Microservice Logs Analysis Employing AI: A Systematic Literature Review

Md Arfan Uddin[a], Shakthi Weerasinghe[a], Darek Gajewski[a], Melika Akbarsharifi[a], Roxana Akbarsharifi[a], Christopher Stoner[a], Tomas Cerny[a], Sen He[a]

[a]*Department of Electrical and Computer Engineering, University of Arizona, Tucson, Arizona, USA*

Abstract

Background: Microservice architectures generate massive volumes of fragmented log data. While complicating unified monitoring and diagnosis, traditional approaches are typically overwhelmed, leading to delayed incident detection and costly system failures. Artificial Intelligence (AI) techniques, particularly machine learning and large language models, have emerged as promising solutions for automating log analysis and addressing these operational challenges.
Objective: The objective of this study is to systematically review and synthesize existing research on AI techniques for microservice log analysis, evaluating their enterprise deployment readiness and identifying current capabilities, limitations, and priority areas for future investigation.
Method: We conducted a systematic literature review of 2208 papers from peer-reviewed academic literature, published between 2018 and 2025. Through a rigorous filtering process, we identified 82 primary studies that thoroughly examine the use of AI for microservice log analysis across tools, techniques, datasets, and challenges.
Results: While AI techniques are frequently applied to anomaly detection and root cause analysis, this review reveals a critical disconnect between research advances and industry needs. 65 studies rely on synthetic or private datasets that poorly reflect production complexities. Approximately two-thirds (65.85%) of papers adopt standardized evaluation benchmarks, with 59.76% using the standard Precision-Recall-F1 classification metrics. Hence, unresolved scalability, generalizability, and data accessibility remain the key challenges that prevent widespread adoption of AI-based microservice log analysis in enterprise environments.
Conclusion: AI has strong potential for advancing log analysis in microservices. Future research should prioritize open datasets, robust benchmarking, and the use of these AI methods for deeper contextual understanding such as understanding why anomalies propagate across services and how they may impact business operations.

Keywords: Systematic Literature Review, Microservices, Log Analysis, Artificial Intelligence, Machine Learning, Large Language Models

Email addresses: `arfan@arizona.edu` (Md Arfan Uddin), `syweerasinghe@arizona.edu` (Shakthi Weerasinghe), `dgajewski@arizona.edu` (Darek Gajewski), `akbarsharifi@arizona.edu` (Melika Akbarsharifi), `roxanaa@arizona.edu` (Roxana Akbarsharifi), `cstoner@arizona.edu` (Christopher Stoner), `tcerny@arizona.edu` (Tomas Cerny), `senhe@arizona.edu` (Sen He)

1. Introduction

Microservice Architecture has become the predominant paradigm for building enterprise-scale applications, driven by demands for enhanced scalability, resilience, and deployment agility [81]. Unlike monolithic systems, microservice deployments are composed of hundreds and sometimes thousands of, loosely coupled, independently versioned (involving independent development teams) services that span across heterogeneous environments including containers, virtual machines, and cloud platforms.

This architetcural shift has fundamentally transformed operational monitoring: where monolithic applications generate centralized, structurally consistent logs, microservice ecosystems produce distributed log streams characterized by semantic heterogeneity, varying formats, and complex inter-service dependencies. Large enterprise applications routinely produce giga to terabytes of log data daily, characterized by varying levels of structure, such as, semi-structured stack traces and unstructured natural language diagnostics [64]. These logs are essential for ensuring observability, enabling anomaly detection, root cause analysis, and system uptime. However, as enterprise systems grow in complexity, so does the scale and fragmentation of log data. It creates unprecedented challenges for traditional monitoring approaches, necessitating fundamentally new methodologies for effective log analysis and operational intelligence.

1.1. Terminology and Scope

To ensure clarity and consistency throughout this review, we define the following key terminology in Table 1.

Table 1: Key terminology and their definitions used throughout this systematic review.

Term	**Definition**
MSA	Microservice Architecture: A distributed software design pattern that structures applications as collections of independently deployable, loosely coupled services, each responsible for specific business capabilities.

Term	**Definition**
MSE	Microservice Environment: Distributed runtime system encompassing deployed microservices and their supporting infrastructure.
LA	Log Analysis: The process of examining, parsing, and extracting insights from system logs.
RCA	Root Cause Analysis: The systematic process of identifying underlying source(s) of system failures or anomalies.
AD	Anomaly Detection: The process of identifying patterns in data that deviate from normal operational behavior.
FD&EP	Fault Diagnosis and Error Prediction: Techniques used to anticipate and identify failures or abnormal error conditions in enterprise MSEs.
I&DA	Interaction and Dependency Analysis: Methods for identifying and understanding inter-service relationships, communication dependencies, and behavioral correlations among microservices.
AI++	Advanced AI techniques encompassing traditional machine learning (ML), deep learning (DL), graph neural networks (GNNs), transformer-based models, large language models (LLMs), and hybrid approaches.
SLR	Systematic Literature Review: A structured, reproducible methodology for identifying, analyzing, and synthesizing relevant research literature on a specific topic.
Primary Studies	The 82 papers that directly address AI++ techniques for microservice log analysis, selected from 2208 initially reviewed papers.
Generated Datasets	Log data produced by microservices that researchers have utilized in their analysis.
Synthesized Datasets	Publicly available datasets used for benchmarking and validation.

Manual log analysis is no longer feasible in modern MSEs. The volume, velocity, and variability of log data make it difficult for human operators to identify failures, trace dependencies, or respond to incidents in real time. This inefficiency becomes especially critical during outages, where delays in fault localization can result in costly downtime. Traditional monitoring frameworks such as self-adjusting log observability for cloud-native applications [64] and the real-time monitoring and user analytics system [82], and rule-based approaches such as Linnaeus [57] and automating microservices test failure Analysis using Kubernetes cluster logs [74], are inadequate to handle the complexity and scale of contemporary microservice systems

AI++ techniques have emerged as a promising avenue for automated LA in MSEs. These approaches can process heterogeneous and high-volume log data, uncovering patterns, anomalies, and context-rich insights beyond human capability [20, 21]. As AI continues to advance, its application to LA opens new possibilities for predictive diagnostics [15, 10], fault detection [17, 9], and improved software reliability [26]. Yet, the landscape remains fragmented due to inconsistent evaluation practices, lack of standardized benchmarks, and limited availability of representative datasets, hindering systematic assessment and practical adoption.

This SLR consolidates current research on AI++-driven LA in MSEs. Building upon a review of 2208 papers from academic-literature published through December 31, 2024, we identified 82 primary studies that specifically address AI++ techniques for microservice log analysis and observability. Our complete replication package is available on Zenodo[1] for transparency and reproducibility.

1.2. Objective and Contributions

This systematic literature review aims to comprehensively analyze the application of AI++ techniques to address the emerging challenges with the ever-growing enterprise microservice logs that require analysis. It is essential to examine the state-of-the-art AI++ techniques and tools used for microservice LA to avoid reinventing the wheel and to catalyze new technique applications that are enterprise-ready. We identify datasets, evaluation metrics, and benchmarks against enterprise operational requirements, and

[1]`https://zenodo.org/records/17497780`

identify existing limits and open challenges that constrain practical deployment of AI++ LA solutions.

Our contributions present a comprehensive analysis of AI++ techniques in microservice LA, organized around four key contributions:

- **Enterprise-ready AI++ model taxonomy:** A structured classification of machine learning models and tools (open-source and proprietary) assessed against enterprise deployment criteria; we also distinguish between research prototypes and production-ready solutions with detailed assessment of scalability and integration requirements.
- **Production-context methodological analysis:** Comparative analysis of supervised, unsupervised, and hybrid approaches that examines their effectiveness in enterprise MSEs; based on different studies and use cases, we identify the production-reality gap between research results and operational performance.
- **Standardized evaluation gap analysis:** Systematic characterization of datasets, evaluation metrics, and benchmarks used for validation; we reveal critical gaps in standardization.
- **Enterprise adoption readiness assessment:** Evidence-based analysis of advantages, organizational barriers, deployment challenges,and technical limitations in AI++ LA in enterprise MSEs.

These contributions provide a novel systematic analysis into the current landscape of AI++-enabled observability, designed to accelerate enterprise adoption of AI-driven microservice LA, moving beyond technical feasibility to address practical deployment realities. Hence, we outline future research directions for both researchers and practitioners in the field.

1.3. Organization of the Paper

The remainder of this paper is organized as follows: Section 2 discusses related work and motivation. Section 4 discusses the research questions (RQs) that drive this study and outlines our review methodology to address them. Section 5 describes the data synthesis approach. Section 6 through Section 8 presents our findings w.r.t. to each RQ. Section 9 summarizes our findings. Section 10 discusses implications and research gaps. Section 11 addresses threats to validity. Section 12 concludes the paper.

2. Motivation and Related Work

Several literature reviews explored various aspects of LA, microservice LA, and monitoring approaches, and AI++ applications in MSEs. While a body of research exists exploring various aspects of LA and AI applications, this systematic literature review (SLR) is the first to provide a comprehensive, enterprise-focused analysis of the intersection between AI techniques and microservice log analysis from a crucial, often-overlooked, operational deployment perspective.

Existing studies can be broadly categorized into three main areas: (1) general LA methods and practices, (2) microservice-specific LA approaches, and (3) AI++ applications in MSE analysis. We reviewed multiple related literature surveys and their research questions, with detailed comparisons provided in Appendix D to Appendix E.

In thier study of MSE analysis and monitoring, Bushong et al. [1] highlight the challenges posed by the distributed nature of MSEs. Giamattei et al. [2] reviewed monitoring tools for DevOps and microservices, analyzing 71 tools to reveal a highly fragmented landscape with diverse goals and technical constraints that enterprises have to navigate.

He et al. [3] conducted a comprehensive survey on automated LA for reliability engineering, examining logging practices, log parsing, and mining techniques for AD, failure prediction, and diagnosis. Their work identifies open-source toolkits, datasets, and industry best practices while highlighting future research directions. However, their focus remains on traditional automated LA approaches without specifically addressing the unique complexities of distributed MSEs or the emerging AI++ paradigm that has gained prominence since their publication.

While Cândido et al. [4] and Korzeniowski and Goczyła [5] reviewed LA approaches and identified ML techniques ranging from neural networks to clustering algorithms, they focus primarily on "*what techniques*" exist rather than "*how*" they perform in production environments. Hence, the critical production-reality gap—the difference between research results and operational performance, remains critically underexamined.

Similarly, despite having identified 22 LA techniques used for smart troubleshooting in Industry 4.0 focused research, Partovian et al. [6] highlight the potential performance implications, limited versatility, and limited interoperability in existing systems if they are inadequatly adapted into production environments.

Notably, no previous work has systematically characterizes the datasets, evaluation metrics, and benchmarks used across LA in MSEs. This gap is compelling for investigation since Wang and Qi [7] and Zhang et al. [8] suggest that system scale makes AD and RCA increasingly challenging. Yet, without standardized evaluation frameworks, enterprises may not reliably compare approaches or validate the effectiveness of their LA deployments.

Recent studies have only recently started exploring AI++ integration in MSEs. Moreschini et al. [9] investigates AI techniques across the microservice lifecycle, addressing quality attributes and DevOps workflows, while Zhou and Fokaefs [10] examined AI assistants across the incident lifecycle. Akhtar et al. [11] recently surveyed LLM-based event log analysis techniques, examining the developing body of knowledge, commonalities in methods and models, and research gaps in this emerging domain. While their work addresses the latest AI++ developments in log analysis, it focuses on event logs broadly rather than the specific challenges of distributed MSEs. Furthermore, none of these studies provides evidence-based analysis of organizational barriers, deployment challenges, technical limitations, or practical adoption pathways that enterprises demand. Abdelfattah et al. [12]'s multivocal study on microservice dependencies highlighted that effective LA requires understanding inter-service dependencies in large enterprise MSEs. The study identifies key types of dependencies and their implications for LA, observability, and system diagnostics. Yet, it stopped short of evaluating how AI++ approaches address this complexity in practice.

The ML-based AD review by Nassif et al. [13] identified 29 models, 22 datasets, and numerous performance metrics across various domains. While their work offers valuable insights into the broader field of ML-based AD, it does not elaborate on MSEs. Extending these findings, Zhang et al. [8]'s comprehensive work on failure diagnosis in MSEs offers technical insights into the enterprise implementation of these AI-driven approaches. But it focuses narrowly on cascading failures in decentralized environments without broader coverage of LA tasks (e.g., log parsing, AD, and performance monitoring). Similarly, Steidl et al. [14]'s industry-focused study examines runtime AD approaches but does not extend to the full spectrum of LA activities or AI++ model taxonomies.

Overall, Table 2 systematically compares similar surveys against four key dimensions that enterprises require to adopt AI++ techniques for LA in MSEs: enterprise-ready taxonomy (ERT), production-environment analysis (PEA), evaluation standardization (ES), and adoption readiness (AR).

Table 2: Comparison of related surveys against enterprise-critical dimensions.

Survey	**ERT**	**PEA**	**ES**	**AR**
Zhou and Fokaefs [10]	×	N/A	×	-
Bushong et al. [1]	×	×	×	×
Giamattei et al. [2]	×	×	×	-
Korzeniowski and Goczyła [5]	-	×	-	×
Partovian et al. [6]	×	-	×	-
Nassif et al. [13]	-	×	-	×
Zhang et al. [8]	×	-	-	-
Cândido et al. [4]	×	×	×	×
Abdelfattah et al. [12]	×	×	×	-
Moreschini et al. [9]	×	-	×	-
Steidl et al. [14]	×	-	×	-
He et al. [3]	-	-	-	-
Akhtar et al. [11]	×	×	-	×
Soldani and Brogi [15]	×	×	×	-
Our Survey	✓	✓	✓	✓

Key: ✓ = Discussed, - = Partially, × = Not discussed

Most of these studies also predate the widespread adoption of modern AI approaches or explore specific aspects. Accordingly, no previous work has comprehensively addressed all four dimensions simultaneously. Critically, none of them distinguishes between research prototypes and production-ready solutions or systematically analyzes the production-reality gap. Therefore, none comprehensively addresses the intersection of AI++ techniques with microservice LA from an enterprise perspective, which motivates our work. Our study bridges these gaps by examining AI++ tools, evaluation methods, and challenges, specifically in microservice LA, where the distributed nature and complexity of systems make traditional approaches insufficient.

3. Research Questions

To systematically address our research objective, this study is guided by the following research questions (RQs):

RQ1 Which AI++ techniques constitute the state-of-the-art for LA in enterprise MSEs?

RQ2 What datasets, evaluation metrics, and benchmarks are utilized to validate the performance of AI++ techniques in microservice LA, and to what extent do they reflect real-world enterprise complexity?

RQ3 What are the operational benefits, challenges, and risks associated with different AI++ techniques for microservice LA in enterprise production environments?

Each question follows Kitchenham et al. [16] RQ framework of **P**opulation, **I**ntervention, **C**omparison, and **O**utcome (PICO). PICO helps refine research questions by identifying the target population, the intervention being studied, relevant comparisons, and expected outcomes.

RQ1 seeks to analyze the techniques and tools created for LA while assessing their enterprise deployment readiness. If there is a population of tools utilizing AI++ techniques for LA, it is important to determine whether certain algorithms or characteristics are common across these AI++ LA tools in MSEs. Establishing common goals among these tools is also essential for comparison. By identifying common processes and goals across tools, there is potential to reveal what has been attempted, what has worked or failed, and highlight techniques not yet explored for future research.

RQ2 focuses on the collection of data from techniques applied in MSEs and how well they reflect enterprise operational requirements. Each approach uses datasets, evaluation metrics, and benchmarks, and it is necessary to examine how these approaches and their data compare to one another so to determine their limitations. Some approaches may also prove more effective in recognizing key anomalous patterns of a MSE from operational logs and such distinctions must be identified to formalize a resilient, advanced LA method.

RQ3 evaluates the benefits, challenges, and risks of applying AI++ techniques to enterprise-level microservice LA. It examines the balance between improvements (e.g., accuracy, real-time processing) and persistent challenges (e.g., performance constraints, data limitations, implementation complexity) across different AI++ approaches, highlighting gaps that must be addressed to enable more reliable and efficient solutions in the future.

4. Methodology 1: Literature Search

Figure 1 summarizes our research methodology in addressing the RQs. Each stage in this process are discussed in detail in the subsequent sections.

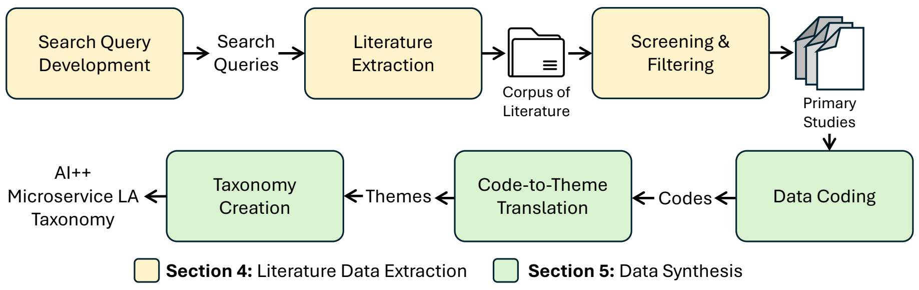

Figure 1: The methodology used in this study.

4.1. Data Sources and Search Strategy

The data sources for this study make use of four digital libraries commonly used for SLRs: Scopus [2], IEEE Xplore [3], ACM Digital Library [4], and SpringerLink [5]. Google Scholar [6] was initially used in the search queries but was later excluded in favor of *SpringerLink* for consistency.

Search strings were constructed through an iterative process documented in Appendix B. The PICO framework guided the general clauses, combined with logical `AND`, and additional aliases identified in the literature were incorporated. The scope of relevant

[2]`https://www.scopus.com/`
[3]`https://ieeexplore.ieee.org/`
[4]`https://dl.acm.org/`
[5]`https://link.springer.com/`
[6]`https://scholar.google.com/`

literature includes studies on *microservices* or *cloud-native* systems that address *LA* tasks using AI++ techniques. Through this incremental process, driven by identified aliases, we derived the following query:

```
( "Microservice" OR "Micro*service*" OR "Cloud*native" OR "Cloud*comput*" )
AND ("Log Analysis" OR "Log Mining" OR "Log Management" OR "Log*"
    OR "Observability" OR "Monitoring" OR "Log Monitoring"
    OR "Trace analysis" OR "Tracing" OR "Telemetry")
AND ( "NLP" OR "Natural Language Processing" OR "Large Language Model"
    OR "LLM" OR "Machine Learning" or "ML"
    OR "Artificial Intelligence" OR "AI")
```

The second and third clauses cover a broad set of aliases identified in the literature as relevant to this topic. For *SpringerLink*, the query was slightly modified to account for differences in wildcard interpretation. Overall, this study focuses on research published from January 2018 till (excluding) January 2025.

4.2. Inclusion and Exclusion Criteria

To narrow the study focus to relevant, high-quality literature, we define a set of inclusion and exclusion criteria that directly address each RQ (Table 3). The exclusion criteria filter out studies with limited relevance or insufficient quality, ensuring the overall reliability of the review. As outlined in Section 4.3, a study must meet the quality assessment criteria to be considered for the full text review discussed in Section 4.4.

Table 3: Inclusion and Exclusion Criteria

ID	**Inclusion Criteria**	**ID**	**Exclusion Criteria**
I1	Microservice log/trace analysis with AI++	E1	SLR, SMS, Review, or Vision papers
I2	Use cases of AI++ tools/techniques for logs	E2	Duplicate entry
I3	Risks or benefits of applying AI++ to Logs	E3	Gray literature
		E4	Outside the scope of Microservices + AI++ logs
		E5	Extended versions of earlier papers
		E6	Entire proceedings volumes
		E7	Non-English papers
		E8	Full text not institutionally accessible

4.3. Quality Assessment

To ensure methodological rigor and reduce selection bias, we employed a structured quality assessment (QA) protocol guided by the following criteria:

- **QA1 Research rigor:** Does the study present a clear methodology, experimental design, and implementation details?
- **QA2 Evaluation adequacy:** Does the study employ appropriate datasets, evaluation metrics, and benchmarks?
- **QA3 Result validity:** Are the findings supported by sufficient evidence and analyzed to yield clear outcomes?
- **QA4 Relevance to practice:** Does the study address practical implementation challenges, scalability, or enterprise deployment barriers?
- **QA5 Reproducibility:** Does the study provide sufficient technical details for validation or replication in enterprise MSEs?

Each study was scored on a 3-point scale (0=Poor, 1=Adequate, 2=Good) for each criterion. Studies that scored more than 5 points `AND` met at least two inclusion criteria without violating any exclusion criteria were considered for review.

4.4. Primary Studies Identification

We conducted a three-round identification process guided by multiple reviewers (Figure 2). At each step, reviewers determined whether a study was suitable for inclusion. These steps progressed from general analysis to increasingly focused evaluation, minimizing unnecessary effort on irrelevant search results. The steps included: *search query execution*, *title and abstract screening*, *inclusion and exclusion criteria filtering*, *full-text review*, and *snowballing.*

The **search query execution** step centralizes all results for subsequent processing. This step is necessary for systematic data gathering and organization before any review can begin. Additionally, it consolidates studies from the different digital libraries.

In **title and abstract screening** (Round 1 of QA), each paper was independently evaluated by two randomly assigned, blinded reviewers who submitted binary decisions (1 for inclusion, 0 for exclusion) based on relevance to microservices and AI++ techniques.

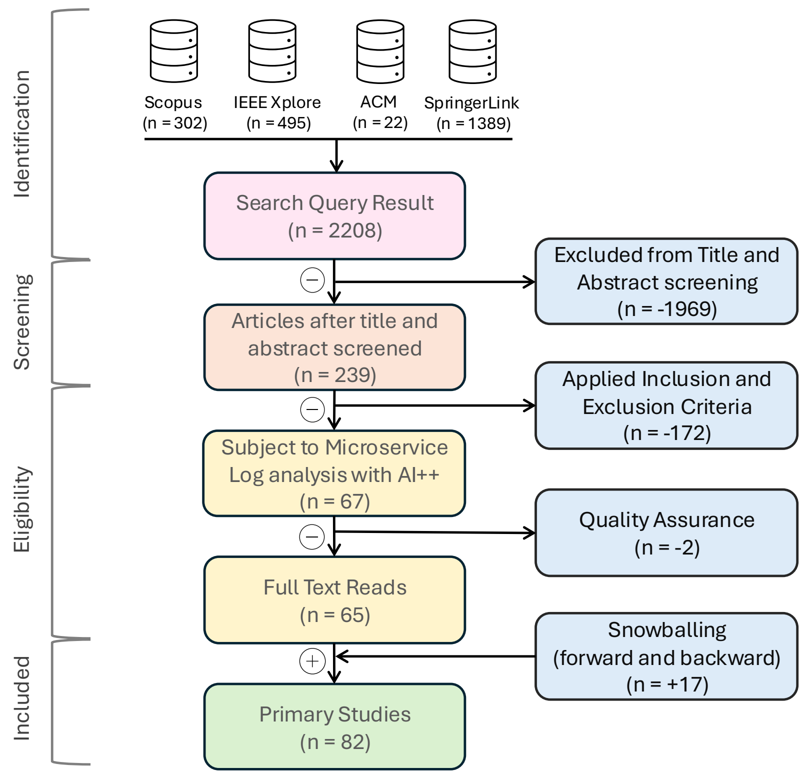
Identification
Screening
Eligibility
Included
Scopus (n = 302)
IEEE Xplore (n = 495)
ACM (n = 22)
SpringerLink (n = 1389)
Search Query Result (n = 2208)
Excluded from Title and Abstract screening (n = -1969)
Articles after title and abstract screened (n = 239)
Applied Inclusion and Exclusion Criteria (n = -172)
Subject to Microservice Log analysis with AI++ (n = 67)
Quality Assurance (n = -2)
Full Text Reads (n = 65)
Snowballing (forward and backward) (n = +17)
Primary Studies (n = 82)

Figure 2: Study Identification Workflow.

Disagreements were resolved by a third reviewer. Reviewers examine only the title and abstract to determine whether a study is relevant to the review topic. This is the most general review step, intended to remove irrelevant results such as literature unrelated to software engineering.

Inclusion and exclusion criteria filtering (Round 2 of QA) applied the predefined criteria from Table 3 alongside the QA. Studies meeting the quality threshold and inclusion criteria proceeded to full-text review. At this stage, notes and data were extracted from studies addressing our RQs, yielding 65 high-quality studies for data extraction.

The following step is the **full-text review**, in which the filtered studies are analyzed in full. Notes and data are extracted from studies that address our RQs. Studies may be excluded at this stage, but none were due to their relevance to our focus areas.

Snowballing (Round 3 of QA) involved bidirectional citation analysis to identify additional literature potentially missed by the initial search queries. *Backward snowballing* considers studies referenced in the primary study bibliography, while *forward snowballing* considers studies that cite a primary study. Litmaps facilitated this process by returning interconnected studies based on the primary study bibliography [17]. Newly identified studies underwent the same evaluation process, adding 17 papers and resulting in a final dataset of 82 primary studies.

Through this rigorous, consistent process, we ensured that the academic literature synthesis using both qualitative and quantitative methods are crediable, and insights about AI++ LA techniques and their applications in entreprise contexts are valid.

5. Methodology 2: Data Synthesis using Thematic Analysis

Our literature search yielded 2208 papers from academic databases (302 Scopus, 495 IEEE Xplore, 22 ACM Digital Library, 1,389 SpringerLink) focusing on MSAs and AI++ applications to logging data from the past decade. The summary is provided in Table 4

Table 4: Results of search and selection.

Activity	**# Papers**
Evidence collection	2208
302 Scopus, 495 IEEE Xplore, 22 ACM DL, and 1389 SpringerLink	
Inclusion and exclusion criteria to title and abstract	239 (-1,969)
Full-text review	65 (-174)
Snowballing	82 (+17)
Quality Assessment	82 (0)
Selected Primary Studies	**82**

Following the systematic screening methodology described in Section 4, we identified 82 primary studies through the 3-round process: title/abstract screening (Round 1 of QA), inclusion/exclusion criteria application (Round 2 of QA), and full-text review with snowballing (Round 3 of QA).

A structured thematic analysis was applied [18] to develop a taxonomy of dependencies, techniques, and artifacts extracted from the selected studies. This process was conducted in three stages:

Stage 1 Data Coding: *Open coding* techniques [19] were applied to analyze AI++ techniques, tools, and other artifacts in enterprise environments. Codes were assigned as concepts emerged, focusing on dependencies centered around similar artifacts, which allowed categories to form naturally from the data.

Stage 2 Translating codes into themes: Using *axial coding* [20], initial codes were systematically grouped into abstract categories addressing the research questions. We connected and refined the relationships between codes to create meaningful clusters. This process identified core themes and clarified connections between AI++ techniques, evaluation approaches, and enterprise deployment factors.

Stage 3 Creating a taxonomy: A formal classification system was developed, organizing AI++ techniques by their enterprise deployment maturity, evaluation frameworks by production relevance, and deployment barriers by their impact on enterprise adoption. Axial coding ensures a consistent structure to this process and highlights connections among dependency types and artifacts. This taxonomy directly supports the enterprise-focused analysis in the next section. The complete dataset is available in Zenodo package[7].

6. RQ1 Results: Current state-of-the-art for AI++ techniques and tools

MSEs generate extensive log data that is inherently distributed, dynamic, and continually evolving, creating significant challenges for analysis. To address these issues, recent research increasingly employs AI++ techniques—encompassing Machine Learning (ML), Deep Learning (DL), Graph Neural Networks (GNNs), transformer-based models, Large Language Models (LLMs), and hybrid approaches. In this section, we analyze 87 distinct AI++ techniques identified in the literature and organize them into matrices based on algorithm type and application domain (e.g., AD, RCA). Our findings highlight a growing emphasis on methods—particularly transformers, LLMs, and GNN-based hybrids—that capture structural and contextual dependencies in log data, reflecting the requirements of enterprise MSEs.

6.1. Taxonomy and Categorization of Techniques

Based on the open and axial coding methodologies described in Section 5, we constructed a multi-dimensional taxonomy that classifies the primary studies along two key

[7]`https://zenodo.org/records/17497780`

dimensions: (1) the algorithmic hierarchy of AI techniques, and (2) their operational viability in enterprise environments. As visualized in Figure 3, the algorithmic dimension is structured into a three-level hierarchy:

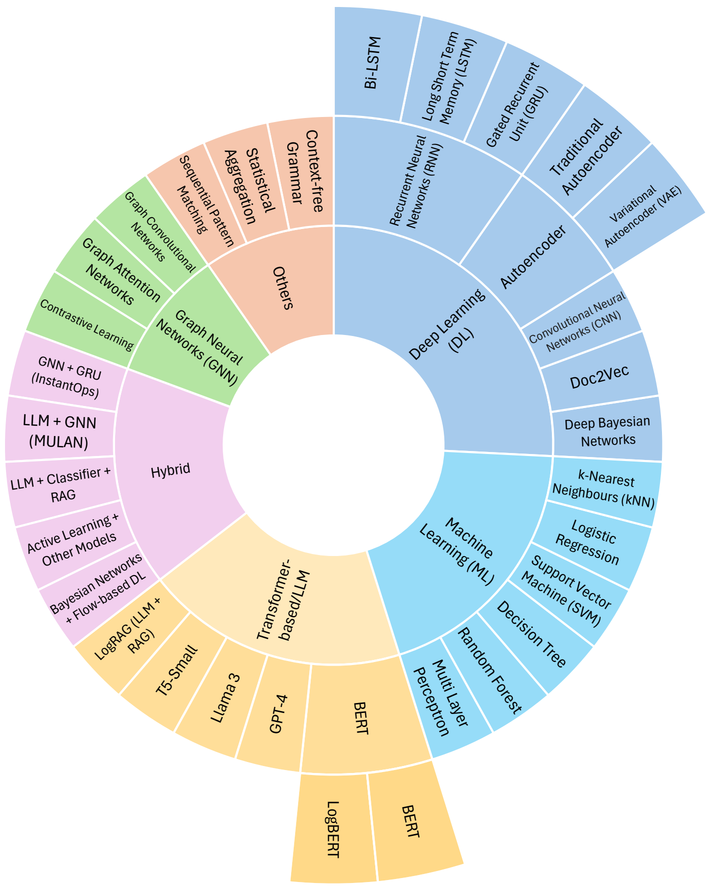

Figure 3: Taxonomy of AI techniques for Microservice LA

1. **Technique:** The six broad AI paradigm (e.g., DL, Transformer-based/LLM).
2. **Technology:** The model family within that paradigm (e.g., RNN, Autoencoders).
3. **Specific Implementation:** The precise algorithms instantiated in the research (e.g., LSTM, LogBERT, Llama 3).

ML techniques remain widely adopted. While not inherently "transparent" in the same way as rule-based systems, they are favored for their established maturity and lower training overhead compared to DNNs. Primary studies utilize these methods for tasks requiring efficient inference including AD, fault prediction, and service dependency analysis. For instance, Zhou et al. [P1] and Aktacs et al. [P2] demonstrate that Random Forests (RF) and Support Vector Machines (SVM) can effectively model latent errors and service interactions without the heavy computational footprint of generative models. These methods are particularly valuable in enterprise scenarios where explainability and fast inference are essential operational requirements.

Evaluations by Dodda et al. [P3] and Darwesh et al. [P4] further reinforce this value, showing that when coupled with robust feature engineering (e.g., discretization, imputation), traditional classifiers (e.g., RF, SVM, K-means etc.) can achieve accuracies around 99.81% in cloud and Kubernetes environments. These findings suggest that for well-structured log data, classical ML remains a highly effective baseline when properly tuned.

DL models address the limitations of traditional ML in handling sequential logs. Common DL models include Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM), which studies such as Khudyakov et al. [P5] and Chen et al. [P6] utilize to model temporal dependencies of MSE behavior. They achieved AD time improvements over statistical approaches.

To address high-dimensionality, researchers employ Autoencoders (AEs) in unsupervised AD. As investigated by Gulmez et al. [P7], these models detect failures by learning to compress and reconstruct normal log patterns, flagging deviations as anomalies. Yet, there a notable trade-off. Multimodal DL framework including of Nedelkoski et al. [P8] denote the superioriority of feature extraction using models such as Convolutional Neural Networks (CNNs), however for a significant computational demands for training.

GNNs are specifically applied to model the graph-structured dependencies between microservices. Techniques such as Graph Convolutional Networks (GCNs) and Graph Attention Networks (GATs) are commonly used to capture these service relationships for tasks such as trace AD and RCA. Unlike sequence-based models, GNNs explicitly map the topology of distributed traces.

Research such as TraceCRL by Zhang et al. [P9] and InstantOps by Rouf et al. [P10], demonstrates that combining GNNs with temporal models (such as GRUs) significantly improves accuracy of AD in large microservice systems by correlating topological structural anomalies with temporal spikes. Further, Zhang et al. [P11] used a hybrid approach integrating GNNs with Positive–Unlabeled (PU) Learning for trace AD on partially labeled datasets, addressing a common challenge in enterprise environments where labeled data is limited.

Recent advances in GNN architectures focus on heterogeneous graph representations for microservice LA. Studies by Nguyen Ba-Hung et al. [P12] and Wang et al. [P13] show that fusing logs, metrics, and traces into a unified graph structure enables finer-grained RCA, allowing the detection of subtle failures that impact end-users without triggering simple threshold alerts.

Transformer-based models and LLMs represent the shift toward semantic-aware LA. By leveraging self-attention, these models capture long-range dependencies that RNNs often miss. Current research applies these models in two ways: self-supervised AD, such as LogBERT [P14], and complex reasoning. The integration of LLMs with structural data is a key trend. Han et al. [P15] introduced a GPT-4-based hybrid system that combines LLM-generated insights with traditional classifiers to improve RCA performance. Ding et al. [P16] demonstrated that aligning graph-level representations with Llama-7B (LLM4MST) allows for few-shot learning, achieving high F1-scores even with minimal training data. This highlights the potential of LLMs to facilitate cross-domain knowledge transfer.

Hybrid techniques are increasingly adopted to resolve the limitations of single-model architectures. Recognizing that no single model captures topology, semantics, and time perfectly, researchers have proposed fusion architectures. Prominent examples include MULAN [P17] and LogRAG [P18], which combine the reasoning capabilities of LLMs with the structural grounding of GNNs. These hybrids consistently outperform single-model baselines in complex RCA tasks, albeit at the cost of increased architectural complexity.

Other methods encompass approaches outside typical AI++ pipelines, including rule-based systems, statistical analysis, and heuristic-driven models. While offering speed, primary studies such as Wijesinghe et al. [P19] suggest that grammar-based AD systems rely on predefined patterns that may struggle to adapt to the dynamic updates of enterprise MSEs. Consequently, the literature positions these methods as effective for stable environments, but less viable for the dynamic discovery required in complex enterprise LA and CI/CD workflows.

To complement the algorithmic dimension shown in Figure 3, our taxonomy incorporates a second dimension that evaluates the operational viability of each AI++ technique category for enterprise MSE LA(Table 5). This dimension addresses practical considerations that are critical for enterprise adoption decisions.

Table 5: Operational viability of AI++ techniques for enterprise LA.

Category	**Microservice-Specific Operational Trade-offs**
ML	(+) Low latency inference latency suitable for high-throughput log streams; interpretable (e.g., trees); low training cost. (−) Heavily dependent on static log parsers/templates; Requires manual feature engineering; ignores semantics.
DL	(+) Captures long-term sequential dependencies (e.g., detecting “slow” failures over time); handles high-dimensional data. (−) Black-box decisions complicate RCA; high training overhead.
Transformers / LLM	(+) Understands unstructured log text directly (no parsing needed); deep semantic understanding; handles unseen logs via few-shot learning. (−) Prohibitive GPU requirements for real-time ingestion of large volumes of logs; latency risks.
GNN	(+) Explicitly maps failure propagation across microservice dependencies (using traces found in logs). (−) Constructing dynamic graphs from traces in real-time introduces significant latency.
Hybrid	(+) Best-in-class RCA by fusing structural data (traces) with semantic data (logs); models structure and semantics. (−) Requires maintaining multiple pipelines (graph + text), increasing “technical debt”; difficult to maintain.
Others	(+) Deterministic behavior is preferred for known, critical alerts; no training required; fast execution. (−) Rigid; fails on unseen patterns in scaling MSEs.

Further, it also reveals that ML techniques dominate the reviewed methods (34.5%), followed by DLs (27.6%), GNN approaches (19.5%), and LLMs (12.6%), with hybrid

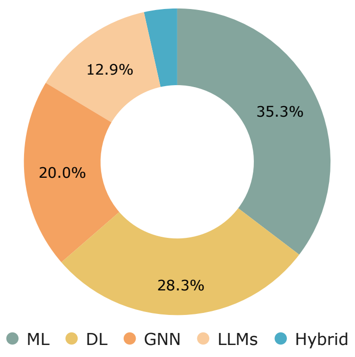

Figure 4: Classification of AI++ methods for LA in MSEs

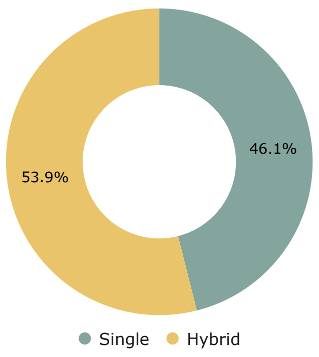

Figure 5: Proportion of single vs. hybrid AI++ approaches for LA in MSEs

approaches accounting for 3.4% (Figure 4). As shown in Figure 5, 46.1% of studies employ single-model architectures for their simplicity and lower computational cost, while 53.9% adopt hybrid methods—such as GNN+LLM, CNN-LSTM, or Knowledge Graph+LSTM—to address the multifaceted challenges of enterprise MSE LA (e.g., the costs utilizing CPU vs GPUs). This trend highlights a shift toward integrating diverse AI paradigms to model service-level dependencies alongside semantic and temporal dynamics. The growing adoption of hybrid methods reflects recognition that microservice complexity demands models capable of capturing structural relationships (via GNNs), temporal patterns (via RNNs/LSTMs), and semantic understanding (via transformers/LLMs) simultaneously.

6.2. Techniques Mapped to Microservice Use Cases

AI++ techniques align with specific microservice LA use cases. By systematically mapping each AI++ category to these practical scenarios, we provide clearer insights into their suitability and effectiveness for enterprise-scale applications. Based on our analysis, we identify five major use cases: AD, FD&EP, RCA, I&DA, and Debugging and Troubleshooting. From *Discussion*, *Threats to Validity*, and *Future Work* sections, we extracted explicitly stated limitations (e.g., "graph construction latency") and synthesized using the open and axial coding methodology described in Section 5 (Stage 3). This process allowed us to aggregate individual constraints into high-level deployment barriers and map them to the dominant use case of each respective study.

Table 6: AI++ Technique Distribution Across Use Cases

Use Case	Studies	Dominant Techniques	Key Challenges
Anomaly Detection	59 (72%)	GNNs, LLMs, Hybrid	Data labeling, false positives
Root Cause Analysis	20 (24%)	ML, Hyrid	Computational complexity
Fault Prediction	3 (4%)	Time series, CNNs	Environment-specific tuning
Dependency Analysis	1 (1%)	Statistical models, ML	Limited generalizability
Other Applications	5 (6%)	Mixed approaches	Diverse requirements

6.2.1. Anomaly and Fault Detection

Anomaly and Fault Detection is the most frequent use case in microservice LA, addressed by 59 studies. These approaches aim to uncover abnormal execution behavior, latent faults, or system failures by analyzing logs, traces, and service dependencies taking tools in Figure 6 to use. Thereby, Table 7 summarizes techniques and tools developed for AD in the primary studies. We organize them based on their methodological strategies and the modalities they operate on.

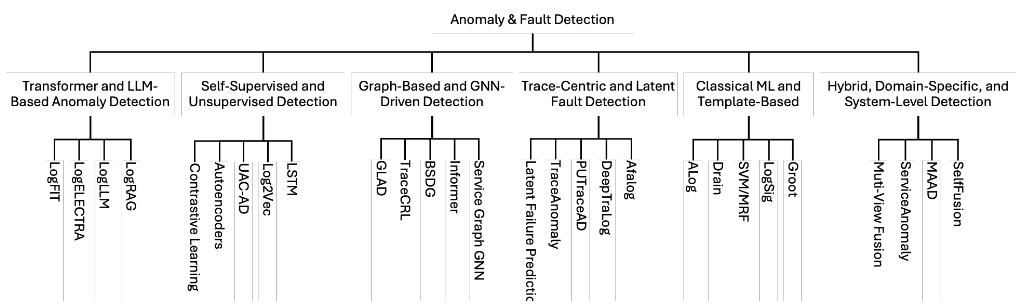
Anomaly & Fault Detection
Transformer and LLM-Based Anomaly Detection
Self-Supervised and Unsupervised Detection
Graph-Based and GNN-Driven Detection
Trace-Centric and Latent Fault Detection
Classical ML and Template-Based
Hybrid, Domain-Specific, and System-Level Detection
LogFIT
LogELECTRA
LogLLM
LogRAG
Contrastive Learning
Autoencoders
UAC-AD
Log2Vec
LSTM
GLAD
TraceCRL
BSDG
Informer
Service Graph GNN
Latent Failure Predictic
TraceAnomaly
PUTraceAD
DeepTraLog
Afalog
ALog
Drain
SVM/MRF
LogSig
Groot
Muti-View Fusion
ServiceAnomaly
MAAD
SelfFusion

Figure 6: AI++ tools and techniques used in anomaly & fault detection.

Table 7: **Techniques and tools developed for AD in MSEs**

Technique Category	**Representative Tools**	**Key Insight**
Sequential & Representation Learning (LSTM-based)	DeepLog[P20], LogAnomaly[P21], LogRobust[P22], PLELog[P23], TICAD[P24], LTTng-LSTM[P25]	Learns sequential log patterns using LSTMs and attention mechanisms; robust to noise and instability
Sequential Learning (CNN-based)	MAAD[P26], Fault Linkage Locator[P27], Event Call Chain RCA[P6]	Uses CNNs for feature extraction from logs and metrics; enables distributed and real-time detection

Continued on next page

Table 7 (continued): Techniques and tools developed for AD in MSEs

Technique Category	**Representative Tools**	**Key Insight**
Statistical & Dimensionality Reduction	LogPCA[P28], LogCluster[P29], Minerva[P30], LongTale[P31]	Applies PCA, clustering, and regression for outlier detection and causal reasoning; interpretable and efficient
Transformer & Pre-trained Language Models	LogBERT[P14], LogFiT[P32], LogLLM[P33], LogELECTRA[P34], LogAttention[P35]	Fine-tunes transformers (BERT, RoBERTa, ELECTRA) for contextual log understanding; high generalization with limited labels
LLM-based Reasoning & RAG	LogRAG[P18], LasRCA[P15], MULAN[P17]	Leverages LLMs (GPT-4, LLaMA) with retrieval-augmented generation for explainable RCA and few-shot anomaly detection
Graph Neural Networks (GNNs)	TraceCRL[P9], PUTraceAD[P11], AdvGraLog[P36], BertHTLG[P37], BSDG[P38], GLAD-PAW[P39], GraphLog-Anomaly[P40], HGT-DGNN[P12]	Uses GNNs to model inter-service dependencies and trace graphs; captures structural and causal relationships
Causal Graph & Dependency Analysis	GROOT[P41], TraceAnomaly[P42], DeepTraLog[P43], MicroScope[P44], InstantOps[P10], DiagFusion[P45]	Constructs causal graphs for RCA; supports multimodal fusion
Temporal & Workflow Modeling	TWLog[P46], TICAD[P24], Informer[P47]	Models execution workflows and temporal semantics; captures RPC traffic patterns
Active Learning & Few-shot	AcLog[P48], AFALog[P49]	Incorporates human feedback for label-efficient anomaly detection
Ensemble & Multi-model Fusion	MEPFL[P1], TraceRCA[P44], CDMF[P50], M3 Framework[P3]	Combines multiple models for robust predictions; fuses trace and metric signals
Hybrid Multi-modal Architectures	InstantOps[P10], MicroHECL[P51], ServiceAnomaly[P52], DiagFusion[P45], DeepTraLog[P43], LasRCA[P15]	Integrates logs, metrics, and traces for end-to-end failure diagnosis
Federated & Privacy-preserving Learning	FedTag[P53], GLAD-PAW[P39]	Enables anomaly detection without raw log sharing
Real-time & Distributed Detection	MAAD[P26], Informer[P47], LongTale[P31], Minerva[P30], MADMM[P54]	Supports low-latency distributed monitoring
Optimization-based Methods	DQN-DBO Monitoring[P55], MADMM[P54]	Uses RL and ADMM for scalable detection
Regression Testing & CI/CD Integration	MRTS-BP[P56], Linnaeus[P57]	Prioritizes regression tests using log trees; CI/CD pipeline integration
Knowledge Graph Integration	Automated KG-LSTM Monitoring[P58]	Uses knowledge graphs for contextual RCA; achieves high accuracy
Dependency & Interaction Modeling	MULAN[P17], RCSF[P44], BSDG[P38]	Models inter-service dependencies and propagation paths

Continued on next page

Table 7 (continued): Techniques and tools developed for AD in MSEs

Technique Category	Representative Tools	Key Insight
Rule-based & Lightweight Alternatives	Linnaeus[P57]	Uses simple heuristics for explainable anomaly detection

Transformer and LLM-Based AD: To overcome the limitations of rigid log templates, recent scholarship increasingly shifts toward transformer architectures and LLMs that learn contextual embeddings directly from raw log sequences. For instance, Almodovar et al. [P32] demonstrate that fine-tuning models such as RoBERTa and Longformer models for log AD (into LogFiT) avoids explicit parsing. As a reuslt, these models adapt effectively to changes in log formats. This trend toward semantic understanding is further advanced by in LogRAG [P33], which integrates LLMs with structural log encodings to enhance robustness in multi-tenant platforms. Beyond AD accuracy, Zhang et al. [P18] leverage retrieval-augmented generation (RAG) to address the interpretability gap, producing contextual explanations of anomalies, while Yamanaka et al. [P34] confirm that pretraining schemes (such as ELECTRA) yield promising results under noisy and evolving log structures.

Self-Supervised and Unsupervised Detection Methods: Given the scarcity of labeled log data, a significant number of research adopts self-supervised or unsupervised techniques to model normal behavior distributions. Bogatinovski et al. [P59] utilize masked span prediction combined with multi-head attention to learn trace dependencies without supervision. Similarly, Liang et al. [P40] and Liu et al. [P42], leverage autoencoders to build representations of normal log sequences, flagging outliers as anomalies.

Addressing the challenge of distinguishing "hard samples" in multi-modal microservice data, Liu et al. [P60] introduced UAC-AD. It automatically identifies them and fine-tunes training weights across log and metric modalities. By combining adversarial learning with contrastive techniques and reconstruction autoencoders, they demonstrated F1-score improvements of 4.5-19.3% over baselines such as DeepLog, LogRobust, and OmniAnomaly. Bridging research and practice, Xu et al. [P61] demonstrated that GAN models integrated into an ELK+Kafka platform can achieve high F1-scores (0.829) in log-based AD, while remaining viable for enterprise deployment. Conversely, for scenarios requiring strictly real-time AD, Khudyakov et al. [P5] argue for sliding window embeddings,. Zhang et al. [P45] and Munir et al. [P35] (using Log2Vec) focused on robustness against varying log structures via sequence modeling and unsupervised semantic embeddings.

Graph-Based and GNN-Driven Detection: Since microservice systems inherently exhibit graph-like structures, researchers increasingly model anomalies as deviations in interaction graphs. He et al. [P36] and Shi et al. [P38] demonstrate that GNNs can effectively capture anomalous communication flows in MSEs. Adding a temporal dimension, Chen et al. [P47] show that diffusion convolutional RNNs (DCRNN) can model irregular RPC traffic spatially and temporally. This structural approach to AD is refined in TraceCRL [P9] using contrastive learning, GLAD [P39] using dynamic attention, and by Kohyarnejadfard et al. [P25] using attention-enhanced node embeddings to isolate anomalous subpaths.

Trace-Centric and Latent Fault Detection: While log messages offer local perspectives, execution traces provide a global view of system behavior. Accordingly, several methods target trace-level AD. Du et al. [P24] (TraceAnomaly) and Duan et al. [P49] (AfaLog) utilize attention mechanisms to align fault propagation patterns across services. Moving from AD to anomaly *prediction*, Zhou et al. [P1] combined Random Forest, KNN, and MLP to forecast latent errors before system failure. DeepTraLog [P43] fuses convolutional and recurrent layers to detect structural anomalies in trace graphs. To address class imbalance, Zhang et al. [P11] using PUTraceAD, prove that positive-unlabeled learning significantly improves the recall of rare but critical anomalies.

Classical ML and Template-Based Methods: Despite the rise of DL, classical ML models remain relevant due to their efficiency. This pipeline fundamentally depends on log parsing, advanced by robust algorithms such as LogSig [P41] and Drain [P60, P26] whose fixed-depth tree structure made it highly efficient and widely adopted. Once parsed, classical classifiers has been sufficient; Streiffer et al. [P30] utilized SVMs on engineered features, while Sheluhin et al. [P62] used tree-based models for multi-class distinction of different anomaly types. More recently, Lu et al. [P63] combined parsing with Bi-LSTMs to achieve 95% precision in power grids, and Chen et al. [P6] using Drain3 demonstrated that unifying trace and log data into event chains improves root cause localization by upto 20%.

Hybrid, Domain-Specific, and System-Level Detection: Practical realities in enterprise MSEs demand hybrid models that integrate domain knowledge and system-specific design to enhance AD. Hence, Panahandeh et al. [P52] and Nedelkoski et al. [P8] (combining LSTM-based log sequence analysis with Isolation Forests) argue for multi-view fusion (logs + metrics) to diagnose service-specific issues in complex ecosystems. To handle concept drift, Tan et al. [P26] propose adaptive frameworks, while Aktacs et al. [P2] combine tail-based sampling with rules for service-specific faults isolation efficiency. Addressing data volume, Pathak et al. [P64] designed a system reducing log volume by 90% without sacrificing accuracy, validating early findings by Xu et al. [P28] that sequential heuristics can be both lightweight and effective.

Benchmarking, Meta-Learning, and System Evaluation: Finally, the maturity of the field is evidenced by a focus on benchmarking and reproducibility of AD systems. D'Angelo et al. [P65] and Nedelkoski et al. [P66] contributed grammar-based synthesis of structured log anomalies and fault injection platforms for controlled benchmarking of AD systems under realistic microservice workloads, respectively. Simultaneously, Hadadi et al. [P67] and Nobre et al. [P68] provide comparative analyses that standardize performance metrics across AD system architectures.

6.2.2. Root Cause Analysis (RCA)

RCA plays a crucial role in identifying the underlying causes of anomalies or failures in MSAs. Unlike AD, which flags deviations, RCA must pinpoint the *source*, [P20] facilitating remediation. A total of 20 studies fully (a further 14 partially with other use cases) focused on RCA. This section categorizes these approaches into 5 themes based on the underlying methodologies used (Figure 7).

Graph-Based RCA and Service Dependency Modeling: Graph structures are the dominant paradigm for RCA since they directly map interactions between services. It enables isolating root causes by analyzing service dependencies and behavior propagation. Brandon et al. [P69] and Wang et al. [P41] demonstrate that representing interactions as dynamic graphs allows for influence scoring to rank candidate services. While effective for indirect dependencies, this method relies on predefined patterns. To mitigate reliance on perfect data, Zhang et al. [P45, P22] incorporate robust learning methods to trace failures through cascaded component interactions, even with noisy or incomplete logs.

Transformer and LLM-Based RCA: Representing a shift toward semantic infer-

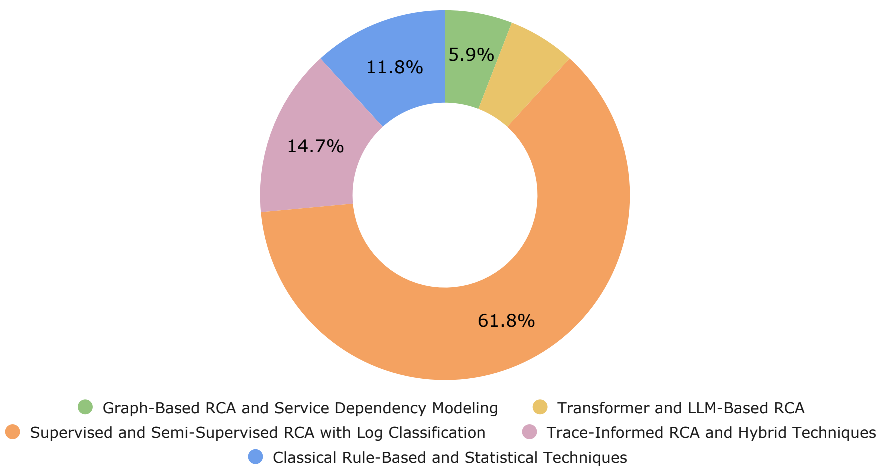

Figure 7: The distribution of RCA techniques.

ence, Han et al. [P15] show that one-shot RCA using GPT-4 can identify fault sources with minimal supervision, dramatically reducing labeling effort. Similarly, Zheng et al. [P17] and Zhu et al. [P70] confirm that LLM-enhanced LA outperform statistical baselines in log summarization and triage. However, these benefits trade off against inference cost and latency concerns from LLM APIs.

Supervised and Semi-Supervised RCA: Traditional supervised learning pipelines or semi-supervised frameworks classify log events and predict causal services. Catovic et al. [P57] emphasize modularity with Linnaeus, a reusable classification pipeline that combines Bag-of-Words, TF-IDF, and multiple classifiers to map logs to root causes. Streiffer et al. [P30] employ classical SVM and Random Forests. Addressing the labeling bottleneck, Li et al. [P71, P44] propose semi-supervised learning for RCA via dynamic windows and instance clustering, enhancing precision in operational settings without requiring fully curated datasets. Building on this foundation, Li et al. [P31] demonstrates detecting hidden root causes involving a long-tail fault analysis model.

Trace-Informed RCA and Hybrid Techniques: Combining traces with logs provides the necessary execution context for fine-grained RCA. Tang et al. [P50] and Liu et al. [P51] argue for correlating anomaly symptoms using graph clustering (across traces) to isolate cause-effect chains. Taking a multi-modal approach, InstantOps [P10] integrates logs, traces, and metrics via a GNN-GRU, suggesting that holistic data fusion reduces diagnostic latency, despite the higher computational training cost.

Classical Rule-Based and Statistical Techniques: Foundational statistical methods laid the groundwork for modern RCA. Early PCA-based approaches [P28] and LSTM-based sequence modeling such as DeepLog [P20] established that deviations in event sequences signal anomalies. While primarily developed for AD, it has also been applied to RCA by correlating anomalies with their sequential context. Subsequent works by Toka et al. [P72] and Wang et al. [P73] refined these into lightweight ranking systems that prioritize suspicious components based on log emission patterns.

6.2.3. Error Detection and Faliure Prediction (ED&FP)

Distinct from reactive ED, FP emphasizes predicting service failures in advance, enabling proactive interventions before anomalies escalate into outages. But, the related literature highlights a trade-off between accuracy and data granularity in this domain. Gulmez et al. [P7] achieved a 99.41% true positive rate with CNNs, but it required structured logs and platform-specific tuning. Alternatively, Toka et al. [P72] leveraged time-series clustering with Dynamic Time Warping and Time-Lagged Cross Correlation to uncover pre-failure patterns, which is effective for high-frequency data but sensitive to irregular sampling. Eventhough Rouf et al. [P10] attempted to bridge this gap with a GNN-GRU hybrid model, it incurs significant computational overhead.

6.2.4. Debugging and Dependency Understanding

Debugging and Dependency Understanding focuses on how microservices interact, which is essential for effective debugging, root cause isolation, and overall system observability. Hence, effective debugging requires understanding causal dependencies among components in a MSA. Aktacs et al. [P2] addressed this by combining interaction prediction with traditional ML (logistic regression, SVMs, and decision trees) to surface hidden dependencies. It aids developers with tracing request flows and identifying potential failures. However, the need for platform-specific training may limit its generalizability.

Taking a formal modeling approach, Wang et al. [P73] argued for the use of Granger causality (both linear and neural) to construct evolving dependency graphs. By integrating Vector Auto Regression (VAR) and Bayesian Linear Regression (BLR), they constructed evolving dependency graphs that reveal both direct and indirect influences.

While this method offers a high-resolution view of service interactions, its effectiveness depends on high-frequency data and precise parameter tuning. Together, these studies highlight that combining statistical modeling with log mining offers a higher resolution view of service interactions than standard tracing alone.

6.2.5. Other Emerging Use Cases

Beyond standard fault management, a small subset of studies repurposes AI++ LA for distinct security and operational goals.

Security and Forensics: Unlike traditional AD which targets performance faults, Chen et al. [P53] addresses data privacy in cross-organizational monitoring. They proposed *FedTag*, a federated learning framework that allows organizations to train collaborative diagnostic models without sharing sensitive raw logs–a critical capability for regulated enterprise environments.

Tooling and Automation: Moving beyond runtime monitoring to development support, Sarika et al. [P74] utilized log mining to build a configuration recommender system that suggests parameter tuning for microservices. Similarly, Wahono et al. [P75] developed a specialized debugging tool that employs keyword matching to accelerate the brute force debugging process practitioners often face when navigating large, distributed log files.

In summary, the diverse range of use cases and techniques explored spans from foundational tasks such as AD and RCA to more advanced objectives like failure forecasting, debugging, and system-level optimization. These studies reflect a growing convergence of traditional ML, DL, graph reasoning, and LLM to address the operational and analytical demands of modern distributed systems. Many of these contributions have materialized as practical tools and platforms that support scalable, intelligent LA in production settings. The following section examines these tools in detail.

6.3. Tools and Implementation

The reviewed literature has produced 61 tools implementing AI++ techniques for microservice LA. Unlike the theoretical models discussed in Section 6.1, these artifacts represent concrete implementations designed for deployment. Table 8 provide comprehensive details on these tools, including their underlying techniques, primary use cases, key innovations, and open-source availability. We categorize them into four by their deployment strategies and implementation focus.

Table 8: AI-based Tools for Microservice LA

Tool	**Technique**	**Use Case**	**Key Insight**	**Open Source**	**Ref**
AcLog	DL – LSTM + Active Learning	AD	Integrates human knowledge into AD for few-shot learning	Yes[8]	[P48]
AFALog	Active Learning + Transformer/LSTM	AD	Mitigates class imbalance with active feedback-based learning strategies	Yes[9]	[P49]
AdvGraLog	GNN + GAN	AD	Combines GNNs and adversarial training to improve robustness to noise	No	[P36]
BertHTLG	GNNs – Sentence-BERT + RGCN	AD	Combines Sentence-BERT for semantic feature extraction with structural learning	No	[P37]
Bi-LSTM-ATT	Bi-LSTM	AD	Combines Bi-LSTM networks with an attention mechanism, capturing bidirectional sequential dependencies in logs.	No	[P63]
BSDG	DualGCN + Bi-level Optimization	AD	Uses dualGCN with mutual attention to integrate intra/inter-service patterns	No	[P38]
CDMF	RF + FP-Growth + Hierarchical Clustering	RCA	Identifies fault correlations with fine-grained cause grouping	No	[P50]
DeepLog	LSTM Neural Networks	AD	Learns sequential log patterns to detect abnormal events	Yes[10]	[P20, P36, P35]
DeepTraLog	GGNN + DeepSVDD	AD, I&DA	Models spatial-temporal trace event graphs with DL-based scoring	Yes[11]	[P43, P40]
DGNN-FD	Deep Graph Neural Networks (DGNN)	AD, FD&EP	Uses DGNN to capture detect hidden service faliures.	No	[P12]
DiagFusion	GNN + FastText + Fusion	RCA, I&DA	Multi-source observability fusion with neural graph modeling	Yes[12]	[P45]

Continued on next page

[8]https://github.com/Dcm-dcm-dcm/AcLog
[9]https://github.com/Dcm-dcm-dcm/AFALog
[10]https://github.com/wuyifan18/DeepLog
[11]https://github.com/FudanSELab/DeepTraLog
[12]https://anonymous.4open.science/r/DiagFusion-AD52

Table 8 (continued): AI-based Tools for Microservice LA

Tool	Technique	Use Case	Key Insight	Open Source	Ref
DQN-DBO Monitoring	Hybrid – Deep Q Network + Dung Beetle Optimization	FD&EP, RCA, AD, Uptime Management	Optimizes DQN hyperparameters to avoid sub-optimal solutions; improves adaptability	No	[P55]
Event Call Chain RCA	DL – LSTM + Drain3 Parser	RCA	Unifies trace and log events via call chains; 18–20% accuracy gain, 0.10–0.18s faster	No	[P6]
Fault Linkage Locator	Hybrid – CNN-LSTM + Enhanced Random Forest	RCA, AD, System Stability	Detects fault linkage sets using time-based correlation analysis on metric sequences	No	[P27]
FedTag	ML – Federated Learning + BERT + One-class Classifier	AD	Protects log privacy using federated training while preserving performance	No	[P53]
GLAD-PAW	GNN + Position Aware Weighting	AD	Uses weighted GNN on session graphs for microservice failure detection	No	[P39, P36]
GraphLog-Anomaly	GNNs – GIN + Global Attention Mechanism	AD	Constructs log event graphs from microservice logs and uses GNNs to detect anomalies	No	[P40]
GROOT	Causality Graphs + PageRank	RCA	Real-time root cause graph propagation based on event causality	No	[P41]
Informer	Hybrid – DBSCAN + DCRNN	AD	Utilizes RPC chain pattern mining with DBSCAN clustering and RNN modeling	No	[P47]
InstantOps	GNN + GRU + Rule Matching	RCA	Combines real-time signals with causal graphs for ops automation	No	[P10]
LasRCA	GPT-4 + Prompt Tuning	AD	Uses in-context LLM reasoning to explain and classify log anomalies	Yes[13]	[P15]
LDMG	Graph-Based DL + Time-aware LSTM + Log-Metric Fusion	AD	Fuses logs and metrics into a service dependency graph, refined by a time-aware LSTM.	No	[P76]

Continued on next page

[13]https://github.com/AmanecerTrio/LasRCA

Table 8 (continued): AI-based Tools for Microservice LA

Tool	**Technique**	**Use Case**	**Key Insight**	**Open Source**	**Ref**
Linnaeus	Naive Bayes + Supervised Classifier	AD	Lightweight RCA tool designed for portability and reuse across systems	No	[P57]
LLM4MST	LLM + GNN + Service Invocation Graphs	AD, RCA	Aligns service dependency representations with a fine-tuned LLM effectively detect anomalies and localize root causes.	No	[P16]
LogAnomaly	LSTM + Template2Vec + Word2Vec	AD	Template2Vec captures structure and semantics of logs for better learning	No	[P21, P35, P40]
LogAttention	Transformer-based / LLMs – Transformer + Ensemble of Weak Classifiers	AD	Combines heuristic weak classification, unsupervised training, ensemble voting	No	[P35]
LogBERT	BERT Language Model	AD	Applies pre-trained BERT to learn contextual log representations	Yes[14]	[P14]
LogCluster	Log clustering + TF-IDF	AD	Cluster-based LA for reducing alert duplication and noise	No	[P29, P36]
LogELECTRA	ELECTRA + Language Modeling	AD	Employs self-supervised transformer architecture with minimal labels	No	[P34]
LogFiT	RoBERTa + Fine-tuning	AD	Adapts pre-trained transformer models to microservice log formats	No	[P32]
LogLLM	BERT + LLaMA Transformer	AD	Leverages hybrid transformer encoders for contextualized AD	Yes[15]	[P33]
LogPCA	PCA + Feature Extraction	AD	Applies statistical modeling over parsed logs using PCA	No	[P28, P36]
LogRAG	Hybrid – RAG + DeepSVDD	AD	Combines semi-supervised learning and RAG to handle unseen anomaly types	No	[P18]
LogRobust	Attention Bi-LSTM	AD	Handles log instability using attention-based bidirectional LSTM models	No	[P22, P36, P35]
LongTale	ElasticNet + LASSO + Ridge	AD, RCA	Correlates application-level traces and system-level signals for RCA	No	[P31]

Continued on next page

[14] `https://github.com/HelenGuohx/logbert`

[15] `https://anonymous.4open.science/r/LogLLM-B7EA`

Table 8 (continued): AI-based Tools for Microservice LA

Tool	**Technique**	**Use Case**	**Key Insight**	**Open Source**	**Ref**
LTTng - LSTM anomaly detection	DL – LSTM Neural Networks	AD, Debugging and Troubleshooting	Uses LTTng for distributed tracing, applies NLP models for event analysis	Yes[16]	[P25]
MAAD	DL – CNN + LSTM + FCN (multi-agent)	AD	Distributed multi-agent architecture enables lightweight AD	Yes[17]	[P26]
MADMM	GCN + GAT + ADMM Optimization	AD	Merges graph and temporal models for scalable distributed AD	No	[P54]
M3 Monitoring Framework	ML – SVM, K-means, Random Forest	AD, Reliability Enhancement	Comprehensive ML comparison; SVM achieves 99.81% accuracy on cloud microservices	No	[P3]
MEPFL	RF + KNN + Multi-model Fusion	AD	Ensemble-based detection using combined trace-level and metric-based signals	Yes[18]	[P1, P44]
MicroHECL	OC-SVM, Random Forest + Metrics	AD, RCA	Extracts service-level metrics and log patterns to localize root causes	No	[P51]
MicroScope	DAG + Dependency Analysis	Debugging and Troubleshooting	Graph-based microservice dependency analysis for debugging flow	No	[P44]
Minerva	k-Shape + Granger Causality	AD, RCA	Combines temporal correlation and clustering for RCA and visualization	No	[P30]
MRCA	Multi-modal data fusion + causal graph construction	RCA	Fuses multi-modal data for root cause localization, while a reward-based causal graph pruning strategy improves analysis speed.	No	[P13]
MRTS-BP	Belief Propagation + Pattern Mining	AD	Uses microservice transaction sequences and BP for anomaly scoring	No	[P56]
MULAN	Log-tailored LLM + Contrastive Learning	RCA	Multi-modal fusion of logs, metrics, and traces for RCA	No	[P17]
PLELog	Attention GRU + HD-CNN	AD	Integrates label estimation and classification using hierarchical models	Yes[19]	[P23]

Continued on next page

[16]`https://github.com/kohyar/LTTng_LSTM_Anomaly_Detection`
[17]`https://github.com/jzxycsjzy/multi_agent_anomaly_detection`
[18]`https://fudanselab.github.io/Research/MEPFL`
[19]`https://github.com/LeonYang95/PLELog`

Table 8 (continued): AI-based Tools for Microservice LA

Tool	**Technique**	**Use Case**	**Key Insight**	**Open Source**	**Ref**
PUTraceAD	GNNs – GNN + Positive-Unlabeled Learning	AD	Utilizes span causal graphs for trace representation with PU learning	Yes[20]	[P11]
RCSF	Sequential Pattern Mining	RCA	Applies frequent pattern mining over trace trees for RCA	No	[P44]
ServiceAnomaly	Annotated CPG + Linear Classifier	AD, I&DA	Constructs causal context graphs to detect service anomalies	Yes[21]	[P52]
TICAD	Bi-LSTM + Template Matching	AD, RCA	Leverages context-aware template grouping and attention for RCA	No	[P24]
TraceAnomaly	Deep Bayesian Network + Posterior Flow	AD, I&DA	Uses posterior attention flow to unify invocation path and response time	Yes[22]	[P42, P44]
TraceCRL	GNNs – GNNs + Contrastive Learning	AD, I&DA	Preserves graph structure in traces using operation-level embeddings	Yes[23]	[P9]
TraceRCA	Forest Ensemble + Multi-metric Learning	AD, RCA	Combines multiple anomaly detectors and metrics in service call graphs	Yes[24]	[P44]
TWLog	Transformer-based / LLMs – Transformer + Trace Relation Graphs	AD	Constructs trace relation graphs to capture task-level relationships	No	[P46]
UAC-AD	Transformer-based / LLMs – Transformer + CNN + GAN	AD	Combines adversarial training to identify difficult AD patterns with contrastive learning.	Yes[25]	[P60]
Unnamed	RNN	FD&EP	Analyzes historical trace logs and monitoring metrics for the accurate forecasting of faults.	No	[P77]
Unnamed	DL + ELK Stack	AD	Integrates DL with ELK+Kafka for scalable AD.	No	[P61]

Continued on next page

[20]`https://github.com/PUTraceAD/PUTraceAD`

[21]`https://github.com/M-panahandeh/ServiceAnomaly`

[22]`https://github.com/NetManAIOps/TraceAnomaly`

[23]`https://fudanselab.github.io/TraceCRL/`

[24]`https://github.com/NetManAIOps/TraceAnomaly`

[25]`https://github.com/lhysgithub/UAC-AD`

Table 8 (continued): AI-based Tools for Microservice LA

Tool	Technique	Use Case	Key Insight	Open Source	Ref
Unnamed	Random Forest + AdaBoost	AD	Model trained to detect DoS attacks and benign workload overloads in Kubernetes.	No	[P4]
Unnamed	Knowledge Graphs + NLP + LSTM	AD, FD&EP	Uses a knowledge graph to structure real-time; NLP and LSTM models understand complex service relationships.	No	[P58]
Unnamed	GNN + ML + Latency distribution prediction	AD	Predicts the full latency distribution using GNNs, rather than a single average value.	No	[P78]

6.3.1. Centralized LA Frameworks

The most common implementation pattern involves centralized engines that ingest aggregated logs for offline or batch processing.

Sequence-based Tooling: Foundational tools such as DeepLog [P20] and LogAnomaly [P21] implement LSTM-based architectures as standalone Python modules (using PyTorch or TensorFlow). These tools typically sit downstream from log aggregators (e.g., ELK,Fluentd, Fluent Bit), parsing raw logs to learn sequential patterns. While effective, their reliance on static templates demands the development of LogCluster [P29], which implements clustering to reduce alert noise. Further, LogRobust [P22] adds contrastive pre-training to handle unstable log data streams.

Transformer and LLM-based Tooling: To overcome parsing fragility, recent tools integrate LLMs. LogBERT [P14] and LogFiT [P32] adapt pre-trained Transformers (i.e., BERT/RoBERTa) to learn log semantics directly. Implementation-wise, these require significant GPU resources, prompting optimized versions such as LogELECTRA [P34], which re-engineers the ELECTRA framework for more efficient masked discrimination. LogLLM [P33] and LasRCA [P15] represent a contemporary approach, employing GPT-4 and LLaMA adapters to generate human-readable incident reports. They bridge the gap between machine-level events and operator dashboards.

6.3.2. Graph-Based and Multi-Modal Pipelines

Addressing the complexity of microservice dependencies, a subset of tools implements multi-modal ingestion pipelines that fuse logs, metrics, and distributed traces.

Trace-Log Fusion: TraceCRL [P9] and DeepTraLog [P43] require rigid data pipelines to construct dynamic service graphs. DiagFusion [P45] and MULAN [P17] advance this by implementing multi-modal encoders that process text (logs) and topology (traces) concurrently. While offering high fidelity (e.g., HGT-DGNN [P12] achieves 88.9% accuracy in hidden failure detection), they impose a high "instrumentation tax"—hidden costs and inefficiencies that arise from inadequate or poor-quality software instrumentation, requiring target systems to comform to logging standards such as OpenTelemetry[26].

Causal Discovery Engines: For RCA, tools such as GROOT [P41] and MicroScope [P44] implement real-time graph construction to visualize failure propagation. InstantOps [P10] extends this architecture by integrating a remediation module, closing the loop between GNN-based detection and automated recovery actions. Similarly, TICAD [P24] and RCSF [P44] implement causal inference over trace trees to identify cascading bottlenecks.

6.3.3. Distributed and Privacy-Aware Architectures

To address the latency and privacy bottlenecks of centralized analysis, several tools propose distributed implementation patterns.

Edge/Agent-Based Processing: MAAD (Multi-Agent Anomaly Detection) [P26] distributes inference workloads across multiple agents residing on individual microservice nodes. The *AI at the edge* design minimizes network overhead and enables real-time detection in large clusters. LTTng-LSTM [P25] adopts a similar low-level approach, integrating directly with the LTTng kernel tracer for highly efficient, low-intrusion monitoring.

Federated & Secure Learning: Regulatory constraints in enterprise environments have driven the development of privacy-preserving tools. FedTag [P53] implements a federated learning architecture where local BERT models are trained on-site, sharing only weight updates rather than raw logs. GLAD-PAW [P39] similarly uses privacy-aware graph learning, ensuring that sensitive service interaction data remains localized. This design specifically addresses enterprise privacy requirements (i.e., GDPR and HIPAA), allowing organizations to utilize AI++ LA without exposing sensitive raw logs to a central

[26]`https://opentelemetry.io/docs/specs/otlp/`

server or third-party cloud.

6.3.4. DevOps and CI/CD Integration

Finally, a distinct category of tools integrates observability directly into the software development lifecycle (SDLC), shifting analysis "left".

Testing and Deployment: MRTS-BP [P56] connects to CI/CD pipelines to analyze API gateway logs, prioritizing regression tests based on recent changes. MADMM [P54] focuses on the deployment phase, monitoring transitional states to detect faults before full rollout.

Active Learning and Feedback: To handle the scarcity of labeled data in production, tools including AcLog [P48] and AFALog [P49] implement "Human-in-the-Loop" interfaces. These tools query operators only for uncertain anomalies, using active learning strategies to progressively refine the model with minimal manual effort. In contrast, LogCluster applies unsupervised sequence clustering to enable AD in environments where labeled logs are limited [P29].

Overall, however, approximately 35% of the tools (e.g., DeepLog, LogBERT, and TraceAnomaly) are open-source. They leverage ecosystem libraries such as HuggingFace Transformers[27] and PyTorch Geometric[28]. Therefore, the level of productization varies significantly. While tools such as DeepLog provide CLI interfaces and config-driven setups, recent implementations (e.g., LogLLM) are proof-of-concepts potentially requiring refactoring to be enterprise-ready.

6.4. Emerging Trends and Insights

A substantial majority of the reviewed studies concentrate on AD, which accounts for 67.8% of the papers (Figure 8). This trend reflects the critical role AD plays in maintaining the stability of microservice-based systems. RCA follows at 23%, indicating an increasing interest in diagnosing and localizing faults. Less frequently addressed are fault diagnosis and error prediction (2.3%) and interaction and dependency analysis (1.1%), with 5.7% of the papers covering use cases that fall outside these primary categories.

[27]`https://huggingface.co/docs/transformers/en/index`

[28]`https://pytorch-geometric.readthedocs.io/en/latest/`

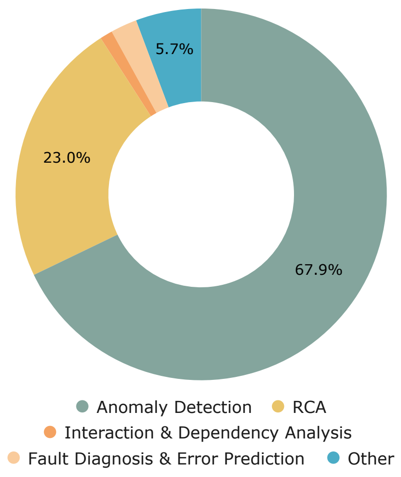

Figure 8: Distribution of analysis types.

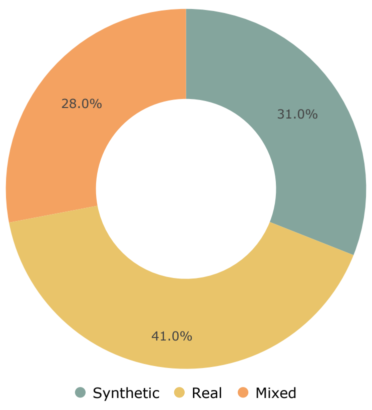

Figure 9: Distribution of papers by datasets.

Traditional ML remains most prevalent (34%), valued for its interpretability and deployment efficiency. However, advanced techniques are gaining ground: DL (28%) for temporal patterns, GNNs (20%) for structural modeling, and LLMs (13%) for semantic understanding. Hybrid approaches. i.e., LLM+other types (3%) remain relatively rare but represent the frontier of research, combining complementary strengths—for instance, GNNs for service topology with LLMs for log semantics.

The single-model versus hybrid distribution 46.1% vs. 53.9%) reflects a field in transition. While single-model approaches dominate due to their simplicity and lower resource requirements, hybrid methods are rapidly emerging as researchers recognize that microservice complexity demands multi-faceted modeling. The convergence toward hybrid architectures suggests future tools will increasingly integrate structural, temporal, and semantic reasoning.

Looking forward, several directions emerge as promising: (1) improved data efficiency through self-supervised and few-shot learning to address labeling scarcity, (2) enhanced interpretability to bridge the gap between model predictions and operator understanding, (3) better generalization across diverse MSE deployments to reduce platform-specific tuning, (4) tighter integration of logs, traces, and metrics for holistic observability, and (5) more sophisticated handling of system evolution and concept drift in long-running production environments. The next section examines the datasets, evaluation metrics, and benchmarks that support this ongoing evolution.

7. RQ2 Results: Datasets, Evaluation Metrics and Benchmarks

This section analyzes the experimental rigor of primary studies. We examine three dimensions: (1) dataset provenance, (2) evaluation metrics and benchmarks, and (3) availability of software artifacts.

7.1. Dataset Taxonomy and Distribution

Datasets fall into two main themes, *Generated* datasets and *Synthesized* datasets. Generated datasets come directly from microservices the original authors have access to, and generate logs using a variety of different public or commercial/private services. Synthesized datasets are publicly available and are discussed in detail in the next sections.

Using these themes from the taxonomy, papers were grouped into three categeories:

- **Fully synthetic lab-generated** (`Generated.Lab`) datasets in a controlled, non production environment, often using open-source microservices; they were used in 31% of primary studies,
- **Industry-generated** (real-world) (`Generated.Real`) datasets sourced from live, commercial, or production systems; they were used in 41% of primary studies, and
- **Publically available** synthetic (`Synthesized.Public`) datasets were used in 18% of the primary studies; these datasets typically originate from open-source projects, such as Hadoop's public HDFS logs, or are generated by researchers using open-source microservices such as TrainTicket.

Notably, 28% of studies used mixed datasets - typically injecting sythentic data into `Generated.Real` data. Note that these figures include overlap, as some studies used multiple dataset types. Summary of the different dataset usage is illustrated in Figure 9.

7.2. Evaluation Metrics and Benchmarks

Metrics quantify the performance of AI++ LA techniques, while benchmarks represent the baseline methods used for comparison.

The most common evaluation metrics are Precision (70.73%), Recall (67.07%), and F1-Score (64.63%). Threshold-independent metrics are less common: Accuracy (10%), AUC/ROC (7.32%), RMSE (3.66%), Top-K Metrics (3.66%), and MAPE (1.22%).

This heavy reliance on threshold-dependent metrics (precision, recall, F1-score) versus threshold-independent metrics (AUC/ROC) indicates that primary studies focused on *subjective* operational targets rather objective ones. In practical MSEs, defining a decision boundary is mandatory to trigger binary alerts; however, the scarcity of AUC/ROC reporting obscures how sensitive these models are to threshold tuning. This is a critical gap for enterprise adoption. Trade-offs between alarm fatigue (precision) and missed incidents (recall) require dynamic threshold adjustment rather than static operating targets discussed in academic benchmarks.

Benchmarking Trends: Further, while these metrics are consistent across studies, differences in datasets and experimental design preclude direct comparisons between studies. However, standard baselines have emerged. Studies frequently compared against Bayesian Linear Regression, Long-Short Term Memory (LSTM), Multi-Layer Perceptron (MLP), Granger Causality, Random Forest, Support Vector Machine, Decision Tree, Naive Bayes, Gated Recurrent Unit (GRU), k-Nearest Neighbor (kNN), Graph Neural Networks (GNN), and Convolutional Neural Networks (CNN).

Recent work demonstrates a shift toward more sophisticated comparisons:

- **Model Improvement:** Duan et al, [P49] demonstrate that after embedding the AFALog framework, the F1 scores of GRU, LSTM, Bidirectional LSTM, and Transformer models increased by an average of 6.9%, 6.3%, 6.4%, and 7.0%, respectively.
- **Competitive Benchmarking:** Liu et al. [P60] benchmarked UAC-AD against seven benchmarks (including DeepLog and LogRobust), demonstrating F1-score improvements of 4.5-19.3% across three datasets. Similarly, Ding et al. [P16] compared LLM4MST against six benchmarks for root cause localization, establishing new state-of-the-art performance with 0.978 F1-score. Wang et al. [P13] benchmarked MRCA against CloudRanger, MicroCause, TraceDiag, Nezha, and epsilon-Diagnosis, achieving 0.93 precision for metric-level RCA. Liu et al. [P76] compared LMGD against DeepTraLog (Deep SVDD), TraceAnomaly (Posterior Flow-Based VAE), DeepLog (LSTM), and LogAnomaly (LSTM), showing consistent improve-

Table 9: Dataset Sources

Dataset	**Study**	**URL**
HDFS	[P5], [P48], [P49], [P19], [P46], [P36], [P22], [P20], [P21], [P14], [P32], [P67], [P33], [P28], [P29], [P39], [P23]	`https://github.com/logpai/loghub/tree/master/HDFS`
BGL	[P18], [P48], [P49], [P36], [P21],[P20], [P14], [P32], [P33], [P34], [P39], [P23]	`https://www.usenix.org/cfdr-data` or `https://github.com/logpai/loghub/tree/master/BGL`
Thunderbird	[P14], [P32], [P33], [P34]	`https://www.usenix.org/cfdr-data` or `https://github.com/logpai/loghub/tree/master/Thunderbird`
Spirit	[P34]	`https://www.usenix.org/cfdr-data`
Liberty	[P33]	`https://www.usenix.org/cfdr-data`
Cloudwise	[P45], [P61], [P6]	`https://github.com/CloudWise-OpenSource/GAIA-DataSet`
AI Ops Challenge	[P38], [P43]	`https://github.com/NetManAIOps/AIOps-Challenge-2020-Data`
OpenStack	[P46], [P79], [P20]	`https://github.com/logpai/loghub/tree/master/OpenStack`
ATLAS	[P53], [P61]	`https://github.com/purseclab/ATLAS`
Hades	[P60], [P16]	`https://github.com/NetManAIOps/Hades`

ments in both recall (2.63%) and F1-score (1.05%).

These comprehensive benchmarking efforts reflect the field's maturation, with researchers increasingly comparing against multiple established baselines rather than single reference methods.

7.3. State-of-the-Art Public Datasets

Public datasets refer to open-access log repositories used to standardize reproducibility. Ten `Synthesized.Public` datasets appear across studies: AIOps Challenge, ATLAS, BGL, CLoudwise, HDFS, Liberty, OpenStack, Hades, Spirit, and Thunderbird. LogHub provides a large number of these datasets for use in these studies and sources summarized in Tale 9. LogHub provides most of these datasets, compiled by Zhu et al. [21] to facilitate AI-driven log analytics research.

7.3.1. USENIX Supercomputing Datasets

BGL, Thunderbird, Liberty, and Spirit are part of the USENIX public datasets, for which, links are provided in Table 9. These open datasets of logs from supercomputing systems (2004-2006) containing alert and non-alert messages with category tags amenable to alert detection research. These systems varied from 512 to 131072 processors and vary by two orders of magnitude in processors and one order of magnitude in memory. The datasets also vary in size: Thunderbird dataset is 1.9GB, Spirit is 864MB, Liberty is 641MB, and the BGL dataset is the smallest at 60MB.

Yamanaka et al. [P34] compared LogELECTRA against DeepLog, LogAnomaly, PLELog, and RobustLog using BGL, Thunderbird, and Spirit. Guan et al.'s [P33] LogLLM achieved the highest F1-score across BGL, Thunderbird, Liberty, and HDFS datasets, outperforming NeuralLog by an average of 6.6%, surpassing FastLogAD, LogBERT, and RAPID.

Guo el al. [P14] present the experimental results on the HDFS, BGL, and Thunderbird datasets against seven different methods: LogCluster, DeepLog, LogAnomaly, Logbert and established methods, Isolation Forest, PCA, and OCSVM, with LogBERT achieving superior F1 scores on all three datasets. Three years later, Almodovar et al. [P32] introduced LogFit, which outperformed DeepLog and LogBERT on HDFS for Recall, F1, and Specificity, and marginally (only 1-4%) exceeded them on BGL and Thunderbird.

7.3.2. HDFS Dataset Versions

HDFS [29] has been the most widely studied, with 17 papers among the primary studies. Loghub contains three public versions of these datasets:

- **HDFS v1** is a log set generated in a private cloud environment using benchmark workloads and manually labeled through handcrafted rules to identify the anomalies. The logs are sliced into traces according to block IDs. Then, each trace associated with a specific block ID is assigned a label for anomalies.
- **HDFS v2** is a log set collected by aggregating logs from the HDFS system in a lab for research purposes. The logs are large, at over 16GB, and are provided as-is without further modification or labeling, which may involve both normal and abnormal cases. Making this version a potentially inconsistent dataset.

[29] `http://hadoop.apache.org/hdfs`

- **HDFS V3 from TraceBench** is an open HDFS monitoring dataset collected in a real IaaS environment using MTracer. It includes traces from various cluster sizes, request types, and workload speeds. Beyond normal operation data, it contains traces with 17 injected faults (both functional and performance-related). The dataset comprises over 370,000 traces across 364 files collected during 180+ hours of monitoring.

Khudyakov et al. [P5] evaluated all three versions and identified HDFS_v3 (TraceBench) as the only suitable dataset with distributed tracing.

7.3.3. Cloud-Native Public Datasets

OpenStack [30] dataset is a cloud operating system that controls large pools of compute, storage, and networking resources. This dataset was generated on CloudLab and contains both normal logs and abnormal cases with failure injection, making the data amenable to AD research.

Du et al. [P20] introduced DeepLog in 2017 and made use of the OpenStack dataset in their experiments, along with the BGL and HDFS datasets. Zhang et al. [P46] later introduced TWLog, comparing it against DeepLog, Primary Component Analysis (PCA), and Support Vector Machine (SVM) using both OpenStack and a generated Kubernetes dataset, extending beyond supercomputer logs to container-based environments.

CloudWise, also known as the **Generic AIOps Atlas** (GAIA) dataset, contains the data from a small microservice called MicroSS. It contains 6,500+ metrics and 7,000,000 log items including simulated anomalies and injection records for fair RCA algorithm evaluation. This dataset is used by Della et al. [P80] and Xu et al. [P61].

Six studies extends datasets from previous research. The **AIOps Challenge** dataset, constructed by Li et al. [22], was used by themselves [P71], Shi et al. [P38] and Zhang et al [P43]. The **ATLAS** dataset from Alsaheel et al. [23] was used by Chen et al. [P53] for testing FedTag.

7.4. State-of-the-Art Generated Datasets

Generated datasets are created by researchers running specific microservice applications in a controlled MSE to produce logs. Researchers generated their own microservice

[30] `https://www.openstack.org`

Table 10: Distribution of Microservices in use to Generate Logs.

Microservice	#	Studies
TrainTicket	19	[P9],[P10],[P1],[P64],[P65],[P17],[P73],[P26],[P44],[P40],[P42],[P43],[P52], [P16], [P76], [P13], [P78], [P27], [P12]
Online Boutique	3	[P17], [P13], [P6]
E-Shopper	2	[P65], [P16]
MicroSS	2	[P10], [P26]
QoTD	2	[P10], [P64]
Sock Shop	2	[P1], [P31]
Hipstershop	2	[P15], [P77]
Product Review	1	[P17]
Teastore	1	[P52]
Book-shop	1	[P3]
Other	8	[P81],[P82], [P72],[P11],[P59],[P37],[P56],[P62]

Table 11: Distribution of papers using production datasets.

Availability	Papers
Public & Private	[P49], [P23], [P19], [P22], [P45], [P61]
Private Only	[P2], [P74], [P75], [P81], [P25], [P30], [P51], [P57], [P41], [P35], [P66], [P4], [P58], [P63], [P3], [P55]

log datasets in 35 studies (representing 43 distinct generation instances). This involved deploying open-source microservices (summarized in Table 10) or capturing logs from private production systems (Table 11).

7.4.1. Log Generation Using Open-Source Microservices

TrainTicket is the most popular open-source microservice (19 uses), followed by Online Boutique (3 uses), E-Shopper, MicroSS, QoTD, Sock Shop, and Hipstershop (2 uses each). Product Review, Teastore, and Bookshop each appear once. Eight papers did not explicitly identify their log generation services.

7.4.2. Private/Industrially Generated Datasets

A further 22 studies generated logs from commercial services unavailable for public research. While 6 papers used this data alongside public benchmarks, 16 studies relied exclusively on their private data. They are summarized in Table 11.

Industrial applications: Sarika et al. [P74] and Catovic et al. [P57] reported experiences at Ericsson, with Catovic deploying Linnaeus across a dozen projects. Liu et al. [P51] deployed MicroHECL on Alibaba's e-commerce system (846 million monthly active users, 30,000+ microservices). Wang et al. [P41] analyzed 952 eBay incidents over 15 months (1/2020 - 4/2021) with SRE team assistance.

Yi et al. [P55] evaluated DQN-DBO fusion on logs from a Marketing 2.0 system. Lu et al. [P63] deployed Bi-LSTM-ATT on a power grid microservice network in Guizhou, China, achieving 95% precision. Hu et al. [P58] tested Knowledge Graph-based monitoring on simulated enterprise networks with 88.74% prediction accuracy. Dodda et al. [P3] benchmarked ML algorithms on AWS EC2 and Azure using a book-shop application, with SVM achieving 99.81% accuracy. Darwesh et al. [P4] collected Kubernetes telemetry for DoS detection, achieving F1 scores exceeding 0.999.

Additional private sources: Zhang et al. [P22] evaluate LogRobust on three datasets, including two HDFS datasets, and one real-world industrial dataset collected from Microsoft. Kohyarnejadfard et al. [P25] make use of logs that come from a microservice developed by Ciena Co. Zhang et al. [P45] collected from the management system of a top-tier commercial bank. Nedelkoski et al. [P66] evaluated production cloud data from Huawei. Lu et al. [P63] collected from internal microservices network of a power grid company in Guizhou, China.

Mixed approaches: Yang et al. [P23] combined 2 public datasets, HDFS and BGL, and two industry datasets from Tianjin University. Wijesinghe et al. [P19] compared LogBert against LogAnomaly using HDFS and AWS CloudWatch data. Duan et al. [P49] experiments on three datasets, including two open datasets (HDFS, BGL) obtained from LogHub, and one log dataset (KUB) collected from an industrial system based on Kubernetes. Xu et al. [P61] also combines GAIA dataset with logs from a large-scale microservice system operated by an e-commerce company.

Several work fuse sythenthetic data with public datasets. Wang et al. [P13] and Nguyen Ba-Hung et al. [P12] used TrainTicket and synthetic e-commerce datasets with injected faults for validation. Liu et al. [P76] evaluated log-metric fusion on TrainTicket with synthetic anomalies.

Other sources: Streiffer et al. [P30] collect metrics from a system composed of 21 nodes running 24 different services and deployed on IBM Bluemix. Aktacs et al. [P2] logs from an unknown Telecom provider. Lin et al. [P29] compare 22 different open-source logging systems using Project Darkstar, a discontinued open-source framework for Massive Multiplayer Online Game development, written in Java and deployed as game engine middleware [31]. Wahono et al. [P75] and Munir et al. [P35] used private datasets without explicit sources. Tang et al. [P27] simulated a prototypical microservice architecture (propriatory) for fault linkage evaluation.

7.5. Emerging Trends for Dataset

The trends for experiments are primarily to use a `Generated.Lab` dataset or mixed dataset, with 59% (47) of the papers making use of one of these methods to run their experiments. The use of `Generated.Real` datasets appears in 41% (33) of the studies, with only 27.5% (22) actually been generated industrially. However, those datasets are typically under non-disclosure agreements and are not accessible, making it difficult to reproduce results or extend the research. This accessibility gap represents a significant barrier to advancing the field, as evidenced by the fact that only a handful of recent studies [P60, P16] have committed to releasing both their datasets and implementation code.

LogHub [21] maintains a set of different log types for various AI-driven log analytics and is freely available for academic work. These logs have been extensively used as benchmarks by various AI algorithms, though concerns about dataset saturation and model overfitting to these well-studied logs are emerging. Notably, the Hades dataset from the AIOps community and the expanded GAIA dataset represent newer public resources that offer fresh evaluation opportunities [P60, P61].

The studies that provide generated datasets from real-world industry `Generated.Real` are of most importance for validating practical applicability. However, the gap between lab-generated synthetic faults and actual production failures remains a critical challenge. Several recent studies [P12, P27, P4] acknowledge that their synthetic fault injection may not capture the full complexity of real-world failure scenarios, suggesting that future work must prioritize collection and sharing of production-grade datasets with realistic failure patterns. The key challenges and benefits of different dataset types are discussed in the following sections.

[31]`https://en.wikipedia.org/wiki/Project_Darkstar`

8. RQ3 Results: What are the benefits, challenges, and risks associated with different AI++ techniques for microservice LA?

Our systematic review of 82 studies revealed a complex landscape of benefits, challenges, and risks associated with implementing AI++ approaches for enterprise microservice LA. This section presents a comprehensive analysis of these factors from an operational deployment perspective to guide researchers and practitioners in navigating this evolving domain.

8.1. Benefits across different AI++ approaches in microservice log analysis

Our SLR identified five main categories of benefits associated with AI++ based microservice LA: (1) performance and accuracy, (2) operational adaptability, (3) real-time processing efficiency, (4) architectural advantages, and (5) enahanced operational visibility. These benefit themes appeared consistently in all the studies surveyed, with their frequency (summarized in the Appendix Table D.16).

Performance and accuracy improvements were widely observed when comparing AI-driven methods to traditional rule-based and earlier ML systems. Advanced techniques using DNNs, multimodal data fusion, or hybrid architectures consistently outperformed baselines such as AnoFusion and JLT. For instance, Rouf et al. [P10], achieved an F1-score of 0.98 on a production system - MicroSS, setting a new benchmark. It indicates substantial reduction in false positives and missed incidents that directly impact operational efficiency and mean time to resolution (MTTR). Similarly, Ding et al. [P16] achieved 0.978 F1-score with few-shot learning capabilities, while Liu et al. [P60] reported 4.5-19.3% F1-score improvements over established baselines across multiple datasets.

Operational adaptability is critical for enterprise environments where system configurations and service dependencies evolve continuously. Self-supervised and domain-agnostic models showed adaptability to evolving log patterns with minimal labeling. Models developed in studies such as of Kohyarnejadfard et al. [P25] and DeepLog [P20] showed cross-infrastructure transferability, crucial for multi-cloud enterprise deployments. Hybrid approaches further demonstrate adaptability: Wang et al. [P13] achieved metric-level RCA with 0.93 precision through multi-modal causal discovery, while Hu et al. [P58] used Knowledge Graphs with LSTM to achieve 88.74% fault prediction accuracy across evolving system states.

Real-time processing efficiency saw major gains, with many models enabling sub-second AD and RCA for meeting enterprise SLAs. Wang et al. [P41] notably diagnosed faults across 5,000 microservices in under five seconds, showcasing real-time scalability required for large-scale enterprise microservice architectures. Chen et al. [P6] improved diagnosis time by 0.10-0.18 seconds through event call chain construction, while Kaushik et al. [P77] demonstrated 15% lower MAPE with RNN for proactive fault prediction, enabling preventive action before system failures.

Architectural advantages arose from preserving inter-service dependencies. Methods that maintain service dependency relationships and trace topologies, such as [P9, P38, P56, P12, P76] provide holistic system insights without requiring extensive infrastructure modifications. Nguyen Ba-Hung et al. [P12] demonstrated this through HGTs that achieved 88.9% accuracy by preserving user impact propagation patterns.

Enhanced operational visibility addresses enterprise requirements for regulatory compliance and operational transparency. Non-intrusive AI++ observability is advantageous over traditional black-box methods because techniques such as grammar-based detection [P65] and context-aware visualizations [P41] enable operations teams to understand and trust automated decision-making processes.

8.2. Distribution of Benefit Themes by AI++ Technique and Enterprise Use Case

To strictly map the primary studies to specific benefit themes, we utilized the data synthesis methodology detailed in Section 5. During the *data coding* stage (Stage 1), we extracted explicit claims of operational improvement from the *Results*, *Discussion*, and *Conclusion* sections of each primary study. These claims were validated against the experimental evidence provided in the papers (e.g., quantitative metrics for performance or qualitative evidence for adaptability) rather than relying solely on abstract keywords.

In the *theme mapping* stage (Stage 2), these codes were aggregated into five high-level benefit categories. A single study was assigned to a benefit category only if it demonstrated a distinct, evidence-backed improvement in that area.

Building on this classification, we analyzed the distribution of benefit themes across AI++ methods and primary use cases to better understand how different techniques contribute to enterprise microservice LA. (Appendix Tables D.17 and D.18 summarize the findings).

AI++ method analysis. Figure 10 summarizes the distribution of benefit categories between AI++ techniques. Neural networks (NN) are the most versatile approach, contributing prominently to *performance and accuracy*, *operational adaptability*, and *architectural advantages*. However, they are notably absent in *real-time processing* and *operational visibility*, where other methods take the lead.

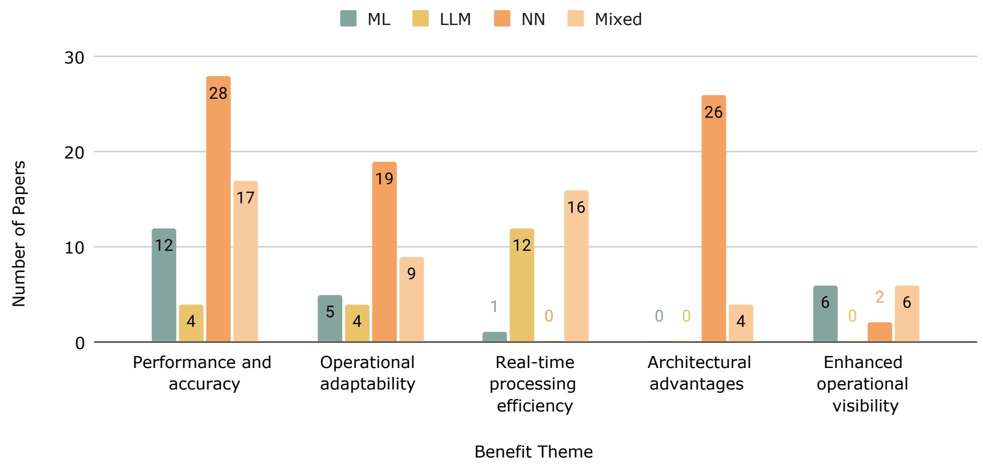

Figure 10: Distribution of Benefits by AI++ Method Type

LLMs are the most prominent in *real-time processing*, showing the highest count of methods level in this category. They also contribute to *operational adaptability* and, to a lesser extent, *performance and accuracy*, particularly in scenarios that require minimal labeling and adaptation to dynamic log patterns. However, LLMs are not represented in architecture-related or monitoring-oriented benefits.

Traditional ML methods appear most frequently under *performance and accuracy*, followed by *enahanced operational visibility* and *operational adaptability*. They do not appear in *architectural advantages*, and their role in *real-time processing* is relatively minor. Despite these limitations, ML techniques remain operationally viable for many enterprise scenarios due to their simplicity, interpretability, and reliability for common prediction and classification tasks.

Mixed approaches, those that combine multiple AI paradigms, show broad and balanced contributions across all five benefit categories. They are especially strong in *real-time processing* and *performance and accuracy*, and they contribute significantly to *operational adaptability*, *operational visibility*, and even *architectural advantages*, although less prominently. These methods benefit from their ability to integrate structural patterns (e.g., traces, graphs) with semantic context (e.g., log events, metrics), making them well-suited for complex enterprise environments where multiple operational requirements must be satisfied simultaneously. However, this versatility comes at the cost of increased implementation complexity and operational overheads.

Operational benefits across enterprise use cases. Figure 11 illustrates how different benefit categories align with core use cases in microservice LA. The most prominent theme across the dataset is AD, which contributes meaningfully to all five benefit categories. It is especially influential in driving improvements in *performance and accuracy*, *operational adaptability*, and *architectural advantages*.

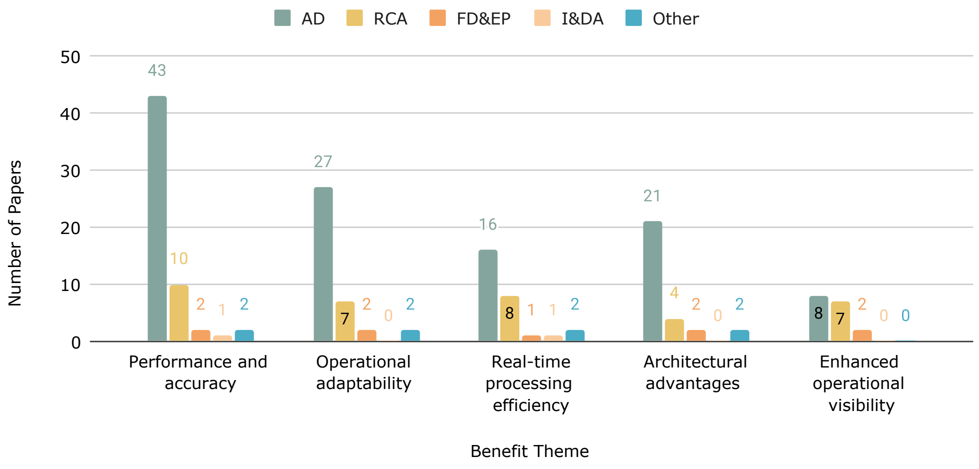

Figure 11: Distribution of Use Cases Across Benefit Types

RCA also shows strong coverage across benefit themes, particularly in *performance*, *real-time efficiency*, *operational adaptability*, and *operational visibility*. RCA use cases frequently leverage architectural or temporal patterns to improve diagnostic precision, aligning well with enterprise needs for rapid incident resolution and MTTR improvement.

Fault Diagnosis and Error Prediction contributes mainly to *performance and accuracy* and *operational adaptability*, supporting tasks that require early detection of system degradation. While less mature than AD or RCA, this category remains relevant for enterprises seeking to transition from reactive to predictive operational models.

Interaction and Dependency Analysis demonstrates a narrower scope, with contributions concentrated in operational contexts such as *performance* and *real-time processing.* These methods often focus on capturing service call patterns and dependencies but are less common in architectural or visibility-oriented applications.

Lastly, other use cases appear across several benefit categories, albeit with lower frequency. These applications often emphasize complementary insights that enhance broader system understanding rather than direct fault management.

In Summary, while ML methods dominate practical deployment scenarios due to their simplicity and high fault prediction performance, the growing trend toward DL, GNN, LLM, and hybrid methods reflects a shift toward achieving greater structural awareness, flexibility, and scalability in increasingly complex enterprise MSEs. However, this evolution must balance advanced capabilities with operational considerations including deployment complexity, maintenance overhead, and integration requirements.

8.3. Operational challenges and enterprise risks across AI++ approaches in microservice LA

Our systematic review identified five main categories of challenges and risks associated with AI++-based microservice LA: (1) performance and resource constraints, (2) data limitations and dependencies, (3) reliability and accuracy issues, (4) cross-environment portability and security risks, and (5) enterprise integration complexities. These challenge themes appeared consistently throughout the studies surveyed. Further details are available in the Appendix table D.19).

Performance and resource constraints represent the most frequently reported operational barrier (41 studies, 50%), particularly affecting computationally intensive methods. This directly impacts enterprise deployment feasibility and operational costs. LLMs faced the highest proportion of these constraints among all techniques examined, making real-time analysis challenging in production environments. They translate to increased infrastructure costs for log management and analysis, potential latency issues affecting real-time incident response, and scalability limitations that may prevent deployment across large-scale MSAs (e.g., cross-cloud deployments). When analysing the work of Wang et al. [P13], it could be argued that PC causal discovery adds considerable computational overhead for real-time RCA, while HGT's complexity in [P12] may hinder scalability beyond small experimental datasets.

Data limitations and dependencies were almost equally prevalent (46 studies, 56%), with this challenge manifesting in the requirement for extensive labeled log datasets, which are often scarce in production environments. This directly affects enterprise-level AI++ LA adoption timelines, as it nessecitates investmenting in data curation, annotation, and ongoing data quality management. As a result, LA will rely heavily on domain expertise.

Reliability and accuracy issues were reported in 42 studies (51%), affecting the practical deployment and trustworthiness of AI++ based LA systems. This directly impact enterprise risk management, compliance readiness, and operational team acceptance. Operational consequences include increased manual oversight requirements and extended testing and validation for maintaining enterprise SLAs and system reliability. Hence, synthetic fault injection used in [P27, P4, P13], may not fully capture production failure complexity, leading to optimistic performance estimates.

Cross-environment portability and security risks appeared in 28 studies (34%), affecting organizations with multi-cloud, hybrid, or diverse MSAs. They manifest as difficulties in transferring trained LA models between different MSEs and potential security vulnerabilities in AI++ systems. The operational impact includes increased deployment complexity (including extended security assessments) and constraints on cross-organizational model sharing or reuse. Espeically, Liu et al. [P76]'s work underlines synchronization challenges when fusing logs and metrics across distributed systems with clock drift and network delays.

Enterprise integration complexities were explicitly reported less frequently (10 studies), but almost exclusively in LLM-based approaches. These challenges relate to the complexity of integrating AI++ methods into existing monitoring systems and operational workflows. Hence, implication include extended integration timelines involving training and from possible disruptions to already established processes involving the LA dependent use cases (e.g., fault response, RAC). The work of Ding et al. [P16] also highlights that LLM fine-tuning requires substantial GPU resources, limiting adoption for organizations without specialized infrastructure.

8.4. Operational risk profile by AI++ technique and use case

We analyze the distribution of operational challenge themes across both AI++ methods and primary use cases to better understand how specific challenges manifest themselves in different contexts. While Figure 12 and Figure 13 summarizes the risks, analysis of these distributions is in Appendix Tables D.20 and D.21.

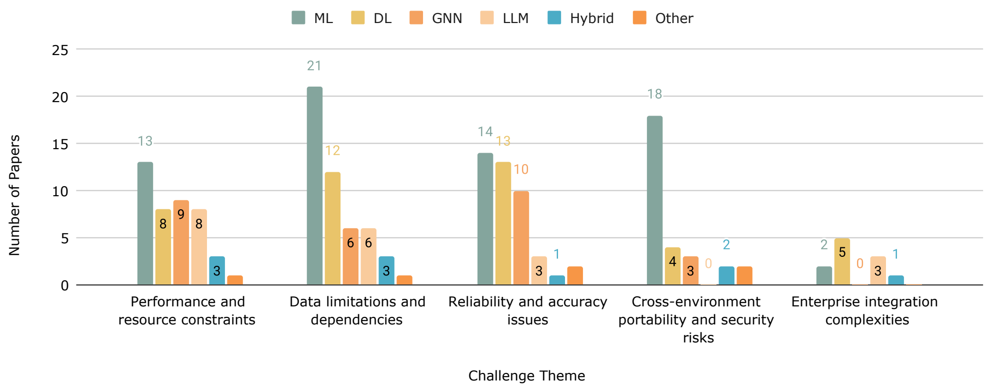

Figure 12: The distribution of operational risks in using AI++ techniques microservice LA.

AI++ Method Analysis. Traditional ML approaches (34.5% of studies) faced the highest proportion of operational risks from data availability (70%) and cross-environment portability (60%), suggesting difficulties in adapting to diverse and evolving MSEs. DL methods (28% of the studies) showed more distributed operational risks, with 54% reporting reliability issues, 50% reporting data limitations, and 33% reporting resource constraints. GNNs (20% of the studies) encountered significant performance limitations (53%) and reliability issues (59%) when modeling complex service relationships. LLMs (13% of the studies) faced the highest proportion of performance and resource constraints (73%) of any method, alongside unique enterprise intergration (i.e., technical implementation, data availability, reliability) challenges not reported in other methods; including complex deployment requirements, specialized infrastructure needs, and integration complexities with existing enterprise observability platforms.

Hybrid approaches combining multiple paradigms showed more balanced risk profiles but introduced increased implementation complexity [P58, P16].

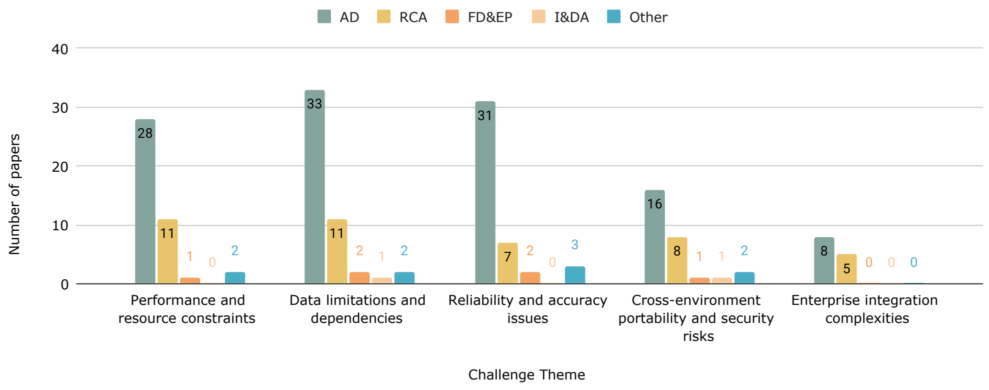

Figure 13: The distribution of operational risks among use cases of microservice LA.

Operational risk distribution across use case analysis. Across use cases, AD (59 studies) shows distributed operational risks across all challenge categories, with performance constraints, data limitations, and reliability issues, each affecting approximately 55% of the studies. RCA (20 studies) was particularly affected by performance constraints (55%), data limitations (55%), and cross-environment portability concerns (40%). Fault diagnosis and error prediction (2 studies) exhibit critical operational risks, with 100% of implementations reporting both data availability limitations and reliability issues. A single study on interaction and dependency analysis reported both data limitations and cross-environment portability concerns.

9. Discussion: Learnings on AI++ for log analysis in microservices

We synthesize key findings from our systematic review of 82 primary studies on the application of AI techniques to enterprise MSE LA. Guided by our three RQs, we critically examine: (1) the state-of-the-art approaches and their enterprise deployment readiness, (2) standard datasets, evaluation practices and their alignment with operational requirements, and (3) known challenges of adopting AI++ microservice LA techniques in production environments. We identify emerging trends—including the increasing role of LLMs, the shift toward hybrid and multi-modal analysis techniques, and the demand for explainability and enterprise-scale real-world validation. These insights reveal critical gaps between current research approaches and enterprise deployment realities, informing both researchers and practitioners seeking to bridge this divide in AI-enhanced observability for production MSEs.

9.1. Reflections on the RQs

To contextualize the results of our review within enterprise deployment realities, we revisit the three research questions that guided this study (Figure 14). Each subsection synthesizes key patterns, techniques, and limitations observed across the literature, combining quantitative and qualitative insights on AI++ micorservice LA.

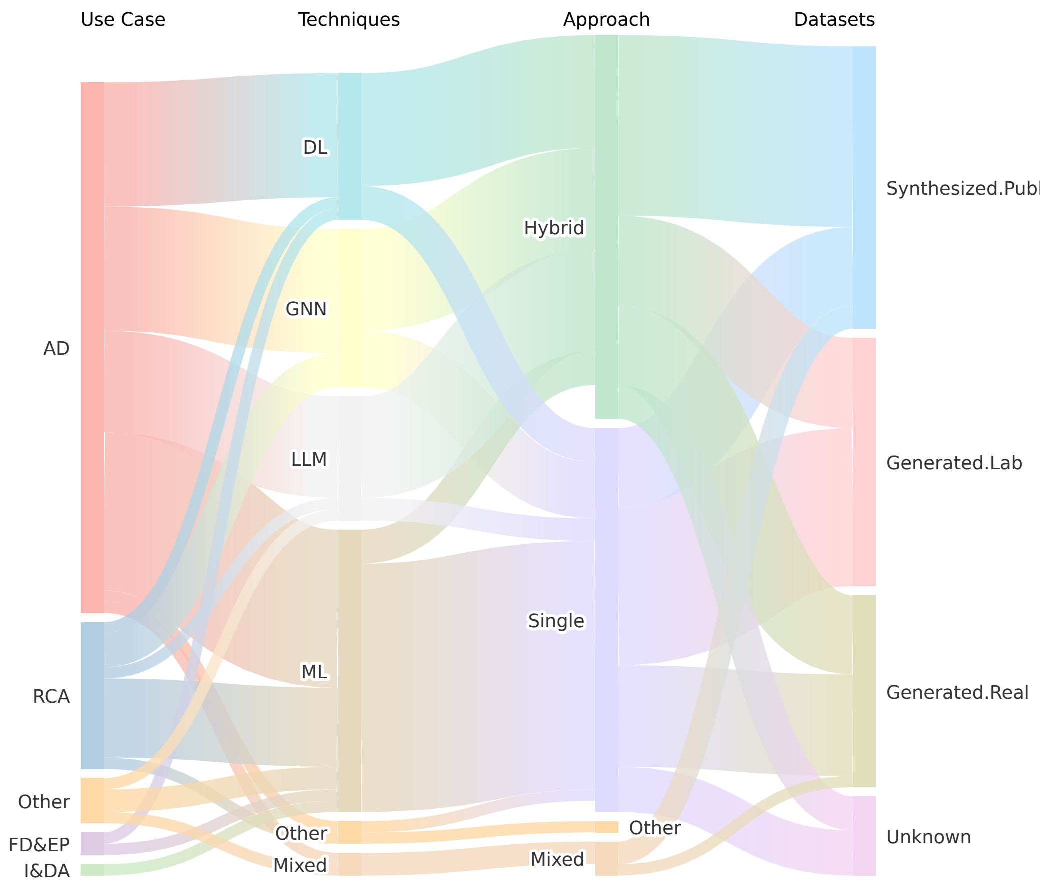

Figure 14: Mapping of AI++ techniques to microservice use cases to approaches used to datasets used in evaluating the efefctiveness of the approach.

RQ1: Current State-of-the-Art AI Tools and Techniques.

AI-based microservice LA research predominantly targets AD (72%), followed by RCA (24%) and fault diagnosis (2.4%), reflecting the primary operational priorities of enterprise microservice environments. Among AI++ techniques, traditional ML models—such as SVMs, logistic regression, and decision trees—remain the most widely adopted (37%) due to their interpretability, low resource requirements, and ease of deployment. These approaches align well with enterprise production needs, as shown by Aktacs et al. [P2], who used ML classifiers for microservice interaction prediction, and Catovic et al. [P57], who demonstrated portable, reusable ML pipelines for RCA.

However, the field is moving toward more expressive models that better capture the structural and temporal complexity of distributed systems. GNNs (21%) model inter-service relationships crucial to dependency reasoning and trace-based analysis [P46, P37]. DL methods (29%)—notably LSTM and Transformer variants—support temporal and contextual pattern learning in evolving log streams [P48, P5]. Meanwhile,LLMs (13%) are emerging for semantic understanding of unstructured logs: Han et al. [P15] introduced a one-shot RCA system using GPT-4 for rapid incident triage, and Zhang et al. [P18] proposed LogRAG, which employs retrieval-augmented generation to mitigate false positives in AD.

More than half of the reviewed studies (53.9%) employ hybrid architectures that combine complementary models—such as GNNs with Transformers—to integrate structural, temporal, and semantic signals. Systems such as InstantOps [P10] and MULAN [P17] exemplify this movement toward multi-modal, enterprise-ready log analysis frameworks capable of end-to-end monitoring and RCA. Despite these advances, most works still rely on controlled datasets or synthetic environments. Broader validation under production-scale workloads, system evolution, and resource constraints remains limited, signaling that empirical deployment readiness continues to lag behind methodological innovation.

RQ2: Datasets, evaluation metrics, and benchmarks vs. enterprise operational requirements.

A critical gap exists between research evaluation practices and enterprise operational requirements. Reseach dependence on some form of generated data (synthetic or small system-based)—used in 83% of studies—severely limits enterprise applicability, while 41% utilize real-world logs, and 27.5% combine both. This disconnect undermines the validity of research claims for enterprise deployment. Wijesinghe et al. [P19] finds that models such as DeepLog and LogBERT perform worse on industrial logs due to structural inconsistencies not captured in synthetic datasets. Catovic et al. [P57] further stress that ML systems often lack reusability in operational enterprise settings due to these evaluation limitations.

Enterprise privacy constraints and data ownership discourage dataset sharing, creating a fundamental barrier to enterprise-relevant research. While Rouf et al. [P10] and Duan et al. [P48] use anonymized or semi-public datasets, such efforts are rare and insufficient for representing true enterprise complexity. In contrast, evaluation practices lack standardization. While precision, recall, and F1-score are used, metric definitions vary. Khudyakov et al. [P5] emphasize that unsupervised models need more robust thresholds and evaluation design that refelct practical conditions. Zhang et al. [P18] highlight the sensitivity of log parsing and the value of error-aware evaluations to align with operational realities.

This evaluation-reality gap significantly hampers enterprise adoption progress. Even innovative work such as Pathak et al.'s [P64] log volume reduction method lack better evaluation protocols and enterprise-relevant benchmarks for more robust comparison.

RQ3: Challenges, Risks, and Benefits.

Several recurring challenges limit the effective application of AI in complex microservice LA. First, data quality and labeling present fundamental operational barriers. Zhang et al. [P18] demonstrates that parsing noise in logs causes unignorable false positive rates. Liang et al. [P40] highlight the impracticality of manual labeling in high-volume MSAs and the need for adaptable unsupervised methods that require least human intervention.

Scalability is another concern. Tan et al. [P26] introduce MAAD, where each service runs a local agent to distribute log-based event detection tasks. Pathak et al. [P64] reduce log volume with adaptive observability, lowering computational overhead while preserving utility—essential for cost-effective enterprise deployment.

Model interpretability remains a critical barrier to enterprise adoption because operational teams require transparent decision-making processes. Nobre et al. [P68] stress that opaque models hinder trust, and Catovic et al. [P57] recommend pipelines that are easier to validate and maintain. Han et al. [P15] note that LLMs offer semantic insights but require additional explainability mechanisms for production use.

Similarly, handling dynamic topologies is also difficult. Li et al. [P71] propose using sliding windows and call chains to adapt to topological changes, while Liu et al. [P51] employ real-time call graphs for fault localization in evolving microservice systems.

Despite these issues, benefits are evident. Active learning improves accuracy with minimal labeled data [P48]. Multi-modal systems using logs, metrics, and traces can enhance RCA and observability [P10]. Together, these findings indicate that, while challenges around data, scale, and trust persist, AI techniques can deliver substantial benefits when well integrated into monitoring pipelines.

10. Recommendations and Future Works

Our findings offer important insights for both researchers aiming to bridge the research-practice gap, advance the state-of-the-art, and practitioners seeking to implement AI-driven solutions in real-world MSEs. We outline key implications drawn from common challenges, reported limitations, and future work directions identified across the primary studies.

10.1. Summary of Recommendations

To consolidate our findings, Table 12 compares all recommendations across both audiences. Each recommendation is rated by its *criticality*, estimated *difficulty of adoption*, anticipated *benefits*, and key *challenges to adoption*. These evaluations are based on our synthesis and expert judgment.

10.2. Recommendations for Researchers

Enterprise-realistic datasets and benchmarks. Overreliance on synthetic logs limits fundamentally undermines enterprise applicability and deployment confidence. Existing datasets such as HDFS, BGL, and Thunderbird also do not fully represent microservice systems but rather more representative of high-performance monlith systems. As demonstarted by Wijesinghe et al. [P19], future work should prioritize releasing anonymized, enterprise-representative realistic datasets along with standardized benchmarks that reflect enterprise evaluation criteria.

Table 12: Comparison matrix summarizing recommendations for researchers and practitioners with their criticality, adoption difficulty, benefits, and challenges.

Recommendation	Criticality	Difficulty	Benefits if Adopted	Challenges to Adoption
Enterprise-realistic datasets and benchmarks	High	High	Improves external validity, enhances model generalizability, and enables reproducible enterprise-grade evaluation	Enterprise privacy, data sensitivity, and lack of standardized logging formats
Broader use cases beyond anomaly detection	Medium –High	Medium	Expands AI application scope (e.g., fault prediction, regression testing) and aligns with practical enterprise goals	Lack of labeled data for rare events; difficulty defining ground truth for RCA and dependency modeling
Multi-modal integration of logs, traces, and metrics	High	High	Enhances RCA accuracy and observability by leveraging cross-source correlations	Integration complexity, synchronization overhead, and tool heterogeneity across enterprise stacks
Enterprise-grade explainability	High	Medium –High	Increases user trust, supports compliance and debugging, enables adoption in regulated sectors	Trade-off between transparency and model complexity; absence of standard interpretability frameworks
Hybrid architectures for enterprise scale (e.g., GNN+LLM)	High	High	Captures structural and semantic patterns for large, dynamic MSAs; improves detection accuracy	Increased computational and maintenance overhead; complex model coordination
Evaluation in production-like settings	High	Medium	Improves robustness, reduces overfitting to lab datasets, validates real-world feasibility	Limited access to production environments; risk of system impact during testing
Prioritize interpretability in operations	Medium –High	Low –Medium	Facilitates faster incident resolution, compliance reporting, and stakeholder confidence	Possible reduction in raw performance; balancing clarity with precision
Domain- and architecture-specific customization	Medium	Medium	Improves precision and efficiency through tailored models and domain fine-tuning (e.g., API-level logs, LLMs for semantic LA)	Requires domain expertise and ongoing retraining; limited generalizability

Broader use cases. Current research in AD is research heavy. Beyond AD, more attention is needed on fault diagnosis, proactive prediction, and service dependency modeling that reflect real enterprise operational needs. Recent work [P56] indicates that API gateway logs can support regression testing, indicating untapped potential for enterprise-relevant applications.

Multi-modal integration. Enterprise MSEs generate diverse observability data streams that require integrated analysis approaches. Yet, most research examines these data sources in isolation, limiting enterprise deployment value. Future work should prioritize holistic approaches combining logs, traces, and metrics as in [P10, P51].

Enterprise-grade explainability. As deep and language models proliferate, their explainability must improve for operational acceptance and regulatory compliance. Model outputs need to be understood and verifiable under audit requirements [P68, P15]. Hence, research must develop explainability frameworks tailored to enterprise microservice LA.

10.3. Recommendations for Practitioners

Adopt hybrid architectures for enterprise scale. GNN+RNN or LLM-integrated pipelines, exemplified by InstantOps and MULAN [P10, P17], are better suited for capturing more complex microservice dependency patterns and temporal patterns. Practitioners should conduct pilot deployments comparing hybrid versus single-model approaches on representative workloads, measuring not only AD accuracy but also inference latency, memory footprint, and integration complexity with existing observability stacks (e.g., Prometheus, Jaeger). Decision criteria should balance enhanced AD capabilities against operational overhead—including model maintenance, version control for multiple components, and expertise requirements for debugging multi-stage pipelines.

Evaluate in production-like settings. Tools should be tested on real-world or staging environments since unrealistic, synthetic data skews performance claims [P19]. Concretely, this requires: (1) training and validating on logs from actual production systems with real failure modes rather than synthetic injections; (2) evaluating under production-realistic conditions including log volume variability, schema evolution, and infrastructure heterogeneity; and (3) measuring operationally relevant metrics such as time-to-detect, false positive rates during business hours, and mean time to resolution (MTTR) improvement. Synthetic benchmarks as found the analyzed studies only establish baseline capabilities and cannot substitute for validation against the complexity of production environments.

Prioritize interpretability. In compliance or incident management contexts, clarity often trumps marginal performance gains [P57, P68]. This requires selecting models that provide actionable explanations, such as attention weights highlighting specific log lines, causal graphs showing service dependencies leading to failures, or natural language summaries of detected anomalies. For incident response, interpretable outputs enable on-call engineers to quickly validate alerts, prioritize remediation actions, and document root causes for post-mortems. For compliance scenarios (e.g., System and Organization Controls 2, General Data Protection Regulation audits), interpretable decisions create audit trails showing why specific events were flagged. It supports regulatory requirements on algorithmic transparency.

Customize to domain and architecture. Generic ML models underperform in domain-specific enterprise environments with unique service architectures, business logic, and operational patterns. Customization strategies may include (not limited to):

- fine-tuning on organization-specific log schemas and error patterns,
- incorporating domain knowledge through feature engineering (e.g., microservice-level objectives, deployment topology), and
- adapting model architectures to match system characteristics, such as using API-level logs for request-response tracing or message-queue logs for asynchronous workflows [P56].

With the rise of ChatGPT, Claude, Gemini, and DeepSeek, future work must investigate domain-specific-fine-tuning and deployment of models for semantic log summarization, automated RCA, and interactive LA.

Collectively, trends in this area signal a shift from fragmented analysis tools to unified, intelligent observability platforms. Advances in LLM integration, explainability, and privacy-aware methods will be key to enabling scalable, production-ready log intelligence systems.

11. Threats to Validity

We systematically address potential threats to validity following Wohlin et al.'s framework [24] and outline the strategies implemented to mitigate each.

Construct Validity: The evolving complexity of AI methods presented inherent challenges to achieving complete coverage for categorization. To mitigate this risk, we employed an iterative search strategy, systematically refining terms to reflect microservice and AI-specific concepts. Data extraction was conducted independently by multiple researchers to minimize subjective bias. The categorization of AI approaches introduced additional threats, which we addressed by independently classifying a subset of studies, comparing results, and refining the classification schema through multiple iterations.

Internal Validity: Major threats include bias in study selection and inconsistencies in data extraction. To mitigate these risks, we employed a multi-stage review process with clearly defined inclusion and exclusion criteria, where multiple researchers independently assessed studies and a third reviewer resolved disagreements through arbitration, supported by a structured voting mechanism (Figure 2). Consistency in data extraction was addressed through structured extraction forms tailored to each research question, with multiple researchers independently extracting and reconciling data to minimize subjective interpretation, particularly given the technical complexity of AI methods. Nonetheless, certain threats persist: the classification of AI techniques (Table 8) required interpretive judgment, potentially introducing bias, and despite our snowballing process identifying 21% additional papers, it remains possible that some relevant studies were missed, affecting the completeness of our findings.

External Validity: Several factors may limit generalizability of our findings. 43 out of the 82 studies relied on synthetic or lab-generated datasets rather than real-world production data, potentially affecting applicability to operational MSEs where log patterns and complexity differ substantially. Additionally, our findings reflect a strong emphasis on AD tasks within the existing literature, with comparatively less coverage of RCA, fault diagnosis, and interaction analysis, suggesting that our findings may generalize more effectively to AD scenarios. Although we included studies from multiple major digital libraries we excluded gray literature and focused primarily on recent studies to reflect current practices.

Conclusion Validity: Although we employed a rigorous and iterative search strategy across four major digital libraries, the complexity and rapidly evolving nature of AI and microservice terminology raises the possibility that some relevant studies were overlooked. Our study selection process, involving multiple screening stages, necessarily applied aggressive filtering to manage the large volume of literature, which may have excluded studies whose relevance was not immediately evident from titles or abstracts. Additionally, by excluding gray literature such as technical reports, white papers, and blog posts, we enhanced the scientific rigor of the review but may have missed insights from emerging industry practices not yet captured in academic publications.

12. Conclusion

This systematic literature review underscores the transformative potential of AI techniques in advancing LA within MSEs. Through rigorous analysis of 82 primary studies from an initial pool of 2208 papers, the review categorized AI techniques—such as ML, DL, GNNs, and LLMs—and examined their applications in addressing operational challenges such as AD, RCA, and fault prediction. Our study identified 61 tools, 10 publicly available datasets, and 10 open-source microservices benchmarks, establishing critical connections between techniques, tools, and datasets to support future comparative research and benchmark-driven evaluations. While AI techniques offer benefits in performance, adaptability, and real-time processing capabilities, challenges persist, including performance constraints, data size and quality limitations, and reliability issues in production environments. The review highlights an emerging paradigm shift toward hybrid AI architectures that combine structural, semantic, and temporal reasoning to address the complexity of modern microservice systems. Our findings highlight a critical gap between research focus and enterprise operational needs, with current work concentrating on AD while other essential tasks such as fault diagnosis and service dependency modeling remain underexplored. By systematically mapping the use of public datasets and linking them with AI approaches and tools, this study provides a consolidated foundation for understanding the role of modern AI techniques in LA and offers a practical roadmap for benchmarking and developing reproducible, enterprise-aware AI solutions in micorservice observability research. These insights are essential for bridging the gap between academic innovation in AI++ LA and practical deployment in complex production environments.

Funding

This material is based upon work supported by the National Science Foundation under Grant No. 2409933. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Declaration of competing interest

The authors declare the following financial interests or personal relationships which may be considered as potential competing interests: Tomas Cerny reports financial support was provided by National Science Foundation. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] V. Bushong, A. S. Abdelfattah, A. A. Maruf, D. Das, A. Lehman, E. Jaroszewski, M. Coffey, T. Cerny, K. Frajtak, P. Tisnovsky, M. Bures, On microservice analysis and architecture evolution: A systematic mapping study, Applied Sciences 11 (2021). URL: https://www.mdpi.com/2076-3417/11/17/7856.

[2] L. Giamattei, A. Guerriero, R. Pietrantuono, S. Russo, I. Malavolta, T. Islam, M. Dînga, A. Koziolek, S. Singh, M. Armbruster, J. Gutierrez-Martinez, S. Caro-Alvaro, D. Rodriguez, S. Weber, J. Henss, E. F. Vogelin, F. S. Panojo, Monitoring tools for devops and microservices: A systematic grey literature review, Journal of Systems and Software 208 (2024) 111906. URL: https://www.sciencedirect.com/science/article/pii/S0164121223003011. doi:https://doi.org/10.1016/j.jss.2023.111906.

[3] S. He, P. He, Z. Chen, T. Yang, Y. Su, M. R. Lyu, A survey on automated log analysis for reliability engineering, ACM Computing Surveys 54 (2022) 1–37. URL: https://dl.acm.org/doi/10.1145/3460345. doi:10.1145/3460345.

[4] J. Cândido, M. Aniche, A. van Deursen, Log-based software monitoring: a systematic mapping study, Empirical Software Engineering 26 (2021) 1–47.

[5] L. Korzeniowski, K. Goczyła, Landscape of automated log analysis: A systematic literature review and mapping study, IEEE Access 10 (2022) 21892–21913. doi:10.1109/ACCESS.2022.3152549.

[6] S. Partovian, A. Bucaioni, F. Flammini, J. Thornadtsson, Analysis of log files to enable smart-troubleshooting in industry 4.0: A systematic mapping study, IEEE Access 12 (2024) 147640–147658. doi:10.1109/ACCESS.2023.3342365.

[7] T. Wang, G. Qi, A comprehensive survey on root cause analysis in (micro) services: Methodologies, challenges, and trends, 2024. URL: https://arxiv.org/abs/2408.00803. arXiv:2408.00803.

[8] S. Zhang, S. Xia, W. Fan, B. Shi, X. Xiong, Z. Zhong, M. Ma, Y. Sun, D. Pei, Failure diagnosis in microservice systems: A comprehensive survey and analysis, 2024. URL: https://arxiv.org/abs/2407.01710. arXiv:2407.01710.

[9] S. Moreschini, S. Pour, I. Lanese, D. Balouek-Thomert, J. Bogner, X. Li, F. Pecorelli, J. Soldani, E. Truyen, D. Taibi, Ai techniques in the microservices life-cycle: A survey, 2023. URL: https://arxiv.org/abs/2305.16092. arXiv:2305.16092.

[10] D. Z. Zhou, M. Fokaefs, Ai assistants for incident lifecycle in a microservice environment: A systematic literature review, arXiv preprint arXiv:2410.04334 (2024). doi:10.48550/arXiv.2410.04334.

[11] S. Akhtar, S. Khan, S. Parkinson, LLM-based event log analysis techniques: A survey, 2025. URL: http://arxiv.org/abs/2502.00677. doi:10.48550/arXiv.2502.00677. arXiv:2502.00677 [cs].

[12] A. S. Abdelfattah, T. Cerny, M. S. H. Chy, M. A. Uddin, S. Perry, C. Brown, L. Goodrich, M. Hurtado, M. Hassan, Y. Cai, R. Kazman, Multivocal study on microservice dependencies, Journal of Systems and Software 222 (2025) 112334. doi:10.1016/j.jss.2025.112334.

[13] A. B. Nassif, M. A. Talib, Q. Nasir, F. M. Dakalbab, Machine learning for anomaly detection: A systematic review, IEEE Access 9 (2021) 78658–78700. doi:10.1109/ACCESS.2021.3083060.

[14] M. Steidl, B. Dornauer, M. Felderer, R. Ramler, M.-C. Racasan, M. Gattringer, How industry tackles anomalies during runtime: Approaches and key monitoring parameters, in: 2024 50th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), IEEE, NJ, USA, 2024, pp. 364–372. URL: https://ieeexplore.ieee.org/document/10803340/. doi:10.1109/SEAA64295.2024.00062.

[15] J. Soldani, A. Brogi, Anomaly detection and failure root cause analysis in (micro) service-based cloud applications: A survey, ACM Computing Surveys 55 (2023) 1–39. URL: https://dl.acm.org/doi/10.1145/3501297. doi:10.1145/3501297.

[16] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, S. Linkman, Systematic literature reviews in software engineering – a systematic literature review, Information and Software Technology 51 (2009) 7–15. doi:https://doi.org/10.1016/j.infsof.2008.09.009.

[17] M. Kisley, ”search algorithms in litmaps”, 2024. URL: https://docs.litmaps.com/en/articles/9029858-search-algorithms-in-litmaps?_gl=1*nf2fz2*_gcl_au*ODAyNzc1NzE2LjE3MzU1ODI2Mjk.*_ga*NzEzODgwNjU4LjE3MzU1ODI2Mjk.*_ga_6JT7JKRQ1D*MTczNjI2ODEzMS43LjEuMTczNjI2ODEzNi41NS4wLjA., accessed: 2025-01-12.

[18] D. S. Cruzes, T. Dyba, Recommended steps for thematic synthesis in software engineering, in: 2011 international symposium on empirical software engineering and measurement, IEEE, 2011, pp. 275–284.

[19] A. L. Strauss, J. Corbin, Open coding, Social research methods: A reader (2004) 303–306.

[20] J. Kendall, Axial coding and the grounded theory controversy, Western journal of nursing research 21 (1999) 743–757.

[21] J. Zhu, S. He, P. He, J. Liu, M. R. Lyu, Loghub: A large collection of system log datasets for ai-driven log analytics, 2023. URL: https://arxiv.org/abs/2008.06448. arXiv:2008.06448.

[22] Z. Li, N. Zhao, S. Zhang, Y. Sun, P. Chen, X. Wen, M. Ma, D. Pei, Constructing large-scale real-world benchmark datasets for aiops, arXiv preprint arXiv:2208.03938 (2022).

[23] A. Alsaheel, Y. Nan, S. Ma, L. Yu, G. Walkup, Z. B. Celik, X. Zhang, D. Xu, {ATLAS}: A sequence-based learning approach for attack investigation, in: 30th USENIX security symposium (USENIX security 21), 2021, pp. 3005–3022.

[24] C. Wohlin, A. Rainer, Challenges and recommendations to publishing and using credible evidence in software engineering, Information and Software Technology 134 (2021) 106555. URL: https://www.sciencedirect.com/science/article/pii/S0950584921000409. doi:https://doi.org/10.1016/j.infsof.2021.106555.

Primary Studies

[P1] X. Zhou, X. Peng, T. Xie, J. Sun, C. Ji, D. Liu, Q. Xiang, C. 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, 2019, pp. 683–694.

[P2] K. Aktaş, H. H. Kilinc, Interaction prediction and anomaly detection in a microservices-based telecommunication platform, in: Proceedings of the 2024 International Conference on Software and Systems Processes, 2024, pp. 56–65.

[P3] S. Dodda, S. Chintala, N. Kunchakuri, N. Kamuni, Enhancing microservice reliability in cloud environments using machine learning for anomaly detection, in: 2024 International Conference on Computing, Sciences and Communications (ICCSC), IEEE, NJ, USA, 2024, pp. 1–5. doi:10.1109/ICCSC62048.2024.10830437.

[P4] G. Darwesh, J. Hammoud, A. Vorobeva, Enhancing kubernetes security with machine learning: a proactive approach to anomaly detection, Scientific and Technical Journal of Information Technologies, Mechanics and Optics 24 (2024) 1007–1015. doi:10.17586/2226-1494-2024-24-6-1007-1015.

[P5] D. A. Khudyakov, G. E. Yakhyaeva, Unsupervised anomaly detection on distributed log tracing through deep learning, in: 2024 IEEE 25th International Conference of Young Professionals in Electron Devices and Materials (EDM), IEEE, 2024, pp. 1830–1833.

[P6] Y. Chen, Z.-M. Xiao, F. Teng, A root cause localization method based on event call chains for microservices, in: 2024 16th International Conference on Communication Software and Networks (ICCSN), IEEE, NJ, USA, 2024, pp. 43–48. doi:10.1109/ICCSN63464.2024.10793332.

[P7] S. Gulmez, A. Duman, S. A. Duman, I. Sogukpinar, Log mining-based online failure prediction in client-server architecture, in: 2023 8th International Conference on Computer Science and Engineering (UBMK), IEEE, 2023, pp. 492–497.

[P8] S. Nedelkoski, J. Cardoso, O. Kao, Anomaly detection from system tracing data using multimodal deep learning, in: 2019 IEEE 12th International Conference on Cloud Computing (CLOUD), IEEE, 2019, pp. 179–186.

[P9] C. Zhang, X. Peng, T. Zhou, C. Sha, Z. Yan, Y. Chen, H. Yang, Tracecrl: contrastive representation learning for microservice trace analysis, in: Proceedings of the 30th ACM joint European software engineering conference and symposium on the foundations of software engineering, 2022, pp. 1221–1232.

[P10] R. Rouf, M. Rasolroveicy, M. Litoiu, S. Nagar, P. Mohapatra, P. Gupta, I. Watts, Instantops: A joint approach to system failure prediction and root cause identification in microserivces cloud-native applications, in: Proceedings of the 15th ACM/SPEC International Conference on Performance Engineering, 2024, pp. 119–129.

[P11] K. Zhang, C. Zhang, X. Peng, C. Sha, Putracead: Trace anomaly detection with partial labels based on gnn and pu learning, in: 2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE), IEEE, 2022, pp. 239–250.

[P12] N. Ba-Hung, H. Yabusaki, T. Sagara, A deep graph neural networks approach for service failure analytics, in: 2024 11th International Conference on Future Internet of Things and Cloud (FiCloud), IEEE, NJ, USA, 2024, pp. 298–301. doi:10.1109/FiCloud62933.2024.00053.

[P13] Y. Wang, Z. Zhu, Q. Fu, Y. Ma, P. He, MRCA: Metric-level root cause analysis for microservices via multi-modal data, in: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering, ACM, New York, NY, USA, 2024, pp. 1057–1068. doi:10.1145/3691620.3695485.

[P14] H. Guo, S. Yuan, X. Wu, Logbert: Log anomaly detection via bert, in: 2021 international joint conference on neural networks (IJCNN), IEEE, 2021, pp. 1–8.

[P15] Y. Han, Q. Du, Y. Huang, J. Wu, F. Tian, C. He, The potential of one-shot failure root cause analysis: Collaboration of the large language model and small classifier, in: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering, 2024, pp. 931–943.

[P16] S. Ding, Y. Xu, Z. Lu, F. Tang, T. Li, J. Ge, Power microservices troubleshooting by pretrained language model with multi-source data, in: 2024 IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA), IEEE, NJ, USA, 2024, pp. 1768–1775. doi:10.1109/ISPA63168.2024.00241.

[P17] L. Zheng, Z. Chen, J. He, H. Chen, Mulan: Multi-modal causal structure learning and root cause analysis for microservice systems, in: Proceedings of the ACM on Web Conference 2024, 2024, pp. 4107–4116.

[P18] W. Zhang, Q. Zhang, E. Yu, Y. Ren, Y. Meng, M. Qiu, J. Wang, Lograg: Semi-supervised log-based anomaly detection with retrieval-augmented generation, in: 2024 IEEE International Conference on Web Services (ICWS), IEEE, 2024, pp. 1100–1102.

[P19] N. Wijesinghe, H. Hemmati, Log-based anomaly detection of enterprise software: An empirical study, in: 2023 IEEE 23rd International Conference on Software Quality, Reliability, and Security (QRS), IEEE, 2023, pp. 12–23.

[P20] M. Du, F. Li, G. Zheng, V. Srikumar, Deeplog: Anomaly detection and diagnosis from system logs through deep learning, in: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security, 2017, pp. 1285–1298.

[P21] W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, S. Tao, P. Sun, et al., Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs., in: IJCAI, volume 19, 2019, pp. 4739–4745.

[P22] X. Zhang, Y. Xu, Q. Lin, B. Qiao, H. Zhang, Y. Dang, C. Xie, X. Yang, Q. Cheng, Z. Li, et al., Robust log-based anomaly detection on unstable log data, in: Proceedings of the 2019 27th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering, 2019, pp. 807–817.

[P23] L. Yang, J. Chen, Z. Wang, W. Wang, J. Jiang, X. Dong, W. Zhang, Semi-supervised log-based anomaly detection via probabilistic label estimation, in: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), IEEE, 2021, pp. 1448–1460.

[P24] Q. Du, L. Zhao, F. Tian, Y. Han, Trace-based anomaly detection with contextual sequential invocations, in: International Conference on Database and Expert Systems Applications, Springer, 2023, pp. 113–118.

[P25] I. Kohyarnejadfard, D. Aloise, S. V. Azhari, M. R. Dagenais, Anomaly detection in microservice environments using distributed tracing data analysis and nlp, Journal of Cloud Computing 11 (2022) 25.

[P26] R. Tan, Z. Li, Maad: A distributed anomaly detection architecture for microservices systems, in: 2024 IEEE International Parallel and Distributed Processing Symposium (IPDPS), IEEE, 2024, pp. 1009–1021.

[P27] J. Tang, Q. Hu, Y. Long, W. Wei, G. Zhang, Z. Lin, Algorithm for locating fault linkage sets in end-to-end microservice networks under multiple business scenarios, in: 2024 4th International Conference on Energy Engineering and Power Systems (EEPS), IEEE, NJ, USA, 2024, pp. 1367–1372. doi:10.1109/EEPS63402.2024.10804554.

[P28] W. Xu, L. Huang, A. Fox, D. Patterson, M. I. Jordan, Detecting large-scale system problems by mining console logs, in: Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles, 2009, pp. 117–132.

[P29] Q. Lin, H. Zhang, J.-G. Lou, Y. Zhang, X. Chen, Log clustering based problem identification for online service systems, in: Proceedings of the 38th international conference on software engineering companion, 2016, pp. 102–111.

[P30] C. Streiffer, R. Raghavendra, T. Benson, M. Srivatsa, Learning to simplify distributed systems management, in: 2018 IEEE International Conference on Big Data (Big Data), IEEE, 2018, pp. 1837–1845.

[P31] R. Li, M. Du, Z. Wang, H. Chang, S. Mukherjee, E. Eide, Longtale: Toward automatic performance anomaly explanation in microservices, in: Proceedings of the 2022 ACM/SPEC on International Conference on Performance Engineering, 2022, pp. 5–16.

[P32] C. Almodovar, F. Sabrina, S. Karimi, S. Azad, Logfit: Log anomaly detection using fine-tuned language models, IEEE Transactions on Network and Service Management 21 (2024) 1715–1723.

[P33] W. Guan, J. Cao, S. Qian, J. Gao, C. Ouyang, Logllm: Log-based anomaly detection using large language models, arXiv preprint arXiv:2411.08561 (2024).

[P34] Y. Yamanaka, T. Takahashi, T. Minami, Y. Nakajima, Logelectra: Self-supervised anomaly detection for unstructured logs, arXiv preprint arXiv:2402.10397 (2024).

[P35] S. Munir, H. Ali, J. Qureshi, Log attention–assessing software releases with attention-based log anomaly detection, in: International Conference on Service-Oriented Computing, Springer, 2021, pp. 139–150.

[P36] Z. He, Y. Tang, K. Zhao, J. Liu, W. Chen, Graph-based log anomaly detection via adversarial training, in: International Symposium on Dependable Software Engineering: Theories, Tools, and Applications, Springer, 2023, pp. 55–71.

[P37] L. Chen, Q. Dang, M. Chen, B. Sun, C. Du, Z. Lu, Berthtlg: Graph-based microservice anomaly detection through sentence-bert enhancement, in: International Conference on Web Information Systems and Applications, Springer, 2023, pp. 427–439.

[P38] K. Shi, J. Li, Y. Liu, Y. Chang, X. Li, Bsdg: Anomaly detection of microservice trace based on dual graph convolutional neural network, in: International Conference on Service-Oriented Computing, Springer, 2022, pp. 171–185.

[P39] Y. Wan, Y. Liu, D. Wang, Y. Wen, Glad-paw: Graph-based log anomaly detection by position aware weighted graph attention network, in: Pacific-asia conference on knowledge discovery and data mining, Springer, 2021, pp. 66–77.

[P40] X. Liang, L. Li, H. Peng, Unsupervised microservice log anomaly detection method based on graph neural network, in: International Conference on Swarm Intelligence, Springer, 2024, pp. 197–208.

[P41] H. Wang, Z. Wu, H. Jiang, Y. Huang, J. Wang, S. Kopru, T. Xie, Groot: An event-graph-based approach for root cause analysis in industrial settings, in: 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE), IEEE, 2021, pp. 419–429.

[P42] P. Liu, H. Xu, Q. Ouyang, R. Jiao, Z. Chen, S. Zhang, J. Yang, L. Mo, J. Zeng, W. 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), IEEE, 2020, pp. 48–58.

[P43] C. Zhang, X. Peng, C. Sha, K. Zhang, Z. Fu, X. Wu, Q. Lin, D. Zhang, Deeptralog: Trace-log combined microservice anomaly detection through graph-based deep learning, in: Proceedings of the 44th International Conference on Software Engineering, 2022, pp. 623–634.

[P44] Z. Li, J. Chen, R. Jiao, N. Zhao, Z. Wang, S. Zhang, Y. Wu, L. Jiang, L. Yan, Z. Wang, et al., Practical root cause localization for microservice systems via trace analysis, in: 2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS), IEEE, 2021, pp. 1–10.

[P45] S. Zhang, P. Jin, Z. Lin, Y. Sun, B. Zhang, S. Xia, Z. Li, Z. Zhong, M. Ma, W. Jin, et al., Robust failure diagnosis of microservice system through multimodal data, IEEE Transactions on Services Computing 16 (2023) 3851–3864.

[P46] S. Zhang, D. Fan, L. He, Y. Liu, D. Chen, Twlog: Task workflow-based log anomaly detection, in: Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International

Conference on Web and Big Data, Springer, 2024, pp. 3–16.

[P47] J. Chen, H. Huang, H. Chen, Informer: Irregular traffic detection for containerized microservices rpc in the real world, in: Proceedings of the 4th ACM/IEEE Symposium on Edge Computing, 2019, pp. 389–394.

[P48] C. Duan, T. Jia, Y. Li, G. Huang, Aclog: An approach to detecting anomalies from system logs with active learning, in: 2023 IEEE International Conference on Web Services (ICWS), IEEE, 2023, pp. 436–443.

[P49] C. Duan, T. Jia, H. Cai, Y. Li, G. Huang, Afalog: A general augmentation framework for log-based anomaly detection with active learning, in: 2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE), IEEE, 2023, pp. 46–56.

[P50] Y. Tang, N. Chen, Y. Zhang, X. Yao, S. Yu, Fine-grained diagnosis method for microservice faults based on hierarchical correlation analysis, in: CCF Conference on Computer Supported Cooperative Work and Social Computing, Springer, 2021, pp. 14–28.

[P51] D. Liu, C. He, X. Peng, F. Lin, C. Zhang, S. Gong, Z. Li, J. Ou, Z. Wu, Microhecl: High-efficient root cause localization in large-scale microservice systems, in: 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), IEEE, 2021, pp. 338–347.

[P52] M. Panahandeh, A. Hamou-Lhadj, M. Hamdaqa, J. Miller, Serviceanomaly: An anomaly detection approach in microservices using distributed traces and profiling metrics, Journal of Systems and Software 209 (2024) 111917.

[P53] M. Chen, Z. Dai, Y. Li, Z. Lu, Fedtag: Towards automated attack investigation using federated learning, in: International Artificial Intelligence Conference, Springer, 2023, pp. 112–126.

[P54] P. Wang, X. Zhang, Z. Cao, Z. Chen, Madmm: microservice system anomaly detection via multi-modal data and multi-feature extraction, Neural Computing and Applications (2024) 1–19.

[P55] D. Yi, L. Xuesong, S. Ge, Z. Lei, Z. Lisha, Research on the integrated model construction of business perspective and IT system monitoring, in: 2024 4th International Signal Processing, Communications and Engineering Management Conference (ISPCEM), IEEE, NJ, USA, 2024, pp. 790–795. doi:10.1109/ISPCEM64498.2024.00141.

[P56] L.-z. Chen, J. Wu, H.-y. Yang, K. Zhang, A microservice regression testing selection approach based on belief propagation, Journal of Cloud Computing 12 (2023) 20.

[P57] A. Catovic, C. Cartwright, Y. T. Gebreyesus, S. Ferlin, Linnaeus: A highly reusable and adaptable ml based log classification pipeline, in: 2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN), IEEE, 2021, pp. 11–18.

[P58] Q. Hu, Y. Long, Y. Zhong, G. Zhang, W. Wei, Automated monitoring method for enterprise microservices network operation status based on database knowledge graph, in: 2024 6th International Conference on Electronics and Communication, Network and Computer Technology (ECNCT), IEEE, NJ, USA, 2024, pp. 567–572. doi:10.1109/ECNCT63103.2024.10704355.

[P59] J. Bogatinovski, S. Nedelkoski, J. Cardoso, O. Kao, Self-supervised anomaly detection from distributed traces, in: 2020 IEEE/ACM 13th International Conference on Utility and Cloud

Computing (UCC), IEEE, 2020, pp. 342–347.

[P60] H. Liu, X. Huang, M. Jia, T. Jia, J. Han, Z. Wu, Y. Li, UAC-AD: Unsupervised adversarial contrastive learning for anomaly detection on multi-modal data in microservice systems, IEEE Transactions on Services Computing 17 (2024) 3887–3900. doi:10.1109/TSC.2024.3411481.

[P61] S. Xu, Z. Meng, H. Wang, Research and implementation of log-based anomaly detection platform based on ELK+kafka, in: 2024 10th International Conference on Computer and Communications (ICCC), IEEE, NJ, USA, 2024, pp. 1928–1932. doi:10.1109/ICCC62609.2024.10941763.

[P62] O. I. Sheluhin, D. V. Kostin, M. G. Gorodnichev, Multiclass classification of anomalous states of computer systems by means of intellectual analysis of system journals, Automatic Control and Computer Sciences 54 (2020) 549–559.

[P63] R. Lu, X. Zhu, X. Li, N. Long, G. Zhang, An anomaly monitoring and early warning method for power grid microservice network based on log visualisation and analysis, in: 2024 5th International Conference on Machine Learning and Computer Application (ICMLCA), IEEE, NJ, USA, 2024, pp. 584–590. doi:10.1109/ICMLCA63499.2024.10753853.

[P64] D. Pathak, M. Verma, A. Chakraborty, H. Kumar, Self adjusting log observability for cloud native applications, in: 2024 IEEE 17th International Conference on Cloud Computing (CLOUD), IEEE, 2024, pp. 482–493.

[P65] A. D'Angelo, G. d'Aloisio, Grammar-based anomaly detection of microservice systems execution traces, in: Companion of the 15th ACM/SPEC International Conference on Performance Engineering, 2024, pp. 77–81.

[P66] S. Nedelkoski, J. Cardoso, O. Kao, Anomaly detection and classification using distributed tracing and deep learning, in: 2019 19th IEEE/ACM international symposium on cluster, cloud and grid computing (CCGRID), IEEE, 2019, pp. 241–250.

[P67] F. Hadadi, Q. Xu, D. Bianculli, L. Briand, Anomaly detection on unstable logs with gpt models, arXiv preprint arXiv:2406.07467 (2024).

[P68] J. Nobre, E. S. Pires, A. Reis, Anomaly detection in microservice-based systems, Applied Sciences 13 (2023) 7891.

[P69] Á. Brandón, M. Solé, A. Huélamo, D. Solans, M. S. Pérez, V. Muntés-Mulero, Graph-based root cause analysis for service-oriented and microservice architectures, Journal of Systems and Software 159 (2020) 110432.

[P70] X. Zhu, R. Lu, X. Li, G. Zhang, L. Pan, Methods for correlation analysis of alarm information in multi-microservice application environments, in: 2024 9th International Symposium on Computer and Information Processing Technology (ISCIPT), IEEE, 2024, pp. 699–704.

[P71] M. Li, D. Tang, Z. Wen, Y. Cheng, Microservice anomaly detection based on tracing data using semi-supervised learning, in: 2021 4th International Conference on Artificial Intelligence and Big Data (ICAIBD), IEEE, 2021, pp. 38–44.

[P72] L. Toka, G. Dobreff, D. Haja, M. Szalay, Predicting cloud-native application failures based on monitoring data of cloud infrastructure, in: 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM), IEEE, 2021, pp. 842–847.

[P73] Q. Wang, L. Shwartz, G. Y. Grabarnik, V. Arya, K. Shanmugam, Detecting causal structure on cloud application microservices using granger causality models, in: 2021 IEEE 14th International Conference on Cloud Computing (CLOUD), IEEE, 2021, pp. 558–565.

[P74] P. K. Sarika, D. Badampudi, S. P. Josyula, M. Usman, Automating microservices test failure analysis using kubernetes cluster logs, in: Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering, 2023, pp. 192–195.

[P75] B. H. V. Wahono, I. Mahfud, B. Y. I. Exshadi, A. M. Shiddiqi, et al., Brute force detection system based on machine learning classifier algorithm in cloud-based infrastructure, in: 2024 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS), IEEE, 2024, pp. 939–943.

[P76] X. Liu, Y. Liu, M. Wei, P. Xu, LMGD: Log-metric combined microservice anomaly detection through graph-based deep learning, IEEE Access 12 (2024) 186510–186519. doi:10.1109/ACCESS.2024.3481676.

[P77] N. Kaushik, H. Kumar, V. Raj, P. Garg, Proactive fault prediction in microservices applications using trace logs and monitoring metrics, in: 2024 International Conference on Progressive Innovations in Intelligent Systems and Data Science (ICPIDS), IEEE, NJ, USA, 2024, pp. 410–415. doi:10.1109/ICPIDS65698.2024.00070.

[P78] H. Hussain, G. Abuoda, M. Litoiu, Exploring approaches to integrate performance prediction and anomaly detection in microservices systems, in: 2024 34th International Conference on Collaborative Advances in Software and COmputiNg (CASCON), IEEE, NJ, USA, 2024, pp. 1–4. URL: https://ieeexplore.ieee.org/document/10838219/. doi:10.1109/CASCON62161.2024.10838219.

[P79] B. Jasmin, N. Sasho, Multi-source anomaly detection in distributed it systems, in: International Conference on Service-Oriented Computing, 2020, pp. 201–13.

[P80] R. Della Corte, R. Pietrantuono, Towards log-driven testing through transformers: A preliminary study, in: 2024 19th European Dependable Computing Conference (EDCC), IEEE, 2024, pp. 103–106.

[P81] M. Khanahmadi, A. Shameli-Sendi, M. Jabbarifar, Q. Fournier, M. Dagenais, Detection of microservice-based software anomalies based on opentracing in cloud, Software: Practice and Experience 53 (2023) 1681–1699.

[P82] C. H. Zhang, M. O. Shafiq, A real-time, scalable monitoring and user analytics solution for microservices-based software applications, in: 2022 IEEE International Conference on Big Data (Big Data), IEEE, 2022, pp. 6125–6134.

Appendix A. Research Questions of Related Works

Table A.13 summarizes the research questions addressed by these studies, highlighting their diverse focuses and methodological approaches.

Table A.13: Research Questions in Related Studies

Title & Research Questions	**Paper**
AI Assistants for Incident Lifecycle in a Microservice Environment: A Systematic Literature Review RQ1: What is the current landscape of AI assistants across the incident lifecycle of microservices? RQ2: What methods and techniques are used to implement AI assistants? RQ3: What are the challenges and limitations of current AI assistants?	[10]
On Microservice Analysis and Architecture Evolution: A Systematic Mapping Study RQ1: What methods and techniques are used in microservice analysis? RQ2: What are the problems or opportunities that are addressed using microservice analysis techniques? RQ3: Does microservice analysis overlap with other areas of software analysis, or are new methods or paradigms needed? RQ4: What potential future research directions are open in the area of microservice analysis?	[1]
Monitoring tools for DevOps and microservices: A systematic grey literature review RQ1: What are the main characteristics of monitoring tools for microservice-based systems? RQ2: What information is gathered to characterize the behavior of the monitored system? RQ3: How does the tool implement the monitoring process?	[2]
Landscape of Automated Log Analysis: A Systematic Literature Review and Mapping Study RQ1: What different log analysis goals have been identified in the literature? RQ2: What log types and their attributes have been studied in the literature? RQ3: What data extraction knowledge has been discovered in the analyzed literature?	[5]
Analysis of Log Files to Enable Smart-Troubleshooting in Industry 4.0: A Systematic Mapping Study RQ1: What are the publication trends in analyzing log files for smart-troubleshooting in Industry 4.0? RQ2: Which attributes can be used for describing log files to enable smart-troubleshooting in Industry 4.0? RQ3: What are the main techniques used to analyze log files for smart-troubleshooting?	[6]

Continued on next page

Table A.13: Research Questions in Related Studies

Title & Research Questions	**Paper**
RQ4: What are the main challenges in analyzing log files for smart-troubleshooting?	
Machine Learning for Anomaly Detection: A Systematic Review RQ1: What is the main prediction about research work done in anomaly detection? RQ2: What kinds of ML algorithms are being applied in anomaly detection? RQ3: What is the overall estimation and accuracy of machine learning models? RQ4: What is the percentage of papers that address unsupervised, semi-supervised, or supervised anomaly detection?	[13]
Failure Diagnosis in Microservice Systems: A Comprehensive Survey and Analysis RQ1: What is the granularity of failure diagnosis in modern microservice systems? How to formulate the problem of failure diagnosis? RQ2: What are the most important characteristics and methods that define each failure diagnosis technique? Based on this, how to taxonomize each technique and qualitatively analyze each class? RQ3: Are there publicly available datasets and toolkits for failure diagnosis in the AIOps domain of microservice systems? What are their evaluation metrics?	[8]
Log-based software monitoring: a systematic mapping study RQ1: What are the publication trends in research on log-based monitoring over the years? RQ2: What are the different research scopes of log-based monitoring?	[4]
Multivocal study on microservice dependencies RQ1: What types of dependencies are prevalent in microservice architecture, and how do they impact system design and functionality? RQ2: What techniques and tools are available to detect and manage these dependencies? RQ2.1: Which benchmarks have been used in literature to test and demonstrate dependencies? RQ3: Does the existing literature consider dependencies in addressing software architecture quality attributes?	[12]
AI Techniques in the Microservices Life-Cycle: A Survey RQ1: When? - How has the number of AI4MS publications evolved over the years? RQ2: Who? - Which countries and institutions perform AI4MS research? RQ3: Where? - In which industry domains is AI used for MSs? RQ4: Why? - What is the rationale for using AI to improve MS systems? RQ4.1: Which are the improved quality attributes? RQ4.2: Which are the improved DevOps phases? RQ4.3: Which quality attributes are improved in which DevOps phases? RQ5: How? - Which approaches to AI are used for MSs? RQ6: Which AI4MS challenges need to be addressed by future research?	[9]

Continued on next page

Table A.13: Research Questions in Related Studies

Title & Research Questions	**Paper**
How Industry Tackles Anomalies during Runtime: Approaches and Key Monitoring Parameters RQ1: What are the prevailing interpretations, characteristics, and examples of anomalies within the industry? RQ2: What factors influence the selection of anomaly detection approaches in industrial settings? RQ3: Which runtime monitoring data is used to identify anomalies by industry?	[14]
LLM-based event log analysis techniques: A survey RQ1: What is the developing body of knowledge regarding LLM-based event log analysis techniques? RQ2: What are the commonalities (e.g., methods, models, factors) between existing LLM-based event log analysis works? RQ3: What are the gaps in previous research, key challenges, and potential solutions for further developments in the LLM-based event log analysis domain? RQ4: What are the potential for future research in LLM-based event log analysis?	[11]
A Survey on Automated Log Analysis for Reliability Engineering RQ1: How can logging practices be automated or improved, including where-to-log, what-to-log, and how-to-log? RQ2: What are the approaches to manage raw log volume and structure logs, specifically focusing on log compression and parsing (automatically extracting event templates and key parameters)? RQ3: What automated log mining techniques are employed to enhance system reliability, specifically in the areas of anomaly detection, failure prediction, and failure diagnosis? RQ4: What open-source toolkits and datasets are available to facilitate automated log analysis research? RQ5: What are the best current practices in industry, and what are the promising future directions and open challenges for real-world and next-generation automated log analysis?	[3]
Anomaly Detection and Failure Root Cause Analysis in (Micro) Service-Based Cloud Applications: A Survey RQ1: What are the currently available techniques for detecting anomalies in multi-service/microservice applications? RQ2: What are the existing techniques for identifying the root causes of detected anomalies? RQ3: What are the open challenges and future research directions in the field of microservice anomaly detection and RCA?	[15]

Appendix B. Search Query Development

Search strings were built in an iterative process throughout their development in Table B.14. Table B.14 utilizes differential syntax between Sets #2 - #7 to explain the changes from the previous iteration without rewriting the same keywords and allows for better representation of how differences change the total number of papers gathered. The syntax is explained as follows: ”///” means that the section remains the same from the previous iteration with no changes. ”+++keywords+++” means that the section contains an addition of keywords from the previous iteration. ”—keywords—” means that the section contains a subtraction of keywords from the previous iteration. The sections that do not utilize this syntax are Sets #1 and #8 as they are the first and final iteration.

Each iteration of the search string developed the review’s focus on what is relevant literature. The goal of a search string is to gather as much of the relevant literature as possible while minimizing the amount of irrelevant literature. The initial set – Set #1 in Table B.14 – was created out of an educated guess from initially developed research questions. Three major ideas emerged out of this initial Set #1: Microservices, Log Analysis, and Natural Language Processing (NLP). These three major points are what drive the future iterations of the search string. These major points are separated by an ”AND” statement, as each of these three points must be a part of the paper to be considered for the review process.

The initial set was an ideal start but lacked the amount of results and focus on log/trace analysis that was needed for the review. The set needed to be more inclusive of the idea of logs and log analysis; thus, Set #2 in Table B.14 added more related keywords to logging, such as ”Log Management” and ”Log*”. This proved to be a somewhat correct step as the results expanded, but some results did not relate to machine learning as was hoped, and too many nuances arose from using ”Log*”. The next set – Set #3 in Table B.14 – sought to address this lack of machine learning tools and correct the irrelvancy from using ”Log*”, however, when adding ”ML” to the section related to AI and ”Logging” and ”Logs”, the results grew to an unacceptable amount for the review. The next iteration would have to get more specific and remove some keywords that were being utilized to get better results with less irrelevancy.

Table B.14: Search Strings

Set	Search String	# of Papers
#1	(”Microservice” OR ”Micro*service*” OR ”Cloud-native”) AND (”Log Analysis” OR ”Log Mining”) AND (”NLP” OR ”Natural Language Processing” OR ”LLM”)	414
#2	(///) AND (+++OR ”Log Management” OR ”Log*”+++) AND (///)	2736
#3	(///) AND (—OR ”Log*”—, +++OR ”Logging” OR ”Logs”+++) AND (+++OR ”ML”+++)	6005
#4	(///) AND (—OR ”Logging” OR ”Logs”—) AND (—OR ”ML”—, +++OR ”Large Lanauge Model”+++)	498
#5	(+++OR ”Cloud*compute*”+++) AND (+++”Logging” OR ”Observability” OR ”Monitoring” OR ”Log Monitoring”+++) AND (///)	3869
#6	(///) AND (+++OR ”Trace analysis” OR ”Tracing”+++) AND (+++OR ”Machine Learning” or ”ML” OR ”Artificial Intelligence” OR ”AI”+++)	2608
#7	(///) AND (+++OR ”Log*” OR “Telemetry”+++) AND (///)	2649
#8	(++++OR ”Cloud*comput”) AND (///) AND (///)	2208
#8 (Final query)	(”Microservice” OR ”Micro*service*” OR ”Cloud*native” OR ”Cloud*comput*”) AND (”Log Analysis” OR ”Log Mining” OR ”Log Management” OR ”Log*” OR ”Observability” OR ”Monitoring” OR ”Log Monitoring” OR ”Trace analysis” OR ”Tracing” OR ”Telemetry”) AND (”NLP” OR ”Natural Language Processing” OR ”Large Language Model” OR ”LLM” OR ”Machine Learning” or ”ML” OR ”Artificial Intelligence” OR ”AI”)	2208
#8 (Springer-Link)	(”Microservice” OR ”Micro?service” OR ”Cloud?native” OR ”Cloud?comput”) AND (”Log Analysis” OR ”Log Mining” OR ”Log Management” OR ”Log” OR ”Logging” OR ”Observability” OR ”Monitoring” OR ”Log Monitoring” OR ”Trace analysis” OR ”Tracing” OR ”Telemetry”) AND (”NLP” OR ”Natural Language Processing” OR ”Large Language Model” OR ”LLM” OR ”Machine Learning” or ”ML” OR ”Artificial Intelligence” OR ”AI”)	2208

Set #4 in Table B.14 removes most of the previous iterations’ added keywords to reduce the scope. The set adds the keyword ”Large Language Model” in addition to the abbreviation ”LLM” to get more specific results relating to that specific AI tool. This set, however, gives a similar number to Set #1, but more of these results are relevant to the review. This is a signal that this is the correct path, and keywords should be added in the next iteration instead of reducing scope. Set #5 in Table B.14 adds more keywords with respect to logging/tracing and cloud computing to allow the inclusion of more studies. Results were more accurate than that of set #3 when inclusion was last expanded, but it still contained unacceptable amounts of irrelevant studies with a lack of AI tools in the studies. This signaled that an expansion of the AI tool keywords was needed with the supplementation of trace analysis in the logging topic.

Set #6 in Table B.14 attempts to include more relevant literature to the results by adding more keywords that are related to logging and AI. It adds in trace analysis as well as expansions into common terms for AI to expand the search for more valid literature and reduce the amount of literature that is irrelevant. The next Set #7 in Table B.14 is a minor revision of Set #6; the results of this revision were still in acceptable margins for relevance and amount of literature with little change in the number of resulting studies. There was some expansion in the area of the string regarding logging and tracing analysis with the addition of "Log*" and Telemetry".

Set #8 and the SpringerLink specific variant in Table B.14 are the same as set #7 but queried by the indexers at different times. This allowed more studies to be added into consideration as more results were published between the time of set #7 and set #8. All sets are searched between the dates of 2024, September 12, and 2024, October 31. Set #7 was queried on 1 October 2024, and set #8 was queried on 20 October 2024. The amount of time that had passed made it necessary to re-query results.

Sets #1 through #5 in Table B.14 were not searched with SpringerLink, and Google Scholar was utilized instead. Google Scholar was replaced in favor of SpringerLink after set #5 as results are more relevant and easier to replicate. Set #8 is the final settled search string for further literary review and analysis. Set #8, however, needed to be redefined separately for SpringerLink as the indexer syntax of set #8 without readjustment resulted in a large amount of unrelated papers. This can be identified by the decrease in the total number of papers from the final version of set #8 and the SpringerLink-specific version in Table B.14.

SpringerLink had issues beyond that of the search string needing modification. The SprinerLink indexer had updated their indexing web interface, which no longer included mass exportation of citations of a search results. This caused the need to utilize the older web interface, which is hidden; this interface allows mass exportation critical to the systematic literature review. However, mass exportation is limited to just 1000 results, while the results gathered from SpringerLink were 1260. The strategy to overcome this issue was to limit the search query by year, which lowered the results gathered to less than 1000. The search results less than 1000 were extracted into their own spreadsheet and then stitched back together utilizing a Python script tool (merge.py)[32] to one master spreadsheet. However, exported citations to the search results did not include the abstract – which is important to study identification steps defined in Section 4.4. This final issue with SpringerLink was resolved with another Python script tool (abstract_-scrape.py)[33] that scraped the abstracts based on links to the SpringerLink entries. There were few errors identified where the abstracts could not be scraped, but inspection of the errors determined results with lack of abstracts to be irrelevant on manual search of error entries. After this process, the review is left with results from SpringerLink that could be used to identify studies of interest.

The total given from the final search string and fix for SpringerLink revealed 1924 pieces of literature that were close to the topic of the review. The most amount of literature resulted was given by SpringerLink with 1260, while every other indexer individually resulted in less than 400. The time of publication for each piece of literature was not limited by the search query, but it is noted that most studies were published after 2018.

Appendix C. Identified Primary Studies

Identified Primary Studies appear in Table C.15.

Table C.15: Primary Literature Results

Title	**Year**	**Reference**
TraceCRL: contrastive representation learning for microservice trace analysis	2022	[P9]
InstantOps: A Joint Approach to System Failure Prediction and Root Cause Identification in Microserivces Cloud-Native Applications	2024	[P10]
Interaction Prediction and Anomaly Detection in a Microservices-based Telecommunication Platform	2024	[P2]
Latent error prediction and fault localization for microservice applications by learning from system trace logs	2019	[P1]

Continued on next page

[32]https://zenodo.org/records/15127457

[33]https://zenodo.org/records/15127457

Table C.15: Primary Literature Results

Title	**Year**	**Reference**
The Potential of One-Shot Failure Root Cause Analysis: Collaboration of the Large Language Model and Small Classifier	2024	[P15]
Self Adjusting Log Observability for Cloud Native Applications	2024	[P64]
Automating Microservices Test Failure Analysis using Kubernetes Cluster Logs	2023	[P74]
Anomaly Detection in Microservice-Based Systems	2023	[P68]
MULAN: Multi-modal Causal Structure Learning and Root Cause Analysis for Microservice Systems	2024	[P17]
Brute Force Detection System Based on Machine Learning Classifier Algorithm in Cloud-Based Infrastructure	2024	[P75]
Grammar-Based Anomaly Detection of Microservice Systems Execution Traces	2024	[P65]
Detection of microservice-based software anomalies based on OpenTracing in cloud	2023	[P81]
Anomaly detection in microservice environments using distributed tracing data analysis and NLP	2022	[P25]
LongTale: Toward Automatic Performance Anomaly Explanation in Microservices	2022	[P31]
Informer: Irregular traffic detection for containerized microservices RPC in the real world	2022	[P47]
Detecting Causal Structure on Cloud Application Microservices Using Granger Causality Models	2021	[P73]
Learning to Simplify Distributed Systems Management	2018	[P30]
Graph-based root cause analysis for service-oriented and microservice architectures	2020	[P69]
MicroHECL: High-Efficient Root Cause Localization in Large-Scale Microservice Systems	2021	[P51]
A Real-time, Scalable Monitoring and User Analytics Solution for Microservices-based Software Applications	2022	[P82]
Microservice Anomaly Detection Based on Tracing Data Using Semi-supervised Learning	2021	[P71]
LogRAG: Semi-Supervised Log-based Anomaly Detection with Retrieval-Augmented Generation	2024	[P18]
Unsupervised Anomaly Detection on Distributed Log Tracing through Deep Learning	2024	[P5]

Continued on next page

Table C.15: Primary Literature Results

Title	**Year**	**Reference**
MAAD: A Distributed Anomaly Detection Architecture for Microservices Systems	2024	[P26]
Methods for Correlation Analysis of Alarm Information in Multi-Microservice Application Environments	2024	[P70]
Practical Root Cause Localization for Microservice Systems via Trace Analysis	2021	[P44]
Log Mining-based Online Failure Prediction in Client-Server Architecture	2023	[P7]
Linnaeus: A highly reusable and adaptable ML based log classification pipeline	2021	[P57]
Towards Log-driven Testing through Transformers: A Preliminary Study	2024	[P80]
AcLog: An Approach to Detecting Anomalies from System Logs with Active Learning	2023	[P48]
Groot: An Event-graph-based Approach for Root Cause Analysis in Industrial Settings	2021	[P41]
AFALog: A General Augmentation Framework for Log-based Anomaly Detection with Active Learning	2023	[P49]
Predicting cloud-native application failures based on monitoring data of cloud infrastructure	2021	[P72]
Log-based Anomaly Detection of Enterprise Software: An Empirical Study	2023	[P19]
PUTraceAD: Trace Anomaly Detection with Partial Labels based on GNN and PU Learning	2022	[P11]
Anomaly Detection from System Tracing Data Using Multimodal Deep Learning	2019	[P8]
Self-Supervised Anomaly Detection from Distributed Traces	2020	[P59]
BertHTLG: Graph-Based Microservice Anomaly Detection Through Sentence-Bert Enhancement	2023	[P37]
A microservice regression testing selection approach based on belief propagation	2023	[P56]
FedTag: Towards Automated Attack Investigation Using Federated Learning	2024	[P53]
MADMM: microservice system anomaly detection via multi-modal data and multi-feature extraction	2024	[P54]
Log Attention – Assessing Software Releases with Attention-Based Log Anomaly Detection	2022	[P35]
Unsupervised Microservice Log Anomaly Detection Method Based on Graph Neural Network	2024	[P40]
TWLog: Task Workflow-Based Log Anomaly Detection	2024	[P46]

Continued on next page

Table C.15: Primary Literature Results

Title	**Year**	**Reference**
Fine-Grained Diagnosis Method for Microservice Faults Based on Hierarchical Correlation Analysis	2022	[P50]
BSDG: Anomaly Detection of Microservice Trace Based on Dual Graph Convolutional Neural Network	2022	[P38]
Multi-source Anomaly Detection in Distributed IT Systems	2021	[P79]
Multiclass Classification of Anomalous States of Computer Systems by Means of Intellectual Analysis of System Journals	2020	[P62]
Graph-Based Log Anomaly Detection via Adversarial Training	2024	[P36]
Trace-Based Anomaly Detection with Contextual Sequential Invocations	2023	[P24]
Robust log-based anomaly detection on unstable log data	2019	[P22]
Robust Failure Diagnosis of Microservice System Through Multimodal Data	2023	[P45]
Anomaly Detection and Classification using Distributed Tracing and Deep Learning	2019	[P66]
LogAnomaly: Unsupervised Detection of Sequential and Quantitative Anomalies in Unstructured Logs	2019	[P21]
Unsupervised Detection of Microservice Trace Anomalies through Service-Level Deep Bayesian Networks	2020	[P42]
DeepTraLog: trace-log combined microservice anomaly detection through graph-based deep learning	2022	[P43]
DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning	2017	[P20]
ServiceAnomaly: An anomaly detection approach in microservices using distributed traces and profiling metrics	2024	[P52]
LogBERT: Log Anomaly Detection via BERT	2021	[P14]
LogFiT: Log Anomaly Detection Using Fine-Tuned Language Models	2024	[P32]
Anomaly Detection on Unstable Logs with GPT Models	2024	[P67]
LogLLM: Log-based Anomaly Detection Using Large Language Models	2024	[P33]
LogELECTRA: Self-supervised Anomaly Detection for Unstructured Logs	2024	[P34]
Detecting large-scale system problems by mining console logs	2019	[P28]
Log clustering based problem identification for online service systems	2016	[P29]
GLAD-PAW: graph-based log anomaly detection by position aware weighted graph attention network	2021	[P39]

Continued on next page

Table C.15: Primary Literature Results

Title	**Year**	**Reference**
Semi-Supervised Log-Based Anomaly Detection via Probabilistic Label Estimation	2021	[P23]
Enhancing Kubernetes security with machine learning: a proactive approach to anomaly detection	2024	[P4]
Automated Monitoring Method for Enterprise Microservices Network Operation Status Based on Database Knowledge Graph	2024	[P58]
Power Microservices Troubleshooting by Pretrained Language Model with Multi-source Data	2024	[P16]
An Anomaly Monitoring and Early Warning Method for Power Grid Microservice Network Based on Log Visualisation and Analysis	2024	[P63]
LMGD: Log-Metric Combined Microservice Anomaly Detection Through Graph-Based Deep Learning	2024	[P76]
MRCA: Metric-level Root cause Analysis for microservices via multi-modal data	2024	[P13]
Research on the Integrated Model Construction of Business Perspective and IT System Monitoring	2024	[P55]
Algorithm for Locating Fault Linkage Sets in End-to-End Microservice Networks Under Multiple Business Scenarios	2024	[P27]
A Deep Graph Neural Networks Approach for Service Failure Analytics	2024	[P12]
Enhancing Microservice Reliability in Cloud Environments Using Machine Learning for Anomaly Detection	2024	[P3]
Proactive Fault Prediction in Microservices Applications Using Trace Logs and Monitoring Metrics	2024	[P77]
UAC-AD: Unsupervised Adversarial Contrastive Learning for Anomaly Detection on Multi-Modal Data in Microservice Systems	2024	[P60]
A Root Cause Localization Method Based on Event Call Chains for Microservices	2024	[P6]
Research and Implementation of Log-Based Anomaly Detection Platform Based on ELK+Kafka	2024	[P61]
Exploring Approaches to Integrate Performance Prediction and Anomaly Detection in Microservices Systems	2024	[P78]

Appendix D. Detailed Analysis of AI++ Benefits

Appendix D.1. Overall Summary of AI++ Benefits

Table D.16 provides a summary of the major benefit themes identified across primary studies. The distribution highlights the prevalence of each benefit across the research landscape.

Table D.16: Frequency of AI++ Benefits in Reviewed Studies

Benefit Theme	**References**
Performance & Accuracy	[P10], [P68], [P17], [P25], [P47], [P30], [P5], [P7], [P48], [P11], [P8], [P59], [P37], [P56], [P54], [P40], [P46], [P38], [P79], [P36], [P66], [P21], [P42], [P43], [P20], [P14], [P39], [P15], [P18], [P32], [P33], [P2], [P1], [P71], [P70], [P81], [P49], [P50], [P62], [P52], [P31], [P28], [P73], [P51], [P26], [P44], [P41], [P19], [P53], [P35], [P24], [P22], [P45], [P34], [P29], [P23], [P3], [P16], [P63], [P76], [P78]
Operational adaptability	[P15], [P18], [P32], [P67], [P17], [P25], [P47], [P7], [P48], [P11], [P8], [P59], [P56], [P46], [P38], [P79], [P36], [P66], [P21], [P42], [P43], [P20], [P14], [P1], [P71], [P82], [P57], [P31], [P75], [P51], [P44], [P41], [P24], [P22], [P34], [P29], [P23]
Real-time processing efficiency	[P2], [P1], [P70], [P82], [P57], [P49], [P72], [P52], [P74], [P65], [P31], [P28], [P75], [P64], [P73], [P51], [P26], [P44], [P41], [P19], [P53], [P35], [P22], [P45], [P34], [P77], [P4]
Architectural advantages	[P9], [P10], [P5], [P7], [P48], [P11], [P8], [P59], [P37], [P56], [P54], [P40], [P46], [P38], [P79], [P36], [P66], [P21], [P42], [P43], [P20], [P14], [P39], [P27], [P12], [P60], [P6], [P13]
Enahanced operational visibility	[P1], [P72], [P50], [P62], [P52], [P65], [P31], [P64], [P73], [P44], [P41], [P35], [P55], [P61], [P58]

Appendix D.2. Benefit Themes by AI++ Method Type

Table D.17 breaks down the distribution of benefit themes by AI++ method type (e.g., ML, DL, GNN, LLM, Hybrid). This view helps illustrate how different AI techniques contribute to specific benefits.

Table D.17: Distribution of Benefit Themes by AI Technique (Compact View)

Benefit Theme	**References (with AI Technique)**
Performance & Accuracy	(NN) [P10, P17, P25, P47, P30, P5, P7, P48, P11, P8, P59, P37, P56, P54, P40, P46, P38, P79, P36, P66, P21, P42, P43, P20, P14, P39, P68, P76]; (LLM) [P15, P18, P32, P33]; (ML) [P2, P1, P71, P70, P81, P49, P50, P62, P52, P31, P28, P3]; (Mixed) [P73, P51, P26, P44, P41, P19, P53, P35, P24, P22, P45, P34, P29, P23, P16, P63, P78]
Operational adaptability	(NN) [P17, P25, P47, P7, P48, P11, P8, P59, P56, P46, P38, P79, P36, P66, P21, P42, P43, P20, P14]; (LLM) [P15, P18, P32, P67]; (ML) [P1, P71, P82, P57, P31]; (Mixed) [P75, P51, P44, P41, P24, P22, P34, P29, P23]
Real-time processing efficiency	(ML)[P4]; (LLM) [P2, P1, P70, P82, P57, P49, P72, P52, P74, P65, P31, P28]; (Mixed) [P75, P64, P73, P51, P26, P44, P41, P19, P53, P35, P22, P45, P34, P29, P23, P77]
Architectural advantages	(NN) [P9, P10, P5, P7, P48, P11, P8, P59, P37, P56, P54, P40, P46, P38, P79, P36, P66, P21, P42, P43, P20, P14, P39, P60, P6, P13]; (Mixed) [P53, P24, P12, P27]
Enahanced operational visibility	(ML) [P1, P72, P50, P62, P52, P31]; (NN) [P61, P55]; (Mixed) [P64, P73, P44, P41, P35, P58]

Appendix D.3. Benefit Themes by Use Case

Table D.18 maps each benefit theme to its corresponding use cases, helping contextualize how AI++ approaches impact different tasks such as anomaly detection, root cause analysis, fault diagnosis, and dependency analysis.

Table D.18: Benefit Themes Mapped to Use Cases

Benefit Theme	**Use Case and References**
Performance & accuracy	Anomaly Detection: [P68, P81, P25, P47, P30, P51, P71, P5, P26, P70, P48, P49, P19, P11, P8, P59, P37, P54, P35, P40, P46, P38, P79, P62, P36, P24, P45, P66, P21, P43, P20, P52, P14, P34, P28, P29, P39, P23, P3, P16, P63, P76, P78]; Root Cause Analysis: [P10, P17, P31, P73, P44, P41, P50, P22, P16, P78]; Fault Diagnosis / Error Prediction: [P1, P7]; Interaction / Dependency Analysis: [P2]; Other: [P56, P53]
Real-time processing efficiency	Anomaly Detection: [P64, P65, P51, P82, P26, P70, P49, P19, P35, P45, P52, P34, P28, P29, P77, P4]; Root Cause Analysis: [P74, P31, P73, P44, P57, P41, P72, P77]; Fault Diagnosis / Error Prediction: [P1]; Interaction / Dependency Analysis: [P2]; Other: [P75, P53]
Operational adaptability	Anomaly Detection: [P68, P25, P47, P51, P82, P71, P18, P48, P11, P8, P59, P46, P38, P79, P36, P24, P66, P21, P43, P20, P14, P32, P67, P33, P34, P29, P23]; Root Cause Analysis: [P15, P17, P31, P44, P57, P41, P22]; Fault Diagnosis / Error Prediction: [P1, P7]; Other: [P75, P56]
Architectural advantages	Anomaly Detection: [P9, P5, P48, P11, P8, P59, P37, P54, P40, P46, P38, P79, P24, P66, P21, P43, P20, P14, P39, P27, P60]; Root Cause Analysis: [P10, P27, P6, P13]; Fault Diagnosis / Error Prediction: [P7, P12]; Other: [P56, P53]
Enahanced operational visibility	Anomaly Detection: [P64, P65, P35, P62, P52, P55, P61, P58]; Root Cause Analysis: [P31, P73, P44, P41, P72, P50, P55]; Fault Diagnosis / Error Prediction: [P1, P58]

Continued on next page

Table D.19: Frequency of AI++ Challenges in Reviewed Studies

Challenge Theme	**Studies Citing & References**
Data Limitations & Dependencies (46 studies)	[P9], [P2], [P1], [P74], [P68], [P31], [P73], [P51], [P82], [P71], [P5], [P70], [P44], [P7], [P57], [P41], [P49], [P19], [P53], [P35], [P40], [P46], [P50], [P79], [P62], [P24], [P45], [P66], [P43], [P20], [P52], [P32], [P67], [P34], [P28], [P29], [P23], [P27], [P12], [P61], [P4], [P58], [P16], [P63], [P13], [P78]
Reliability & Accuracy Issues (42 studies)	[P10], [P1], [P64], [P68], [P65], [P81], [P31], [P47], [P73], [P30], [P51], [P71], [P44], [P7], [P48], [P41], [P11], [P8], [P59], [P56], [P53], [P54], [P40], [P46], [P79], [P62], [P36], [P66], [P21], [P43], [P20], [P14], [P28], [P39], [55], [P12], [P3], [P60], [P61], [P58], [P76], [P13]
Performance & Resource Constraints (41 studies)	[P9], [P10], [P1], [P15], [P64], [P68], [P17], [P75], [P31], [P47], [P73], [P51], [P82], [P18], [P57], [P48], [P41], [P72], [P19], [P11], [P8], [P59], [P53], [P35], [P40], [P50], [P79], [P24], [P45], [P66], [P43], [P20], [P14], [P67], [P33], [P34], [P28], [P23], [P16], [P76], [P13]
Cross-environment Portability and Security Risks (28 studies)	[P2], [P1], [P74], [P75], [P65], [P81], [P31], [P47], [P73], [P51], [P82], [P26], [P70], [P44], [P57], [P41], [P49], [P72], [P19], [P53], [P24], [P22], [P45], [P52], [P29], [P23], [P4], [P76]
Enterprise Integration Complexities (10 studies)	[P15], [P67], [P33], [55], [P77], [P6], [P4], [P58], [P16], [P63]

Table D.20: Distribution of Challenges across AI++ Method Types (with percentage of papers in each method category)

Challenge Theme	**ML (n=30)**	**DL (n=24)**	**GNN (n=17)**	**LLM (n=11)**	**Hybrid (n=3)**	**Other (n=2)**	**References**
Performance & Resource Constraints	13 (43%)	8 (33%)	9 (53%)	8 (73%)	3 (100%)	1 (50%)	**ML**: [P1], [P64], [P68], [P75], [P31], [P73], [P51], [P82], [P57], [P72], [P50], [P28], [P23] **DL**: [P48], [P8], [P59], [P24], [P66], [P20], [P76], [P13] **GNN**: [P9], [P10], [P17], [P47], [P11], [P40], [P45], [P43], [P76] **LLM**: [P15], [P18], [P35], [P79], [P14], [P67], [P33], [P34] **Hybrid**: [P19], [P53], [P16] **Other**: [P41]
Data Limitations & Dependencies	21 (70%)	12 (50%)	6 (35%)	6 (55%)	3 (100%)	1 (50%)	**ML**: [P2], [P1], [P74], [P68], [P31], [P73], [P51], [P82], [P71], [P70], [P44], [P57], [P50], [P62], [P52], [P28], [P29], [P23], [P27], [P4], [P78] **DL**: [P5], [P7], [P49], [P24], [P66], [P20] [P27] [P12], [P61], [P58], [P63], [P13] **GNN**: [P9], [P40], [P45], [P43] [P12], [P78] **LLM**: [P35], [P46], [P79], [P32], [P67], [P34] **Hybrid**: [P19], [P53], [P16] **Other**: [P41]
Reliability & Accuracy Issues	14 (47%)	13 (54%)	10 (59%)	3 (27%)	1 (33%)	2 (100%)	**ML**: [P1], [P64], [P68], [P81], [P31], [P73], [P30], [P51], [P71], [P44], [P56], [P62], [P28], [P3] **DL**: [P7], [P48], [P8], [P59], [P66], [P21], [P20], [P12], [P60], [P61], [P58], [P76], [P13] **GNN**: [P10], [P47], [P11], [P54], [P40], [P36], [P43], [P39] [P12], [P76] **LLM**: [P46], [P79], [P14] **Hybrid**: [P53] **Other**: [P65], [P41]
Enterprise Integration Complexities	2 (7%)	5 (21%)	0 (0%)	3 (27%)	1 (33%)	0 (0%)	**ML**: [P77], [P4] **DL**: [P55], [P77], [P6], [P58], [P63] **LLM**: [P15], [P67], [P33] **Hybrid**: [P16]
Cross-environment Portability & Security Risks	18 (60%)	4 (17%)	3 (18%)	0 (0%)	2 (67%)	2 (100%)	**ML**: [P2], [P1], [P74], [P75], [P81], [P31], [P73], [P51], [P82], [P26], [P70], [P44], [P57], [P72], [P52], [P29], [P23], [P4] **DL**: [P49], [P24], [P22], [P76] **GNN**: [P47], [P45], [P76] **Hybrid**: [P19], [P53] **Other**: [P65], [P41]

Table D.21: Distribution of challenges across Use Cases (with percentage of papers in each use case)

Challenge Theme	**AD (n=59)**	**RCA (n=20)**	**FD&EP (n=2)**	**I&DA (n=1)**	**Other (n=5)**	**References**
Performance & Resource Constraints	28 (47%)	11 (55%)	1 (50%)	0 (0%)	2 (40%)	**Anomaly Detection**: [P9], [P64], [P68], [P47], [P51], [P82], [P18], [P48], [P19], [P11], [P8], [P59], [P35], [P40], [P79], [P24], [P45], [P66], [P43], [P20], [P14], [P67], [P33], [P34], [P28], [P23], [P16], [P76] **RCA**: [P10], [P15], [P17], [P31], [P73], [P57], [P41], [P72], [P50], [P16], [P13] **Fault Prediction**: [P1] **Other**: [P75], [P53]
Data Limitations & Dependencies	33 (56%)	11 (55%)	2 (100%)	1 (100%)	2 (40%)	**Anomaly Detection**: [P9], [P68], [P51], [P82], [P71], [P5], [P70], [P49], [P19], [P35], [P40], [P46], [P79], [P62], [P24], [P45], [P66], [P43], [P20], [P52], [P32], [P67], [P34], [P28], [P29], [P23], [P27], [P61], [P4], [P58], [P16], [P63], [P78] **RCA**: [P74], [P31], [P73], [P44], [P57], [P41], [P50], [P27], [P16], [P13], [P78] **Fault Prediction**: [P1], [P7] **Dependency Analysis** : [P2] **Other**: [P53], [P12]
Reliability & Accuracy Issues	31 (53%)	7 (35%)	2 (100%)	0 (0%)	3 (60%)	**Anomaly Detection**: [P64], [P68], [P65], [P81], [P47], [P30], [P51], [P71], [P48], [P11], [P8], [P59], [P54], [P40], [P46], [P79], [P62], [P36], [P66], [P21], [P43], [P20], [P14], [P28], [P39] [P55], [P3], [P60], [P61], [P58], [P76] **RCA**: [P10], [P31], [P73], [P44], [P41], [P55], [P13] **Fault Prediction**: [P1], [P7] **Other**: [P56], [P53], [P12]
Enterprise Integration Complexities	8 (14%)	5 (25%)	0 (0%)	0 (0%)	0 (0%)	**Anomaly Detection**: [P67], [P33], [P55], [P77], [P4], [P58], [P16], [P63] **RCA**: [P15], [P55], [P77], [P6], [P16]

Continued on next page

Continued from previous page

Challenge Type	**AD (n=59)**	**RCA (n=30)**	**FD&EP (n=2)**	**I&DA (n=1)**	**Other (n=5)**	**References**
Cross-environment Portability & Security Risks	16 (27%)	8 (40%)	1 (50%)	1 (100%)	2 (40%)	**Anomaly Detection**: [P65], [P81], [P47], [P51], [P82], [P26], [P70], [P49], [P19], [P24], [P45], [P52], [P29], [P23], [P4], [P76] **RCA**: [P74], [P31], [P73], [P44], [P57], [P41], [P72], [P22] **Fault Prediction**: [P1] **Dependency Analysis** : [P2] **Other**: [P75], [P53]

Appendix E. Detailed Analysis of Tools used in Anomaly Detection

Table E.22 provides details about tools that utilize AI++ techniques to perform anomaly detection (AD). We summarize the details since a predominent set of studies (72%) focused on AD.

Table E.22: **AI-Based Tools for Anomaly Detection in Microservice Systems**

Tool	**AD Technique**	**Key Insight**	**Benefits & Challenges**	**Refs.**
LogPCA	PCA + Statistical Analysis	Applies PCA and regression for outlier detection and causal reasoning in HDFS and datacenter logs	Interpretable and efficient; scales to tens of millions of lines; sensitive to log quality	[P28]
LogCluster	Clustering	Uses clustering on structured log templates for anomaly detection	Interpretable; relies on log parsing quality	[P29]
Minerva	Statistical & Dimensionality Reduction	Statistical analysis for outlier detection and causal reasoning	Interpretable and efficient; real-time distributed monitoring	[P30]
LongTale	Regression & Dimensionality Reduction	Applies regression for outlier detection with distributed monitoring	Efficient for real-time detection	[P31]
DeepLog	LSTM sequence modeling	Learns sequential log patterns for HDFS, OpenStack, and security logs using unsupervised learning	Simple pipeline with online adaptation; template reliance may flag evolving patterns	[P20]
LogAnomaly	LSTM-based	Sequential log pattern learning using LSTMs; robust to noise and instability	Handles noisy logs well; computational overhead	[P21]
LogRobust	LSTM-based	Sequential learning with attention mechanisms; robust to noise and instability	Strong robustness; complexity in training	[P22]

Continued on next page

Table E.22 (continued): AI-Based Tools for Anomaly Detection in Microservice Systems

Tool	**AD Technique**	**Key Insight**	**Benefits & Challenges**	**Refs.**
PLELog	Attention-GRU + Partial Label Ensemble	Semi-supervised AD using historical anomalies without heavy labeling on HDFS and BGL	Robust to unstable logs; clustering plus GRU adds computational overhead	[P23]
TICAD	LSTM-based	Models execution workflows and temporal semantics for trace-based AD	Captures temporal patterns; moderate complexity	[P24]
LTTng-LSTM	LSTM-based	Learns sequential log patterns using LSTMs and attention mechanisms	Robust to noise; requires labeled data	[P25]
BiLSTM-ATT	Bi-LSTM + Attention with Drain parsing	Real-time AD for power grid microservices with bi-directional dependencies; 95% precision	High precision with bi-directional modeling; parser dependence and drift issues	[P63]
MAAD	CNN-based	Uses CNNs for feature extraction from logs and metrics; adaptive under concept drift	Enables distributed and real-time detection; resilient to evolving behavior	[P26]
Fault Linkage Locator	CNN-based	CNN-based feature extraction for distributed fault detection	Real-time detection capability	[P27]
Event Call Chain RCA	CNN-based with Drain3 parsing	Extracts trace and log events using Drain3; trains LSTM on contextual relationships	Improves root cause localization by 18-20% with faster positioning; parsing dependency	[P6]
LogBERT	BERT with self-supervised learning	Template-free log-sequence AD on HDFS, BGL, and Thunderbird using masked span prediction	Strong performance across datasets without templates; token/sequence length costs	[P14]
LogFiT	Fine-tuned RoBERTa/Longformer	Semantic log AD without templates; adapts to changing log formats across varied environments	Strong F1 with variability handling and no labels; high computational cost	[P32]
LogLLM	BERT encoder + LLaMA classifier	Integrates LLMs with structural log encoding for multi-tenant platforms on unstable logs	Best F1 with robust semantics; highest computational cost with labeling needs	[P33]
LogELECTRA	ELECTRA with replaced token detection	Parser-free point-anomaly detection using ELECTRA pretraining for noisy and evolving logs	Real-time detection without parsing; requires quality normal logs with transformer cost	[P34]
LogAttention	Transformer with Attention	Fine-tunes transformers for contextual log understanding with high generalization	High generalization with limited labels; computational requirements	[P35]

Continued on next page

Table E.22 (continued): AI-Based Tools for Anomaly Detection in Microservice Systems

Tool	**AD Technique**	**Key Insight**	**Benefits & Challenges**	**Refs.**
Log2Vec	Unsupervised Embeddings	Unsupervised embeddings to capture semantic similarities across log events	Captures semantic similarities well; requires sufficient training data	[P35]
LogRAG	LLM + RAG	Combines retrieval-augmented generation with LLMs for explainable RCA and contextual explanations	Enhanced interpretability; few-shot anomaly detection; computational cost	[P18]
LasRCA	LLM-based Reasoning	Leverages LLMs for explainable RCA and few-shot anomaly detection	Explainable results; integrates multi-modal data; expensive fine-tuning	[P15]
MULAN	LLM-based Reasoning	Uses LLMs with RAG for few-shot anomaly detection and dependency modeling	Few-shot capability; models inter-service dependencies; high computational cost	[P17]
TraceCRL	GNN with Contrastive Learning	Applies contrastive learning on trace graphs to distinguish normal from abnormal executions	Captures structural relationships; requires quality trace data	[P9]
PUTraceAD	GNN with Positive-Unlabeled Learning	Leverages positive-unlabeled learning to improve recall of rare but critical anomalies	Improves recall for rare anomalies; training complexity	[P11]
AdvGraLog	GNN-based	Uses GNNs to model inter-service dependencies and trace graphs	Captures structural and causal relationships; graph construction overhead	[P36]
BertHTLG	GNN-based	Models inter-service dependencies using GNNs for trace graphs	Captures causal relationships; complexity in graph modeling	[P37]
BSDG	GNN-based	Uses GNNs to model interaction patterns within MSAs and identify anomalous communication flows	Models inter-service dependencies effectively; graph complexity	[P38]
GLAD-PAW	GNN with Dynamic Attention	Employs dynamic attention over evolving service graphs on HDFS and BGL for log session AD	Outperforms baselines with position awareness; requires good parsing; federated learning support	[P39]
GraphLog-Anomaly	GNN-based	Uses GNNs to model inter-service dependencies and trace graphs	Captures structural relationships; graph construction cost	[P40]
HGT-DGNN	Heterogeneous Graph Transformer	Service failure analytics for hidden failures using multi-modal telemetry on e-commerce systems	30pp improvement over XGBoost with user-impact modeling; small synthetic study with scalability concerns	[P12]

Continued on next page

Table E.22 (continued): AI-Based Tools for Anomaly Detection in Microservice Systems

Tool	**AD Technique**	**Key Insight**	**Benefits & Challenges**	**Refs.**
GROOT	Causal Graph Analysis	Constructs causal graphs for RCA; supports multimodal fusion	Effective RCA; graph construction complexity	[P41]
TraceAnomaly	Deep Bayesian Network	Real-time trace AD using attention-based neural networks to learn comprehensive trace semantics	High precision/recall with RCA localization; daily GPU retraining required and sensitive to unseen paths	[P42]
DeepTraLog	Graph-based Deep SVDD + CNN/RNN	Unified AD over microservice traces and logs using graph representation; fuses convolutional and recurrent layers	Strong F1 with unified trace+log graph; heavy preprocessing with synthetic-scale evaluation	[P43]
MicroScope	Causal Graph & Dependency Analysis	Constructs causal graphs for RCA; supports multimodal fusion	Effective for RCA; complexity in causal inference	[P44]
InstantOps	Causal Graph & Multi-modal Fusion	Constructs causal graphs for RCA; integrates logs, metrics, and traces for end-to-end failure diagnosis	Holistic multi-modal approach; high complexity	[P10]
DiagFusion	Causal Graph & Multi-modal Fusion	Robust sequence modeling that adapts to varying log structures; integrates multi-modal data for RCA	Robust to log structure variations; preprocessing overhead	[P45]
TWLog	Temporal & Workflow Modeling	Models execution workflows and temporal semantics	Captures workflow patterns effectively; moderate complexity	[P46]
Informer	DCRNN for RPC Traffic	Models irregular RPC traffic using diffusion convolutional recurrent neural network; captures spatial and temporal dependencies	Effective for distributed monitoring; models RPC patterns well; training complexity	[P47]
AcLog	Active Learning + Clustering	Combines unsupervised clustering with supervised classifiers for label-efficient AD	Label-efficient; incorporates human feedback effectively	[P48]
AFALog	Active Learning & Few-shot	Aligns fault propagation patterns across services using structural trace features; few-shot learning	Incorporates human feedback; label-efficient; requires human interaction	[P49]
MEPFL	Ensemble Learning	Combines multiple models (Random Forest, KNN, MLP) for latent error forecasting before system failure	Robust predictions via ensemble; computational overhead	[P1]

Continued on next page

Table E.22 (continued): AI-Based Tools for Anomaly Detection in Microservice Systems

Tool	**AD Technique**	**Key Insight**	**Benefits & Challenges**	**Refs.**
TraceRCA	Multi-model Fusion	Combines multiple models for robust predictions; fuses trace and metric signals	Strong robustness; complexity in fusion	[P44]
CDMF	Multi-model Fusion	Fuses trace and metric signals for robust predictions	Effective multi-modal fusion; integration complexity	[P50]
M3 Framework	Ensemble (SVM/RF/NB/LR)	Benchmark study on microservices reliability; SVM achieved 99.81% accuracy on Book-shop testbed	Multi-metric evaluation with strong performance; limited to small synthetic dataset	[P3]
MicroHECL	Hybrid Multi-modal Architecture	Integrates logs, metrics, and traces for end-to-end failure diagnosis	Holistic approach; high system complexity	[P51]
ServiceAnomaly	Trace DAG + Profiling Metrics	Multi-view fusion combining logs and metrics for service-specific AD using context propagation graphs	Interpretable graphs with improved F1; limited real-world diversity with tracing dependency	[P52]
UAC-AD	Adversarial Contrastive Learning + Autoencoder	Unsupervised adversarial contrastive learning with reconstruction AE for multi-modal log+metric AD	F1 improvements of 4.5-19.3% over baselines; label-free; GAN/contrastive training cost with retuning on drift	[P60]
ELK+Kafka Platform	ELK + Kafka + GAN	Enterprise log AD platform on AIOps Atlas and e-commerce; scalable pipeline with massive log handling	F1-scores of 0.792 and 0.829; strong practical fit; template brittleness with infrastructure cost	[P61]
FedTag	Federated Learning	Enables anomaly detection without raw log sharing using federated learning	Privacy-preserving; communication overhead	[P53]
MADMM	Optimization-based (ADMM)	Uses ADMM for scalable detection with distributed monitoring	Scalable for distributed systems; optimization complexity	[P54]
DQN-DBO Monitoring	Reinforcement Learning	Uses RL for scalable detection and monitoring	Adaptive learning; training complexity	[P55]
MRTS-BP	Regression Testing	Prioritizes regression tests using log trees; CI/CD pipeline integration	Effective for testing; requires CI/CD integration	[P56]
Linnaeus	Rule-based	Uses simple heuristics for explainable anomaly detection; prioritizes regression tests	Explainable and lightweight; limited detection capability	[P57]
Automated KG-LSTM Monitoring	Knowledge Graph + LSTM	Uses knowledge graphs for contextual RCA with LSTM; achieves high accuracy	Holistic KG view with proactive alerts; synthetic evaluation with unclear scalability	[P58]

Continued on next page

Table E.22 (continued): AI-Based Tools for Anomaly Detection in Microservice Systems

Tool	**AD Technique**	**Key Insight**	**Benefits & Challenges**	**Refs.**
RCSF	Dependency Modeling	Models inter-service dependencies and propagation paths	Effective dependency modeling; complexity in mapping	[P44]