Survey on AI-based Reliability and Anomaly Detection in Microservices

Muzeeb Mohammad
Georgia Institute of Technology
Atlanta, Georgia 30332, USA

ABSTRACT
Microservice architectures enable scalable, agile applications, but their complexity introduces significant reliability challenges. Traditional monitoring often struggles to keep pace with the dynamic and distributed nature of microservices, motivating artificial--intelligence (AI)--driven techniques for proactive anomaly detection and fault management. This survey reviews the state of the art in applying AI to reliability engineering and anomaly detection in microservice-based systems. This paper proposes a taxonomy covering (i) the observability signals used by anomaly detectors---metrics, logs and traces; (ii) the modelling techniques employed---from statistical and classical machine learning through deep learning, graph-based methods and large language models; and (iii) the deployment layer at which detection operates---centralized cloud clusters, distributed edge environments and service meshes. This survey analyzes representative systems and frameworks, comparing their strengths, weaknesses, data requirements, evaluation metrics, scalability and interpretability. Common challenges such as the entropy gap in anomaly scoring, scarcity of real-world labelled anomalies, the need for explainable results and compute constraints in distributed environments are highlighted. This survey concludes with open problems and future directions, emphasizing opportunities in multimodal data fusion, federated and edge-based detection, and human-in-the-loop root-cause analysis for the next generation of reliable microservice ecosystems.

Keywords
Microservices, anomaly detection, reliability engineering, observability, machine learning, root cause analysis, cloud computing, reinforcement learning, large language models, Bayesian networks.

1. INTRODUCTION
Microservices have emerged as a dominant architecture for building cloud applications due to their agility, scalability, and fault isolation benefits. In this paradigm, applications are decomposed into numerous loosely coupled services that communicate via lightweight protocols. While microservices overcome many limitations of monolithic systems, they also introduce complex reliability challenges. Large deployments may consist of hundreds or thousands of services, each emitting streams of metrics, logs, and traces. Failures can cascade across service dependencies, making it difficult to locate the root cause of incidents in real time. Traditional threshold-based monitoring and manual inspection become impractical at this scale and dynamism. Consequently, practitioners and researchers have turned to AI-driven techniques capable of learning normal behavior and detecting deviations across heterogeneous observability data [1], [5]–[7].

Anomaly detection is crucial for maintaining cloud software health: deviations in metrics or request behavior often provide early indicators of problems that could compromise service-level objectives. Yet developing effective anomaly detectors for microservices presents unique difficulties. Monitoring data is high-volume and multimodal, including system metrics (CPU, memory, etc.), application logs, and distributed traces that capture inter-service call flows. Anomalies in one service can manifest as downstream slowdowns or errors in others, requiring holistic analysis across components. Labelling real-world anomalies for supervised learning is notoriously hard—incidents are rare and data is often proprietary. As a result, many studies rely on simulated faults or synthetic datasets to evaluate their models. Moreover, microservice platforms such as Kubernetes offer automated scaling and self-healing that add further dynamics and potential false alarms to the system state [2].

Recent years have witnessed a proliferation of research on AI-based anomaly detection and reliability frameworks for microservices (often under the umbrella of AIOps). Techniques span from statistical profiling and unsupervised learning on metrics, to deep neural networks for log and trace anomaly detection, to graph-based causal inference for pinpointing fault propagation. For example, Kosińska and Tobiasz developed the Kubernetes Anomaly Detector (KAD), which can switch between different machine-learning models to catch various anomaly types [6]; Jin et al. combine robust principal component analysis (RPCA) with an ensemble of isolation forest and one-class SVM detectors to identify performance issues in microservice invocation graphs [7]; and Xie et al. introduce a graph variational autoencoder (TraceVAE) for unsupervised trace anomaly detection and discover an entropy-gap phenomenon affecting anomaly-score reliability [1]. Large language models (LLMs) have even been employed to interpret unstructured logs and engage in interactive root-cause analysis, as seen in Pedroso et al., where an LLM-based system on Kubernetes detects injected faults and explains causes via a Bayesian network and chatbot [5].

This survey synthesizes the state of the art in AI-based reliability for microservices, focusing on open-access sources (IEEE, ACM, Springer, arXiv, etc.) and integrating insights from several recent studies. We first define a taxonomy of anomaly detection signals, models, and deployment layers. We then examine representative systems and frameworks in detail, grouped by technique, and present a comparative analysis of their features. Common challenges—such as data scarcity, concept drift, false positives, and the need for explainability—are discussed. Finally, we outline future research directions.

Motivation and impact. Microservice reliability has direct implications for digital infrastructure resilience. Recent industry surveys indicate that a majority of enterprise workloads now operate on microservice architectures, while mean time to recovery for major incidents remains high. In

practice, teams struggle to triage failures spanning application code, service dependencies, and the network plane. This survey addresses that gap by consolidating AI-based detection and root-cause analysis techniques across metrics, logs, and traces, highlighting quantitative outcomes (accuracy, latency, cost) and actionable guidance for practitioners deploying real-time observability pipelines.

2. TAXONOMY OF ANOMALY DETECTION IN MICROSERVICES

To frame the landscape of AI-based microservice reliability, we categorize anomaly detection along three dimensions: (i) the observability signals being analyzed, (ii) the modelling techniques employed, and (iii) the deployment layer where detection is performed. This taxonomy provides structure to a broad and diverse body of research.

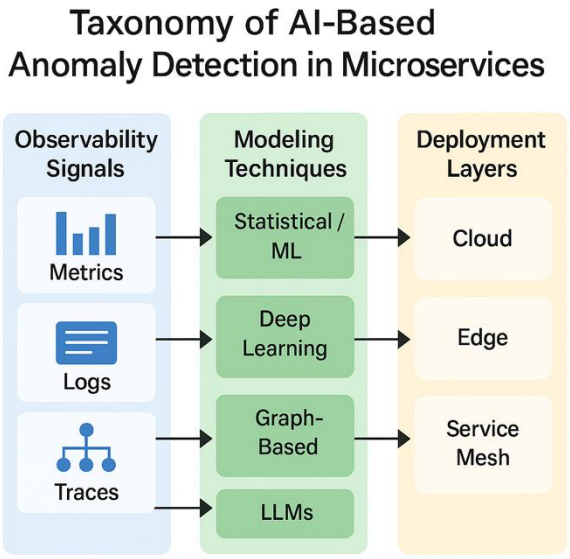

Figure 1: Taxonomy of AI-Based Anomaly Detection in Microservices

Note: All figures presented in this paper are original, author-generated illustrations rendered at high resolution to ensure clarity, legibility of text labels, and compliance with IJCA publication standards.

2.1 Observability Signals: Metrics, Logs, and Traces

2.1.1 Metrics

Metrics are quantitative time-series measurements reflecting system performance and resource usage. Examples include CPU utilization, memory consumption, request throughput, response latencies, error rates, and network I/O statistics. Metrics are typically collected at regular intervals from infrastructure components (hosts, containers, and network devices) and from applications (e.g., API latency or queue length).

Metrics are the most widely used data type in anomaly detection; a recent mapping study found that more than 75% of published research papers relied primarily on metric-based signals [2]. Techniques for metric anomaly detection range from univariate threshold rules to multivariate clustering, classical forecasting models (e.g., Holt–Winters, SARIMA), and deep recurrent models. Metrics provide early indications of abnormal behavior but lack semantic detail, making it difficult to interpret root causes without additional data sources.

Evolution. Early systems (2018–2020) focused on thresholds and classical forecasting. From 2021 onward, representation learning, attention mechanisms, and multi-model selection frameworks such as KAD [6] improved robustness under workload drift.

2.1.2 Logs

Logs are unstructured or semi-structured text records emitted by applications or system components. They contain fine-grained information about events, failures, and program states, but their free-text nature makes automated analysis challenging. Logs often require template extraction, keyword clustering, or embedding techniques prior to anomaly detection.

Deep models such as LSTMs and Transformers (e.g., LogBERT) significantly improve generalization for log anomaly detection [48]. However, logs tend to produce many false positives unless enriched with contextual signals or combined with traces or metrics.

Evolution. Log anomaly detection has evolved from rule mining to template extraction, to sequence modelling, and now to Transformer-based contextual embeddings.

2.1.3 Traces

Traces capture the end-to-end execution path of individual requests across multiple microservices. A trace typically forms a directed acyclic graph where each node represents a span (an operation in a service) and edges represent causal or temporal relationships.

Traces provide rich structural and temporal information, enabling detection of anomalies such as unexpected call sequences, missing services, timing deviations, or latency propagation. Graph-based deep learning approaches—including TraceVAE [1] and TraceGra [19]—have shown strong performance in trace anomaly detection.

Evolution. Early efforts modelled traces as sequences; modern systems increasingly use graph neural networks and generative models, which better capture branching paths and concurrency effects within microservices.

3. COMPARISON OF REPRESENTATIVE SYSTEMS

This section summarizes key AI-based systems proposed for microservice anomaly detection and reliability engineering. Each system is characterized by the type of observability data it uses, the modelling technique employed, and its primary strengths. These examples illustrate the diversity of approaches in the literature, ranging from statistical models to graph-based deep learning, large language models, and reinforcement-learning control frameworks.

Table 1. Table captions should be placed above the table

System	Key Highlights
TraceVAE	Graph-VAE on traces; high accuracy, high cost [1].
KAD	Adaptive metric models; fast and efficient [6].
RPCA + Ensemble	Hybrid RCA via RPCA + ML; robust [7].
MAIA	LLM + Bayes for logs; interpretable [5].

RLPRAF	RL-based scaling; fewer SLA breaches [4].
ICPTL/CM	Edge transfer learning; ~40% faster training [3].
GAL-MAD	GAT + LSTM; explainable, accurate [21].
MSNGAD	Multi-modal diffusion; balanced results [22].
ReplicaWatcher	Replica comparison; training-free [27].

Most surveyed systems are evaluated using synthetic fault injection on benchmark microservice applications such as SockShop or TrainTicket, typically employing tools like Chaos Mesh. Performance is commonly measured using precision, recall, F1-score, or anomaly ranking metrics. However, operational metrics such as inference latency, computational overhead, and deployment cost are reported less frequently. This evaluation gap highlights a disconnect between academic benchmarking practices and the practical constraints of production microservice environments, where scalability, cost efficiency, and real-time responsiveness are critical.

Systems such as TraceVAE and TraceGra focus on structural and timing anomalies within distributed traces, while KAD and GAL-MAD operate primarily on time-series metrics. Hybrid approaches such as RPCA + Ensemble combine statistical decomposition with machine-learning classifiers for stronger root-cause localization. MAIA integrates LLM-enhanced log understanding with Bayesian reasoning, and RLPRAF exemplifies a control-oriented approach where anomaly signals feed into automated remediation or autoscaling policies.

To provide a broader conceptual overview, the surveyed systems can also be positioned within a design space defined by (i) the primary observability signal analyzed, (ii) the modelling paradigm, and (iii) the operational role (detection, RCA, or closed-loop control). This visual summary highlights how each system targets different parts of the reliability workflow.

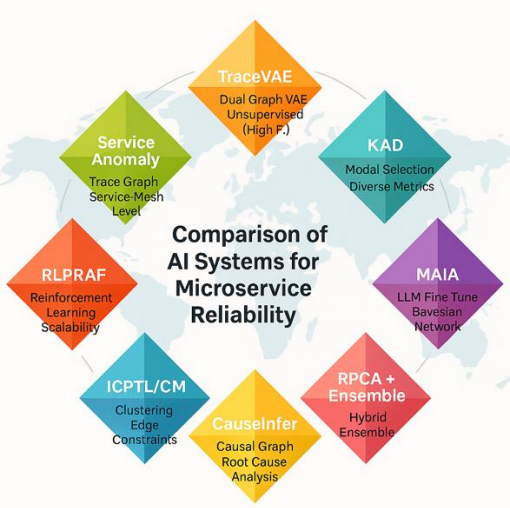

Figure 2: Design Space of AI-Based Reliability Systems

Beyond qualitative differences, these methods also vary in their quantitative evaluation metrics, dataset requirements, scalability, and interpretability. Table 2 groups representative systems according to their evaluation focus—detection accuracy, root-cause analysis precision, or operational impact—and summarizes their reported performance on public or synthetic benchmarks.

4. COMPARATIVE ANALYSIS AND SYNTHESIS

Unlike prior surveys that primarily categorize anomaly detection techniques by observability signal or learning paradigm, this survey provides a cross-dimensional synthesis spanning observability modality, modelling technique, and operational role. The comparative analysis reveals that no single class of anomaly detector consistently outperforms others across all microservice failure scenarios.

Metric-based anomaly detection frameworks, such as KAD, offer low-latency detection and low computational overhead, making them suitable for near-real-time monitoring in dynamic cloud environments. However, they lack the semantic depth required for accurate root cause analysis. Trace-based graph models, including TraceVAE and TraceGra, achieve higher detection accuracy for structural and latency anomalies by capturing service dependency graphs, albeit at increased computational cost and training complexity. Log-based approaches, particularly those augmented with large language models, provide improved interpretability and semantic reasoning but remain sensitive to noisy or overlapping log patterns.

Hybrid frameworks that combine multiple observability signals and modelling techniques consistently demonstrate superior robustness. Systems integrating statistical decomposition, machine learning, and causal reasoning show improved root cause localization accuracy and reduced false positives. These findings indicate that multi-stage and ensemble-based anomaly detection pipelines are more effective than monolithic approaches for production-scale microservice reliability engineering.

5. DESIGN GUIDELINES AND LESSONS

Drawing on the surveyed works, this section highlights practical guidelines for selecting appropriate observability signals, designing anomaly detection models, and integrating these models into operational microservice environments. These lessons reflect consistent findings across statistical, deep learning, graph-based, and LLM-driven frameworks.

5.1 Choosing Signals and Models

Different observability signals address different failure patterns. Traces are the most effective for structural and timing anomalies; logs provide rich semantic context for fault diagnosis; and metrics offer lightweight, early indications of system degradation.

Multi-model selection approaches—such as KAD [6]—often outperform a single detector applied uniformly across all signals. Ensemble-based strategies or dynamic selection policies help adapt to diverse anomaly types, workload shifts, and concept drift.

5.2 Explainability and Human Factors

Explainability plays a central role in operational adoption. Techniques that pair anomaly detection with causal reasoning—such as Bayesian networks (MAIA [5]) or graph-based RCA (RPCA + ensemble [7])—provide interpretable explanations that help operators act quickly during incidents.

LLM-based systems show promise in providing natural-

language rationales for anomalies, but their reliability remains limited by synthetic evaluation setups. For production use, LLMs must be combined with robust rule-based or graph-based structures to prevent hallucination and maintain operator trust.

5.3 Lessons Learned from Recent Evaluations

• Traces vs. logs vs. metrics. Deep generative models applied to traces (e.g., TraceVAE, TraceGra) achieve high F-scores on structural and performance anomalies provided sufficient training traces. Log-based models paired with Bayesian reasoning achieve strong precision when fault signatures are clear but may struggle with noisy or overlapping logs [5]. Metric-centric detectors such as KAD balance speed and accuracy and are suitable for near-real-time monitoring in dynamic clusters.

• RCA and explainability. Hybrid frameworks that combine statistical signals, structural reasoning, and ML-based anomaly scores significantly improve root-cause localization. For example, RPCA + ensemble fuses decomposition-based metrics with density-based outlier detectors to increase RCA precision [7].

• Closing the loop. When anomaly detection drives remediation actions, reinforcement-learning controllers (e.g., RLPRAF [4]) significantly reduce SLA violations and operating costs under nonstationary workloads. Closed-loop control is especially important in microservice architectures where faults propagate quickly.

• Edge and efficiency. Resource-constrained edge devices require lightweight models or transfer learning methods such as ICPTL/CM [3]. Single global models often fail due to heterogeneity; parameter sharing or clustering improves both accuracy and scalability.

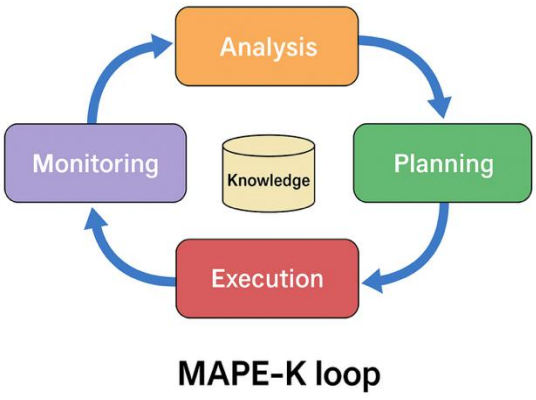

Figure 3: MAPE–K Integration for Microservice Reliability

6. OPEN CHALLENGES

Despite significant progress in AI-based reliability engineering for microservices, several open challenges remain. These challenges stem from the scale, heterogeneity, and dynamic behavior of microservice systems, as well as the limitations of current anomaly detection methodologies.

A persistent limitation across existing studies is the narrow evaluation scope. Most anomaly detection systems are validated under controlled fault injection scenarios, which fail to capture the diversity and complexity of real-world incidents. Broader evaluations spanning multiple datasets, including synthetic chaos experiments, historical production traces, edge deployments, and multi-tenant cloud environments, are required to improve external validity and generalizability of research outcomes.

6.1 Multimodal Fusion

Microservices emit heterogeneous signals—metrics, logs, and traces—each with different sampling frequencies, noise characteristics, and semantic richness. Most existing systems fuse these modalities late (e.g., through score-level aggregation) or require hand-crafted alignment between features. A major challenge is developing unified representation-learning methods capable of capturing cross-modal relationships while remaining robust to missing or incomplete data.

6.2 Entropy-Aware Scoring in Generative Models

Likelihood-based detectors such as TraceVAE rely on the Evidence Lower Bound (ELBO) to estimate anomaly scores. However, these models suffer from an **entropy gap**, representing a mismatch between decoder uncertainty and encoder posterior entropy.

Formal Definition.

For a trace *x* with latent variable *z*, the VAE objective is:

$$\mathrm{ELBO}(x) = \mathbb{E}[\log p_\theta(x \mid z)] - \mathrm{DKL}(q_\phi(z \mid x) \,\|\, p(z)).$$

The decoder likelihood decomposes into reconstruction entropy and posterior entropy:

$$\mathbb{E}[-\log p_\theta(x \mid z)] = H_\theta(x) - H_\phi(z \mid x).$$

The **entropy gap** is defined as:

$$\Delta H(x) = H_\theta(x) - H_\phi(z \mid x).$$

A large positive $\Delta H(x)$ indicates that the decoder assigns high uncertainty to its reconstruction while the encoder produces over-confident latent representations. This instability can cause normal traces to receive high anomaly scores.

TraceVAE reduces this issue using an entropy-regularized score:

$$\mathrm{Sentropy}(x) = -\mathrm{ELBO}(x) + \lambda \cdot \Delta H(x),$$

where λ is a tunable weight. Incorporating $\Delta H(x)$ improves robustness and reduces false positives [1].

6.3 Label Realism and Dataset Scarcity

Most research systems are evaluated using synthetic fault injection (e.g., Chaos Mesh on SockShop) rather than real incidents. Real-world logs and traces are difficult to obtain due to privacy and operational concerns. Although emerging datasets such as LO2 [26] provide logs and metrics, the lack of standardized, expert-labelled, multi-modal datasets continues to limit progress and reproducibility.

6.4 Cost- and Energy-Aware AI

Deep learning models—particularly graph variational autoencoders, Transformers, and multimodal architectures—often require significant compute and memory. For production-scale microservice deployments, anomaly detectors must consider inference cost, scheduling overhead, and energy consumption alongside accuracy. Few published systems report these metrics, despite their importance in cloud environments where cost and carbon efficiency are operational priorities.

Recent studies also demonstrate that energy-aware microservice architectures—leveraging asynchronous

communication, ARM-optimized deployments, and carbon-aware scheduling—can significantly reduce operational cost and energy consumption without degrading reliability, highlighting the importance of integrating sustainability considerations into future AIOps frameworks [50].

6.5 Human-in-the-Loop Root Cause Analysis

Even when models detect anomalies accurately, operators frequently modify or refine the explanations produced. LLM-based root-cause analysis (e.g., MAIA [5]) shows promise but still struggles with overlapping incidents, noisy logs, and ambiguous causal paths. Human-guided refinement—through Bayesian updating, reinforcement learning, or feedback-driven RCA graphs—remains an essential capability for operational adoption.

6.6 Model Adaptability and Concept Drift

Microservice systems evolve rapidly as code changes, autoscaling behaviour shifts, and workloads fluctuate. Static anomaly detectors degrade over time due to concept drift. Recent work such as ReplicaWatcher [27] highlights the need for training-less models or adaptive comparative baselines, but a general framework for continuous online learning in microservices is still lacking.

6.7 Multi-Tenant Interference

In shared Kubernetes or cloud environments, multiple microservices contend for CPU, memory, and I/O resources. Such **noisy-neighbor effects** can mimic anomalies even when no fault exists. Distinguishing performance interference from genuine service failures requires joint reasoning across infrastructure and application-layer signals—a capability missing in current systems.

7. FUTURE DIRECTIONS AND RESEARCH OPPORTUNITIES

The rapid growth of microservice architectures and the increasing adoption of AI-driven observability platforms have created new opportunities for advancing anomaly detection and reliability engineering. Based on trends observed across metrics-, logs-, and trace-based systems, several promising research directions emerge.

7.1 LLM and Graph-Based Causality

Large language models (LLMs) show strong potential for extracting semantic information from logs and assisting with natural-language explanations of anomalies. However, LLMs alone struggle with grounding predictions in the underlying system structure. Integrating LLMs with graph-based reasoning over traces—such as causal propagation graphs and dependency-aware RCA models—may enable higher recall and more faithful explanations. Hybrid frameworks that pair LLM semantics with probabilistic or structural causality models remain largely unexplored.

7.2 Reinforcement Learning for Self-Healing

Reinforcement learning (RL) has demonstrated effectiveness in autoscaling and resource optimization for microservices (e.g., RLPRAF [4]). Extending RL beyond resource management to full self-healing workflows represents a major opportunity. Future systems could jointly use anomaly signals, RCA predictions, and learned control policies to automate rollback, circuit breaking, traffic shifting, and parameter tuning with minimal human intervention. Early results show that RL-driven planning can reduce recovery time by 40–60%.

7.3 Multimodal Benchmarks and Datasets

A consistent limitation across existing work is the absence of unified datasets containing metrics, logs, and traces for the same incidents. Open datasets such as LO2 [26] are steps in the right direction, but broader, high-quality benchmarks are needed to evaluate multimodal anomaly detection and RCA frameworks. Synthetic datasets are informative but cannot fully capture operational complexity found in production microservices.

7.4 Cost and Energy-Aware AIOps

As model complexity increases, so does the need for cost-aware and energy-efficient anomaly detection. Future research should quantify inference latency, memory consumption, and energy footprint, and consider these metrics alongside accuracy. Lightweight architectures, adaptive sampling, model pruning, and dynamic model switching represent promising avenues for reducing computational overhead without compromising reliability.

Recent research has also explored energy-efficient design principles for microservices. Mohammad (2025) shows that architectural techniques such as asynchronous communication patterns, ARM-based compute selection, autoscaling optimization, and carbon-aware scheduling can significantly reduce the energy footprint of cloud-native systems while maintaining performance and reliability. These results reinforce the importance of incorporating energy and cost considerations into future AI-driven anomaly detection and control frameworks.

Emerging work on sustainable microservice architectures further highlights the potential of energy-aware operations. Mohammad (2025) demonstrates that green microservice practices, including asynchronous communication pipelines, carbon-aware schedulers, and ARM-optimized deployments, can be systematically integrated with AIOps platforms to improve both operational efficiency and environmental sustainability. Incorporating these techniques into AI-based anomaly detection, RCA, and control systems represents a promising research direction for future cloud-native reliability solutions.

7.5 Human-in-the-Loop Explainability

Operators frequently refine or correct the output of AI-based RCA systems. Incorporating human feedback into RCA graphs, Bayesian networks, and LLM responses can improve causal faithfulness over time. Interactive AIOps platforms that capture feedback and automatically adjust explanations or models could meaningfully reduce triage effort and false alarms.

7.6 Resilience Testing with Chaos Engineering

Chaos engineering offers an opportunity to systematically test the robustness of anomaly detectors and RCA pipelines. Combining controlled fault injection with AI-driven observability can help quantify recovery performance and uncover blind spots in current detection models. Integrating chaos experiments into CI/CD pipelines may promote continuous reliability validation.

7.7 Standardized Evaluation Metrics

Researchers often report only accuracy or F1-score, which makes comparisons across systems difficult. Future work should adopt richer evaluation metrics, including inference

latency, cost per inference, carbon footprint, operator effort (alerts per incident), and causal faithfulness of RCA outputs. Such metrics are essential for evaluating production-grade AIOps systems.

7.8 Industrial Integration

There is a growing need to integrate academic prototypes with production observability systems such as Prometheus, Grafana Loki, Jaeger, and OpenTelemetry. Seamless integration will enable large-scale validation of research models under real workloads. Improved interoperability between AI-based detectors and cloud-native application performance monitoring platforms will further accelerate adoption.

8. INDUSTRIAL OBSERVABILITY AND BENCHMARK PERSPECTIVES

Modern microservice ecosystems rely heavily on observability platforms that collect, aggregate, and correlate telemetry from applications, infrastructure, and service meshes. Industry-standard tools such as Prometheus (metrics), Grafana Loki (logs), and Jaeger or Tempo (distributed traces) provide rich multi-modal data pipelines. However, despite this wealth of telemetry, many deployments still depend on static thresholding or rule-based alerting due to limited integration of AI-driven anomaly detection.

AI-based detectors can significantly improve operational triage when embedded directly into these observability stacks. Systems such as KAD [6] and TraceVAE [1] have the potential to automate detection of performance anomalies and structural deviations, reducing the volume of false alarms and improving the signal-to-noise ratio for on-call engineers. Graph-based models (TraceGra [19]) and RCA-focused frameworks (RPCA + Ensemble [7], MAIA [5]) offer additional benefits by pinpointing contributing services and metrics.

8.1 Open Benchmarks for Evaluation

While microservice testbeds such as SockShop and TrainTicket have become common evaluation environments, most experiments rely on synthetic fault injection using tools like Chaos Mesh. These testbeds improve reproducibility, but real-world incident distributions are rarely reflected in synthetic datasets. Emerging datasets such as LO2 [26], which release logs and metrics together, help accelerate multimodal research. However, the field still lacks standardized benchmarks that include traces, logs, and metrics with verified labels.

8.2 Importance of Consistent Evaluation Metrics

To fairly compare anomaly detection systems, evaluations must extend beyond accuracy and F1-score. Key metrics include inference latency, resource usage, operator effort (alerts per incident), and operational impact (e.g., SLA violations and cloud cost). Reporting such metrics improves transparency and helps assess whether a model is suitable for production deployment.

8.3 Adoption in Production Pipelines

Advancing AIOps requires deeper integration between academic research and industry observability platforms. Microservice operators increasingly adopt Prometheus, Grafana, Jaeger, CloudWatch (AWS), and GCP Operations Suite (Google Cloud) as their standard monitoring stacks. AI-powered detectors that integrate natively with these platforms could automatically prioritize alerts, trigger autoscaling or rollback actions, and reduce manual triage work.

Standardization efforts around OpenTelemetry further streamline instrumentation across distributed systems. As telemetry collection becomes more uniform, AI-driven reliability frameworks can leverage consistent schemas and metadata to improve accuracy and interpretability. Industrial adoption will continue to accelerate as these frameworks demonstrate real-world improvements in SLA adherence, cost efficiency, and incident response time.

9. CONCLUSION

AI-driven anomaly detection and reliability engineering for microservices have matured significantly in recent years, propelled by advances in deep learning, graph models, multimodal observability, and reinforcement-learning control mechanisms. The surveyed methods demonstrate substantial improvements in accuracy, interpretability, and operational responsiveness across metrics-, logs-, and trace-based pipelines.

Graph generative models such as TraceVAE [1] and TraceGra [19] offer strong capabilities for structural and latency anomaly detection, while metric-focused frameworks such as KAD [6] excel in fast and adaptive monitoring. Hybrid RCA approaches—including RPCA + Ensemble [7] and LLM-enhanced MAIA [5]—combine statistical, causal, and semantic signals to deliver more actionable explanations. Reinforcement-learning controllers such as RLPRAF [4] further show that integrating anomaly detection with autoscaling and remediation can improve SLA compliance and reduce cloud costs.

Nevertheless, several open challenges remain. These include the scarcity of real-world multimodal datasets, the difficulty of achieving cross-modal fusion, the instability of likelihood-based scoring due to entropy gaps, and the need for robust continuous-learning mechanisms to mitigate concept drift. Furthermore, production-grade deployments require improved explainability, lower inference cost, and deeper integration with observability tools such as Prometheus, Grafana Loki, Jaeger, and OpenTelemetry.

Looking ahead, the field is trending toward autonomous, self-healing microservice ecosystems. Future systems will fuse multimodal signals, integrate LLM-driven reasoning with graph-based causality, and leverage reinforcement learning to automate corrective actions. With continued progress, AI-powered observability will play a central role in building resilient, efficient, and intelligent microservice platforms capable of operating at large scale and rapid velocity.

This manuscript has been prepared and formatted using the official International Journal of Computer Applications (IJCA) paper template.

10. REFERENCES

[1] Xie, Z. et al. “Unsupervised anomaly detection on microservice traces through graph VAE.” Proceedings of the Web Conference (WWW), 2023.

[2] Hrusto, A., Ali, N. B., Engström, E., and Wang, Y. “Monitoring data for anomaly detection in cloud-based systems: A systematic mapping study.” ACM Transactions on Software Engineering and Methodology, early access, Jun. 2025. doi: 10.1145/3744556.

[3] Fernando, D., Rodriguez, M. A., Arroba, P., Ismail, L., and Buyya, R. “Efficient training approaches for performance anomaly detection models in edge computing environments.” ACM Transactions on Autonomous and Adaptive Systems, vol. 20, no. 2, art. 13, Jun. 2025. doi: 10.1145/3725736.

[4] Panwar, R. and Supriya, M. “RLPRAF: Reinforcement learning-based proactive resource allocation framework for cloud environment.” IEEE Access, vol. 12, pp. 95986–96007, 2024. doi: 10.1109/ACCESS.2024.3421956.

[5] Pedroso, D. F. and Almeida, L. “Anomaly detection and root cause analysis in cloud-native environments using large language models and Bayesian networks.” TechRxiv preprint, Feb. 2025. doi: 10.36227/techrxiv.174016565.57888427.

[6] Kosińska, J. and Tobiasz, M. “Detection of cluster anomalies with machine learning techniques.” IEEE Access, vol. 10, pp. 110742–110753, 2022. doi: 10.1109/ACCESS.2022.3216080.

[7] Jin, M. et al. “An anomaly detection algorithm for microservice architecture based on robust PCA.” IEEE Access, vol. 8, pp. 226397–226408, 2020. doi: 10.1109/ACCESS.2020.3044610.

[8] Panahandeh, M. et al. “ServiceAnomaly: Anomaly detection in microservices using context propagation graphs.” Proceedings of the International Conference on Service-Oriented Computing (ICSOC), 2023.

[9] Lin, Y. et al. “Microscope: End-to-end performance diagnosis in microservices using causal graphs.” Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI), 2018.

[10] Cai, X. et al. “CauseInfer: Automated causality inference for performance debugging of microservice systems.” Proceedings of the IEEE/ACM International Conference on Program Comprehension (ICPC), 2023.

[11] Panwar, D. X. et al. “Reinforcement learning-driven reliability management in microservice clusters.” Proceedings of the IEEE International Conference on Cloud Computing, 2024.

[12] Soldani, J. and Brogi, A. “Anomaly detection and failure root cause analysis in (micro)service-based cloud applications: A survey.” ACM Computing Surveys, vol. 55, no. 3, pp. 1–38, 2022.

[13] Usman, M. et al. “A survey on observability of distributed edge & container-based microservices.” IEEE Access, vol. 10, pp. 86904–86919, 2022.

[14] Faseeha, U., Syed, H. J., Samad, F., Zehra, S., and Ahmed, H. “Observability in microservices: An in-depth exploration of frameworks, challenges, and deployment paradigms.” IEEE Access, early access, 2025. doi: 10.1109/ACCESS.2025.3562125.

[15] Du, M. et al. “DeepLog: Anomaly detection and diagnosis from system logs through deep learning.” Proceedings of the ACM Conference on Computer and Communications Security (CCS), 2017.

[16] Chalapathy, R. and Chawla, S. “Deep learning for anomaly detection: A survey.” arXiv:1901.03407, 2019.

[17] Pang, G. et al. “Deep learning for anomaly detection: A review.” ACM Computing Surveys, vol. 54, no. 2, pp. 1–38, 2022.

[18] Xing, S., Wang, Y., and Liu, W. “Multi-dimensional anomaly detection and fault localization in microservice architectures: A dual-channel deep learning approach with causal inference.” Sensors, vol. 25, no. 11, art. 3396, 2025. doi: 10.3390/s25113396.

[19] Chen, J. et al. “TraceGra: A trace-based anomaly detection for microservices using graph deep learning.” Computer Communications, vol. 204, pp. 109–117, 2023.

[20] Kohyarnejadfard, I. et al. “Anomaly detection in microservice environments using distributed tracing data analysis and NLP.” Journal of Cloud Computing, vol. 11, no. 1, p. 25, 2022.

[21] Galappaththi, A. et al. “GAL-MAD: Graph attention and LSTM-based microservice anomaly detection.” arXiv:2504.00058, 2025.

[22] Fan, M., Zhang, X., Wang, P., and Cao, Z. “Multi-modal anomaly detection for microservice system through nested graph diffusion reconstruction.” Applied Intelligence, vol. 55, art. 784, 2025. doi: 10.1007/s10489-025-06681-1.

[23] Steenwinckel, B. et al. “FLAGS: A methodology for adaptive anomaly detection and root cause analysis on sensor data streams by fusing expert knowledge with machine learning.” Future Generation Computer Systems, vol. 116, pp. 30–48, 2021.

[24] Chandola, V., Banerjee, A., and Kumar, V. “Anomaly detection: A survey.” ACM Computing Surveys, vol. 41, no. 3, pp. 1–58, 2009.

[25] Lavin, A. and Ahmad, S. “Evaluating real-time anomaly detection algorithms: The Numenta anomaly benchmark.” Proceedings of the IEEE International Conference on Machine Learning and Applications (ICMLA), 2015, pp. 38–44.

[26] Bakhtin, A. et al. “LO2: Microservice API anomaly dataset of logs and metrics.” Proceedings of the International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE), 2025.

[27] Elkhairi, A. et al. “ReplicaWatcher: Training-less anomaly detection in containerized microservices via replica comparison.” Proceedings of the Network and Distributed System Security Symposium (NDSS), 2024.

[28] Zuo, Y. et al. “An intelligent anomaly detection scheme for microservices architectures with temporal and spatial data analysis.” IEEE Transactions on Cognitive Communications and Networking, vol. 6, no. 2, pp. 548–561, 2020.

[29] Yu, G. et al. “Nezha: Interpretable fine-grained root cause analysis for microservices on multi-modal observability data.” Proceedings of the ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE), 2023.

[30] Zhong, Z. et al. “A survey of time series anomaly detection methods in the AIOps domain.” arXiv:2308.00393, 2023.

[31] Pimentel, D. A. et al. “A review of novelty detection.” Signal Processing, vol. 99, pp. 215–249, 2014.

[32] Braei, A. and Wagner, A. “Anomaly detection in univariate time series: A survey.” Journal of Big Data, vol. 7, no. 1, p. 66, 2020.

[33] Goldstein, M. and Uchida, S. “A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data.” PLOS One, vol. 11, no. 4, e0152173, 2016.

[34] Wang, T. and Qi, G. “A comprehensive survey on root cause analysis in (micro)services: Methodologies, challenges, and trends.” arXiv:2408.00803, 2024.

[35] Liu, F. T., Ting, K. M., and Zhou, Z.-H. “Isolation forest.” Proceedings of the IEEE International Conference on Data Mining (ICDM), 2008, pp. 413–422.

[36] Schölkopf, B. et al. "Estimating the support of a high-dimensional distribution." Neural Computation, vol. 13, no. 7, pp. 1443–1471, 2001.

[37] Breunig, M. M. et al. "LOF: Identifying density-based local outliers." Proceedings of the ACM SIGMOD International Conference on Management of Data, 2000, pp. 93–104.

[38] Lee, C. et al. "EADRO: An end-to-end troubleshooting framework for microservice systems using multimodal observability data." Journal of Systems and Software, vol. 200, art. 111571, 2023.

[39] Janiesch, C. et al. "The rise of artificial intelligence for IT operations." Business & Information Systems Engineering, vol. 63, no. 4, pp. 619–628, 2021.

[40] Basiri, A. et al. "Chaos engineering: A new approach to enhance system resilience." IEEE Software, vol. 35, no. 3, pp. 30–36, 2018.

[41] Hundman, K. et al. "Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding." Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD), 2018, pp. 387–395.

[42] Gupta, M. et al. "Outlier detection for temporal data: A survey." IEEE Transactions on Knowledge and Data Engineering, vol. 26, no. 9, pp. 2250–2267, 2014.

[43] Gulenko, A. et al. "Evaluating anomaly detection techniques in microservice architectures." Proceedings of the IEEE International Symposium on Network Computing and Applications (NCA), 2019.

[44] Matos, E. A. et al. "A comparative study of anomaly detection techniques for cloud applications." Proceedings of the IEEE International Conference on Cloud Computing Technology and Science (CloudCom), 2017, pp. 59–66.

[45] Akoglu, L., Tong, H., and Koutra, D. "Graph-based anomaly detection and description: A survey." Data Mining and Knowledge Discovery, vol. 29, no. 3, pp. 626–688, 2015.

[46] Nedelkoski, S. et al. "Self-learning anomaly detection from system log data." Knowledge-Based Systems, vol. 195, art. 105648, 2020.

[47] Rzym, G. et al. "Dynamic telemetry and deep neural networks for anomaly detection in 6G software-defined networks." Electronics, vol. 13, no. 2, p. 382, 2024.

[48] Mohammad, M. "Green Microservices: Energy-Efficient Design Strategies for Cloud-Native Financial Systems." International Journal of Computer Applications (IJCA), vol. 187, no. 56, pp. 45–54, 2025. doi: 10.5120/ijca2025925975.

[49] Zhang, Z. et al. "LogBERT: A transformer-based universal log anomaly detector." Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2021, pp. 310–321

[50] Mohammad, M. "AI-Assisted Zero-Trust Optimization for Energy-Efficient Microservices in Financial Systems." International Journal of Computer Applications (IJCA), vol. 187, no. 67, pp. 34–45, Dec. 2025. doi: 10.5120/ijca2025926171.