International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 06 , June - 2026

State of the Art on Proactive Anomaly Detection in CI/CD Pipelines through Hybrid Analysis of Logs and Metrics.

Oumar DIAGNE[1], Ibrahima FALL[2]

[1] Université Cheikh Anta Diop (UCAD), Dakar, Sénégal
[2] UMI 209 UMMISCO – UCAD, École Supérieure Polytechnique (ESP),
Université Cheikh Anta Diop (UCAD), Dakar, Sénégal

Abstract. Beyond automating repetitive tasks, CI/CD pipelines have become the operational backbone that ensures the continuous integration, validation, and deployment of modern applications. However, the growing complexity of these processing chains, combined with their critical role in service delivery, makes their monitoring and proactive anomaly detection essential. These dynamic environments generate substantial volumes of observability data, primarily textual logs and numerical metrics. This paper presents a systematic literature review of anomaly detection techniques applied to CI/CD workflows. We first examine existing approaches based on log analysis—including syntactic processing, pattern mining, and deep learning—as well as metric-based methods, such as statistical techniques, change detection, and time-series forecasting. We also review the few attempts at hybrid analysis that combine both modalities. From this review, we identify several major gaps: the predominantly reactive nature of current systems, the lack of unified frameworks for correlating logs and metrics, and insufficient adaptation to the dynamic characteristics of CI/CD environments. Finally, we synthesize the key scientific challenges that remain to be addressed and outline promising directions for developing an intelligent, proactive, and explainable framework dedicated to anomaly detection in CI/CD pipelines.

Keywords: CI/CD, Proactive Anomaly Detection, Log Analysis, Metric Analysis, Deep Learning, Visual Analytics, AIOps, DevOps.

1 INTRODUCTION

The adoption of DevOps methodologies has profoundly transformed the software development lifecycle, positioning continuous integration and continuous deployment (CI/CD) pipelines at the core of value delivery [1]. These pipelines automate critical stages ranging from code validation to production deployment, enabling organizations to release software more frequently and with higher quality. However, this agility comes with increased complexity. CI/CD environments are inherently distributed, ephemeral, and heterogeneous, orchestrated by a multitude of tools—such as Jenkins, GitLab CI, and GitHub Actions—and running on cloud-native infrastructures [2].

This complexity results in the massive generation of telemetry data, consisting primarily of two modalities:

- **Logs**: semi-structured textual messages generated at each stage of the pipeline (compilation, testing, deployment), rich in semantic information about the progress of operations and the errors encountered.
- **Metrics**: time series of numerical values (job execution times, CPU/memory consumption, success/failure rates), providing a quantitative view of system health and performance.

  A failure in a CI/CD pipeline can have major consequences, ranging from delivery delays to the unavailability of critical services. Yet traditional monitoring, which is often reactive and based on static thresholds, shows its limitations when faced with the dynamic nature and volume of these data

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 06 , June - 2026

[3]. Incidents are frequently detected only after they have impacted production, and root cause analysis (RCA) is lengthy and laborious due to the lack of correlation between logs and metrics.

In response to this observation, the research community is moving toward more intelligent approaches. These approaches aim to jointly analyze logs and metrics for proactive detection and contextualized explanation of anomalies. However, as our analysis highlights, existing work suffers from several limitations:

- It treats logs and metrics in isolation [4, 5].
- It is rarely adapted to the specificities of CI/CD pipelines (parallel jobs, flaky tests, short life cycles) [6].
- The few hybrid approaches, often originating from the AIOps field [7], do not provide an integrated framework combining deep learning-based detection with explainability through visual analytics for the specific context of software delivery chains.

This article aims to provide a comprehensive and critical state-of-the-art review in order to lay the foundations for such a framework, which we name LogPipeGuard. The remainder of this paper is organized as follows: Section 2 presents the review methodology. Section 3 describes CI/CD pipelines as data-generating systems. Sections 4 and 5 analyze log-based and metric-based detection approaches, respectively. Section 6 explores the emerging field of hybrid analysis and its applications in AIOps. Section 7 synthesizes the identified gaps and proposes a roadmap for LogPipeGuard. Finally, we conclude this state-of-the-art review.

2 REVIEW METHODOLOGY

Our literature review followed a systematic approach to identify, select, and analyze relevant works. Searches were conducted across major academic databases: IEEE Xplore, the ACM Digital Library, SpringerLink, Scopus, Google Scholar, and DBLP. We also explored preprint archives such as arXiv to capture the most recent trends.

Keywords used: Queries combined terms from the fields of software engineering, distributed systems, and machine learning. The main keywords, used individually or in Boolean combinations, were: ("CI/CD pipeline" OR "continuous integration" OR "continuous deployment" OR "DevOps") AND ("anomaly detection" OR "failure prediction" OR "root cause analysis") AND ("log analysis" OR "log parsing" OR "system logs") AND ("metric analysis" OR "time series" OR "monitoring data") AND ("machine learning" OR "deep learning" OR "neural networks") AND ("hybrid" OR "multimodal" OR "correlation") AND ("AIOps" OR "observability").

Inclusion and exclusion criteria: We included articles published between 2010 and 2024, a period covering the rise of DevOps practices and deep learning techniques. The inclusion criteria were: (i) peer-reviewed journal or conference articles; (ii) works proposing anomaly detection methods in software systems; (iii) studies specifically addressing CI/CD pipelines or similar distributed environments. Tutorials, posters, and articles not written in English were excluded. In total, 75 articles were thoroughly reviewed, of which thirty-one were retained for this synthesis.

2 CI/CD PIPELINES AS DATA-GENERATING SYSTEMS

2.1 Architecture and Operation

A CI/CD pipeline can be modeled as a directed graph of interconnected stages, where each node represents an automated task [8]. Typical stages include:

- **Build**: compilation of source code, dependency resolution, and creation of executable artifacts.
- **Test**: execution of automated test suites (unit, integration, end-to-end, performance).
- **Deploy**: deployment of the validated artifact to target environments (staging, production).

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 06 , June - 2026

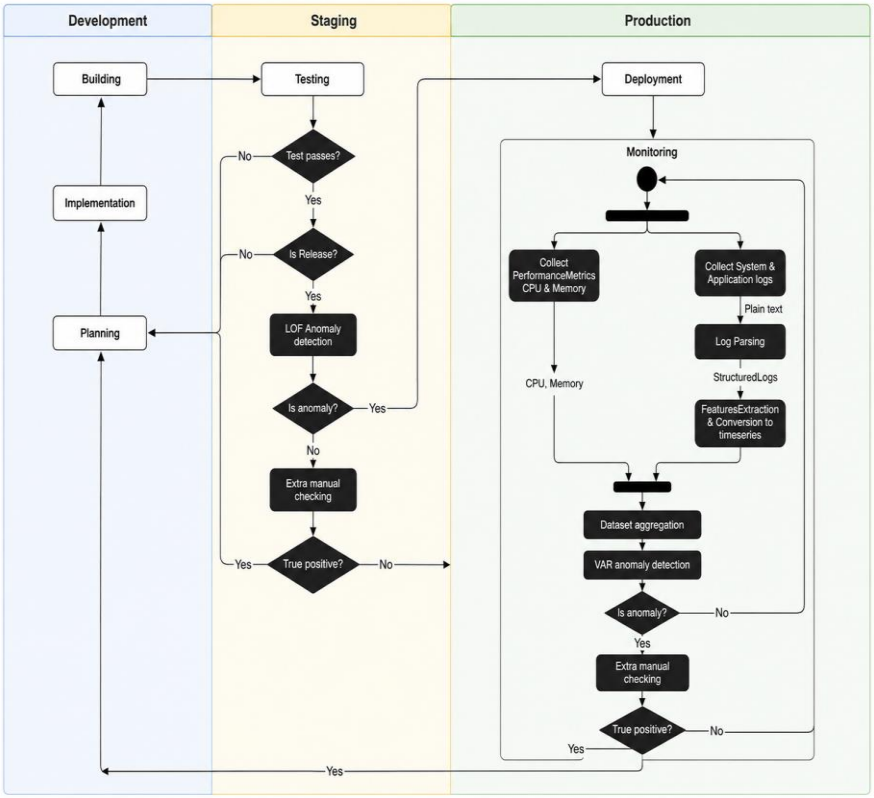

Figure 1. Directed graph of stages

The execution of these stages is orchestrated by CI/CD servers (Jenkins, GitLab CI, GitHub Actions, Tekton), which provision often ephemeral runtime environments (Docker containers, virtual machines). This dynamic and distributed nature constitutes a major source of complexity for data collection and for establishing a baseline of normal behavior.

2.2 Typology of CI/CD Logs

Logs take the form of textual traces produced by tools and scripts at each stage of the pipeline. They can be categorized as follows:

- **Build logs**: compiler messages, warnings, syntax or dependency errors.
- **Test logs**: assertion results, test execution traces, coverage reports. These are particularly prone to the phenomenon of flaky tests (non-deterministic tests) [9].
- **Deployment logs**: outputs from configuration tools (Ansible, Terraform), service startup messages, connection errors to databases or external APIs.

The variability of formats is a major challenge. Each tool (Maven, Gradle, Pytest) produces logs with its own syntax, making generic parsing complex.

2.3 Definition of Anomalies in CI/CD Pipelines

An anomaly in this context is a deviation from the expected behavior of the pipeline, potentially impacting the quality or speed of delivery. A taxonomy can be established along several dimensions:

- **By nature**: functional (build failure) vs. performance (abnormal slowness).
- **By scale**: global (the entire pipeline fails) vs. local (a single test fails in a flaky manner).
- **By persistence**: transient (latency spike), intermittent (unstable network), gradual (memory leak).
- **By causality**: related to code, infrastructure, configuration, or an external dependency.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 06 , June - 2026

Proactive detection aims to identify precursor signals of these anomalies before they cause an explicit failure (e.g., a gradual increase in build time preceding a timeout).

3 Log-Based Anomaly Detection

Log analysis is a cornerstone of system reliability. Its evolution has followed that of machine learning, progressing from rule-based approaches to complex deep learning models.

3.1 From Parsing to Structured Representations

The first and crucial step is the transformation of raw textual logs into an exploitable structure. Traditional parsers, based on regular expressions, are fragile and require constant manual adaptation. Automatic parsers such as Drain [10] (based on a fixed-depth tree) and Spell [11] (based on the longest common subsequence) have become established for efficiently extracting log templates (the constant part of the message) and their parameters. This parsing step is a recognized point of fragility for subsequent analysis stages [12].

3.2 Classical Approaches (Clustering, Markov)

Before the advent of deep learning, anomaly detection in logs relied primarily on clustering and Markov models. Clustering-based approaches [13] group similar log messages in order to identify majority patterns (considered normal) and flag as anomalies isolated messages or those belonging to very small clusters. Markov models, on the other hand, model the sequential ordering of events by learning transition probabilities between log templates; a sequence is then considered anomalous if its likelihood falls below a threshold [14]. Although these methods are interpretable and computationally inexpensive, they struggle to capture long-term dependencies and adapt poorly to dynamic environments such as CI/CD pipelines. The table below summarizes the relative strengths and weaknesses of these classical approaches compared to deep learning methods.

Table 1. Comparison of classical and modern approaches.

Criterion	Clustering	Markov Models	Deep Learning
Interpretability	High	High	Low
Computational cost	Low	Low	High
Temporal order	No	Yes (local)	Yes (global)
Long-range dependencies	No	No	Yes
Adaptation to CI/CD logs	Moderate	Low	High

3.3 Deep Learning (DeepLog, LogAnomaly, Autoencoders, Transformers)

The introduction of deep learning marked a turning point. DeepLog [15] is a foundational work that models log streams as sequences and uses a recurrent neural network (LSTM) to predict the next log template. An anomaly is detected when the actual observed template is not among the most probable templates predicted by the model. DeepLog also incorporates an online update mechanism to adapt to new log patterns, which is essential in evolving environments.

Since DeepLog, numerous variants have emerged:

- **LogAnomaly [16]**: Combines an LSTM sequence model with semantic representation of logs via word embeddings to detect semantic anomalies (an abnormal log message even if it follows a normal sequence).

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 06 , June - 2026

- **Autoencoders for reconstruction**: Models such as autoencoders [17] are trained to reconstruct "normal" log sequences. A sequence with a high reconstruction error is classified as anomalous. These models are known to be more robust to noise.
- **Transformers**: More recently, Transformer-based architectures [18] have been explored for their ability to capture long-range dependencies through the attention mechanism, outperforming LSTMs on certain detection tasks [19].

3.4 Limitations of Logs-Only Approaches

Given that metrics provide quantitative visibility into system health and performance, relying exclusively on them for anomaly detection presents significant limitations:

- **Lack of semantic context**: Metrics indicate *that* a problem is occurring (e.g., increased CPU usage, extended build time) but rarely *why*. They do not provide information about the specific operations, errors, or configuration changes that may have caused the anomaly.
- **Threshold sensitivity**: Metric-based detection often relies on static or dynamic thresholds. Setting these thresholds appropriately is challenging: overly sensitive thresholds generate numerous false positives, while overly permissive ones allow genuine anomalies to go undetected. This difficulty is particularly problematic in dynamic CI/CD environments where normal behavior varies across pipelines and over time.
- **Inability to distinguish anomaly types**: A metric deviation alone does not differentiate between a performance degradation, a functional failure, or a transient issue such as a flaky test. This lack of discriminative power complicates root cause analysis and incident prioritization.
- **Limited capacity for gradual degradation detection**: While metrics are well-suited for capturing trends, detecting slow, incremental changes—such as memory leaks or gradual increases in build time—requires sophisticated forecasting models. Simple threshold-based methods fail in such scenarios.
- **Reactive detection**: Like log-based approaches, metric-based methods often detect anomalies after they have occurred or when a critical threshold has been breached, offering little proactive warning.
- **Data sparsity and alignment challenges**: In CI/CD pipelines, metrics may be collected at varying granularities or only during specific stages. Sparse or irregularly sampled metrics complicate time-series analysis and make it difficult to correlate with log events.

4 Hybrid Log/Metric Analysis

Given the evident complementarity of logs and metrics, research is moving toward hybrid analysis, often within the broader framework of AIOps (Artificial Intelligence for IT Operations

4.1 Correlation and Aggregation

The first steps toward hybridization are often simple yet effective for root cause analysis.

- **Temporal windowing**: When a metric becomes anomalous, the corresponding time window in the logs is isolated to search for error messages. This is the classical approach of many monitoring platforms (e.g., correlation in Grafana) [26]. However, it is reactive and manual.
- **Score aggregation**: An anomaly score is computed independently for logs and for metrics. These scores are then combined (by average, maximum, or a logical model) to produce a final score. This approach does not capture fine-grained interactions between modalities.

4.2 Feature Fusion and Multimodal Models

More advanced integration aims to create shared representations.

- **Early fusion (feature fusion)**: Features are extracted from logs (e.g., template embeddings via Word2Vec [27]) and from metrics (e.g., sliding window statistics) over the same time window.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 06 , June - 2026

These vectors are concatenated to form a hybrid representation, which is then used to train a classifier (SVM, Random Forest) [28]. The challenge lies in perfect temporal alignment and dimensionality management.

- **Late fusion and two-tower models**: One network (e.g., CNN or LSTM) processes logs, while another processes metrics. Their latent representations are then combined (concatenation, weighted sum) before the final decision layer. This allows each network to learn representations specific to its modality [29].
- **Cross-attention mechanisms**: Inspired by Transformers, these mechanisms enable the model to dynamically learn which parts of the logs are most relevant for explaining a variation in metrics, and vice versa. This is a highly active research direction for truly contextual fusion [30].

4.3 Focus on ChronoCorrelator (Explainability)

The ChronoCorrelator paper [29] proposes an original visual analytics approach for log/metric correlation. It introduces local correlation measures (such as δfront*δfront* and fi*fi*) to quantify, at a given instant, the strength and temporal lag of the influence between a log pattern and a metric. The interactive interface allows engineers to visually explore these correlations to diagnose incidents. Although its application is not specific to CI/CD pipelines, this work is fundamental for the explainability component of LogPipeGuard. It demonstrates that it is possible to go beyond mere detection to provide operators with causal understanding.

4.4 Application to CI/CD Pipelines (State of the Art)

The application of these hybrid techniques to the specific domain of CI/CD pipelines is still in its infancy. A few recent works are beginning to emerge:

- **Root cause analysis in pipelines**: Approaches based on causal graphs [6] attempt to link infrastructure metrics to CI/CD job failures in order to identify the primary cause of a failure.
- **Flaky test detection**: Methods combine test log analysis (failure messages) with execution metrics (duration, memory consumption) to characterize and predict the non-deterministic nature of tests [9].

However, to the best of our knowledge, there is no unified and intelligent framework that offers:

- Proactive detection through multimodal deep learning.
- Anomaly explainability through visual log/metric correlation.
- Adaptation to the specificities of CI/CD pipelines (parallelism, flaky tests, ephemerality).

5 Synthesis, Challenges, and Proposal of a LogPipeGuard Framework

5.1 Synthesis of Gaps in the State of the Art

Our in-depth analysis of the literature allows us to identify the following major gaps, which constitute the scientific challenges of our thesis work:

Table 2. Synthesis of identified gaps in the state of the art.

Domain	Identified Gaps
Log-based detection	Lack of quantitative context, reactivity, parsing dependency, difficulty in modeling gradual degradations.
Metric-based detection	Lack of semantic context, difficulty in explaining the *why* of an anomaly, sensitivity to threshold tuning.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 06 , June - 2026

Hybrid analysis	Rarity of works applied to CI/CD, absence of a formal framework for real-time fusion, lack of publicly annotated datasets, focus on detection rather than proactive prediction.
Explainability	Deep learning models are often "black boxes." Few works integrate visual analytics mechanisms to make detection interpretable for DevOps engineers.

To the best of our knowledge, only a few public datasets exist for anomaly analysis in CI/CD pipelines, such as LogHub (CI/CD logs) or Google's Flaky Tests dataset, but none combine annotated logs and metrics.

5.2 Architecture of the Proposed Framework

To address these gaps, we propose the conceptual framework LogPipeGuard, whose overall architecture is illustrated below. This framework aims to integrate the strengths of existing approaches while mitigating their weaknesses.

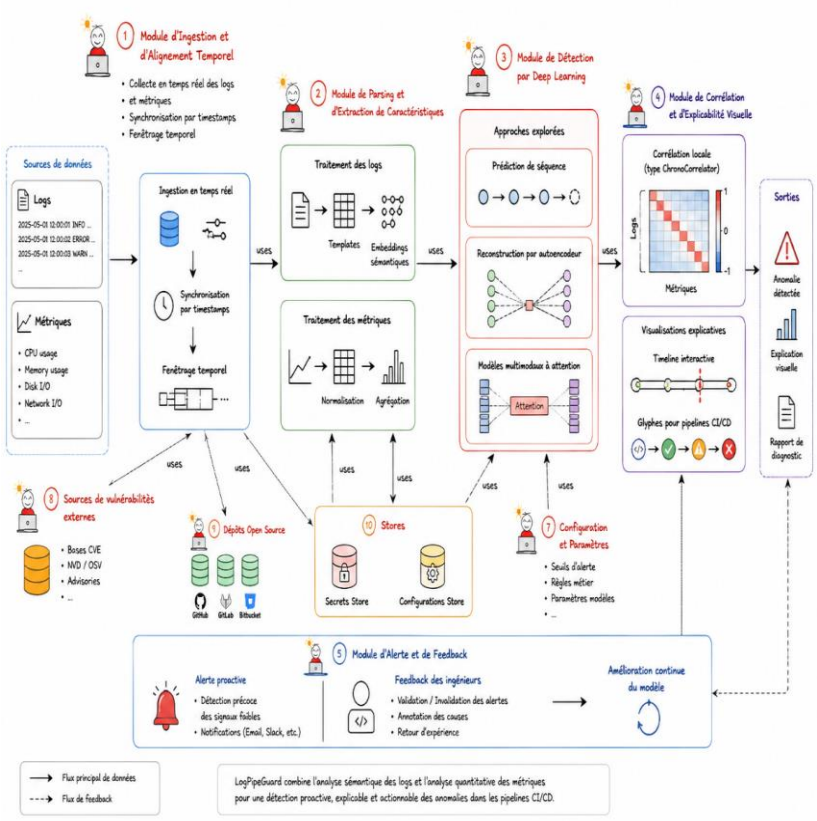

Figure 2: Overall system architecture

LogPipeGuard is built upon three fundamental principles:

- **Hybridization**: Fusing the semantic information of logs with the quantitative dynamics of metrics for more robust and contextual detection.
- **Proactivity**: Moving beyond mere detection to identify precursor signals of anomalies, enabling alerts before explicit failure occurs.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 06 , June - 2026

- **Explainability**: Providing DevOps engineers with visual and intuitive explanations of the probable causes of detected anomalies, drawing inspiration from ChronoCorrelator.

The architecture is composed of several key modules:

- **Ingestion and Temporal Alignment Module**: Real-time collection of logs and metrics, synchronization via timestamps, and windowing.
- **Parsing and Feature Extraction Module**: Transformation of logs into templates and semantic embeddings; normalization and aggregation of metrics.
- **Deep Learning Detection Module**: Implementation and comparison of multiple architectures (sequence prediction, autoencoder reconstruction, multimodal attention-based models) for anomaly detection.
- **Correlation and Visual Explainability Module**: Adaptation of local correlation measures (ChronoCorrelator-type) to link detected anomalies to the underlying logs and metrics. Creation of dedicated visualizations (glyphs for CI/CD pipelines, interactive timelines) to facilitate diagnosis.
- **Alerting and Feedback Module**: Generation of proactive alerts and a feedback mechanism from engineers for continuous model improvement.

5.3 Roadmap

The implementation of LogPipeGuard will follow the roadmap below, which corresponds to a series of publishable scientific contributions:

- **State of the art and positioning**: This article constitutes the first building block.
- **Development of a hybrid detection model**: Design and implementation of a deep learning architecture fusing logs and metrics, with validation on initial datasets.
- **Robustness study of models**: Comparative analysis of forecasting vs. reconstruction approaches in the presence of noise and the specificities of CI/CD logs (flaky tests).
- **Design of the visual explainability module**: Adaptation of ChronoCorrelator to the CI/CD context and evaluation through a user study with DevOps engineers.
- **Full integration and validation of the framework**: Presentation of LogPipeGuard as a whole, with evaluation on industrial use cases and publication of an open-source prototype.

6 Conclusion

Proactive anomaly detection in CI/CD pipelines is a strategic challenge for ensuring the reliability of modern software delivery chains. This state-of-the-art review has highlighted the richness and complementarity of log and metric data, but also the limitations of approaches that treat them separately. Hybrid analysis, although still emerging in the specific context of CI/CD, appears to be the most promising path toward accurate, early, and interpretable detection.

Building upon a critical analysis of the scientific challenges identified, we have proposed the LogPipeGuard framework. This framework aims to address the identified gaps by integrating multimodal deep learning models for proactivity and visual analytics techniques for explainability, all adapted to the unique challenges posed by CI/CD environments. The proposed research roadmap will guide our future work toward the development and validation of this framework, with the objective of providing both a significant scientific contribution and a practical tool for the DevOps community.

REFERENCES

1. N. Forsgren, J. Humble, and G. Kim, Accelerate: The Science of Lean Software and DevOps. IT Revolution Press, 2018.
2. M. Shahin, M. A. Babar, and L. Zhu, "Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices," IEEE Access, 2017.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 06 , June - 2026

3. M. Trevelin, et al., "A survey on monitoring and observability in microservices-based systems," Journal of Systems and Software, 2023.
4. M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS), pp. 1285-1298, ACM, 2017.
5. K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Söderström, "Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding," in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD), pp. 387-395, 2018.
6. L. Wang, et al., "Graph-based root cause analysis for CI/CD pipelines," in IEEE/ACM International Conference on Automated Software Engineering (ASE), 2023.
7. Y. Dang, Q. Lin, and P. Huang, "AIOps: Real-World Challenges and Research Innovations," in IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), 2019.
8. J. Humble and D. Farley, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley, 2010.
9. Q. Luo, F. Hariri, L. Eloussi, and D. Marinov, "An empirical analysis of flaky tests," in Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE), pp. 643-653, 2014.
10. P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An online log parsing approach with fixed depth tree," in IEEE International Conference on Web Services (ICWS), pp. 33-40, IEEE, 2017.
11. M. Du and F. Li, "Spell: Streaming parsing of system logs," in IEEE 16th International Conference on Data Mining (ICDM), pp. 859-864, IEEE, 2016. doi: 10.1109/ICDM.2016.0103.
12. X. Wu, H. Li, and F. Khomh, "On the effectiveness of log representation for log-based anomaly detection," Empirical Software Engineering, vol. 28, no. 6, p. 137, 2023. doi: 10.1007/s10664-023-10364-1.
13. Q. Lin, H. Zhang, J.-G. Lou, Y. Zhang, and X. Chen, "Log clustering based problem identification for online service systems," in Proceedings of the 38th International Conference on Software Engineering Companion (ICSE), pp. 102-111, ACM, 2016.
14. A. Haque, A. DeLucia, and E. Baseman, "Markov chain modeling for anomaly detection in high performance computing system logs," in Proceedings of the Fourth International Workshop on HPC User Support Tools (HUST), pp. 1-8, ACM, 2017. doi: 10.1145/3152493.3152559.
15. W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, and L. Song, "LogAnomaly: Unsupervised Detection of Sequential and Quantitative Anomalies in Unstructured Logs," in Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI), pp. 4739-4745, 2019.
16. R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Deep autoencoder based anomaly detection for system logs," in IEEE International Conference on Soft Computing and Machine Intelligence (ISCMI), 2021.
17. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017.
18. S. Nedelkoski, J. Bogatinovski, A. Acker, J. Cardoso, and O. Kao, "Self-attentive classification-based anomaly detection in unstructured logs," in IEEE International Conference on Data Mining (ICDM), pp. 1196-1201, IEEE, 2020.
19. A. Ganesan, M. Almasri, and S. Bagchi, "Detecting performance anomalies in cloud-based systems: A survey," ACM Computing Surveys, vol. 54, no. 6, pp. 1-35, 2021.
20. R. Killick, P. Fearnhead, and I. A. Eckley, "Optimal detection of changepoints with a linear computational cost," Journal of the American Statistical Association, vol. 107, no. 500, pp. 1590-1598, 2012.
21. S. J. Taylor and B. Letham, "Forecasting at scale," The American Statistician, vol. 72, no. 1, pp. 37-45, 2018.
22. F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in IEEE International Conference on Data Mining (ICDM), pp. 413-422, IEEE, 2008.
23. J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, "USAD: UnSupervised Anomaly Detection on multivariate time series," in Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD), 2020.
24. Y. Zhang, et al., "A systematic approach to performance monitoring and root cause analysis for microservices," in IEEE International Conference on Web Services (ICWS), 2019.
25. T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," arXiv preprint arXiv:1301.3781, 2013.
26. S. Nedelkoski, J. Bogatinovski, A. Acker, and O. Kao, "Self-supervised log parsing and anomaly detection," in IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 231-238, IEEE, 2020.
27. H. Guo, J. Yang, J. Liu, J. Bai, B. Wang, Z. Li, T. Zheng, B. Zhang, J. Peng, and Q. Tian, "LogFormer: A Pre-train and Tuning Pipeline for Log Anomaly Detection," in Proceedings of the AAAI Conference on Artificial Intelligence, 2023.
28. Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "A transformer-based framework for multivariate time series representation learning," in Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), 2022.
29. M. A. M. M. van Dortmont, S. van den Elzen, and J. J. van Wijk, "ChronoCorrelator: Enriching Events with Time Series," Computer Graphics Forum, vol. 38, no. 3, pp. 387-399, 2019.
30. T. M. Mäntylä, Y. Wang, and J. Nyyssölä, "LogLead: Fast and Integrated Log Loader, Enhancer, and Anomaly Detector," arXiv preprint arXiv:2311.11809, 2023.