significant impact on machine-2-6, and the best-f1 drops by 7.4% after removing the GRU. The GRU can extract the time dependency in time series and output the predicted value considering the hidden status of previous data. The gated structure ensures that important information will not disappear during long-term propagation, which makes GRU improve the abnormal detection ability of CGNN-MHSA-AR.

The influence of AR model. We show the effect of an AR model on the anomaly detection ability of CGNN-MHSA-AR in Table 4.7 w/o ar. We can see that the best-f1 is reduced by 23% on average, with the greatest impact on machine-1-3, which has a 38% reduction in best-f1 value. The AR model

predicts data based on previous data linearly. We add the AR model because the output after GRU in CGNN-MHSA-AR is not sensitive to the original data. Experimental results show that the AR model improves the anomaly detection performance of our model.

4.4 Conclusions

This chapter focuses on performance anomaly detection of distributed applications, which need to satisfy two challenging requirements: high detection accuracy and robustness. Based on our survey, many machine learning-based methods have been developed for performance anomaly detection. However, these detection methods have inconsistent performance for different datasets and rarely simultaneously solve the three requirements. Therefore, based on existing performance anomaly detection methods, we provide the ELBD framework that integrates existing detection methods, and the CGNN-MHSA-AR for unsupervised anomaly detection.

Considering that base detection methods (IForest, KNN, LOF, OCSVM) perform differently on datasets with different data patterns, we develop an ELBD framework (maximum, average, weighted average, and deep ensemble) that integrates existing detection methods for improving detection performance. Our experiments show that methods in the ELBD framework significantly improve detection accuracy and robustness, especially the deep ensemble method. In addition, the deep ensemble method has the multi-step prediction ability, which can predict anomalies in the next four minutes with high accuracy. We also evaluate detection performance with our indicator, and the results show that the deep ensemble method has the highest ARP_score 5.166, which is much better than other methods.

Accurately capturing the relationship between features and time series is crucial for multivariate time series anomaly detection. Therefore, in CGNN-MHSA-AR, we use two parallel GNNs to resolve false positive detection caused by irregular fluctuations in the data. Furthermore, considering complex data patterns in multivariate time series, we integrate multi-head self-attention, GRU, and the AR model to extract multiple-dimensional information and improve detection robustness. CGNN-MHSA-AR also provides the function of abnormal explanation. In experiments, we use seven public datasets to evaluate our model. In terms of best-f1 score, CGNN-MHSA-AR outperforms all baseline methods in seven datasets. Compared with the state-of-the-art baseline method, CGNN-MHSA-AR increases the best-f1 up to 31.3%. Furthermore, on the seven datasets, the precision of CGNN-MHSA-AR is above 0.7, and the recall is above 0.85, reflecting the excellent robustness of our model. The model has also been shown to be able to correctly determine the root causes of 74.1% of detected anomalies, a higher percentage than the state-of-the-art models.

This chapter provides two solutions for performance anomaly detection of distributed applications, and the results show that the AI-based method has superior performance in terms of detection accuracy, robustness. However, some aspects of this research can still be improved. For example, for applying AI methods to help operators and developers better implement performance management of cloud applications, detection efficiency can be explored more in the future [80].

Chapter 5

Root Cause Localization with Gradient-based Causal Inference Method

In chapter 3, we have demonstrated the feasibility of real-time metric-level root cause localization. However, our approach relied on the traditional method PC algorithm (named after its authors, Peter and Clark) for building the causal directed acyclic graph (DAG), and our experiments were conducted with only one type of anomaly. In this chapter, we aim to expand our exploration of root cause localization within the performance diagnosis framework by incorporating advanced methods and conducting comprehensive experiments. Causal Inference (CI) based methods have gained popularity recently for root cause localization, but currently used CI methods have limitations, such as the linear causal relations assumption and strict data distribution requirements. In addition, existing research primarily focuses on coarse-grained localization (only faulty services can be localized), but there is a growing interest in fine-grained root cause localization (localize indicative metrics on the faulty service). To tackle these challenges, we propose a root cause localization framework working with causal inference and named CausalRCA. The CausalRCA uses a gradient-based causal structure learning method to generate weighted causal graphs and a root cause inference method to localize root cause metrics, achieving fine-grained, automated, and real-time root cause localization. We conduct coarse- and fine-grained root cause localization to evaluate the localization performance of CausalRCA. Experimental results show that CausalRCA significantly outperforms baseline methods in localization accuracy.

This chapter is based on:

- **Ruyue Xin**, Peng Chen, and Zhiming Zhao. "Causalrca: Causal inference based precise fine-grained root cause localization for microservice applications." Journal of Systems and Software 203 (2023): 111724.

5.1 Introduction

Microservices architecture [13] builds cloud applications by decomposing the system functionalities into multiple independently deployable units, making distributed applications more resilient, robust, and adaptable to dynamic cloud environments. The performance of a microservice application is vital to guarantee the quality of user experience and service [251]. However, performance anomalies, such as degraded response time, are inevitable due to the large scale and complex dependencies of services, causing enormous economic loss and user dissatisfaction [137]. Furthermore, the performance of applications heavily depends on the underlying resources [84]; for example, high CPU usage results in a congested queue and growing latency [108]. In order to enable application operations to take actions to resolve performance anomalies effectively, root cause localization to identify faulty services or resources is at the core of software maintenance for online service systems [45].

A microservice application can be observed by monitoring tools, which help operators to detect performance anomalies[40, 212]. However, performance anomaly detection only notifies operators when an anomaly occurs. To effectively handle performance anomalies, operators need to be informed about where the anomaly occurs (e.g., the faulty service) and what causes the anomaly (e.g., the memory leak). Root causes of a performance anomaly can be localized at different granularity: coarse-grained and fine-grained [42]. Coarse-grained means that only faulty services can be identified, and the corresponding action will be the migration or restart of the entire service [244], which is simple and straightforward but may not recover anomalies and has a higher risk of affecting other services and longer recovery times [245]. The developer of Instana Autotrace[1] emphasized the importance of identifying anomaly locations and root causes to avoid delays associated with restarting services, as this may not solve anomalies[68].

At a fine granularity, root cause localization will identify not only the faulty service but also the underlying resources through monitoring metrics of the service [248]. Operators can choose accurate actions to mitigate the performance anomaly using fine-grained root cause when pinpointing indicative metrics on the faulty service [42, 247]. For example, service scale-out has a positive effect and shorter recovery time compared with service restart in the case of underlying resources being insufficient [245]. In industry, fine-grained root cause localization attracts much attention. The CCF AIOps Challenge[2], jointly organized by industry and academia, aims to solve problems in real IT operations scenarios based on production systems of industrial companies (e.g., Sougo, eBay, Tencent) [138], providing the performance diagnosis challenge for microservice systems and requires localizing root causes at the metric level in 2020[3]. Instana Autotrace[1], Google Cloud

[1]https://www.instana.com/
[2]http://iops.ai/
[3]https://competition.aiops-challenge.com/home/competition/1484441527290765368

Operations[4] are commercial platforms to identify root causes to help developers and operators fix performance issues [233]. However, these platforms work with trace data that requires integrating tracking codes into applications and require time and expert technologies to analyze data. Monitoring data, which is different from trace data, can be readily collected and utilized for fine-grained root cause localization in microservice applications, aiding service operators in efficiently and cost-effectively identifying faulty services and pinpointing faulty metrics to resolve performance anomalies [42].

To track the root cause localization problem of microservice applications, some research has developed in recent years. We can classify them into log-based, trace-based, and metric-based according to the data sources [209]. Log-based [4] and trace-based [121, 240] research have limitations, such as complex real-time processing and information extraction. On the other hand, metric-based research uses real-time monitoring data, including service latency and system resources, and focuses on localizing faulty services and resource metrics. This kind of research can assist anomaly recovery in taking actions like resource scaling easily without intervention of application source code [209]. Nowadays, most metric-based research is coarse-grained [85, 141, 157, 158, 159, 235], and fine-grained root cause localization is starting to catch the attention of researchers [165, 246]. As for localization methods in metric-based research, causal inference (CI) based methods that can model causal effects between services have been developing recently. For example, CauseInfer [41] applies the PC algorithm (named after its authors, Peter and Clark) [214], and MicroDiag [246] uses the linear non-Gaussian acyclic model (LiNGAM) [202] to obtain causal graphs of metrics, which can be seen as anomaly propagation paths. However, currently used CI methods have limitations, such as uncertainty about some causal relations between metrics and strict assumptions about input data and causal relations [248]. Therefore, advanced CI methods can be considered for fine-grained root cause localization to discover anomaly propagation paths and improve localization performance.

Fine-grained root cause localization in a microservice application is challenging because 1) services are often heterogeneous and have different characteristics, which may result in diverse anomaly symptoms for the same issue; 2) the complex dependency between microservices makes it difficult to model the anomaly propagation resulting from faulty services; 3) a large number of anomalous metrics introduced in a system makes it hard to find out the root one for a performance anomaly. To address these challenges, we formulate our main research question: *how to pinpoint the root cause of performance anomalies at a fine granularity based on monitoring data?* Three sub-questions are proposed:

- How to model anomaly propagation between monitoring metrics using CI methods?

[4] https://cloud.google.com/products/operations

- How to precisely determine the root cause based on the propagation model?

- How to evaluate the performance of root cause localization result?

To answer the research question, we propose a CI-based fine-grained root cause localization framework named CausalRCA for microservice applications in this chapter. The framework activates when an anomaly is detected. Based on real-time monitoring data, CausalRCA will perform automatic root cause localization, including modeling anomaly propagation paths as a causal graph and ranking metrics to localize the root cause by traversing along the graph. Finally, CausalRCA outputs predicted root causes, which can be used by operators to determine strategies and recovery actions to solve the anomaly. We evaluate the localization performance of CausalRCA on the Sock Shop microservice benchmark. When a performance anomaly in the Sock Shop, such as the high response time of user requests, is detected, we can input monitoring metrics, including service latency and resource metrics of each service, to CausalRCA. After processing, the faulty service and root cause metric, for example, the memory usage metric in the order service, will be identified. Our experimental results show that CausalRCA improves localization accuracy. For example, the average improvement of $AC@5$ for the fine-grained root cause metric localization in the faulty service is 9.4% compared with baseline methods.

Our contributions can be summarized below:

- We propose an automated, fine-grained root cause localization framework named CausalRCA, which analyzes monitoring data and localizes faulty services and system resources in real-time.
- We provide a gradient-based causal structure learning method in CausalRCA, which can automatically capture linear and non-linear causal relations between monitoring metrics.
- We conduct coarse- and fine-grained experiments to evaluate the localization performance of CausalRCA and demonstrate that the proposed framework has the best localization accuracy compared with baseline methods. For example, the average $AC@3$ is 0.719, which is a 10% improvement compared with baseline methods.

The rest of the chapter is organized as follows. In Section 5.2, we review existing root cause localization research and CI methods. In Section 5.3, we propose a framework for root cause localization and a detailed introduction of each method. In Section 5.4, we design experiments from coarse-grained to fine-grained to evaluate the localization performance of our framework. Finally, discussion and conclusion are provided in Section 5.5 and Section 5.6.

5.2 Related works

In recent years, research has developed for root cause localization in distributed system [79], clouds [240, 247]. Based on data sources, we can categorize these researches into three groups: log-based, trace-based, and metric-based [209]. Log-based research [4] mainly localizes root causes based on text logs parsing, which is hard to work in real time. Trace-based research [121, 240] gathers information through the complete tracing of the execution paths and then identifies root causes along those paths. However, trace data only focuses on service level, and it is time-consuming for developers to understand source code well enough to extract trace information. In contrast, metrics-based research uses monitoring data collected from applications and underlying infrastructures to construct causal graphs and infer root causes. Metric-based research can achieve automated, real-time root cause localization based on multi-dimensional information.

Most metric-based research is focusing on coarse-grained root cause localization [85, 141, 157, 158, 159, 235, 247]. However, fine-grained root cause localization was proposed early and has begun to attract the attention of more researchers in recent years [42, 165, 246]. As for CI methods, we can see that causal structure learning methods like PC and LiNGAM are applied. At the same time, root cause inference methods, such as BFS and random walk, are popular. Based on these works, we consider precise root cause localization is more helpful for microservice application recovery from performance anomalies. At the same time, the localization accuracy of existing works can be improved; for example, the success rate of accurately identifying the root cause may be under 20% [141, 235]. Therefore, we are motivated to explore fine-grained root cause localization with advanced CI methods.

Causal inference methods, especially causal structure learning, have been researched for several years, and they play a vital role in many areas, such as genetics [179] and biology [191]. The causal structure learning problem can be formulated as to learn a directed acyclic graph (DAG) from observational data. Methods can be classified into constrained-based, score-based, function-based, and gradient-based. Constrained-based methods, such as PC and FCI [214], use conditional independence tests to learn the skeleton of the casual graph and then orient the edges based on pre-defined orientation rules. Score-based methods, like GES [46], assign scores to different causal graphs based on a pre-defined score function and then search over the space of DAGs to find the optimal one. Finally, function-based methods, like LiNGAM [201, 202], construct a linear Structural Equation Model (SEM) based on linear and non-Gaussian assumptions, and solve it to get the DAG. These traditional methods contribute much to causal structure learning but have limitations. PC usually has ambiguous causal relations in causal graphs. GES takes a long time to match graphs, which makes it inappropriate to be applied to large-scale data. LiNGAM has strict linear and non-Gaussian assumptions, which makes it impractical.

With the development of deep neural networks, gradient-based methods are developed. Zheng et al.[282] propose an equality constraint to the linear SEM, which enables a suite of continuous optimization techniques such as gradient descent. After that, Yu et al.[264] provide a deep generative model and apply a variant of the structural constraint to learn the DAG. Gradient-based methods have no limitation of input data, can deal with linear and non-linear causal relations in data, and can automatically generate a weighted DAG. Gradient-based methods have been applied to medical [232] and biology [56]. However, to the best of our knowledge, no research has applied gradient-based methods to root cause localization of microservice applications.

Based on DAGs generated by causal structure learning methods, researchers apply graph methods, like BFS [15], random walk [215], and PageRank [188], for root cause inference. The BFS is to traverse the graph and determine the abnormal node without descendants or with no abnormal descendants as a root cause. A random walk is walking through paths and choosing neighbors randomly in a graph. It determines the node most visited as the root cause. PageRank improves the random walk by adding the possibility of jumping to a random node, which will be used in this work.

In conclusion, metric-based research can achieve automated and real-time root cause localization compared with log- and trace-based research. However, most existing research is about coarse-grained faulty service localization, while fine-grained root cause metric localization can be more helpful for rapid recovery and loss mitigation. CI-based methods are popular, but currently used methods have their limitations. Therefore, this chapter will mainly focus on fine-grained root cause localization and explore gradient-based methods to build causal graphs.

5.3 Root cause localization framework

In this section, we propose a root cause localization framework named CausalRCA, including causal structure learning and root cause inference, and we will introduce detailed methods in the framework. All codes and data can be found in our Github repository CausalRCA[5].

5.3.1 Framework overview

The CausalRCA can automatically build anomaly propagation paths and localize root causes in real-time based on observable metrics. The CausalRCA framework consists of three components: monitoring metrics, causal structure learning, and root cause inference, as shown in Figure 5.1.

The CausalRCA works when an anomaly occurs, such as the high latency of user requests, and it then automatically builds anomaly propagation paths

[5]https://github.com/AXinx/CausalRCA_code.git

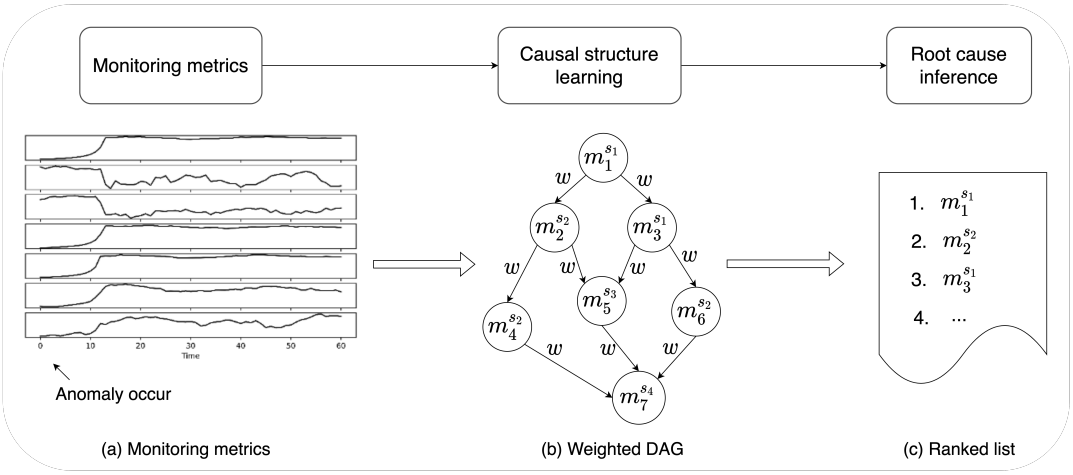


Figure 5.1: CausalRCA: details of the root cause localization framework

and localize root causes in real-time based on observable metrics. We first collect monitoring data, including service-level data, that is, service latency, and resource-level data, such as container CPU/memory usage. We use $m_i^{s_j}$ to represent a monitoring metric in the service s_j, and all monitoring data is time-series data as shown in Figure 5.1(a). Based on monitoring data, we then start the causal structure learning. The causal structure learning will automatically build a causal graph of metrics, which can be seen as anomaly propagation paths. We develop the causal structure learning with a gradient-based CI method, which can output a weighted DAG to represent causal relations between metrics as shown in Figure 5.1(b). With the DAG, we start root cause inference to localize root causes. We apply PageRank to the weighted DAG and output a ranked list of all metrics, as shown in Figure 5.1(c). Depending on the input data, the CausalRCA can be used for coarse- or fine-grained root cause localization. Coarse-grained works when input service latency, and CausalRCA will output the faulty service. Fine-grained works when inputting resource metrics, and CausalRCA will output the root cause metric. We evaluate the localization performance of CausalRCA in experiments in Section 5.4.

5.3.2 Causal structure learning

The causal structure learning component aims to build a causal graph of monitoring metrics. The causal graphs can be seen as anomaly propagation paths between metrics. We can use a DAG to represent the causal graph, in which each node represents a metric, and each edge represents a cause-effect relationship. Based on related work, we know that traditional causal structure learning methods have strict limitations on input data and relations. Therefore, we implement the causal structure learning in CausalRCA with a gradient-based method, DAG-GNN [264]. DAG-GNN provides a deep generative model, which is a variational

autoencoder (VAE) parameterized by a novel graph neural network (GNN) [27], and applies a variant of the structural constraint to learn DAGs. Unlike other causal structure learning methods, the gradient-based method has no limitation of input data, can extract linear or non-linear causal relations between metrics, and automatically outputs a weighted DAG.

We use $X \in \mathbb{R}^{m \times n}$ (m is metrics, n is samples of each metric) to represent input data. To get a DAG from X, Zheng et al.[282] adopt a linear SEM as a data generation model, which is $X = A^T X + Z$ ($A \in \mathbb{R}^{m \times m}$ is the weighted adjacency matrix. $Z \in \mathbb{R}^{m \times n}$ is the noise matrix). To ensure the acyclicity of the DAG, a constraint of A is proposed as:

$$h(A) = tr\left[(I + \alpha A \circ A)^m\right] - m = 0 \tag{5.1}$$

Based on the linear SEM, we can get $X = (I - A^T)^{-1} Z$, which can be written as $X = f_A(Z)$. This equation is a general form recognized as an abstraction of parameterized GNNs [123]. We can also see that X is generated from a latent representation Z by defining a probabilistic graphical model. The generative model can be developed based on a VAE, and Z follows a standard Gaussian distribution [122] as shown in Figure 5.2.

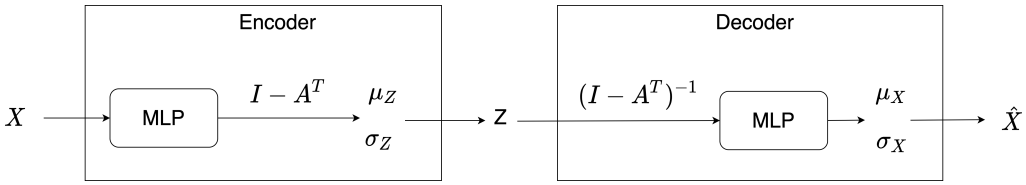


Figure 5.2: Architecture of the causal structure learning method

With latent representation Z, we can define the decoder to reconstruct X as:

$$X = f_2((I - A^T)^{-1} f_1(Z)) \tag{5.2}$$

Then, the corresponding encoder can be defined as:

$$Z = f_4((I - A^T) f_3(X)) \tag{5.3}$$

Combining with deep neural networks, we use multilayer perceptron (MLP) to simulate f_1, f_2, f_3, and f_4, which all are parameterized functions. Based on VAE, the output of encoder and decoder are data distributions, so we get Z by sampling from μ_Z and σ_Z, and $\hat{X}$ by sampling from μ_X and σ_X.

For a VAE model, with a variational posterior $q(Z|X)$ to approximate the actual posterior $p(Z|X)$, evidence lower bound (ELBO) can be represented as:

$$\begin{aligned} L_{ELBO} &= E_{Z \sim q}\left[\log p(X|Z)\right] - KL(q(Z|X), p(Z)) \\ &= E_{Z \sim q}\left(-\frac{1}{2c}\|X - \hat{X}\|\right) - KL(q(Z|X), p(Z)) \end{aligned} \tag{5.4}$$

Thus, the learning problem can be defined as:

$$\begin{aligned} \min_{A,\theta} f(A,\theta) &= -L_{ELBO} \\ s.t.\ h(A) &= 0 \end{aligned} \tag{5.5}$$

where θ is all the parameters of the VAE. For a nonlinear equality-constrained problem, we can use augmented Lagrangian method [19] to solve it.

$$L_c(A,\theta,\lambda) = f(A,\theta) + \lambda h(A) + \frac{c}{2}|h(A)|^2 \tag{5.6}$$

where λ is the Lagrange multiplier and $c > 0$ is the penalty parameter. The following update rules are defined:

$$\begin{aligned} A^k, \lambda^k &= \arg\min_{A,\theta} L_{c^k}(A,\theta,\lambda^k) \\ \lambda^{k+1} &= \lambda^k + c^k h(A^k) \\ c^{k+1} &= \begin{cases} \eta c^k, \ if\ |h(A^k)| > \gamma |h(A^{k-1})| \\ c^k, \ otherwise \end{cases} \end{aligned} \tag{5.7}$$

In the augmented Lagrangian, the penalty parameter c is typically updated using an exponentially increasing function of the iteration number, and the Lagrange multiplier λ is correspondingly updated to converge to the optimal condition. The update rule for the penalty parameter c is important to balance the trade-off between feasibility and optimality in the optimization problem. The rule states that if the constraint violation at the next iteration is larger than the current violation, the value of c should be increased. Conversely, if the constraint violation at the next iteration is smaller than the current violation, the value of c should be kept the same. To achieve faster convergence and find optimal solutions, the update rule depends on two tuning parameters, η and γ. Usually, we set $\eta > 1$ to induce fast convergence and $\gamma < 1$ to limit the convergence speed [264]. If γ is set too high, the convergence will be slow, while if η is set too high, the convergence will be fast, but the results may oscillate. Parameter analysis is provided in our experiments in Section 5.4.2.

During training, parameters A and θ will be updated after every epoch. After training, we can get the A, which is the adjacency matrix of a DAG. For root cause localization in microservice applications, we define $X = [m_1^{s_1}, m_2^{s_1}, ..., m_i^{s_j}, ...]$. With this causal structure learning method, we can get a weighted DAG (G) which represent causal relations between metrics as shown in Figure 5.1(b). Each node in G represent a metric, for example, $m_1^{s_1}$ means a metric in service s_1. The edge from $m_1^{s_1}$ to $m_2^{s_1}$ indicates that a change in $m_1^{s_1}$ will result in a change in $m_2^{s_1}$ with the weight w. Weight w represents the degree of the impact. If w is large, it indicates that a small change in $m_1^{s_1}$ will result in a large change in $m_2^{s_1}$. Furthermore, w can be either negative or positive, implying that an increase in $m_1^{s_1}$ may result in an increase or decrease in $m_2^{s_1}$. Based on the weighted DAG, we then use a root cause inference method to pinpoint the root cause metric.

5.3.3 Root cause inference

For the weighted DAG (G), we can rank metrics with the PageRank algorithm. PageRank works according to the number of incoming edges and the probability of anomalies spreading through the graph. We define P_{ij} as the transition probability of node i to j:

$$P_{ij} = \begin{cases} \frac{w_{ij}}{\sum_j w_{ij}}, \; if \; w_{ij} \neq 0 \\ 0, \; otherwise \end{cases} \tag{5.8}$$

Here, w_{ij} is the weight between node i and j. We define P as the transition probability matrix. Then, we can get the PageRank vector v as proposed by [175] as:

$$v = \alpha P v + \frac{1-\alpha}{n} \tag{5.9}$$

Here, n is the number of nodes, $\alpha \in (0, 1)$ is the teleportation probability, and it means that the random walk will continue with probability α and jump to a random node with probability $1 - \alpha$. We use the default setting $\alpha = 0.85$ [16]. To get results of the root cause inference method better, we first reverse edges in G and use the absolute value of all weights. After running the root cause inference method, we rank the PageRank scores of all nodes and get the ranked list as shown in Figure 5.1(c). The higher the ranking on the list, the more likely the root cause is.

5.4 Experiments and results

To evaluate the root cause localization framework CausalRCA, we conduct experiments on both coarse-grained and fine-grained root cause localization. As for coarse-grained root cause localization, we design experiments to identify faulty services. As for fine-grained root cause localization, we first localize root cause metrics in the faulty service. In addition, taking into account the lack of understanding of services and underlying infrastructures of an application, we provide another fine-grained experiment to localize the root cause metric with all monitoring metrics in all services. In this section, we will introduce experimental settings and experimental results.

5.4.1 Experimental settings

Testbed

To evaluate our framework, we deploy the Sock Shop[6], which simulates an e-commerce website that sells socks. It is widely used as a microservice benchmark designed to aid demonstration and test microservices and cloudnative technologies [141, 247, 248]. The Sock Shop consists of 13 services, which are implemented in heterogeneous technologies and communicate via REST over HTTP. Except for communication services, it contains 7 functional services, which are, *frontend* serves as the entry of user requests; *catalogue* provides product catalogue and information; *carts* holds shopping carts; *user* stores user accounts, including paymenet cards and addresses; *orders* place orders of login users from carts, and it consumes memory a lot; finally, *payment* and *shipping* services are provided for orders, which require network for processing transactions.

We deploy the Sock Shop with Kubernetes on several VMs in the cloud, as shown in Figure 5.3. In the Kubernetes cluster, we have one master node and three worker nodes. Their configurations are Ubuntu 18.04, 4vCPU, 16G RAM Memory, and 80G Disk. On the master node, we deploy open-source monitoring and visualization tools, Prometheus and Grafana, respectively. Prometheus and Grafana are widely used for monitoring in microservice applications [172, 237]. Prometheus can keep monitoring the whole system and collecting both service-level and resource-level data [247]. In addition, we deploy a load generation tool, Locust[7], on the master node to simulate workloads for the microservice application. On worker nodes, we deploy 13 services of the Sock Shop application, and they are allocated to different VMs automatically by Kubernetes.

Anomaly injection

Microservice applications are deployed and distributed in clouds, and their performance is highly dependent on the resources of the underlying infrastructures. There are several common and widespread real performance anomalies in distributed systems [161]. Anomalous CPU consumption in VMs due to infinite loops, busy waits, or deadlocks of competing actions in applications can cause a slowdown of user requests [196]. Memory leak, one of the most prominent software bugs that severely threaten the availability and security of systems [117], happen when allocated chunks of memory are not freed after their use. Accumulations of unfreed memory may exhaust the system resource and lead to memory shortage and system failures. In addition, network resources are vulnerable to being attacked because of the frequent communication between servers and clients. Network latency anomalies usually originate from queuing or processing delays

[6] https://github.com/microservices-demo/microservices-demo
[7] https://locust.io/

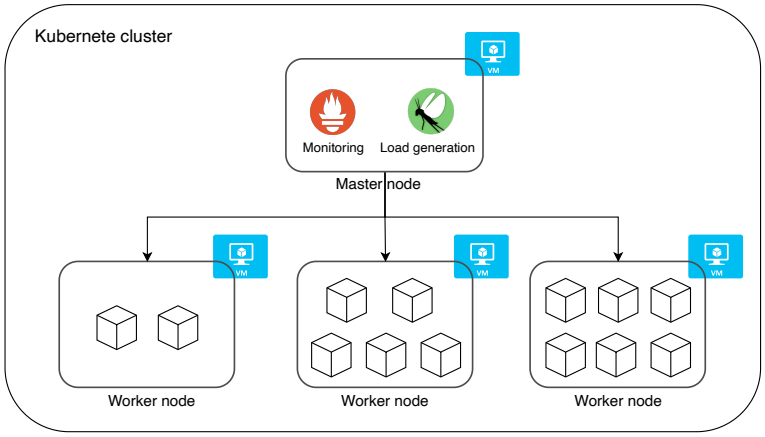


Figure 5.3: The microservice application Sock Shop deployed on VMs with Kubernetes

of packets on gateways [196]. The three anomalies are common and frequent in microservice application [141, 247], which will be used to evaluate our framework.

Our method can be applied to any anomaly that manifests as increased microservice response time. In this evaluation, we inject the three common anomalies: CPU hog, memory leak, and network delay. We inject CPU hog by consuming CPU resources of each service. For memory leak, we allocate memory continuously for each service. For network delay, we enable traffic control to delay the network packets. We implement anomaly injection with the tool Pumba[8], which can emulate network failures and stress-testing resources for Docker containers. Based on anomaly detection research [104, 111], anomalies usually last several minutes, so each anomaly of each service we injected lasts 5 minutes, and the application will have 10 minutes to cool down before another injection.

Data collection

We deploy Prometheus to monitor the microservice application and collect monitoring data in real-time. Prometheus is configured to collect data every 5 seconds. We collect both service-level and resource-level data. At the service level, we collect the latency of each service. At the resource level, we collect container resource-related metrics, including CPU usage, memory usage, disk read and write, and network received and transmitted bytes, as shown in Table 5.1.

Baseline methods

Related work in Section 5.2 shows that CI-based root cause localization uses different causal structure learning methods. Our CausalRCA is developed based

[8]https://github.com/alexei-led/pumba

Metrics type	Metrics
Service-level	Service latency
Resource-level	CPU usage
	Memory usage
	Disk read
	Disk write
	Network received
	Network transmitted

Table 5.1: Collected monitoring metrics

on a gradient-based causal structure learning method. Therefore, to evaluate the localization performance of our CausalRCA, we design baseline methods by combining different causal structure learning methods with PageRank. We chose the constraint-based method PC, the score-based method GES, and the function-based method LiNGAM.

For these baseline structure learning methods, we use their default parameter settings in causal-learn[9]. In CausalRCA, we use 2-layers MLP in the encoder and decoder, respectively. We set the learning rate as $1e-3$, and training epochs as 1000. In addition, we train the model with the Adam optimizer. We use $\eta = 10$ and $\gamma = 0.25$ as default in our experiments, which is proven to work well in [264], and perform parameter analysis with $\eta = 100, 1000$ and $\gamma = 0.5, 0.75$. We run CausalRCA 10 times and take the average as the result of each experiment.

Evaluation metrics

To evaluate localization accuracy, we use two performance metrics: $AC@k$ and $Avg@k$, which are the most commonly used metrics to evaluate rank results[247]. $AC@k$ represents the probability that the top k results given by a method include the real root cause. $Avg@k$ evaluates the overall performance of a method by computing the average $AC@k$. Their formulas can be found in Section 3.3.2. We use AC@1, AC@3, and Avg@5 in our experiments. AC@1 evaluates if the top localized root cause is the real one, and it is the most restrictive and accurate metric. AC@3 is used to determine if the top three localized results have the real root cause. This metric is less accurate than $AC@1$, but it can still help operators quickly reduce root-cause candidates and localize the real ones. Finally, Avg@5 represents the average localization ability. The three metrics are commonly used in the root cause localization task, and they can fairly evaluate localization performance [165, 247].

[9]https://github.com/cmu-phil/causal-learn

Statistical testing

To assess the statistical significance of different RCA methods, we use the one-way analysis of variance (ANOVA) to test the difference between all RCA methods and the t-test to check the pairwise differences [57]. We use $Avg@5$ as the performance score of each RCA method. ANOVA is a hypothesis-testing framework for determining whether the between-group variation is significant. The F-statistic, calculated as the ratio of the between-group variation to the within-group variation, is used in ANOVA. The p-value associated with the F-statistic indicates the probability of obtaining an F-statistic as extreme as the observed one, assuming the null hypothesis is true. If the p-value is less than the significance level (usually 0.05), we reject the null hypothesis and conclude that there is a statistically significant difference among the RCA methods.

If the ANOVA test indicates a significant performance difference among these methods, we then use a t-test to determine the differences between each pair of RCA methods. We calculate the t-value to measure the difference between the average performance scores of two methods relative to the variability within each method. If the t-value is less than the critical value, we fail to reject the null hypothesis and conclude that there is insufficient evidence to suggest a performance difference between the two methods.

5.4.2 Experimental results

We provide the results of three experiments as below:

- Coarse-grained faulty service localization based on service latency of all services.
- Fine-grained root cause metric localization in the faulty service based on system-level metrics in the faulty service.
- Fine-grained root cause metric localization with all monitoring metrics in all services

We compare the localization performance of CausalRCA with baseline methods and explain the results.

Coarse-grained faulty service localization

We evaluate the performance of CausalRCA on localizing the faulty service that initiates performance anomalies. This localization is conducted based on service-level data, which is the latency of all services. Table 5.2 shows the localization accuracy compared with baseline methods for different anomalies. We can see that, when compared to baseline methods, CausalRCA has improved localization accuracy in terms of $AC@1$, $AC@3$, and $Avg@5$ in different anomalies by up to 10%. In addition, for CPU hog, causalRCA has the best performance in terms of $AC@1$, $AC@3$, and $Avg@5$. The $AC@3$ is 0.718, which means that

there is a 71.8% chance of finding the root cause in the top three metrics on the ranked list, which is slightly higher than the LiNGAM-based method. For memory leak, CausalRCA continues to outperform in terms of $AC@1$, $AC@3$, and $Avg@5$. There is a 62.1% possibility of localizing the root cause in the top 3 metrics. For network delay, $AC@3$ is not good enough, but $AC@1$ and $Avg@5$ are higher than baseline methods. In general, the average $AC@1$ of CausalRCA is 0.2, which means that there is an average 20% possibility that the top 1 metric on the ranked list can be identified as the root cause. The averages $AC@3$ and $Avg@5$ of CausalRCA for the three anomalies are 0.575 and 0.581, respectively. The increase of average $Avg@5$ is 6.7%, showing the improvement in localizing accuracy compared with baseline methods. We provide statistical testing to show the significant difference between these RCA methods. We obtained a p-value of 0.0003 using the ANOVA method first, showing a significant performance difference between the four RCA methods. We further utilized t-tests to compare the performance differences between each pair of methods, and the resulting p-values are shown in Figure 5.4. We can see that CausalRCA has a significant difference from baseline methods, while PC-based and GES-based methods have no significant difference.

Table 5.2: Localization accuracy of CI-based methods on localizing faulty services (Coarse-grained) for different anomalies

Methods	PC-based	GES-based	LiNGAM-based	CausalRCA	Increase
CPU hog					
AC@1	0.143	0.143	0.143	**0.187**	4.4%
AC@3	0.286	0.429	0.714	**0.718**	0.4%
Avg@5	0.429	0.400	0.571	**0.624**	5.3%
Memory leak					
AC@1	0	0	0.143	**0.243**	10.0%
AC@3	0.429	0.143	0.571	**0.621**	5.0%
Avg@5	0.429	0.229	0.543	**0.614**	7.1%
Network delay					
AC@1	0.143	0	0.143	**0.171**	2.8%
AC@3	**0.571**	0	0.429	0.386	-18.5%
Avg@5	0.486	0.171	0.429	**0.506**	2.0%
Average Avg@5	0.448	0.267	0.514	**0.581**	6.7%

We analyze the impact of parameters γ and η in causal structure learning on the root cause localization performance of CausalRCA. We use $\gamma = 0.25$ and $eta = 10$ as default, and also set $\gamma = 0.5, 0.75$, and $\eta = 100, 1000$. The results can be found in Figure 5.5. We can see that $\gamma = 0.25, \eta = 10$ has the best performance in CPU hog and Network latency, and it also has the best average performance of different anomalies. In addition, $\gamma = 0.75, \eta = 10$ performs best for memory leak, because a high γ can prevent too fast convergence and find better solutions, but it usually takes more time. Furthermore, we can see that $\gamma = 0.25, \eta = 1000$ performs poorly on CPU hog anomaly, but relatively well

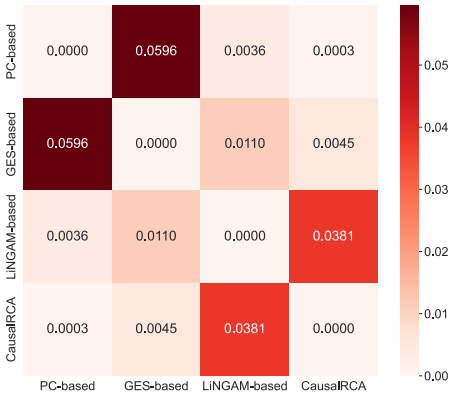


Figure 5.4: P-value of RCA methods (Coarse-grained experiment)

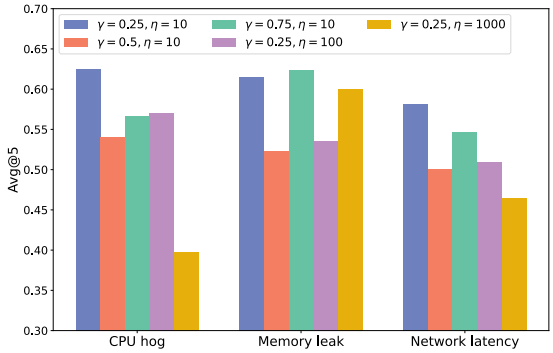
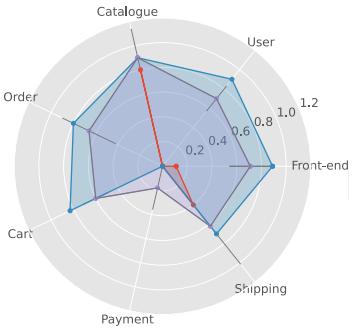
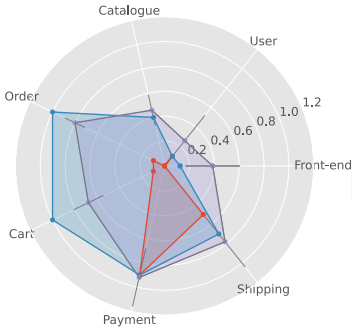


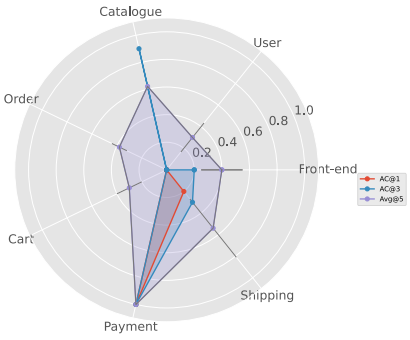
Figure 5.5: Localization accuracy with different γ and η (Coarse-grained experiment)



(a) CPU hog



(b) Memory leak



(c) Network delay

Figure 5.6: Performance of CausalRCA on localizing faulty services with different anomalies

on memory leak anomaly, suggesting that a high η can lead to more variance in localization results.

We then provide a detailed performance of CausalRCA on localizing faulty services with different anomalies in Figure 5.6. For CPU hog in Figure 5.6(a), we can see that localization accuracy performs well on services except *payment*, because *payment* is not a CPU-intensive service. For the memory leak in Figure 5.6(b), we can see that *front-end*, *user*, and *catalogue* perform worse than other services. The memory leak issues in these services do not affect their service latency much, making it difficult to identify cause-effect relations between services. In terms of network delay in Figure 5.6(c), only service *payment* performs well, which explains the poor average localization performance of $AC@3$ in Table 5.2. *Payment* service relies heavily on the network, making it easy to localize the network delay issue. We plot the errorbar for $Avg@5$ in Figure 5.6 to represent the variation of our results, and we can see that the standard deviations of many results are not high.

Fine-grained root cause metric localization in the faulty service

Given the faulty service, we apply CausalRCA to container resource metrics and evaluate its performance on localization accuracy for different anomalies. Table 5.3 shows the localization accuracy of CausalRCA on localizing root cause metric in faulty service compared with baseline methods. For different anomalies, we can see that CausalRCA has improved localization accuracy compared with baseline methods in terms of $AC@1$, $AC@3$, and $Avg@5$ by up to 14.3%. In addition, for CPU hog, CausalRCA has the best performance in terms of $AC@3$ and $Avg@5$, while the $AC@1$ is worse than PC-based methods. For memory leak, the $AC@1$ of CausalRCA is worse than LiNGAM-based method, but it still has the best $AC@3$ and $Avg@5$. Finally, for network delay, CausalRCA outperforms in terms of $AC@1$, $AC@3$, and $Avg@5$. In general, the average $AC@1$ of CausalRCA is 0.248, which means there is a 24.8% possibility of determining the top 1 metric on the ranked list as the root cause. For the three anomalies, the average $AC@3$ is 0.719, which means there is a 71.9% possibility to localize the root cause metric in the top 3 metrics on the ranked list, and the average improvement is 10% compared with baseline methods. The average $Avg@5$ of CausalRCA is 0.668, and the average increase is 9.4%. We consider the outperformance of CausalRCA is because resource metrics, such as CPU/memory usage, affect each other, which makes it easier to identify anomaly propagation with CI methods.

We also provide statistical testing to show the significant difference of these RCA methods. We first obtained a p-value of 0.0013 using the ANOVA method, showing a significant performance difference between the four RCA methods. The p-values obtained from t-tests are shown in Figure 5.7. We can see that CausalRCA has a significant difference from baseline methods. In comparison, GES-based method has no significant difference with PC-based and LiNGAM-

Table 5.3: Localization accuracy of CI-based methods on localizing root cause metrics (Fine-grained) in faulty services for different anomalies

Methods	PC-based	GES-based	LiNGAM-based	CausalRCA	Increase
CPU hog					
AC@1	**0.429**	0.143	0	0.229	-20.0%
AC@3	0.429	0.571	0.714	**0.729**	1.5%
Avg@5	0.429	0.600	0.571	**0.670**	7.0%
Memory leak					
AC@1	0	0	**0.429**	0.271	-15.8%
AC@3	0.143	0.429	0.571	**0.714**	14.3%
Avg@5	0.343	0.400	0.629	**0.677**	4.8%
Network delay					
AC@1	0	0.143	0.143	**0.243**	10.0%
AC@3	0.286	0.571	0.429	**0.714**	14.3%
Avg@5	0.221	0.514	0.521	**0.657**	13.6%
Average Avg@5	0.331	0.505	0.574	**0.668**	9.4%

based methods.

We evaluate the impact of parameters γ and η and present the findings in Figure 5.8. The results indicate that $\gamma = 0.25$ and $\eta = 10$ perform the best in identifying CPU hog anomalies, while $\gamma = 0.25$ and $\eta = 100$ are most effective in detecting memory leak and network latency anomalies. In addition, $\gamma = 0.5$ and $\eta = 10$ outperform $\gamma = 0.5$ and $\eta = 100$ in detecting network latency anomalies and perform better than $\gamma = 0.75$ and $\eta = 10$ across all three types of anomalies. Overall, $\gamma = 0.25$ and $\eta = 10$ have the highest average localization performance. However, increasing γ and η could potentially lead to better solutions and improve localization accuracy.

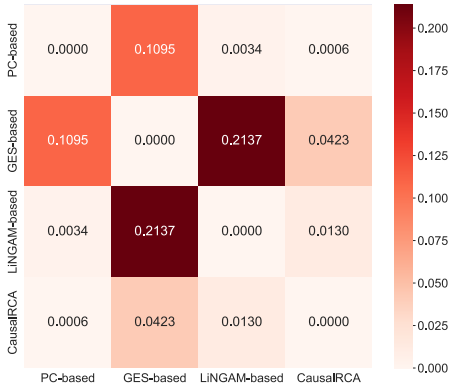


Figure 5.7: P-value of RCA methods (Fine-grained experiment)

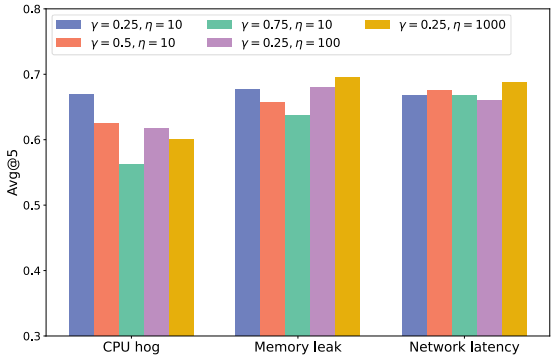
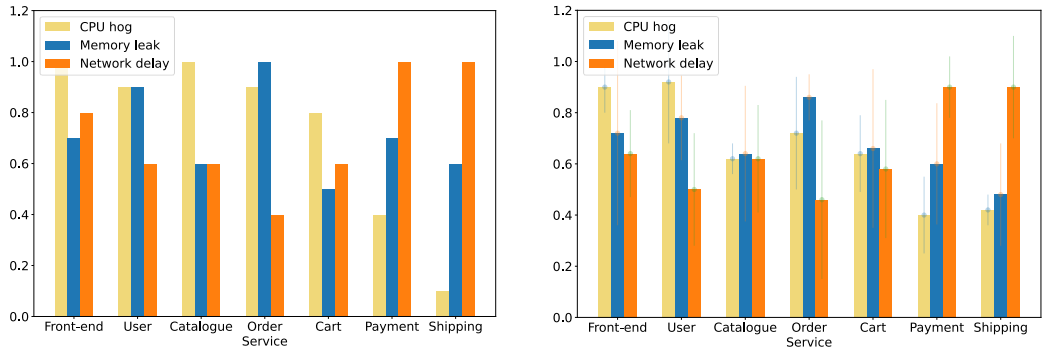


Figure 5.8: Localization accuracy with different γ and η (Fine-grained experiment)

We then provide the performance of CausalRCA on localizing root cause metrics in faulty services with the three anomalies in Figure 5.9. We can see that *Ac*@3 and *AC*@5 have consistent performance. For CPU hog, we can see that



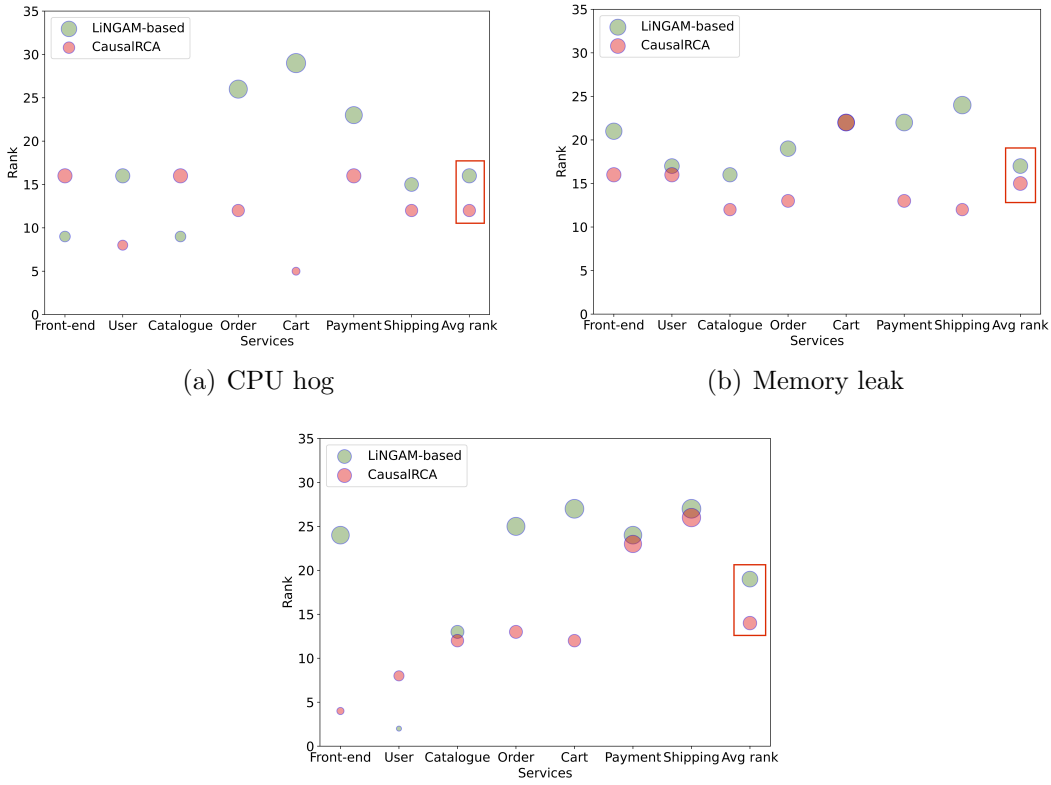
(a) AC@3 (b) Avg@5

Figure 5.9: Performance of CausalRCA on localizing root cause metrics in faulty services with different anomalies

localization accuracy is low for *payment* and *shipping* services because they are insensitive to CPU resources. Because the memory leak issue manifests in multiple resource metrics, all services perform well for the memory leak. *Order* service has the best performance because it is highly related to memory usage. For network delay, we can see that *payment* and *shipping* have the best performance because they rely heavily on the network. We plot the errorbar for *Avg*@5 in Figure 5.9(b), and we can see there are some variances in the localization results, maybe caused by the dynamic nature of cloud environments or random fluctuation of resources in services, showing that the generality of CausalRCA can be explored more in the future.

Fine-grained root cause metric localization with all monitoring metrics

Considering that we do not know services and underlying infrastructures of an application, we conduct the fine-grained root cause localization with all monitoring metrics. We apply CausalRCA on all monitoring metrics to localize the root cause metric. We mainly show the ranks of comparison between the LiNGAM-based method and CausalRCA, because PC sometimes fails to extract causal relations between metrics, while GES takes too long to build a causal graph with too many nodes. For this fine-grained root cause localization, it is hard to identify the root cause metric in the top 1 or top 3 metrics, so we use the rank of root cause metrics to evaluate localization performance as shown in Figure 5.10. We can see that the average rank of CausalRCA is about 13, which is lower than the LiNGAM-based method. The result shows that CausalRCA has better localization performance than the LiNGAM-based method. However, it is still challenging to extract causal relations between metrics and pinpoint the root cause metric from multiple observable metrics. We apply the t-test to LiNGAM-based and CausalRCA methods and obtain the p-value of 0.0335, showing the



(a) CPU hog

(b) Memory leak

(c) Network delay

Figure 5.10: Ranks of root cause metrics identified by CI-based methods

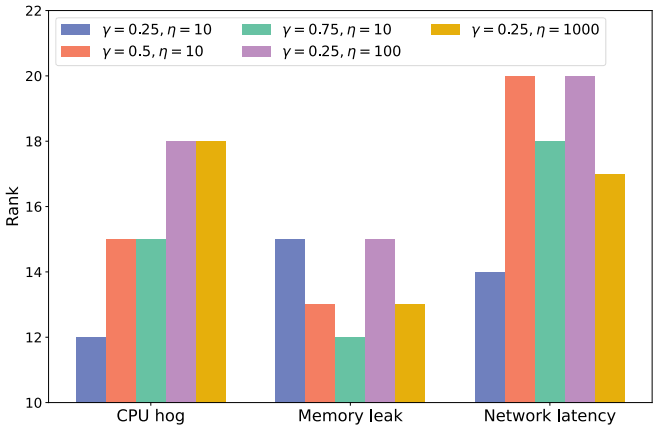


Figure 5.11: Ranks of root cause metrics with different parameters γ and η

significant difference between them. The impact of parameters γ and η is presented in Figure 5.11. The results indicate that $\gamma = 0.25, \eta = 10$ performs the best in identifying CPU hog and network latency anomalies. For the memory leak anomaly, higher γ and η have better localization accuracy, and $\gamma = 0.75, \eta = 10$ is most effective. On average, the $\gamma = 0.25, \eta = 10$ has the best localization performance, while adjusting γ and η for the memory leak anomaly can improve localization accuracy effectively.

In conclusion, CausalRCA has a significant difference from baseline methods and better localization accuracy for coarse-grained and fine-grained root cause localization. In addition, we find that $\gamma = 0.25, \eta = 10$ in CausalRCA has the best localization performance on average. However, adjusting parameters can provide more potential for improving localization accuracy. Based on CI methods, we can see that anomaly propagation performs differently in different services; for example, the *payment* service is sensitive to the network delay issue but nonsensitive to the CPU hog issue. As for fine-grained root cause localization, it is still challenging to pinpoint the root cause metric in all monitoring metrics. Therefore, it is more practical to consider the drill-down localization, i.e., identify the faulty service first and then determine the root cause metric in the faulty service.

5.4.3 Threats to validity

We analyze threats to our framework from the four categories: construct, internal, conclusion, and external validity based on [242]. The **construct threat** to validity mainly lies in the hyperparameters and evaluation metrics. We provide parameter analysis for two hyperparameters in CausalRCA, and results show that default parameters works well as provided in [264] but tuning parameters carefully has the potential of improving localization accuracy. In addition, we use widely used evaluation metrics and provide statistical testing to evaluate the performance difference of different RCA methods.

The **internal threat** to validity mainly lies in the implementation of the framework, as errors or bugs in the implementation could affect the accuracy of the results. To reduce it, we have used established Python packages and conducted thorough testing. We have also repeated the experiments multiple times to ensure the reliability and consistency of our results. The **conclusion threat** to validity of our framework is related to the types of anomalies used in experiments. As microservice applications have a variety of performance anomalies that can affect the localization results [161], we injected three different types of common and frequent anomalies to evaluate the effectiveness of our framework. We report and discuss the localization results for each individual anomaly type, and the experimental results demonstrate the superior performance of our framework on these anomalies.

The **external threat** relies on the configuration of microservice applications

and the data collection strategies. In this section, we investigate a specific configuration of a microservice application to evaluate the performance of CausalRCA, which may limit the generality of our framework. However, building complex infrastructures and repeating the experiments on multiple testbeds is extremely expensive, which is impractical for our experiments. In addition, the benchmark microservice application Sock Shop is widely used in academia to aid the testing of microservices in clouds [141, 247], and it helps us mitigate this threat. On the other hand, the localization performance of our framework heavily relies on input data. To mitigate the threat, we adopt Prometheus, an open-source tool for real-time monitoring, and collect service-level and resource-level metrics that present the status of a running microservice application. Currently, our framework performs well with anomalies injected over fixed time range anomalies, but the effect of different time ranges for CausalRCA can be explored more.

5.5 Discussion

This chapter provides a framework called CausalRCA for root cause localization of microservice applications. The framework is developed with CI-based methods, including causal structure learning and root cause inference. We provide coarse-grained and fine-grained experiments to evaluate the localization performance of the framework. Our experimental results show that the framework has the best localization accuracy compared with baseline methods. However, some aspects can still be improved.

Our experiments show that CausalRCA performs well on localizing faulty services and root cause metrics in faulty services. However, localizing root cause metrics from all monitoring metrics is very hard. The average rank of root cause metrics is outside the top ten. We consider the improvement of localization accuracy can be researched more. First, data preprocessing, such as feature reduction, can be considered to reduce training time and improve localization accuracy. Next, we apply a gradient-based method to learn causal structures. The gradient-based method is applied to time-series monitoring data, which may ignore time lags in the original data. We consider that time lags in the data may help improve causal structure learning. For the root cause inference method PageRank, a personalized PageRank [112], which considers the preferences of nodes, can be applied.

This chapter mainly focuses on monitoring data to implement root cause localization. Monitoring data has multi-dimensional information and is easy to collect compared with trace and log data. However, trace data and log data contain accurate deployment information and service interactions, which can be used to calibrate the causal graph generated based on monitoring data. At the same time, the causal graph generated with CI methods can extract hidden relations between metrics. Therefore, we can consider combining different data resources

to improve localization accuracy in the future.

This chapter mainly focuses on improving localization accuracy of microservices, while efficiency is also important for achieving fast recovery. For the Sock Shop benchmark application, we roughly estimate the time spent of our framework takes tens seconds, showing the cost of time may be lower than service migration [43]. However, for large-scale microservice, e.g., hundreds/thousands of services, the time cost for building the causality graph may be far greater than service migration. We will test the scalability and exact time cost of our framework and pay more attention to reducing training time in the future. For now, we mainly use the data collected in five minutes after the anomaly is detected. In the future, we will consider testing localization performance with different time ranges based on our CausalRCA.

5.6 Conclusion

This chapter tackles the challenge of localizing the root causes of performance anomalies in microservice applications. Root cause localization can be used to help operators achieve fast recovery of microservice applications. Therefore, it is important to guarantee localization accuracy at first. In addition, fine-grained root cause localization, which means identifying both faulty service and resource-related metrics in a faulty service, is necessary. With monitoring data, we provide a CI-based framework named CausalRCA, which can automate localizing root causes with fine granularity and in real-time. The CausalRCA works with causal structure learning and root cause inference components. For causal structure learning, we propose a GNN-based method that uses a deep generative model and applies a variant of the structural constraint to learn the weighted DAG. The gradient-based method can extract non-linear causal relations between metrics compared with other CI methods. For root cause inference, we apply PageRank to visit the weighted DAG and return a ranked list of all metrics. We then provide experiments to evaluate the localization performance of CausalRCA.

To evaluate CausalRCA, we conduct three experiments: coarse-grained faulty service localization, fine-grained root cause metrics localization in faulty services, and fine-grained root cause metrics localization with all monitoring metrics. Our experimental results show that CausalRCA has better localization accuracy than baseline methods. Furthermore, based on CI methods, we can see that anomaly propagation performs differently between services, which gives operators a better understanding of microservice applications. In addition, it is difficult for fine-grained root cause localization with all monitoring metrics because anomalous metrics manifest diverse symptoms in different services. Therefore, fine-grained root cause localization with all monitoring metrics still needs to be improved. However, for microservice applications, we can still consider the drill-down localization, first identifying the faulty service, and then pinpointing the root cause

metric in the faulty service to identify fine-grained causes.

In the future, we will continue to improve the localization performance of CausalRCA. Hyperparameter tuning can be tested more in the future. The causal structure learning can consider time lag in monitoring data, and the root cause inference can be improved by adding the preferences of nodes. In addition, employing knowledge from trace and log data to calibrate the causal graph may improve localization accuracy and make the causal graph more reasonable. Finally, localization efficiency needs to be tested and improved to achieve fast recovery.

Chapter 6

Conclusions

Cloud computing provides elastic and on-demand resources for distributed applications to deliver high-quality services. However, the dynamism of underlying cloud infrastructures and complex dependencies between services introduce abnormal performance phenomena, e.g., degradation, which severely affect the quality of services and the user experience. To make services in applications continuously operational, performance diagnosis systems are required to detect performance anomalies, such as slow response times, and localize their root causes. Such kinds of systems have been studied in recent years. A typical performance diagnosis system comprises components for collecting and preprocessing monitoring data, detecting performance anomalies, and localizing root causes. However, each component of a performance diagnosis system presents unique challenges. The data collection and preprocessing components should collect real-time performance data and reduce noise to make it available for subsequent analysis. Effective detection methods must be accurate for anomaly identification and robust in fitting different data distributions in real scenarios. The root cause localization component aims to accurately identify the underlying causes of performance anomalies, such as resource-related metrics in faulty services. However, many anomalous metrics and complex anomaly propagation paths make it challenging to determine the root cause.

To tackle the above challenges, we propose a comprehensive performance diagnosis system that can effectively detect performance anomalies and localize their root causes to provide actionable insights to operators. Our contributions include the following:

- We reviewed the state-of-the-art research and methods for creating a reliable performance diagnosis system from a technical perspective.

- We developed an effective framework for run-time distributed applications. We evaluated its monitoring tool, data preprocessing, performance anomaly

detection, and root cause localization models, presenting its ability for accurate and effective performance diagnosis.

- We developed a weakly-supervised approach that integrates existing detection methods through ensemble learning to improve the accuracy, robustness, and predictive ability of anomaly detection. We proposed a deep learning-based unsupervised method to improve the trade-off between detection accuracy and robustness.

- We developed a gradient-based causal structure learning model and a root cause inference model to build anomaly propagation paths and determine the root cause effectively.

In this chapter, we conclude the thesis by answering the research questions proposed in Section 1.1. Then, we provide lessons learned when developing methods in the performance diagnosis framework.

6.1 Answer to research questions

With the above-mentioned outcomes and the analysis of evaluation results, we answer our main research question defined in Chapter 1:
RQ: How to effectively diagnose the performance of distributed applications in cloud environments at runtime?

In the thesis, we concluded that effective diagnosis could be achieved by a well-designed performance diagnosis framework, including proper data collection and preprocessing, advanced performance anomaly detection and root cause localization methods. We completed our research works with several steps: state-of-the-art technologies review, framework design, and advanced model development.

In Chapter 2, we summarized trustworthiness requirements and existing technologies of each component for a general performance diagnosis framework. For example, the robustness requirement is needed for anomaly detection and can be achieved by ensemble learning. For the performance diagnosis framework, we then determined the technologies of each component and evaluated the effectiveness of the framework in Chapter 3. Our results show that data preprocessing can reduce noise and improve detection accuracy, existing detection methods perform varies for different performance data, and localization accuracy needs to be improved. To meet trustworthiness requirements, specifically accuracy and robustness for performance anomaly detection, we developed an ensemble-based weakly supervised learning method and a deep learning-based unsupervised learning method in Chapter 4. We concluded that they have superior performance compared with baseline methods. A concrete explanation of performance anomalies can help operators take action quickly and mitigate economic loss, requiring accurate root cause localization methods. Therefore, we proposed the fine-grained root cause

localization framework with the gradient-based CI method in Chapter 5. We concluded that the framework has the best coarse- and fine-grained localization accuracy.

To be more specific, we answer the sub-research questions with details.

RQ1: What are the state-of-the-art technologies for achieving trustworthy performance diagnosis systems?

We conducted a thorough, state-of-the-art survey of the research topic to address this question. We concluded ten trustworthiness requirements and 21 detailed technologies for a performance diagnosis system. Based on existing research on diagnosis systems, we established a general framework comprising data collection, preprocessing, anomaly detection, and root cause localization. Furthermore, we extracted five technical requirements (data privacy, fairness, robustness, explainability, and human intervention) from seven trustworthy AI requirements proposed by the EU and integrated them separately into the performance diagnosis framework. We reviewed related works for each component in the framework and presented state-of-the-art technologies that satisfy trustworthiness requirements. For example, ensemble learning can be used to enhance the robustness of performance anomaly detection. This survey work provided us with a comprehensive understanding of trustworthy performance diagnosis systems and offered guidance on developing advanced methods for detection and localization components.

RQ2: How can a performance diagnosis framework be developed to identify performance issues and determine root causes effectively?

We answered this question by designing a performance diagnosis framework, FIED, including a monitoring tool, data preprocessing, anomaly detection, and root cause localization methods. The monitoring tool can capture performance data that reflects the running status of distributed applications. To decrease data noise and dimensions of collected performance data, we performed metrics selection in data preprocessing and observed that it could significantly improve detection accuracy. Furthermore, we employed different unsupervised detection methods and concluded that they have varying performances on different datasets because they focus on different characteristics in data. For root cause metrics localization of detected performance anomalies, we demonstrated the possibility of metric-level real-time root cause localization.

RQ3: How to improve the accuracy and robustness of detecting performance anomalies?

The improvement of accuracy and robustness for performance anomaly detection can be achieved by advanced machine learning-based detection methods. In this thesis, we provided two solutions: weakly-supervised and unsupervised methods. We developed an Ensemble Learning-Based Detection (ELBD) framework that integrates classic detection methods instead of enhancing a single model. The framework includes three classic ensemble methods without training and a deep ensemble method that requires fewer labels to train a neural network. Based on

the experiment results, ensemble learning could improve detection accuracy and robustness by combining extracted information from existing detection methods. Moreover, the deep ensemble method outperformed other methods in terms of accuracy and robustness, along with multi-step prediction ability. However, ensemble learning-based methods have limited improvement in detection accuracy because it relies heavily on base detection methods. Therefore, we developed a GNN-based unsupervised detection method which focuses on extracting information from both time and feature dimensions and ensures the enhancement of detection accuracy and robustness with multi-dimensional information extraction ability.

RQ4: How to localize root causes of detected performance anomalies at a fine-grained level?

We answered this question by modelling anomaly propagation paths with a causal structure learning method and determining the root cause by traversing these paths. We identified that existing CI methods in metric-based research have limitations due to assumptions about data distribution and inaccurate causal relation extraction. We addressed them by developing a gradient-based CI method to establish causal relations between metrics and a ranking algorithm to identify the root cause metric. Through experiments, we evaluated the localization accuracy of our framework on different levels of granularity and observed its effectiveness in fine-grained root cause localization. The gradient-based CI method outperforms other baseline methods. Furthermore, we found that drill-down localization is more practical, starting with identifying the faulty service and then pinpointing the root cause metrics within the service.

6.2 Lesson learned

This section will discuss some of the lessons learned while developing methods in the performance diagnosis framework.

6.2.1 Performance data

This thesis primarily concerns the performance diagnosis of distributed applications based on monitoring data, which is multi-dimensional time series data. Section 2.3.1 highlighted the abundance of log and trace data available for distributed applications, which provide valuable insights into application running status and service interactions. Despite their drawbacks, such as being challenging to parse and process in real time, we should explore the possibility of integrating these data sources into our diagnosis framework. We believe that the integration has the potential to improve detection and localization accuracy significantly.

6.2.2 Prediction ability

Section 4 introduced the deep ensemble method within the ELBD framework. Our experiments demonstrated that this method significantly enhances detection accuracy and robustness and can predict anomalies with remarkable accuracy up to four minutes in advance. While most detection methods concentrate on identifying anomalies in historical data as quickly as possible, predicting potential anomalies and taking preemptive measures to prevent them can lead to reduced economic loss and greater user satisfaction. While some existing detection methods mainly focus on one-step prediction, we believe further exploration of multi-step prediction of performance anomalies is warranted.

6.2.3 Localization accuracy

We introduced the root cause localization framework in Section 5, , which models anomaly propagation using CI methods and traverses the propagation to identify the root cause. Our experiments demonstrate that this framework can successfully localize coarse-grained faulty services and fine-grained metrics within faulty services, outperforming other localization methods. However, we acknowledge that root cause localization remains challenging, particularly given that the top 1 localization accuracy is currently low. In addition, the fine-grained metrics localization with all monitoring data performs unsatisfactorily. Consequently, we believe that further efforts to improve localization accuracy are vital. Trace data provides insights into service interactions, while CI methods extract hidden causal relationships between services and metrics. Therefore, we suggest exploring trace data to calibrate causal graphs to improve localization accuracy.

Chapter 7

Future Directions

Our work can be explored in the following directions in the future.

- *Different data sources integration.* Utilizing different performance data from distributed applications is a promising direction for improving detection and localization accuracy. Previous research has explored using log and trace data for the performance diagnosis of distributed applications. For example, Zhang et al. [271] combine log and trace data to detect anomalies in microservices. Integrating multiple data sources can provide more information, leading to a better understanding of the application's behaviour and identifying root causes. However, several challenges need to be addressed, such as the heterogeneity of data sources and complex information alignment. Therefore, we leave the exploration of integrating different data sources to improve diagnosis performance as our future work.

- *Upcoming performance anomalies detection.* While existing methods mainly focus on detecting past anomalies, predicting future anomalies can provide proactive insights into application behaviour and help avoid economic loss in advance. In our deep ensemble method, we implemented the prediction ability using neural networks. However, we believe that further improvements can be made by using more advanced models such as LSTM, which can capture long-term dependencies in time series data and are known to perform well in prediction tasks. Therefore, future research could investigate the use of advanced deep learning methods to improve the prediction ability of performance diagnosis methods.

- *Precise root cause localization.* By incorporating trace data, which captures service interactions, we can fine-tune the causal relations between services extracted by CI methods. This refinement leads to a more precise causal graph, thereby significantly enhancing the accuracy of root cause localization.

- *Effective application adaptation.* To effectively address performance anomalies in distributed applications, the diagnosis results should provide actionable insights to support effective adaptation strategies. For example, if the root cause of an anomaly is related to high CPU usage, adding more CPU resources could be a potential adaptation strategy. However, there may be multiple strategies for the same root cause, and the effectiveness of each strategy could depend on the specific application and underlying environment. Therefore, future research could focus on exploring the choice of appropriate adaptation strategies based on diagnosis results for different types of performance anomalies in various distributed applications and evaluation of the effectiveness of these strategies in real-world scenarios.

- *Diagnosis framework validation in complex real-time applications.* We have proposed a performance diagnosis framework and evaluated its effectiveness using performance data collected from benchmark distributed applications, including a DApp and a sockshop microservice application. However, real-world distributed applications can be more complex, with hundreds or thousands of services, large-scale performance data, and intricate service interactions. As a result, our framework may have limitations that need to be addressed before it can be applied to real-world scenarios. Future research can focus on enhancing the scalability and adaptability of our framework to accommodate the complexities of real-world distributed applications.

Bibliography

[1] ISO 24028:2020. Information technology–artificial intelligence–overview of trustworthiness in artificial intelligence. *Standard. International Organization for Standardization.*, 2020.

[2] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[3] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[4] Pooja Aggarwal, Ajay Gupta, Prateeti Mohapatra, Seema Nagar, Atri Mandal, Qing Wang, and Amit Paradkar. Localization of operational faults in cloud applications by mining causal dependencies in logs using golden signals. In *International Conference on Service-Oriented Computing*, pages 137–149. Springer, 2020.

[5] Diana Laura Aguilar, Miguel Angel Medina Perez, Octavio Loyola-Gonzalez, Kim-Kwang Raymond Choo, and Edoardo Bucheli-Susarrey. Towards an interpretable autoencoder: a decision tree-based autoencoder and its application in anomaly detection. *IEEE Transactions on Dependable and Secure Computing*, 2022.

[6] Auday Al-Dulaimy, Javid Taheri, Andreas Kassler, M Reza Hoseiny Farahabady, Shuiguang Deng, and Albert Zomaya. Multiscaler: A multi-loop auto-scaling approach for cloud-based applications. *IEEE Transactions on Cloud Computing*, 2020.

[7] Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special lecture on IE*, 2(1):1–18, 2015.

[8] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias. In *Ethics of Data and Analytics*, pages 254–264. Auerbach Publications, 2016.

[9] Yoshinori Aono, Takuya Hayashi, Lihua Wang, Shiho Moriai, et al. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13(5):1333–1345, 2017.

[10] Mohammad S Aslanpour, Sukhpal Singh Gill, and Adel N Toosi. Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research. *Internet of Things*, 12:100273, 2020.

[11] Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A Zuluaga. Usad: Unsupervised anomaly detection on multivariate time series. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3395–3404, 2020.

[12] Bahadir Ismail Aydin, Yavuz Selim Yilmaz, Yaliang Li, Qi Li, Jing Gao, and Murat

Demirbas. Crowdsourcing for multiple-choice question answering. In *AAAI*, pages 2946–2953. Citeseer, 2014.

[13] Armin Balalaie, Abbas Heydarnoori, and Pooyan Jamshidi. Microservices architecture enables devops: Migration to a cloud-native architecture. *Ieee Software*, 33(3):42–52, 2016.

[14] Sukarna Barua, Md Monirul Islam, Xin Yao, and Kazuyuki Murase. Mwmote–majority weighted minority oversampling technique for imbalanced data set learning. *IEEE Transactions on knowledge and data engineering*, 26(2):405–425, 2012.

[15] Scott Beamer, Krste Asanovic, and David Patterson. Direction-optimizing breadth-first search. In *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pages 1–10. IEEE, 2012.

[16] Luca Becchetti and Carlos Castillo. The distribution of pagerank follows a power-law only for particular values of the damping factor. In *Proceedings of the 15th international conference on World Wide Web*, pages 941–942, 2006.

[17] Hal Berghel. Equifax and the latest round of identity theft roulette. *Computer*, 50(12):72–76, 2017.

[18] Christoph Bergmeir, Rob J Hyndman, and José M Benítez. Bagging exponential smoothing methods using stl decomposition and box–cox transformation. *International journal of forecasting*, 32(2):303–312, 2016.

[19] Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

[20] Alina Beygelzimer, Sanjoy Dasgupta, and John Langford. Importance weighted active learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 49–56, 2009.

[21] Ranjita Bhagwan, Rahul Kumar, Ramachandran Ramjee, George Varghese, Surjyakanta Mohapatra, Hemanth Manoharan, and Piyush Shah. Adtributor: Revenue debugging in advertising systems. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, pages 43–55, 2014.

[22] Jasmin Bogatinovski, Sasho Nedelkoski, Jorge Cardoso, and Odej Kao. Self-supervised anomaly detection from distributed traces. In *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*, pages 342–347. IEEE, 2020.

[23] Andrea Borghesi, Andrea Bartolini, Michele Lombardi, Michela Milano, and Luca Benini. A semisupervised autoencoder-based approach for anomaly detection in high performance computing systems. *Engineering Applications of Artificial Intelligence*, 85:634–644, 2019.

[24] Michael Brackett and Production Susan Earley. The dama guide to the data management body of knowledge (dama-dmbok guide). 2009.

[25] Álvaro Brandón, Marc Solé, Alberto Huélamo, David Solans, María S Pérez, and Victor Muntés-Mulero. Graph-based root cause analysis for service-oriented and microservice architectures. *Journal of Systems and Software*, 159:110432, 2020.

[26] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

[27] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[28] JD explore academy CACIT. White paper on trustworthy artificial intelligence. *http://www.caict.ac.cn/kxyj/qwfb/bps/202107/P020210709319866413974.pdf*, 2021.

[29] Cristian S Calude and Giuseppe Longo. The deluge of spurious correlations in big data. *Foundations of science*, 22(3):595–612, 2017.

[30] Jose Camacho, Gabriel Macia-Fernandez, Noemí Marta Fuentes-García, and Edoardo Saccenti. Semi-supervised multivariate statistical network monitoring for learning security

threats. *IEEE Transactions on Information Forensics and Security*, 14(8):2179–2189, 2019.

[31] Hong Cao, Vincent YF Tan, and John ZF Pang. A parsimonious mixture of gaussian trees model for oversampling in imbalanced and multimodal time-series classification. *IEEE transactions on neural networks and learning systems*, 25(12):2226–2239, 2014.

[32] Yousra Chabchoub, Maurras Ulbricht Togbe, Aliou Boly, and Raja Chiky. An in-depth study and improvement of isolation forest. *IEEE Access*, 10:10219–10237, 2022.

[33] Chengliang Chai, Ju Fan, Guoliang Li, Jiannan Wang, and Yudian Zheng. Crowdsourcing database systems: Overview and challenges. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 2052–2055. IEEE, 2019.

[34] Saptarshi Chakraborty and Dhrubajyoti Das. An overview of face liveness detection. *arXiv preprint arXiv:1405.2227*, 2014.

[35] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019.

[36] Varun Chandola, Varun Mithal, and Vipin Kumar. Comparative evaluation of anomaly detection techniques for sequence data. In *2008 Eighth IEEE international conference on data mining*, pages 743–748. IEEE, 2008.

[37] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

[38] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[39] Kuo-Ming Chen, Tsung-Hui Chang, Kai-Cheng Wang, and Ta-Sung Lee. Machine learning based automatic diagnosis in mobile communication networks. *IEEE Transactions on Vehicular Technology*, 68(10):10081–10093, 2019.

[40] Peng Chen, Hongyun Liu, Ruyue Xin, Thierry Carval, Jiale Zhao, Yunni Xia, and Zhiming Zhao. Effectively detecting operational anomalies in large-scale iot data infrastructures by using a gan-based predictive model. *The Computer Journal*, 65(11):2909–2925, 2022.

[41] Pengfei Chen, Yong Qi, and Di Hou. Causeinfer: Automated end-to-end performance diagnosis with hierarchical causality graph in cloud environment. *IEEE transactions on services computing*, 12(2):214–230, 2016.

[42] Pengfei Chen, Yong Qi, Pengfei Zheng, and Di Hou. Causeinfer: Automatic and distributed performance diagnosis with hierarchical causality graph in large distributed systems. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pages 1887–1895. IEEE, 2014.

[43] Xiangyi Chen, Yuanguo Bi, Xueping Chen, Hai Zhao, Nan Cheng, Fuliang Li, and Wenlin Cheng. Dynamic service migration and request routing for microservice in multi-cell mobile edge computing. *IEEE Internet of Things Journal*, 2022.

[44] Xuliang Chen, Jianhui Jiang, Wei Zhang, and Xuzei Xia. Fault diagnosis for open source software based on dynamic tracking. In *2020 7th International Conference on Dependable Systems and Their Applications (DSA)*, pages 263–268. IEEE, 2020.

[45] Zhuangbin Chen, Yu Kang, Liqun Li, Xu Zhang, Hongyu Zhang, Hui Xu, Yangfan Zhou, Li Yang, Jeffrey Sun, Zhangwei Xu, et al. Towards intelligent incident management: why we need it and how we make it. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1487–1497, 2020.

[46] David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554, 2002.

[47] Jin-Hee Cho, Shouhuai Xu, Patrick M Hurley, Matthew Mackay, Trevor Benjamin, and Mark Beaumont. Stram: Measuring the trustworthiness of computer-based systems. *ACM*

Computing Surveys (CSUR), 51(6):1–47, 2019.

[48] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W Kempa-Liehr. Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). *Neurocomputing*, 307:72–77, 2018.

[49] Xu Chu, Ihab F Ilyas, Sanjay Krishnan, and Jiannan Wang. Data cleaning: Overview and emerging challenges. In *Proceedings of the 2016 international conference on management of data*, pages 2201–2206, 2016.

[50] David A Cohn, Zoubin Ghahramani, and Michael I Jordan. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145, 1996.

[51] European Commission, Content Directorate-General for Communications Networks, and Technology. *Ethics guidelines for trustworthy AI*. Publications Office, 2019.

[52] Ian Covert, Scott M Lundberg, and Su-In Lee. Understanding global feature contributions with additive importance measures. *Advances in Neural Information Processing Systems*, 33:17212–17223, 2020.

[53] Lei Cui, Youyang Qu, Gang Xie, Deze Zeng, Ruidong Li, Shigen Shen, and Shui Yu. Security and privacy-enhanced federated learning for anomaly detection in iot infrastructures. *IEEE Transactions on Industrial Informatics*, 18(5):3492–3500, 2021.

[54] Barnan Das, Narayanan C Krishnan, and Diane J Cook. Racog and wracog: Two probabilistic oversampling techniques. *IEEE transactions on knowledge and data engineering*, 27(1):222–234, 2014.

[55] Sjoerd de Vries and Dirk Thierens. A reliable ensemble based approach to semi-supervised learning. *Knowledge-Based Systems*, 215:106738, 2021.

[56] Andac Demir, Toshiaki Koike-Akino, Ye Wang, and Deniz Erdoğmuş. Eeg-gat: Graph attention networks for classification of electroencephalogram (eeg) signals. In *2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 30–35. IEEE, 2022.

[57] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research*, 7:1–30, 2006.

[58] Ailin Deng and Bryan Hooi. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4027–4035, 2021.

[59] Kaize Ding, Jundong Li, and Huan Liu. Interactive anomaly detection on attributed networks. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pages 357–365, 2019.

[60] Francesca Dominici, Aidan McDermott, Scott L Zeger, and Jonathan M Samet. On the use of generalized additive models in time-series studies of air pollution and health. *American journal of epidemiology*, 156(3):193–203, 2002.

[61] Xibin Dong, Zhiwen Yu, Wenming Cao, Yifan Shi, and Qianli Ma. A survey on ensemble learning. *Frontiers of Computer Science*, 14(2):241–258, 2020.

[62] Min Du, Ruoxi Jia, and Dawn Song. Robust anomaly detection and backdoor attack detection via differential privacy. *arXiv preprint arXiv:1911.07116*, 2019.

[63] Min Du, Feifei Li, Guineng Zheng, and Vivek 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*, pages 1285–1298, 2017.

[64] Xiaoyu Duan, Shi Ying, Wanli Yuan, Hailong Cheng, and Xiang Yin. Qllog: A log anomaly detection method based on q-learning algorithm. *Information Processing & Management*, 58(3):102540, 2021.

[65] Elias P Duarte Jr, Roverli P Ziwich, and Luiz CP Albini. A survey of comparison-based system-level diagnosis. *ACM Computing Surveys (CSUR)*, 43(3):1–56, 2011.

[66] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to

sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[67] Naoual El Aboudi and Laila Benhlima. Review on wrapper feature selection approaches. In *2016 International Conference on Engineering & MIS (ICEMIS)*, pages 1–5. IEEE, 2016.

[68] Chris Engelbert. Rebooting your services for monitoring is so 2015. https://www.instana.com/blog/rebooting-your-services-for-monitoring-is-so-2015/, August 3, 2020.

[69] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Adversarial attacks on deep neural networks for time series classification. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[70] Douglas H Fisher. A selected summary of ai for computational sustainability. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[71] Ana LN Fred and Anil K Jain. Combining multiple clusterings using evidence accumulation. *IEEE transactions on pattern analysis and machine intelligence*, 27(6):835–850, 2005.

[72] Yoav Freund and Llew Mason. The alternating decision tree learning algorithm. In *icml*, volume 99, pages 124–133. Citeseer, 1999.

[73] Antonio Galicia, R Talavera-Llames, A Troncoso, Irena Koprinska, and Francisco Martínez-Álvarez. Multi-step forecasting for big data time series based on ensemble learning. *Knowledge-Based Systems*, 163:830–841, 2019.

[74] Yu Gan, Yanqi Zhang, Kelvin Hu, Dailun Cheng, Yuan He, Meghna Pancholi, and Christina Delimitrou. Seer: Leveraging big data to navigate the complexity of performance debugging in cloud microservices. In *Proceedings of the twenty-fourth international conference on architectural support for programming languages and operating systems*, pages 19–33, 2019.

[75] Jiechao Gao, Haoyu Wang, and Haiying Shen. Task failure prediction in cloud data centers using deep learning. *IEEE transactions on services computing*, 2020.

[76] Adrià Gascón, Phillipp Schoppmann, Borja Balle, Mariana Raykova, Jack Doerner, Samee Zahur, and David Evans. Secure linear regression on vertically partitioned datasets. *IACR Cryptol. ePrint Arch.*, 2016:892, 2016.

[77] Deshani Geethika, Malith Jayasinghe, Yasas Gunarathne, Thilina Ashen Gamage, Sudaraka Jayathilaka, Surangika Ranathunga, and Srinath Perera. Anomaly detection in high-performance api gateways. In *2019 International Conference on High Performance Computing & Simulation (HPCS)*, pages 995–1001. IEEE, 2019.

[78] Alexander Geiger, Dongyu Liu, Sarah Alnegheimish, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Tadgan: Time series anomaly detection using generative adversarial networks. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 33–43. IEEE, 2020.

[79] Amir Gholami and Anurag K Srivastava. Comparative analysis of ml techniques for data-driven anomaly detection, classification and localization in distribution system. In *2020 52nd North American Power Symposium (NAPS)*, pages 1–6. IEEE, 2021.

[80] Sukhpal Singh Gill, Minxian Xu, Carlo Ottaviani, Panos Patros, Rami Bahsoon, Arash Shaghaghi, Muhammed Golec, Vlado Stankovski, Huaming Wu, Ajith Abraham, et al. Ai for next generation computing: Emerging trends and future directions. *Internet of Things*, 19:100514, 2022.

[81] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[82] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[83] Adam Goodge, Bryan Hooi, See-Kiong Ng, and Wee Siong Ng. Robustness of autoencoders for anomaly detection under adversarial impact. In *IJCAI*, pages 1244–1250, 2020.

[84] Brendan Gregg. *Systems performance: enterprise and the cloud*. Pearson Education, 2014.

[85] Zijie Guan, Jinjin Lin, and Pengfei Chen. On anomaly detection and root cause analysis of microservice systems. In *International Conference on Service-Oriented Computing*, pages 465–469. Springer, 2018.

[86] David Gunning. Explainable artificial intelligence (xai). *Defense advanced research projects agency (DARPA), nd Web*, 2(2):1, 2017.

[87] Xiaofeng Guo, Xin Peng, Hanzhang Wang, Wanxue Li, Huai Jiang, Dan Ding, Tao Xie, and Liangfei Su. Graph-based trace analysis for microservice architecture understanding and problem diagnosis. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1387–1397, 2020.

[88] Gajanand Gupta. Algorithm for image processing using improved median filter and comparison of mean, median and improved median filter. *International Journal of Soft Computing and Engineering (IJSCE)*, 1(5):304–311, 2011.

[89] Shangbin Han, Qianhong Wu, Han Zhang, Bo Qin, Jiankun Hu, Xingang Shi, Linfeng Liu, and Xia Yin. Log-based anomaly detection with robust feature extraction and online learning. *IEEE Transactions on Information Forensics and Security*, 16:2300–2311, 2021.

[90] Songqiao Han, Xiyang Hu, Hailiang Huang, Minqi Jiang, and Yue Zhao. Adbench: Anomaly detection benchmark. *Advances in Neural Information Processing Systems*, 35:32142–32159, 2022.

[91] Samuel Harford, Fazle Karim, and Houshang Darabi. Adversarial attacks on multivariate time series. *arXiv preprint arXiv:2004.00410*, 2020.

[92] Kelsey Harley and Rodney Cooper. Information integrity: Are we there yet? *ACM Computing Surveys (CSUR)*, 54(2):1–35, 2021.

[93] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[94] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009.

[95] Pinjia He, Jieming Zhu, Shilin He, Jian Li, and Michael 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)*, pages 654–661. IEEE, 2016.

[96] Shilin He, Jieming Zhu, Pinjia He, and Michael R Lyu. Experience report: System log analysis for anomaly detection. In *2016 IEEE 27th international symposium on software reliability engineering (ISSRE)*, pages 207–218. IEEE, 2016.

[97] Patrik O Hoyer, Dominik Janzing, Joris M Mooij, Jonas Peters, Bernhard Schölkopf, et al. Nonlinear causal discovery with additive noise models. In *NIPS*, volume 21, pages 689–696. Citeseer, 2008.

[98] Min Hu, Zhiwei Ji, Ke Yan, Ye Guo, Xiaowei Feng, Jiaheng Gong, Xin Zhao, and Ligang Dong. Detecting anomalies in time series data via a meta-feature based approach. *Ieee Access*, 6:27760–27776, 2018.

[99] Dong Huang, Jian-Huang Lai, and Chang-Dong Wang. Robust ensemble clustering using probability trajectories. *IEEE transactions on knowledge and data engineering*, 28(5):1312–1326, 2015.

[100] Dong Huang, Jianhuang Lai, and Chang-Dong Wang. Ensemble clustering using factor

graph. *Pattern Recognition*, 50:131–142, 2016.

[101] Dong Huang, Chang-Dong Wang, Jian-Sheng Wu, Jian-Huang Lai, and Chee-Keong Kwoh. Ultra-scalable spectral clustering and ensemble clustering. *IEEE Transactions on Knowledge and Data Engineering*, 32(6):1212–1226, 2019.

[102] Shaohan Huang, Yi Liu, Carol Fung, Rong He, Yining Zhao, Hailong Yang, and Zhongzhi Luan. Hitanomaly: Hierarchical transformers for anomaly detection in system log. *IEEE transactions on network and service management*, 17(4):2064–2076, 2020.

[103] Siteng Huang, Donglin Wang, Xuehan Wu, and Ao Tang. Dsanet: Dual self-attention network for multivariate time series forecasting. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 2129–2132, 2019.

[104] Tao Huang, Pengfei Chen, and Ruipeng Li. A semi-supervised vae based active anomaly detection framework in multivariate time series for online systems. In *Proceedings of the ACM Web Conference 2022*, pages 1797–1806, 2022.

[105] Zijie Huang, Yulei Wu, Niccolò Tempini, Hui Lin, and Hao Yin. An energy-efficient and trustworthy unsupervised anomaly detection framework (eatu) for iiot. *ACM Transactions on Sensor Networks (TOSN)*, 2022.

[106] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 387–395, 2018.

[107] Olumuyiwa Ibidunmoye. *Performance anomaly detection and resolution for autonomous clouds*. PhD thesis, Umeå University, 2017.

[108] Olumuyiwa Ibidunmoye, Francisco Hernández-Rodriguez, and Erik Elmroth. Performance anomaly detection and bottleneck identification. *ACM Computing Surveys (CSUR)*, 48(1):1–35, 2015.

[109] Brian Kenji Iwana and Seiichi Uchida. An empirical survey of data augmentation for time series classification with neural networks. *Plos one*, 16(7):e0254841, 2021.

[110] Navdeep Jaitly and Geoffrey E Hinton. Vocal tract length perturbation (vtlp) improves speech recognition. In *Proc. ICML Workshop on Deep Learning for Audio, Speech and Language*, volume 117, page 21, 2013.

[111] Hiranya Jayathilaka, Chandra Krintz, and Rich Wolski. Detecting performance anomalies in cloud platform applications. *IEEE Transactions on Cloud Computing*, 8(3):764–777, 2018.

[112] Glen Jeh and Jennifer Widom. Scaling personalized web search. In *Proceedings of the 12th international conference on World Wide Web*, pages 271–279, 2003.

[113] Minqi Jiang, Chaochuan Hou, Ao Zheng, Xiyang Hu, Songqiao Han, Hailiang Huang, Xiangnan He, Philip S Yu, and Yue Zhao. Weakly supervised anomaly detection: A survey. *arXiv preprint arXiv:2302.04549*, 2023.

[114] Xiao-Yuan Jing, Xinyu Zhang, Xiaoke Zhu, Fei Wu, Xinge You, Yang Gao, Shiguang Shan, and Jing-Yu Yang. Multiset feature learning for highly imbalanced data classification. *IEEE transactions on pattern analysis and machine intelligence*, 43(1):139–156, 2019.

[115] Yeon-Jee Jung, Seung-Ho Han, and Ho-Jin Choi. Explaining cnn and rnn using selective layer-wise relevance propagation. *IEEE Access*, 9:18670–18681, 2021.

[116] Markus Kalisch and Peter Bühlman. Estimating high-dimensional directed acyclic graphs with the pc-algorithm. *Journal of Machine Learning Research*, 8(3), 2007.

[117] Hui Kang, Haifeng Chen, and Guofei Jiang. Peerwatch: a fault detection and diagnosis tool for virtualized consolidation systems. In *Proceedings of the 7th international conference on Autonomic computing*, pages 119–128, 2010.

[118] Fazle Karim, Somshubra Majumdar, and Houshang Darabi. Adversarial attacks on time

series. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3309–3320, 2020.

[119] Davinder Kaur, Suleyman Uslu, Kaley J Rittichier, and Arjan Durresi. Trustworthy artificial intelligence: a review. *ACM Computing Surveys (CSUR)*, 55(2):1–38, 2022.

[120] Jayden Khakurel, Birgit Penzenstadler, Jari Porras, Antti Knutas, and Wenlu Zhang. The rise of artificial intelligence under the lens of sustainability. *Technologies*, 6(4):100, 2018.

[121] Myunghwan Kim, Roshan Sumbaly, and Sam Shah. Root cause detection in a service-oriented architecture. *ACM SIGMETRICS Performance Evaluation Review*, 41(1):93–104, 2013.

[122] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[123] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[124] Bartosz Krawczyk. Cost-sensitive one-vs-one ensemble for multi-class imbalanced data. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 2447–2452. IEEE, 2016.

[125] Abhishek Kumar, Tristan Braud, Sasu Tarkoma, and Pan Hui. Trustworthy ai in the age of pervasive computing and big data. In *2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 1–6. IEEE, 2020.

[126] Ludmila I Kuncheva and Christopher J Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning*, 51(2):181–207, 2003.

[127] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 95–104, 2018.

[128] Bo Li, Peng Qi, Bo Liu, Shuai Di, Jingen Liu, Jiquan Pei, Jinfeng Yi, and Bowen Zhou. Trustworthy ai: From principles to practices. *ACM Computing Surveys*, 55(9):1–46, 2023.

[129] Bohan Li, Yutai Hou, and Wanxiang Che. Data augmentation approaches in natural language processing: A survey. *AI Open*, 2022.

[130] Kaiyu Li, Guoliang Li, Yong Wang, Yan Huang, Zitao Liu, and Zhongqin Wu. Crowdrl: an end-to-end reinforcement learning framework for data labelling. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 289–300. IEEE, 2021.

[131] Ruinian Li, Tianyi Song, Bo Mei, Hong Li, Xiuzhen Cheng, and Limin Sun. Blockchain for large-scale internet of things data storage and protection. *IEEE Transactions on Services Computing*, 12(5):762–771, 2018.

[132] Xiaoyun Li, Pengfei Chen, Linxiao Jing, Zilong He, and Guangba Yu. Swisslog: Robust and unified deep learning based log anomaly detection for diverse faults. In *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, pages 92–103. IEEE, 2020.

[133] Xiaoyun Li, Pengfei Chen, Linxiao Jing, Zilong He, and Guangba Yu. Swisslog: Robust anomaly detection and localization for interleaved unstructured logs. *IEEE Transactions on Dependable and Secure Computing*, 2022.

[134] Zeyan Li, Junjie Chen, Yihao Chen, CHENGYANG LUO, Yiwei Zhao, Yongqian Sun, Kaixin Sui, Xiping Wang, Dapeng Liu, Xing Jin, et al. Generic and robust root cause localization for multi-dimensional data in online service systems. *Available at SSRN 4100992*.

[135] Zeyan Li, Junjie Chen, Rui Jiao, Nengwen Zhao, Zhijun Wang, Shuwei Zhang, Yanjun Wu, Long Jiang, Leiqin Yan, Zikai Wang, et al. Practical root cause localization for mi-

croservice systems via trace analysis. In *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*, pages 1–10. IEEE, 2021.

[136] Zeyan Li, Chengyang Luo, Yiwei Zhao, Yongqian Sun, Kaixin Sui, Xiping Wang, Dapeng Liu, Xing Jin, Qi Wang, and Dan Pei. Generic and robust localization of multi-dimensional root causes. In *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*, pages 47–57. IEEE, 2019.

[137] Zeyan Li, Nengwen Zhao, Mingjie Li, Xianglin Lu, Lixin Wang, Dongdong Chang, Xiaohui Nie, Li Cao, Wenzhi Zhang, Kaixin Sui, et al. Actionable and interpretable fault localization for recurring failures in online service systems. *arXiv preprint arXiv:2207.09021*, 2022.

[138] Zeyan Li, Nengwen Zhao, Shenglin Zhang, Yongqian Sun, Pengfei Chen, Xidao Wen, Minghua Ma, and Dan Pei. Constructing large-scale real-world benchmark datasets for aiops. *arXiv preprint arXiv:2208.03938*, 2022.

[139] Haoran Liang, Lei Song, Jianxing Wang, Lili Guo, Xuzhi Li, and Ji Liang. Robust unsupervised anomaly detection via multi-time scale dcgans with forgetting mechanism for industrial multivariate time series. *Neurocomputing*, 423:444–462, 2021.

[140] Wei Liang, Yongkai Fan, Kuan-Ching Li, Dafang Zhang, and Jean-Luc Gaudiot. Secure data storage and recovery in industrial blockchain network environments. *IEEE Transactions on Industrial Informatics*, 16(10):6543–6552, 2020.

[141] JinJin Lin, Pengfei Chen, and Zibin Zheng. Microscope: Pinpoint performance issues with causal graphs in micro-service environments. In *International Conference on Service-Oriented Computing*, pages 3–20. Springer, 2018.

[142] Qingwei Lin, Jian-Guang Lou, Hongyu Zhang, and Dongmei Zhang. idice: problem identification for emerging issues. In *Proceedings of the 38th International Conference on Software Engineering*, pages 214–224, 2016.

[143] Qingwei Lin, Hongyu Zhang, Jian-Guang Lou, Yu Zhang, and Xuewei Chen. Log clustering based problem identification for online service systems. In *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*, pages 102–111. IEEE, 2016.

[144] Bo Liu, Ming Ding, Sina Shaham, Wenny Rahayu, Farhad Farokhi, and Zihuai Lin. When machine learning meets privacy: A survey and outlook. *ACM Computing Surveys (CSUR)*, 54(2):1–36, 2021.

[145] Dengzhi Liu, Yong Zhang, Dongbao Jia, Qiaosheng Zhang, Xuefeng Zhao, and Huan Rong. Toward secure distributed data storage with error locating in blockchain enabled edge computing. *Computer Standards & Interfaces*, 79:103560, 2022.

[146] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE, 2008.

[147] Haochen Liu, Yiqi Wang, Wenqi Fan, Xiaorui Liu, Yaxin Li, Shaili Jain, Yunhao Liu, Anil K Jain, and Jiliang Tang. Trustworthy ai: A computational perspective. *arXiv preprint arXiv:2107.06641*, 2021.

[148] Ping Liu, Haowen Xu, Qianyu Ouyang, Rui Jiao, Zhekang Chen, Shenglin Zhang, Jiahai Yang, Linlin Mo, Jice Zeng, Wenman Xue, et al. Unsupervised detection of microservice trace anomalies through service-level deep bayesian networks. In *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, pages 48–58. IEEE, 2020.

[149] Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):539–550, 2008.

[150] Yang Liu, Yan Kang, Chaoping Xing, Tianjian Chen, and Qiang Yang. A secure federated transfer learning framework. *IEEE Intelligent Systems*, 35(4):70–82, 2020.

[151] Yi Liu, Sahil Garg, Jiangtian Nie, Yang Zhang, Zehui Xiong, Jiawen Kang, and M Shamim Hossain. Deep anomaly detection for time-series data in industrial iot: A communication-efficient on-device federated learning approach. *IEEE Internet of Things Journal*, 8(8):6348–6358, 2020.

[152] Shao-Yuan Lo, Poojan Oza, and Vishal M Patel. Adversarially robust one-class novelty detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[153] Jian-Guang Lou, Qiang Fu, Shenqi Yang, Ye Xu, and Jiang Li. Mining invariants from console logs for system problem detection. In *2010 USENIX Annual Technical Conference (USENIX ATC 10)*, 2010.

[154] Guanghong Lu, Chunhui Duan, Guohao Zhou, Xuan Ding, and Yunhao Liu. Privacy-preserving outlier detection with high efficiency over distributed datasets. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, pages 1–10. IEEE, 2021.

[155] Meng Lu. Embedded feature selection accounting for unknown data heterogeneity. *Expert Systems with Applications*, 119:350–361, 2019.

[156] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[157] Meng Ma, Weilan Lin, Disheng Pan, and Ping Wang. Ms-rank: Multi-metric and self-adaptive root cause diagnosis for microservice applications. In *2019 IEEE International Conference on Web Services (ICWS)*, pages 60–67. IEEE, 2019.

[158] Meng Ma, Weilan Lin, Disheng Pan, and Ping Wang. Self-adaptive root cause diagnosis for large-scale microservice architecture. *IEEE Transactions on Services Computing*, 2020.

[159] Meng Ma, Jingmin Xu, Yuan Wang, Pengfei Chen, Zonghua Zhang, and Ping Wang. Automap: Diagnose your microservice-based web applications automatically. In *Proceedings of The Web Conference 2020*, pages 246–258, 2020.

[160] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[161] Leonardo Mariani, Cristina Monni, Mauro Pezzé, Oliviero Riganelli, and Rui Xin. Localizing faults in cloud systems. In *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*, pages 262–273. IEEE, 2018.

[162] Joao Marques-Silva and Alexey Ignatiev. Delivering trustworthy ai through formal xai. In *Proc. of AAAI*, pages 3806–3814, 2022.

[163] David Mease, Abraham J Wyner, and Andreas Buja. Boosted classification trees and class probability/quantile estimation. *Journal of Machine Learning Research*, 8(3), 2007.

[164] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)*, 54(6):1–35, 2021.

[165] Yuan Meng, Shenglin Zhang, Yongqian Sun, Ruru Zhang, Zhilong Hu, Yiyin Zhang, Chenyang Jia, Zhaogang Wang, and Dan Pei. Localizing failure root causes in a microservice through causality inference. In *2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*, pages 1–10. IEEE, 2020.

[166] Prapanna Mondal, Labani Shit, and Saptarsi Goswami. Study of effectiveness of time series modeling (arima) in forecasting stock prices. *International Journal of Computer Science, Engineering and Applications*, 4(2):13, 2014.

[167] Nour Moustafa, Marwa Keshk, Kim-Kwang Raymond Choo, Timothy Lynar, Seyit Camtepe, and Monica Whitty. Dad: a distributed anomaly detection system using ensemble one-class statistical learning in edge networks. *Future Generation Computer Systems*, 118:240–251, 2021.

[168] Leland Gerson Neuberg. Causality: models, reasoning, and inference, by judea pearl, cambridge university press, 2000. *Econometric Theory*, 19(4):675–685, 2003.

[169] Bhuvaneswari Amma NG and S Selvakumar. Anomaly detection framework for internet of things traffic using vector convolutional deep learning approach in fog environment. *Future Generation Computer Systems*, 113:255–265, 2020.

[170] Quoc Phong Nguyen, Kar Wai Lim, Dinil Mon Divakaran, Kian Hsiang Low, and Mun Choon Chan. Gee: A gradient-based explainable variational autoencoder for network anomaly detection. In *2019 IEEE Conference on Communications and Network Security (CNS)*, pages 91–99. IEEE, 2019.

[171] Nikolaos Nikolaou, Narayanan Edakunni, Meelis Kull, Peter Flach, and Gavin Brown. Cost-sensitive boosting algorithms: Do we really need them? *Machine Learning*, 104(2):359–384, 2016.

[172] Oren Ninio. Practical guide on setting up prometheus and grafana for monitoring your microservices. https://komodor.com/blog/setting-up-prometheus-and-grafana-for-monitoring-your-microservices/, Sep 28, 2022.

[173] U.S. Government Accountability Office. Artificial intelligence: An accountability framework for federal agencies and other entities. *https://www.gao.gov/products/gao-21-519sp*, 2021.

[174] Izaskun Oregi, Javier Del Ser, Aritz Perez, and Jose A Lozano. Adversarial sample crafting for time series classification with elastic similarity measures. In *International Symposium on Intelligent and Distributed Computing*, pages 26–39. Springer, 2018.

[175] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[176] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. Deep learning for anomaly detection: A review. *ACM Computing Surveys (CSUR)*, 54(2):1–38, 2021.

[177] Daehyung Park, Yuuna Hoshi, and Charles C Kemp. A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder. *IEEE Robotics and Automation Letters*, 3(3):1544–1551, 2018.

[178] Jiaming Pei, Kaiyang Zhong, Mian Ahmad Jan, and Jinhai Li. Personalized federated learning framework for network traffic anomaly detection. *Computer Networks*, 209:108906, 2022.

[179] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

[180] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 427–438, 2000.

[181] Faraz Rasheed, Peter Peng, Reda Alhajj, and Jon Rokne. Fourier transform based spatial outlier mining. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 317–324. Springer, 2009.

[182] Khandakar M Rashid and Joseph Louis. Window-warping: a time series data augmentation of imu data for construction equipment activity identification. In *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, volume 36, pages 651–657. IAARC Publications, 2019.

[183] General Data Protection Regulation. General data protection regulation (gdpr). *Intersoft Consulting, Accessed in October*, 24(1), 2018.

[184] Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou, Tony Xing, Mao Yang, Jie Tong, and Qi Zhang. Time-series anomaly detection service at microsoft. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 3009–3017, 2019.

[185] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B Gupta,

Xiaojiang Chen, and Xin Wang. A survey of deep active learning. *ACM computing surveys (CSUR)*, 54(9):1–40, 2021.

[186] Ana Reyna, Cristian Martín, Jaime Chen, Enrique Soler, and Manuel Díaz. On blockchain and its integration with iot. challenges and opportunities. *Future generation computer systems*, 88:173–190, 2018.

[187] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " 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*, pages 1135–1144, 2016.

[188] Chris Ridings and Mike Shishigin. Pagerank uncovered. *Technical Paper for the Search Engine Optimization Online Community*, 2002.

[189] Daniele Romanini, Adam James Hall, Pavlos Papadopoulos, Tom Titcombe, Abbas Ismail, Tudor Cebere, Robert Sandmann, Robin Roehm, and Michael A Hoeh. Pyvertical: A vertical federated learning framework for multi-headed splitnn. *arXiv preprint arXiv:2104.00489*, 2021.

[190] Benjamin IP Rubinstein, Peter L Bartlett, Ling Huang, and Nina Taft. Learning in a large function space: Privacy-preserving mechanisms for svm learning. *arXiv preprint arXiv:0911.5708*, 2009.

[191] Karen Sachs, Omar Perez, Dana Pe'er, Douglas A Lauffenburger, and Garry P Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721):523–529, 2005.

[192] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd workshop on machine learning for sensory data analysis*, pages 4–11, 2014.

[193] Noelia Sánchez-Maroño, Amparo Alonso-Betanzos, and María Tombilla-Sanromán. Filter methods for feature selection–a comparative study. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 178–187. Springer, 2007.

[194] C Saranya and G Manikandan. A study on normalization techniques for privacy preserving data mining. *International Journal of Engineering and Technology (IJET)*, 5(3):2701–2704, 2013.

[195] Nishant Saurabh, Carlos Rubia, Anandakumar Palanisamy, Spiros Koulouzis, Mirsat Sefidanoski, Antorweep Chakravorty, Zhiming Zhao, Aleksandar Karadimce, and Radu Prodan. The articonf approach to decentralized car-sharing. *Blockchain: Research and Applications*, 2(3):100013, 2021.

[196] Carla Sauvanaud, Kahina Lazri, Mohamed Kaâniche, and Karama Kanoun. Anomaly detection and root cause localization in virtual network functions. In *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, pages 196–206. IEEE, 2016.

[197] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

[198] Patrick Schratz, Jannes Muenchow, Eugenia Iturritxa, Jakob Richter, and Alexander Brenning. Hyperparameter tuning and performance assessment of statistical and machine-learning algorithms using spatial data. *Ecological Modelling*, 406:109–120, 2019.

[199] Arnaldo Sgueglia, Andrea Di Sorbo, Corrado Aaron Visaggio, and Gerardo Canfora. A systematic literature review of iot time series anomaly detection solutions. *Future Generation Computer Systems*, 2022.

[200] Meet Shah, Mohammedhasan Shaikh, Vishwajeet Mishra, and Grinal Tuscano. Decentralized cloud storage using blockchain. In *2020 4th International conference on trends in electronics and informatics (ICOEI)(48184)*, pages 384–389. IEEE, 2020.

[201] Shohei Shimizu, Patrik O Hoyer, Aapo Hyvärinen, Antti Kerminen, and Michael Jordan.

A linear non-gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7(10), 2006.

[202] Shohei Shimizu, Takanori Inazumi, Yasuhiro Sogawa, Aapo Hyvärinen, Yoshinobu Kawahara, Takashi Washio, Patrik O Hoyer, and Kenneth Bollen. Directlingam: A direct method for learning a linear non-gaussian structural equation model. *The Journal of Machine Learning Research*, 12:1225–1248, 2011.

[203] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[204] Min B Shrestha and Guna R Bhatta. Selecting appropriate methodological framework for time series data analysis. *The Journal of Finance and Data Science*, 4(2):71–89, 2018.

[205] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR, 2017.

[206] Md Amran Siddiqui, Alan Fern, Thomas G Dietterich, Ryan Wright, Alec Theriault, and David W Archer. Feedback-guided anomaly discovery via online optimization. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2200–2209, 2018.

[207] Alban Siffer, Pierre-Alain Fouque, Alexandre Termier, and Christine Largouet. Anomaly detection in streams with extreme value theory. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1067–1075, 2017.

[208] Daniel Smilkov, Nikhil Thorat, Charles Nicholson, Emily Reif, Fernanda B Viégas, and Martin Wattenberg. Embedding projector: Interactive visualization and interpretation of embeddings. *arXiv preprint arXiv:1611.05469*, 2016.

[209] Jacopo Soldani and Antonio Brogi. Anomaly detection and failure root cause analysis in (micro) service-based cloud applications: A survey. *ACM Computing Surveys (CSUR)*, 55(3):1–39, 2022.

[210] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE global conference on signal and information processing*, pages 245–248. IEEE, 2013.

[211] Yujia Song, Ruyue Xin, Peng Chen, Rui Zhang, Juan Chen, and Zhiming Zhao. Identifying performance anomalies in fluctuating cloud environments: a robust correlative-gnn-based explainable approach. *Future Generation Computer Systems*, 145:77–86, 2023.

[212] Yujia Song, Ruyue Xin, Peng Chen, Rui Zhang, Juan Chen, and Zhiming Zhao. Identifying performance anomalies in fluctuating cloud environments: A robust correlative-gnn-based explainable approach. *Future Generation Computer Systems*, 2023. doi: 10.1016/j.future.2023.03.020.

[213] Vasilis A Sotiris, W Tse Peter, and Michael G Pecht. Anomaly detection through a bayesian support vector machine. *IEEE Transactions on Reliability*, 59(2):277–286, 2010.

[214] Peter Spirtes, Clark N Glymour, Richard Scheines, and David Heckerman. *Causation, prediction, and search*. MIT press, 2000.

[215] Frank Spitzer. *Principles of random walk*, volume 34. Springer Science & Business Media, 2001.

[216] Cindy Sridharan. *Distributed systems observability: a guide to building robust systems*. O'Reilly Media, 2018.

[217] Hudan Studiawan and Ferdous Sohel. Performance evaluation of anomaly detection in imbalanced system log data. In *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, pages 239–246. IEEE, 2020.

[218] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan 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, pages 2828–2837, 2019.

[219] Peijie Sun, E Yuepeng, Tong Li, Yulei Wu, Jingguo Ge, Junling You, and Bingzhen Wu. Context-aware learning for anomaly detection with imbalanced log data. In *2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 449–456. IEEE, 2020.

[220] Yongqian Sun, Youjian Zhao, Ya Su, Dapeng Liu, Xiaohui Nie, Yuan Meng, Shiwen Cheng, Dan Pei, Shenglin Zhang, Xianping Qu, et al. Hotspot: Anomaly localization for additive kpis with multi-dimensional attributes. *IEEE Access*, 6:10909–10923, 2018.

[221] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[222] Pedro Tabacof, Julia Tavares, and Eduardo Valle. Adversarial images for variational autoencoders. *arXiv preprint arXiv:1612.00155*, 2016.

[223] Bayu Adhi Tama, Lewis Nkenyereye, SM Riazul Islam, and Kyung-Sup Kwak. An enhanced anomaly detection in web traffic using a stack of classifier ensemble. *IEEE Access*, 8:24120–24134, 2020.

[224] Zhihong Tian, Chaochao Luo, Jing Qiu, Xiaojiang Du, and Mohsen Guizani. A distributed deep learning system for web attack detection on edge devices. *IEEE Transactions on Industrial Informatics*, 16(3):1963–1971, 2019.

[225] Ivan Tomek. An experiment with the edited nearest-nieghbor rule. 1976.

[226] Ehsan Toreini, Mhairi Aitken, Kovila Coopamootoo, Karen Elliott, Carlos Gonzalez Zelaya, and Aad Van Moorsel. The relationship between trust in ai and trustworthy machine learning technologies. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 272–283, 2020.

[227] Shreshth Tuli, Giuliano Casale, and Nicholas R Jennings. Pregan: Preemptive migration prediction network for proactive fault-tolerant edge computing. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, pages 670–679. IEEE, 2022.

[228] Shreshth Tuli, Giuliano Casale, and Nicholas R Jennings. Tranad: Deep transformer networks for anomaly detection in multivariate time series data. *arXiv preprint arXiv:2201.07284*, 2022.

[229] Hristos Tyralis, Georgia Papacharalampous, and Andreas Langousis. Super ensemble learning for daily streamflow forecasting: Large-scale demonstration and comparison with multiple machine learning algorithms. *Neural Computing and Applications*, 33(8):3053–3068, 2021.

[230] Ikram Ul Haq, Iqbal Gondal, Peter Vamplew, and Simon Brown. Categorical features transformation with compact one-hot encoder for fraud detection in distributed environment. In *Australasian Conference on Data Mining*, pages 69–80. Springer, 2018.

[231] Terry T Um, Franz MJ Pfister, Daniel Pichler, Satoshi Endo, Muriel Lang, Sandra Hirche, Urban Fietzek, and Dana Kulić. Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks. In *Proceedings of the 19th ACM international conference on multimodal interaction*, pages 216–220, 2017.

[232] Pulakesh Upadhyaya, Kai Zhang, Can Li, Xiaoqian Jiang, and Yejin Kim. Scalable causal structure learning: New opportunities in biomedicine. *arXiv preprint arXiv:2110.07785*, 2021.

[233] Priyanka Vergadia. Introduction to google cloud's operations suite. https://cloud.google.com/blog/topics/developers-practitioners/introduction-google-clouds-operations-suite/, Dec 3, 2021.

[234] Lingzhi Wang, Nengwen Zhao, Junjie Chen, Pinnong Li, Wenchi Zhang, and Kaixin Sui.

Root-cause metric location for microservice systems via log anomaly detection. In *2020 IEEE International Conference on Web Services (ICWS)*, pages 142–150. IEEE, 2020.

[235] Ping Wang, Jingmin Xu, Meng Ma, Weilan Lin, Disheng Pan, Yuan Wang, and Pengfei Chen. Cloudranger: Root cause identification for cloud native systems. In *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 492–502. IEEE, 2018.

[236] Jonathan Waring, Charlotta Lindvall, and Renato Umeton. Automated machine learning: Review of the state-of-the-art and opportunities for healthcare. *Artificial intelligence in medicine*, 104:101822, 2020.

[237] Muhammad Waseem, Peng Liang, Mojtaba Shahin, Amleto Di Salle, and Gastón Márquez. Design, monitoring, and testing of microservices systems: The practitioners' perspective. *Journal of Systems and Software*, 182:111061, 2021.

[238] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020.

[239] Xing Wei and W Bruce Croft. Lda-based document models for ad-hoc retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 178–185, 2006.

[240] Jianping Weng, Jessie Hui Wang, Jiahai Yang, and Yang Yang. Root cause analysis of anomalies of multitier services in public clouds. *IEEE/ACM Transactions on Networking*, 26(4):1646–1659, 2018.

[241] Maranke Wieringa. What to account for when accounting for algorithms: a systematic literature review on algorithmic accountability. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 1–18, 2020.

[242] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[243] Maksymilian Wojtas and Ke Chen. Feature importance ranking for deep learning. *Advances in Neural Information Processing Systems*, 33:5105–5114, 2020.

[244] Canhua Wu, Nengwen Zhao, Lixin Wang, Xiaoqin Yang, Shining Li, Ming Zhang, Xing Jin, Xidao Wen, Xiaohui Nie, Wenchi Zhang, et al. Identifying root-cause metrics for incident diagnosis in online service systems. In *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*, pages 91–102. IEEE, 2021.

[245] Li Wu, Johan Tordsson, Alexander Acker, and Odej Kao. Microras: Automatic recovery in the absence of historical failure data for microservice systems. In *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*, pages 227–236. IEEE, 2020.

[246] Li Wu, Johan Tordsson, Jasmin Bogatinovski, Erik Elmroth, and Odej Kao. Microdiag: Fine-grained performance diagnosis for microservice systems. In *2021 IEEE/ACM International Workshop on Cloud Intelligence (CloudIntelligence)*, pages 31–36. IEEE, 2021.

[247] Li Wu, Johan Tordsson, Erik Elmroth, and Odej Kao. Microrca: Root cause localization of performance issues in microservices. In *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*, pages 1–9. IEEE, 2020.

[248] Li Wu, Johan Tordsson, Erik Elmroth, and Odej Kao. Causal inference techniques for microservice performance diagnosis: Evaluation and guiding recommendations. In *2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*, pages 21–30. IEEE, 2021.

[249] Doris Xin, Litian Ma, Jialin Liu, Stephen Macke, Shuchen Song, and Aditya Parameswaran. Accelerating human-in-the-loop machine learning: Challenges and op-

portunities. In *Proceedings of the second workshop on data management for end-to-end machine learning*, pages 1–4, 2018.

[250] Ruyue Xin, Hongyun Liu, Peng Chen, Paola Grosso, and Zhiming Zhao. Fired: a fine-grained robust performance diagnosis framework for cloud applications. *arXiv preprint arXiv:2209.01970*, 2022.

[251] Ruyue Xin, Hongyun Liu, Peng Chen, and Zhiming Zhao. Robust and accurate performance anomaly detection and prediction for cloud applications: a novel ensemble learning-based framework. *Journal of Cloud Computing*, 12(1):1–16, 2023.

[252] Ke Xu, Meng Xia, Xing Mu, Yun Wang, and Nan Cao. Ensemblelens: Ensemble-based visual exploration of anomaly detection algorithms with multidimensional data. *IEEE transactions on visualization and computer graphics*, 25(1):109–119, 2018.

[253] Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael I Jordan. Detecting large-scale system problems by mining console logs. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pages 117–132, 2009.

[254] Jian Yang, David Zhang, Alejandro F Frangi, and Jing-yu Yang. Two-dimensional pca: a new approach to appearance-based face representation and recognition. *IEEE transactions on pattern analysis and machine intelligence*, 26(1):131–137, 2004.

[255] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

[256] Jian Yin, Chunjing Gan, Kaiqi Zhao, Xuan Lin, Zhe Quan, and Zhi-Jie Wang. A novel model for imbalanced data classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6680–6687, 2020.

[257] Mogeng Yin, Madeleine Sheehan, Sidney Feygin, Jean-François Paiement, and Alexei Pozdnoukhov. A generative model of urban activities from cellular data. *IEEE Transactions on Intelligent Transportation Systems*, 19(6):1682–1696, 2017.

[258] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*, 2017.

[259] Feng Yu, Jun-zhou Luo, and Wei Li. A trustworthy network fault diagnosis approach. In *2009 First International Conference on Information Science and Engineering*, pages 90–94. IEEE, 2009.

[260] Guangba Yu, Pengfei Chen, Hongyang Chen, Zijie Guan, Zicheng Huang, Linxiao Jing, Tianjun Weng, Xinmeng Sun, and Xiaoyun Li. Microrank: End-to-end latency issue localization with extended spectrum analysis in microservice environments. In *Proceedings of the Web Conference 2021*, pages 3087–3098, 2021.

[261] Guangba Yu, Zicheng Huang, and Pengfei Chen. Tracerank: Abnormal service localization with dis-aggregated end-to-end tracing data in cloud native systems. *Journal of Software: Evolution and Process*, page e2413, 2021.

[262] Xiang Yu, Guoliang Li, Yudian Zheng, Yan Huang, Songfan Zhang, and Fei Chen. Crowdota: An online task assignment system in crowdsourcing. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 1629–1632. IEEE, 2018.

[263] Xiao Yu, Pallavi Joshi, Jianwu Xu, Guoliang Jin, Hui Zhang, and Guofei Jiang. Cloudseer: Workflow monitoring of cloud infrastructures via interleaved logs. *ACM SIGARCH Computer Architecture News*, 44(2):489–502, 2016.

[264] Yue Yu, Jie Chen, Tian Gao, and Mo Yu. Dag-gnn: Dag structure learning with graph neural networks. In *International Conference on Machine Learning*, pages 7154–7163. PMLR, 2019.

[265] Zhiwen Yu, Yidong Zhang, CL Philip Chen, Jane You, Hau-San Wong, Dan Dai, Si Wu, and Jun Zhang. Multiobjective semisupervised classifier ensemble. *IEEE transactions on cybernetics*, 49(6):2280–2293, 2018.

[266] Shuhan Yuan and Xintao Wu. Trustworthy anomaly detection: A survey. *arXiv preprint arXiv:2202.07787*, 2022.

[267] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, 30(9):2805–2824, 2019.

[268] Yue Yuan, Wenchang Shi, Bin Liang, and Bo Qin. An approach to cloud execution failure diagnosis based on exception logs in openstack. In *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, pages 124–131. IEEE, 2019.

[269] Fabio Massimo Zanzotto. Human-in-the-loop artificial intelligence. *Journal of Artificial Intelligence Research*, 64:243–252, 2019.

[270] Chen Zhang, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao. A survey on federated learning. *Knowledge-Based Systems*, 216:106775, 2021.

[271] Chenxi Zhang, Xin Peng, Chaofeng Sha, Ke Zhang, Zhenqing Fu, Xiya Wu, Qingwei Lin, and Dongmei Zhang. Deeptralog: Trace-log combined microservice anomaly detection through graph-based deep learning. In *Proceedings of the 44th International Conference on Software Engineering*, pages 623–634, 2022.

[272] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International conference on machine learning*, pages 7472–7482. PMLR, 2019.

[273] Ke Zhang, Jianwu Xu, Martin Renqiang Min, Guofei Jiang, Konstantinos Pelechrinis, and Hui Zhang. Automated it system failure prediction: A deep learning approach. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 1291–1300. IEEE, 2016.

[274] Tianwei Zhang, Zecheng He, and Ruby B Lee. Privacy-preserving machine learning through data obfuscation. *arXiv preprint arXiv:1807.01860*, 2018.

[275] Wei Zhang and Xiang Li. Federated transfer learning for intelligent fault diagnostics using deep adversarial networks with data privacy. *IEEE/ASME Transactions on Mechatronics*, 27(1):430–439, 2021.

[276] Xu Zhang, Yong Xu, Qingwei Lin, Bo Qiao, Hongyu Zhang, Yingnong Dang, Chunyu Xie, Xinsheng Yang, Qian Cheng, Ze Li, et al. Robust log-based anomaly detection on unstable log data. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 807–817, 2019.

[277] Yan-Ping Zhang, Li-Na Zhang, and Yong-Cheng Wang. Cluster-based majority under-sampling approaches for class imbalance learning. In *2010 2nd IEEE International Conference on Information and Financial Engineering*, pages 400–404. IEEE, 2010.

[278] Hang Zhao, Yujing Wang, Juanyong Duan, Congrui Huang, Defu Cao, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, and Qi Zhang. Multivariate time-series anomaly detection via graph attention network. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 841–850. IEEE, 2020.

[279] Hang Zhao, Yujing Wang, Juanyong Duan, Congrui Huang, Defu Cao, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, and Qi Zhang. Multivariate time-series anomaly detection via graph attention network. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 841–850. IEEE, 2020.

[280] Yue Zhao, Zain Nasrullah, and Zheng Li. Pyod: A python toolbox for scalable outlier detection. *arXiv preprint arXiv:1901.01588*, 2019.

[281] Zilong Zhao, Sophie Cerf, Robert Birke, Bogdan Robu, Sara Bouchenak, Sonia Ben Mokhtar, and Lydia Y Chen. Robust anomaly detection on unreliable data. In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 630–637. IEEE, 2019.

[282] Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. Dags with no tears: Continuous optimization for structure learning. *Advances in Neural Information Processing Systems*, 31, 2018.

[283] Yudian Zheng, Guoliang Li, Yuanbing Li, Caihua Shan, and Reynold Cheng. Truth inference in crowdsourcing: Is the problem solved? *Proceedings of the VLDB Endowment*, 10(5):541–552, 2017.

[284] Yudian Zheng, Jiannan Wang, Guoliang Li, Reynold Cheng, and Jianhua Feng. Qasca: A quality-aware task assignment system for crowdsourcing applications. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pages 1031–1046, 2015.

[285] Huan Zhou, Yang Hu, Xue Ouyang, Jinshu Su, Spiros Koulouzis, Cees de Laat, and Zhiming Zhao. Cloudsstorm: a framework for seamlessly programming and controlling virtual infrastructure functions during the devops lifecycle of cloud applications. *Software: Practice and Experience*, 49(10):1421–1447, 2019.

[286] Huan Zhou, Xue Ouyang, Zhijie Ren, Jinshu Su, Cees de Laat, and Zhiming Zhao. A blockchain based witness model for trustworthy cloud service level agreement enforcement. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019.

[287] Lijing Zhou, Licheng Wang, Yiru Sun, and Pin Lv. Beekeeper: A blockchain-based iot system with secure storage and homomorphic computation. *IEEE Access*, 6:43472–43488, 2018.

[288] Xiang Zhou, Xin Peng, Tao Xie, Jun Sun, Chao Ji, Dewei Liu, Qilin Xiang, and Chuan He. Latent error prediction and fault localization for microservice applications by learning from system trace logs. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 683–694, 2019.

[289] Yingjie Zhou, Xucheng Song, Yanru Zhang, Fanxing Liu, Ce Zhu, and Lingqiao Liu. Feature encoding with autoencoders for weakly supervised anomaly detection. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[290] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC, 2019.

[291] Jieming Zhu, Shilin He, Jinyang Liu, Pinjia He, Qi Xie, Zibin Zheng, and Michael R Lyu. Tools and benchmarks for automated log parsing. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 121–130. IEEE, 2019.

[292] Mingyi Zhu, Kejiang Ye, and Cheng-Zhong Xu. Network anomaly detection and identification based on deep learning methods. In *International Conference on Cloud Computing*, pages 219–234. Springer, 2018.

[293] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International conference on learning representations*, 2018.

Abbreviations

AI	Artificial Intelligence
AL	Active Learning
ANM	Additive Noise Model
AR	Autoregressive
BFS	Breadth First Search
CI	Causal Inference
CNN	Convolutional Neural Network
DAG	Directed Acyclic Graph
DApp	Decentralized Application
DFS	Depth First Search
ELBD	Ensemble Learning-Based Detection
EU	European Union
FFT	Fast Fourier Transform
FPR	False Position Rate
GDPR	General Data Protection Regulation
GNN	Graph Neural Network
GRU	Gate Recurrent Unit
IForest	Isolation Forest
IoT	Internet of Things
ISO	International Organization for Standardization
KNN	K-Nearest Neighbor

KPI	Key Performance Indicator
LiNGAM	Linear Non-Gaussian Acyclic Model
LOF	Local Outlier Factor
LSTM	Long Short Term Memory
ML	Machine Learning
MLP	Multi-Layer Perceptron
MSL	Mars Science Laboratory rover
OCSVM	One-Class Support Vector Machine
PC	Peter-Clark
PCA	Principal Components Analysis
SEM	Structural Equation Model
SLO	service level objective
SMAP	Soil Moisture Activate Passive satellite
SVM	Support Vector Machine
VAE	Variational Autoencoder
VM	Virtual Machine
XAI	Explainable AI

Summary

Cloud computing provides elastic and on-demand resources, enabling the development and operation of distributed applications across multiple machines and data centers. Distributed cloud applications are typically decomposed into smaller, specialized services that can be developed, deployed, and scaled independently. Nevertheless, complex service dependencies and the dynamic nature of cloud environments introduce abnormal performance phenomena, such as degraded response times due to resource saturation. These issues can significantly impact service quality and the user experience. Therefore, it is essential to diagnose and mitigate performance problems in distributed applications to ensure continuous operation.

In recent years, research on performance diagnosis systems for distributed applications has developed. AI-based performance diagnosis systems have gained popularity for their superior performance over traditional statistical methods, adaptability to large-scale data, and handling of complex scenarios. These systems typically consist of four main components: data collection, preprocessing, anomaly detection, and root cause localization. However, despite significant advancements, AI-based methods face potential hazards and may lose public trust due to poor robustness and inexplainability. Therefore, this thesis offers a systematic overview of trustworthiness requirements and state-of-the-art technologies in the development of AI-based trustworthy performance diagnosis systems. In addition, it is still challenging to:

- enhance performance data quality due to missing data labels and the existence of data noise and fluctuations.
- perform robust and accurate anomaly detection because of diverse anomalies and data patterns caused by dynamic cloud environments.
- perform precise and fine-grained root cause localization due to large numbers of anomalous metrics and complex anomaly propagation paths.

Om de bovenstaande uitdagingen aan te pakken, hebben we onze belangrijkste onderzoeksvraag geformuleerd als hoe de prestaties van runtime gedistribueerde

toepassingen in cloudomgevingen effectief te diagnosticeren. Om deze vraag te beantwoorden, hebben we een uitgebreid prestatiediagnosesysteem voorgesteld dat prestatiegegevens effectief kan verwerken, prestatieafwijkingen kan detecteren en hun oorzaken kan lokaliseren om bruikbare inzichten aan operators te verschaffen.

We first developed a novel FIne-grained pErformance Diagnosis (FIED) framework to achieve effective performance diagnosis of distributed applications. The FIED framework can collect real-time performance data, handle noise in monitoring metrics, effectively detect performance anomalies, and localize root causes of distributed applications with typical AI methods. To evaluate the framework, we implemented the deployment and monitoring of a DApp and utilized two public datasets. Our experiment results demonstrated the effect of metric selection on improving detection accuracy, the varying performance of existing detection methods on different datasets, and the feasibility of real-time root cause localization based on causal inference algorithms.

For anomaly detection, we observed that existing detection methods have varying detection accuracy and poor robustness. However, effective anomaly detection methods should meet challenging requirements, including high accuracy in detecting anomalies and robustness to changing data patterns. Therefore, we developed a weakly-supervised Ensemble Learning-Based Detection (ELBD) framework that integrates well-selected existing methods to achieve the highest accuracy and robustness for performance anomaly detection in distributed applications. Furthermore, we proposed a deep learning-based unsupervised method, CGNN-MHSA-AR, which leverages temporal and feature information to achieve superior accuracy and robustness for multivariate time series anomaly detection.

To enhance root cause localization in the performance diagnosis framework, we identified limitations in current CI methods, such as only extracting linear causal relations and relying on strict data distribution requirements. Additionally, existing research primarily focuses on coarse-grained localization, which only identifies faulty services. However, fine-grained root cause localization, pinpointing specific metrics on the faulty service, is more beneficial for operators. To address these issues, we developed the CausalRCA, which generates weighted causal graphs using a gradient-based causal structure learning method and employs a root cause inference method to determine root cause metrics. Furthermore, we conducted coarse- and fine-grained experiments, and results demonstrated that the CausalRCA outperforms baseline methods and offers fine-grained, automated, and real-time root cause localization.

Samenvatting

Cloud computing biedt elastische en on-demand middelen, waardoor de ontwikkeling en werking van gedistribueerde toepassingen over meerdere machines en datacenters mogelijk is. Gedistribueerde cloudtoepassingen worden doorgaans opgesplitst in kleinere, gespecialiseerde diensten die onafhankelijk kunnen worden ontwikkeld, ingezet en geschaald. Desondanks introduceren complexe servicemethodes en de dynamische aard van cloudomgevingen abnormale prestatiefenomenen, zoals verslechterde responstijden als gevolg van bronverzadiging. Deze problemen kunnen aanzienlijke invloed hebben op de dienstkwaliteit en de gebruikerservaring. Het is daarom essentieel om prestatieproblemen in gedistribueerde toepassingen te diagnosticeren en te verminderen om continue werking te waarborgen.

De afgelopen jaren heeft het onderzoek naar prestatiediagnosesystemen voor gedistribueerde toepassingen zich ontwikkeld. Op AI gebaseerde prestatiediagnosesystemen hebben aan populariteit gewonnen vanwege hun superieure prestaties ten opzichte van traditionele statistische methoden, hun aanpasbaarheid aan grootschalige gegevens en hun vermogen om met complexe scenario's om te gaan. Deze systemen bestaan doorgaans uit vier hoofdonderdelen: gegevensverzameling, voorverwerking, anomaliedetectie en het lokaliseren van de oorzaak van problemen. Ondanks aanzienlijke vooruitgang kunnen op AI gebaseerde methoden echter potentiële gevaren tegenkomen en het vertrouwen van het publiek verliezen vanwege slechte robuustheid en onverklaarbaarheid. Daarom biedt dit proefschrift een systematisch overzicht van betrouwbaarheidseisen en state-of-the-art technologieën in de ontwikkeling van op AI gebaseerde betrouwbare prestatiediagnosesystemen. Bovendien blijft het een uitdaging om:

- de kwaliteit van prestatiegegevens te verbeteren vanwege ontbrekende gegevenslabels en het bestaan van gegevensruis en schommelingen.
- voer robuuste en nauwkeurige anomaliedetectie uit vanwege diverse anomalieën en datapatronen veroorzaakt door dynamische cloudomgevingen.
- voer nauwkeurige lokalisatie van de hoofdoorzaak uit vanwege het grote aantal afwijkende meetgegevens en complexe voortplantingspaden.

Om deze uitdagingen aan te pakken, hebben we onze belangrijkste onderzoeksvraag geformuleerd als hoe de prestaties van runtime gedistribueerde toepassingen in cloudomgevingen effectief te diagnosticeren. Om deze vraag te beantwoorden, hebben we een uitgebreid prestatiediagnosesysteem voorgesteld dat prestatiegegevens effectief kan verwerken, prestatieafwijkingen kan detecteren en hun oorzaken kan lokaliseren om bruikbare inzichten aan operators te verschaffen.

We hebben eerst een nieuw Fijnmazig Prestatiediagnose (FIED) raamwerk ontwikkeld om effectieve prestatiediagnose van gedistribueerde toepassingen te bereiken. Het FIED-raamwerk kan realtime prestatiegegevens verzamelen, ruis in bewakingsmetingen verwerken, prestatieafwijkingen effectief detecteren en de oorzaken van problemen in gedistribueerde toepassingen lokaliseren met typische AI-methoden. Om het raamwerk te evalueren, hebben we de implementatie en bewaking van een DApp uitgevoerd en twee openbare datasets gebruikt. Onze experimentele resultaten toonden het effect van metrische selectie op het verbeteren van de detectie-accurheid, de variabele prestaties van bestaande detectiemethoden op verschillende datasets en de haalbaarheid van realtime lokalisatie van de oorzaken van problemen op basis van causaliteitsinferentiealgoritmen.

Voor anomaliedetectie hebben we waargenomen dat bestaande detectiemethoden variërende detectie-accurheid en zwakke robuustheid hebben. Effectieve anomaliedetectie zou echter aan uitdagende eisen moeten voldoen, waaronder een hoge accurheid bij het detecteren van afwijkingen en robuustheid tegen veranderende gegevenspatronen. Daarom hebben we een zwak-geassisteerd Ensemble Leer-Gebaseerd Detectie raamwerk ontwikkeld dat goed geselecteerde bestaande methoden integreert om de hoogste accurheid en robuustheid te bereiken voor het detecteren van prestatieafwijkingen in gedistribueerde toepassingen. Bovendien hebben we een op diepgaand leren gebaseerde ongeassisteerde methode voorgesteld, CGNN-MHSA-AR, die tijds- en kenmerkinformatie benut om uitstekende accurheid en robuustheid te bereiken voor de detectie van afwijkingen in multivariate tijdreeksen.

Om de lokalisatie van de oorzaak van problemen in het prestatiediagnosesysteem te verbeteren, hebben we beperkingen geïdentificeerd in huidige CI-methoden, zoals het alleen extraheren van lineaire causale relaties en het vertrouwen op strikte eisen voor gegevensverdeling. Bovendien richt bestaand onderzoek zich voornamelijk op grofmazige lokalisatie, waarbij alleen defecte services worden geïdentificeerd. Echter, fijnmazige lokalisatie van de oorzaak van problemen, waarbij specifieke metingen op de defecte service worden aangewezen, is nuttiger voor operators. Om deze problemen aan te pakken, hebben we de CausalRCA ontwikkeld, die gewogen causale grafieken genereert met behulp van een op gradiënten gebaseerde methode voor causale structuuranalyse en een methode voor oorzaakinferentie om oorzaakmetingen te bepalen. Bovendien hebben we grof- en fijnmazige experimenten uitgevoerd, en de resultaten toonden aan dat de CausalRCA beter presteert dan basismethoden en fijnmazige, geautomatiseerde en real-time lokalisatie van de oorzaak van problemen biedt.

Publications

Journals:

- **Ruyue Xin**, Hongyun Liu, Peng Chen, and Zhiming Zhao. "Robust and accurate performance anomaly detection and prediction for cloud applications: a novel ensemble learning-based framework." Journal of Cloud Computing 12, no. 1 (2023): 1-16.
- **Ruyue Xin**, Peng Chen, and Zhiming Zhao. "Causalrca: Causal inference based precise fine-grained root cause localization for microservice applications." Journal of Systems and Software 203 (2023): 111724.
- Yujia Songa, **Ruyue Xin**, Peng Chen, Rui Zhang, Juan Chen, and Zhiming Zhao. "Identifying performance anomalies in fluctuating cloud environments: a robust correlative-GNN-based explainable approach." Future Generation Computer Systems (2023). (as co-first author)
- **Ruyue Xin**, Hongyun Liu, Peng Chen, Paola Grosso, and Zhiming Zhao. "FIRED: a fine-grained robust performance diagnosis framework for cloud applications." arXiv preprint arXiv:2209.01970 (2022).
- **Ruyue Xin**, Jingye Wang, Peng Chen, and Zhiming Zhao. "Trustworthy AI-based Performance Diagnosis Systems for Cloud Applications: A Review." ACM computing survey (under revision).
- Chen, Peng, Hongyun Liu, **Ruyue Xin**, Thierry Carval, Jiale Zhao, Yunni Xia, and Zhiming Zhao. "Effectively detecting operational anomalies in large-scale IoT data infrastructures by using a gan-based predictive model." The Computer Journal 65, no. 11 (2022): 2909-2925.
- Zhiming Zhao, Spiros Koulouzis, Riccardo Bianchi, Siamak Farshidi, Zeshun Shi, **Ruyue Xin**, Yuandou Wang et al. "Notebook-as-a-VRE (NaaVRE): From private notebooks to a collaborative cloud virtual research environment." Software: Practice and Experience 52, no. 9 (2022): 1947-1966.

Conferences:

- **Ruyue Xin**, Jardenna Mohazzab, Zeshun Shi, and Zhiming Zhao. "CBProf: Customisable Blockchain-as-a-Service Performance Profiler in Cloud Environments." In Blockchain–ICBC 2021: 4th International Conference, Held as Part of the Services Conference Federation, SCF 2021, Virtual Event, December 10–14, 2021, Proceedings, pp. 131-139. Cham: Springer International Publishing, 2022.

- **Ruyue Xin**, Simon Stallinga, Hongyun Liu, Peng Chen, and Zhiming Zhao. "Provenance-enhanced Root Cause Analysis for Jupyter Notebooks." In 2022 IEEE/ACM 15th International Conference on Utility and Cloud Computing (UCC), pp. 327-333. IEEE, 2022.

- Yujia Songa, **Ruyue Xin**, Rui Zhang, Juan Chen, and Zhiming Zhao. "A Robust and Accurate Multivariate Time Series Anomaly Detection in Fluctuating Cloud-Edge Computing Systems." In 2022 IEEE 24th Int Conf on High Performance Computing Communications; 8th Int Conf on Data Science Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud Big Data Systems Application (HPCC/DSS/SmartCity/DependSys), pp. 357-365. IEEE, 2022. (as co-first author)

- Hongyun Liu, **Ruyue Xin**, Peng Chen, and Zhiming Zhao. "Multi-Objective Robust Workflow Offloading in Edge-to-Cloud Continuum." In 2022 IEEE 15th International Conference on Cloud Computing (CLOUD), pp. 469-478. IEEE, 2022.

- Hoogenkamp, Bram, Siamak Farshidi, **Ruyue Xin**, Zeshun Shi, Peng Chen, and Zhiming Zhao. "A Decentralized Service Control Framework for Decentralized Applications in Cloud Environments." In Service-Oriented and Cloud Computing: 9th IFIP WG 6.12 European Conference, ESOCC 2022, Wittenberg, Germany, March 22–24, 2022, Proceedings, pp. 65-73. Cham: Springer International Publishing, 2022.

Acknowledgments

Four years ago, I could never have imagined the incredible journey that awaited me at the UvA. I still recall the challenges and fresh experiences that accompanied me upon my arrival here. Looking back on the four years, I delighted in experiencing the fantastic cultures of Amsterdam. Little did I know that COVID would steal two precious years of my life. I have encountered both frustration and a sense of achievement in my research pursuits. These diverse experiences have collectively shaped the person I have become today, and I am profoundly thankful for the remarkable individuals I have encountered and the unwavering support I have received along the way.

I would like to first thank my Ph.D. promoter Prof. Cees de Laat. You are always funny, kind, and encouraging. I always got helpful suggestions from the meetings and discussions we have had. I deeply appreciate your caring and help in the past four years. I would like to thank my secondary Ph.D. promoter Dr. Paola Grosso. You are always full of energy, and you care about everyone in our lab. I am very grateful for your valuable comments on my research works and thesis. It has been such a pleasure to work with you for the past four years.

I would also like to thank my Ph.D. co-promoter and supervisor, Dr. Zhiming Zhao. Thanks for your guidance and mentorship throughout this journey. Your expertise and dedication have been instrumental in shaping my research and academic growth. Your feedback and the time you spent reviewing my work have been invaluable. I feel fortunate to have had the opportunity to learn from you, and I am appreciative of your support.

I would like to express my gratitude to Prof. Radu Prodan, Prof. Andy Pimentel, Prof. Rob van Nieuwpoort, Dr. Chrysa Papagianni, and Dr. Zoltan Mann for their kindness to be my Ph.D. defense committee members. I really appreciate the time you have taken to evaluate my thesis and the valuable comments you have made.

A special thanks goes to Peng Chen, who played a vital role in advancing my research. Thanks for your invaluable suggestions, expert guidance, and the

insightful discussions we have had. I deeply appreciate your contributions to my academic journey. I would like to thank my former colleagues Zushun Shi and Hongyun Liu, who gave me enough academic suggestions, help, and encouragements. I would also like to thank my colleges in the QCDIS group: Gabriel Pelouze, Na Li, Spiros Koulouzis, Yi Chen, Yuandou Wang, Zijie Liu.

In addition, I would like to thank some of my friends and colleagues, Jamila Kassem, Lu Zhang, Lu-Chi Liu, Na Li, Sara Shakeri, Yuandou Wang. Lu, Jamilar, and Sara, I truly enjoyed our office time, and I was so happy when we met again after two years working from home. Na and Yuandou, I was glad to have a new office life with you. Lu-chi, I always enjoyed getting together with you and had a good time. Thank you all for making my journey more colorful and interesting.

I am also grateful towards my (both former and current) colleagues in the MNS group and the SNE cluster, including but not limited to Adam Belloum, Ana Oprescu, Cyril Hsu, Dolly Sapra, Florian Speelman, Floris-Jan Willemsen, Garazi Muguruza Lasa, Grace Millerson, Henk Dreuning, Huan Zhou, Imre Fodi, Joseph Hill, Leonardo de Almeida, L. Thomas van Binsbergen, Leonardo Boldrini, Llorenc Escola Farras, Lourens Veen, Marco Brohet, Milen Girma Kebede, Misha Mesarcik, Ning Chen, Poojith Umesh Rao, Ralph Koning, Reggie Cushing, Riccardo Bianchi, Saba Amiri, Saeedeh Baneshi, Siamak Farshidi, Uraz Odyurt, Xiaofeng Liao, Xiaotian Guo, Xin Zhou, Yang Hu, Yixian Shen, Yuri Demchenko. You are all very nice, and it was a great experience working with you.

I would like to express my thanks to my closest friends: Chuanyu Lin, Zhen Jin, and Qiwen Shen. Chuanyu and Zhen, I am grateful for having you sharing our life almost everyday and encourage each other. Qiwen, I feel so fortunate to have you in a foreign land. I also want to thank Penghao Wang. Thanks for your enduring companionship, constant support, and efforts of cheering me up everyday. Last but certainly not least, I want to deeply appreciate my parents, brothers, and grandparents. Your unwavering love, support, and encouragement have been the bedrock of my journey, I could not have completed the journey without all of you. I extend a special thanks to my mom, who dedicated three months to be with me during the thesis writing period, cooking for me, and traveling alongside. Your selfless love is the treasure of my life.