Systematic Mapping of Fault Monitoring Techniques Based on Data Mining

Paulo Ricardo Fernandes Rodrigues
Group of Computer Networks, Software Engineering and Systems (GREat)
Federal University of Ceará (UFC)
Fortaleza, Ceará, Brazil
paulofr@alu.ufc.br

Valéria Lelli
Computer Science Department
Group of Computer Networks, Software Engineering and Systems (GREat)
Federal University of Ceará (UFC)
Fortaleza, Ceará, Brazil
valerialelli@ufc.br

ABSTRACT

The increasing complexity and interconnectivity of modern software systems have made fault detection a significant challenge. Traditional testing techniques often fall short, especially in dynamic and large-scale environments. To address this issue, various real-time monitoring approaches based on data mining have been proposed. These approaches, through continuous system analysis and the use of anomaly detection techniques, are capable of identifying faults in software systems. This work presents a systematic mapping of the literature on these monitoring techniques, aiming to identify implementation challenges and provide a comprehensive overview of the current state of research. The mapping was conducted using the Scopus database and 84 studies were selected to help answer the research questions defined in this work.

This analysis identified the main fault detection techniques, such as neural networks and Support Vector Machines (SVMs), the types of systems monitored, including Cloud and Big Data platforms, common fault categories like cybersecurity and performance, and the evaluation metrics used, such as the F1-score and prediction time. Despite these established approaches, the study also revealed significant challenges that hinder progress, including the difficulty of labeling data for supervised models, the complexity of interpreting the generated models, and the need for computationally efficient monitoring.

KEYWORDS

fault detection, anomaly detection, data mining

1 INTRODUCTION

The increasing complexity of modern software systems, driven by factors such as scale and heterogeneity, has made managing their reliability an ever-growing challenge [30]. In this context, traditional verification and validation techniques, which often rely on predefined test cases, have significant limitations, as they rarely consider the dynamic state and history of a running system, preventing them from capturing its actual complexity [72]. This gap leaves systems vulnerable to faults that may only manifest during live operation, highlighting the need for more adaptive and automated approaches.

In: Proceedings of the Brazilian Symposium on Multimedia and the Web (WebMedia'2025). Rio de Janeiro, Brazil. Porto Alegre: Brazilian Computer Society, 2025.

ISSN 2966-2753

Addressing the limitations of traditional testing techniques requires adaptive and automated strategies capable of handling the system's dynamic nature during runtime. In this context, real-time monitoring emerges as a promising solution, enabling the proactive detection of faults.

Among the various fault detection strategies, anomaly detection combined with data mining is particularly relevant, as it aims to identify deviations from expected behavioral patterns by analyzing operational data and extracting meaningful patterns to enable effective and proactive fault detection.

This work investigates fault monitoring systems that leverage anomaly detection and data mining to improve the reliability of complex software systems in real-time environments. We conducted a systematic mapping following the guidelines of Kitchenham and Charters [51]. Initially, we defined the goal and the research questions. Then, we defined the search string, selected the database, and established the selection criteria. We identified 84 studies by executing the search string in the Scopus database and conducted backward snowballing. Based on the analysis of these studies, we characterized the current research landscape, including prevalent techniques, target systems, challenges, and open research directions.

Our analysis reveals that fault monitoring's versatility is applied in diverse areas with distinct patterns and data usage, most notably in the fields of software aging and cybersecurity. Software aging, the progressive degradation of system performance and resource availability, manifests as a series of faults over time (*e.g.,* memory leaks, resource exhaustion), making it an ideal candidate for proactive monitoring. Likewise, cybersecurity threats like intrusions can be framed as fault issues. An intrusion is a critical runtime problem that manifests as an unauthorized or erroneous state during execution, and the anomaly detection techniques used for operational faults can be effectively applied to identify such security breaches by detecting deviations from normal system behavior.

Beyond these application domains, this work also reveals that general solutions, network faults, cybersecurity threats, and performance issues represent the most prevalent categories in the literature. Each fault type demonstrates distinct data requirements: general fault detection predominantly relies on log data, performance monitoring usually utilizes operating system metrics, while network and cybersecurity solutions primarily analyze traffic data. The data mining landscape encompasses diverse approaches, including supervised machine learning algorithms, unsupervised normal state modeling, and time series forecasting, with supervised machine learning algorithms and normal state modeling methods being the most widely adopted. Furthermore, our findings emphasize the critical importance of preprocessing, particularly for log-based solutions, and reveal that evaluation practices favor classification metrics such as precision, recall, and F1-score over accuracy, reflecting the inherent class imbalance challenges in fault detection scenarios. Based on the analysis of these studies, we characterized the current research landscape, including prevalent techniques, target systems, challenges, and open research directions.

This paper is organized as follows. Section 2 establishes the background, defining key concepts and terminology. Section 3 presents the related studies. Section 4 describes the research methodology for systematic mapping. Section 5 presents the results of the research analysis. Section 5 discusses the implications of the findings and outlines the open research directions. Finally, Section 7 concludes the paper and presents future directions.

2 BACKGROUND

This work adopts the terminology defined by the IEEE [44]. An error is a human action that results in incorrect software behavior or state. A fault is the manifestation of an error during the software's execution. The observable consequence of a fault is a failure, which is the software's inability to perform its expected function within established limits. This creates a causal chain where an error can manifest as a fault at runtime, potentially causing a system failure.

Fault detection can be approached in two primary ways: conventional testing and runtime monitoring. Traditional testing relies on test oracles, which, according to Shahamiri et al. [76], are responsible for comparing a system's observed behavior against a predefined expected result to distinguish between correct and incorrect outcomes. In contrast, runtime monitoring does not depend on predefined oracles but instead analyzes the system continuously to detect deviations from its normal behavior. This latter approach is particularly suited for the complex and dynamic systems that are the focus of this study.

The runtime monitoring techniques explored in this mapping are based on anomaly detection. An anomaly is a pattern of behavior that deviates from what is expected. This work adopts the perspective of Chandola et al. [19], which frames anomaly detection as a broad research field focused on identifying such deviations across different domains. Consequently, by analyzing large volumes of data generated during a system's execution, identifying anomalous patterns becomes a practical and effective strategy for detecting faults in software systems.

3 RELATED WORK

Early research in failure prediction established foundational taxonomies that remain conceptually relevant today. Salfner et al. [72] provided a comprehensive overview of failure prediction methods, categorizing approaches into four main types based on the data used: fault tracking, symptom monitoring, reported errors, and undetected error auditing. Since failures typically result from underlying faults, many of these prediction techniques inherently involve detecting fault manifestations, making them conceptually similar to fault detection. While influential, this work predates the significant advancements in machine learning and complex systems of the last decade and does not follow a systematic search protocol.

Building on this foundation, Grohmann et al. [31] conducted a systematic mapping study focused on Service Level Objective (SLO) failure prediction. Their work introduced a three-dimensional classification framework encompassing prediction target, time horizon, and modeling type. Within this framework, they identified two primary anomaly detection paradigms: time-series forecasting and normal state modeling. However, their analysis omits discussion of data preprocessing and practical implementation hurdles.

Recent efforts have produced more focused examinations of specific aspects of fault diagnosis. Jieyang et al. [46] reviewed data-driven approaches, but concentrated on fault diagnosis in industrial Big Data systems, with an emphasis on physical machinery rather than general software systems. Another line of work, exemplified by Bhandari et al. [12], investigates data quality issues, but their focus is on datasets for fault prediction (*i.e.* static analysis before execution), which is distinct from the real-time fault detection during system operation that we address.

4 METHODOLOGY

We conducted this research using a systematic mapping approach, following the guidelines established by Kitchenham and Charters [51]. This section details the systematic mapping protocol, encompassing the formulation of research questions (RQs), the identification of data sources, the development of the search strategy, and the selection criteria.

4.1 Goal and Research questions

The goal of this research is to investigate monitoring approaches for anomaly detection based on data mining. To guide this systematic mapping, we formulated six research questions aimed at characterizing and organizing the existing literature on data mining-based fault monitoring. These questions focus on identifying the contexts, techniques, and challenges involved in applying anomaly detection methods to fault monitoring in software systems. The research questions are as follows:

- **RQ1.** What types of software systems have been shown to benefit from fault monitoring through data mining-based anomaly detection techniques?
- **RQ2.** What categories of faults have been detected through data mining-based anomaly detection, and what types of system artifacts are utilized in the process?
- **RQ3.** Which data mining-based anomaly detection algorithms and techniques are applied for fault monitoring?
- **RQ4.** Which data preprocessing techniques are employed to enhance the effectiveness of data mining-based fault monitoring?
- **RQ5.** How are the data mining-based fault monitoring approaches evaluated?
- **RQ6.** What are the primary challenges and limitations encountered when implementing data mining-based anomaly detection techniques for fault monitoring?

4.2 Search strategy

For the selection of research sources, the Scopus database was chosen for its extensive coverage of software engineering (SE) publications and established indexing quality. While additional databases could provide complementary coverage, the substantial volume of results and required filtering made their inclusion infeasible within the project timeline. To assist the search string formulation, we first defined the PICOC *(*Population, Intervention, Comparison, Outcome and Context) as follows:

- Population: real-time fault monitoring techniques;
- Intervention. data mining;
- Comparison: the purpose of this mapping is to identify findings on a topic rather than to compare them. Thus, we did not use this criterion.
- Outcomes: applications, metrics, artifacts, techniques, methods and tools.
- Context: software

Based on the PICOC, a structured search string was defined to capture studies proposing fault monitoring in software systems using data mining-based anomaly detection techniques operating in real time. Furthermore, this strategy combines synonyms for fault and anomaly detection, execution phase, and software engineering with data mining-related terms. These combinations were designed to ensure the retrieval of studies relevant to runtime fault monitoring in the context of software engineering. The search string is presented below:

```
(("fault detection" OR "fault verification" OR
"fault diagnosis" OR "fault identification" OR
"fault localization" OR "fault monitoring")
OR
("anomaly detection" OR "anomaly verification" OR
"anomaly diagnosis" OR "anomaly identification" OR
"anomaly localization" OR "anomaly monitoring"))
AND
("runtime" OR "run-time" OR "realtime" OR "real time" OR
"real-time" OR "dynamic" OR "online" OR "on-line" OR
"execution time" OR "operational phase")
AND
("software engineering" OR "software system" OR
"software application" OR "software development" OR
"software process")
AND
("data mining" OR "machine learning" OR
"pattern recognition" OR "statistical learning" OR
"statistical analysis" OR "data analysis")
```

To complement the database search and partially mitigate the single-database limitation, *Backward snowballing* was conducted on the selected studies. This snowballing reviewed the reference lists of the selected studies.

4.3 Inclusion and exclusion criteria

To ensure the relevance and quality of retrieved studies, specific inclusion and exclusion criteria were established to guide the systematic selection process. Since fault detection, by the definition [44], occurs at runtime, the criteria ensure the selection of studies on runtime detection using statistical and data mining approaches, while excluding traditional testing methods that depend on test oracles or predefined expected results. Furthermore, the scope is explicitly limited to software engineering contexts, thereby excluding studies that focus on sensor data analysis or hardware-level fault detection.

Inclusion Criteria

(1) The study must present a fault detection approach using data mining.
(2) The study must present a fault detection approach that operates at runtime.
(3) The study must be available in full text.
(4) The study must be written in English.
(5) The study must be within the context of SE.

Exclusion Criteria

(1) The study focuses on hardware fault monitoring approaches.
(2) The study uses sensor data or other hardware-specific data for monitoring.
(3) The study uses test oracles for fault detection.
(4) The study has been retracted or is unavailable at the time of screening or during the review process.

4.4 Conducting

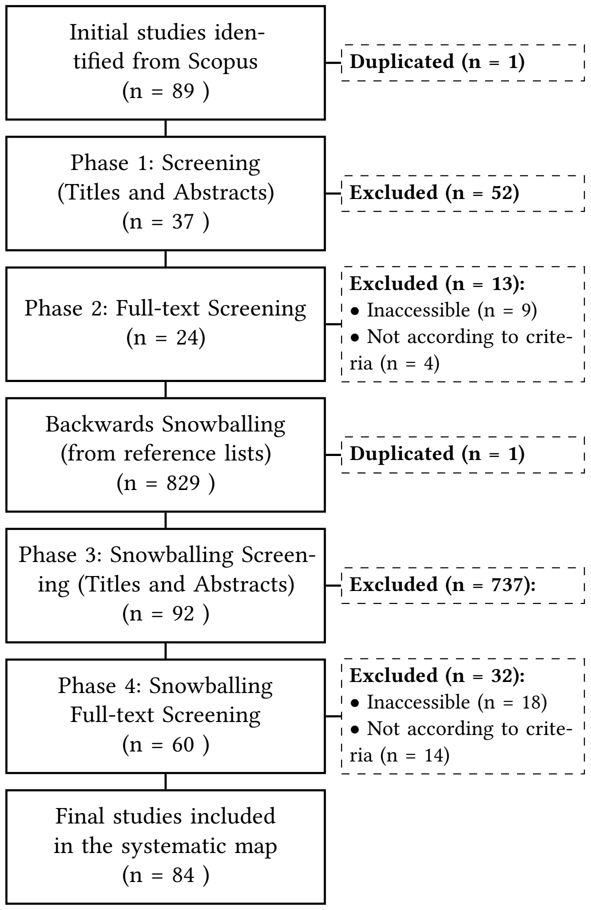
During the first phase, the search string was executed in the Scopus database, yielding 90 studies for initial evaluation. The removal of a single duplicate publication resulted in 89 studies for evaluation. Title, abstract, and keyword screening was performed on all retrieved studies. During this phase, 52 studies were excluded for not meeting the inclusion and exclusion criteria by presenting hardware-related approaches, relying on hardware data, or not aligning with the scope of real-time detection. After this step, 37 studies remained for more detailed evaluation.

Of the 37 studies retained from the first phase, nine studies were excluded due to full-text inaccessibility. The remaining 28 studies underwent full-text screening against the inclusion and exclusion criteria. Following this assessment, four additional studies were excluded for failing to meet the established criteria, resulting in 24 studies selected for inclusion.

Backward snowballing was conducted on the 24 selected studies to identify additional relevant literature from their reference lists. This process generated 830 studies. Title, abstract, and keyword screening was performed on these studies, resulting in the selection of 93 studies for further evaluation. Of these, 18 studies were excluded due to full-text inaccessibility and one duplicate was removed, leaving 74 studies for full-text review. Following assessment, 14 studies were rejected for not meeting the inclusion criteria, yielding 60 additional studies for inclusion.

The systematic mapping study comprised 84 studies in total (24 from the primary search process and 60 from backward citation analysis), representing the final corpus for analysis. Figure 1 presents a flowchart containing all the studies selected in each phase and Figure 2 illustrates the distribution of studies included in this systematic mapping over the years. The complete research protocol and filled data extraction form are publicly available at https://github.com/PauloFRC/Systematic-Mapping-Fault-Monitoring-Appendix.

Figure 1: Flowchart of the Conduction Process


Initial studies identified from Scopus (n = 89)
Duplicated (n = 1)
Phase 1: Screening (Titles and Abstracts) (n = 37)
Excluded (n = 52)
Phase 2: Full-text Screening (n = 24)
Excluded (n = 13):
• Inaccessible (n = 9)
• Not according to criteria (n = 4)
Backwards Snowballing (from reference lists) (n = 829)
Duplicated (n = 1)
Phase 3: Snowballing Screening (Titles and Abstracts) (n = 92)
Excluded (n = 737):
Phase 4: Snowballing Full-text Screening (n = 60)
Excluded (n = 32):
• Inaccessible (n = 18)
• Not according to criteria (n = 14)
Final studies included in the systematic map (n = 84)

5 RESULTS

This section presents the consolidated results of the systematic mapping, structured around the research questions. The discussion offers a comprehensive overview of the findings, advancing knowledge in fault monitoring through data mining-based anomaly detection techniques.

5.1 RQ1. What types of software systems have been shown to benefit from fault monitoring through data mining-based anomaly detection techniques?

The analysis of system types targeted for fault monitoring reveals significant diversity in the application domains of the selected studies. Table 1 summarizes the types of software systems identified in the literature.

As presented in Table 1, several studies (25 out of 84) do not specify the particular system type they target, suggesting either general-purpose approaches or a lack of clarity in defining the target environment. This ambiguity represents a significant limitation, as it complicates comparative evaluation and makes it difficult for practitioners to assess applicability to their specific contexts. Among the studies that do specify their target systems, **Online systems** represent the largest specified category with 21 studies, followed by **Cloud systems** with 12 studies, reflecting the growing importance of cloud-native architectures and web-based applications in modern software engineering. **Big data systems** also constitute a significant category with 9 studies. The distribution also shows emerging trends in fault monitoring for **Container-based systems** (4 studies) and **IoT systems** (3 studies), while **distributed systems** maintain a presence with 3 studies. Additionally, traditional **high-performance computing**, **large-scale industrial systems**, **datacenters**, and **clusters** each account for two studies. The *Others* category encompasses 11 studies focusing on specialized domains such as embedded systems and mobile applications.

The prevalence of studies targeting large-scale and complex system types, including cloud, big data and high-performance systems, reflects that these systems benefit most from data mining-based fault monitoring for fault detection. Studies such as Wang et al. [85] and Monni et al. [64] highlight that the growing complexity of modern software amplifies challenges related to maintainability, citing cloud systems as an example. Uchida et al. [83] argue that modern software systems, due to their complexity, generate enormous volumes of data, which makes analysis challenging. The rise of Internet of Things (IoT) systems further compounds these challenges. Yahyaoui et al. [89] note that IoT systems are being increasingly used, which increases the complexity in the analysis and management of the generated data. As highlighted by Branco et al. [15], a major contributor to this complexity in the IoT domain is the difficulty of ensuring interoperability among heterogeneous devices, while simultaneously managing diverse communication protocols, bandwidth and device resource limitations. These examples illustrate how the complexity and scale of modern systems require advanced monitoring approaches to ensure their reliability and efficiency.

The prevalence of studies targeting large-scale and complex system types, including cloud, big data and high-performance systems, reflects that these systems benefit most from data mining-based fault monitoring for fault detection. Multiple studies highlight how the growing complexity of modern software amplifies challenges related to maintainability and monitoring. Wang et al. [83] and Monni et al. [63] cite cloud systems as prime examples of this complexity, while Uchida et al. [81] argue that modern software systems generate enormous volumes of data that make analysis increasingly challenging. The rise of Internet of Things (IoT) systems further compounds these challenges. Yahyaoui et al. [87] note that IoT systems are being deployed with increasing frequency, adding complexity to data analysis and management. This complexity is particularly evident in the IoT domain, where Branco et al. [16] emphasize the need to ensure interoperability across heterogeneous devices while managing diverse communication protocols and complex architectures. These examples collectively illustrate how the complexity and scale of modern systems necessitate advanced monitoring approaches to ensure their reliability and efficiency.

The infeasibility of traditional testing becomes even more evident in Big Data systems that process streaming data, as described in [8, 23, 75]. These studies highlight that the high complexity and

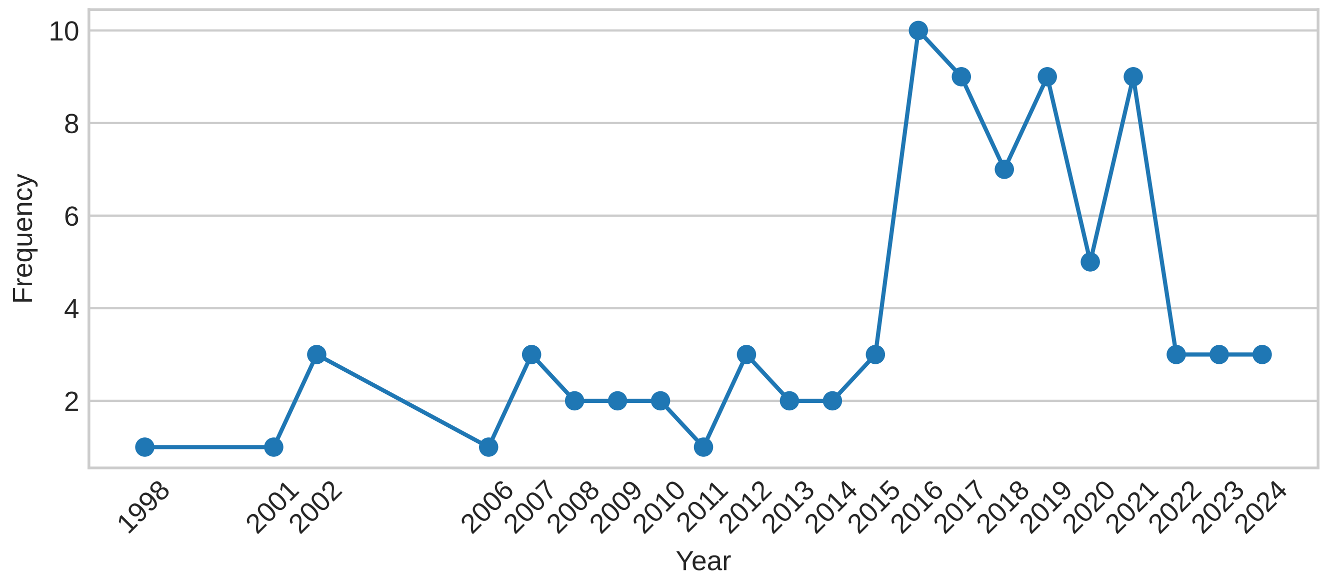


Figure 2: Number of studies per year in the mapping.

massive volume of data make manual or static fault detection impractical, requiring automated approaches. In the same way, Tuncer et al. [82] and Liu et al. [59] reinforce the importance of automated anomaly detection in critical and high-performance systems, in which failures can result in significant economic and operational impacts. However, this data generated on a large scale can be useful for training models that assist in fault monitoring, allowing for failure prevention.

5.2 RQ2. What categories of faults have been detected through data mining-based anomaly detection, and what types of system artifacts are utilized in the process?

To answer this question, fault categories were mapped to system artifacts used as a data source for each study. The complete mapping is presented in Table 2. The table's rows represent the fault types, while the columns denote the monitored artifacts. A citation in a cell indicates that a study addresses the corresponding fault-artifact combination. In this classification, studies that use diverse datasets, artificially generated data or that do not specify a specific type of fault are categorized under the "General" fault type.

Table 2 reveals strong correlations between fault types and the artifacts used to detect them. For instance, **faults related to Network** and **Security** are predominantly detected using **Network Traffic** data. Similarly, **Performance faults** are most frequently diagnosed by analyzing **OS Data** (*e.g.,* CPU usage, system calls) and **Application Logs**, which provide direct insight into resource consumption and application behavior.

It is worth noting that many of the techniques found are generic enough to work with different types of faults, even if experiments have only been carried out for a specific type. For example, Brown et al. [17] test their approach with data on faults related to **Information Security** but highlight that the method could be adapted for other fault categories. Therefore, the map in Table 2 reflects the specific experimental choices made in the literature rather than a rigid limitation of the techniques themselves.

The prominence of **log-based analysis, evident from the densely populated Application Logs column in the table, merits special attention**. Notably, most of these entries refer to general fault types, underscoring the versatility and expressive power of logs in fault detection. Jia et al. [45] show how logs are one of the most important pieces of information for detecting faults in a system. This is because logs are highly efficient at capturing system behavior and its overall state, as they can record events, operations, and potential problems. Therefore, due to their expressive power, logs allow for the detection of almost any type of fault. This combination of the wealth of information contained in logs and advanced techniques for automating analysis makes log-based approaches one of the most promising and applicable for fault detection in software systems. The functioning of log analysis techniques is discussed in the answer to research question RQ4.

In the domain of security, data mining-based monitoring is particularly advantageous. As highlighted by Flanagan et al. [27], unsupervised techniques are capable of detecting "zero-day" vulnerabilities and "Black Swan" events, overcoming the limitations of traditional methods that rely on known threat signatures. As indicated by the large number of studies in this category within the table, the **Information Security** domain has received substantial research focus.

[1]

[1]A "zero-day" vulnerability is a software flaw unknown or without a patch available, leaving it open to exploitation until it is discovered and fixed. A "Black Swan" is a rare, unpredictable event with severe consequences that is often rationalized in hindsight.

Table 1: System types used for fault monitoring

System type	Studies	Quantity
Unspecified type	[10, 11, 13, 17, 22, 24, 26, 32, 35, 38–40, 49, 50, 52, 53, 65, 66, 68, 71, 74, 77, 78, 80, 83]	25
Online systems	[3–6, 20, 25, 27, 28, 33, 41, 43, 54, 56, 59, 62–64, 67, 69, 85, 95]	21
Cloud systems	[4–6, 28, 41, 43, 54, 56, 63, 64, 67, 69]	12
Big Data systems	[8, 9, 16, 23, 57, 60, 61, 70, 75]	9
Container-based systems	[7, 36, 58, 93]	4
IoT systems	[37, 88, 89]	3
Distributed systems	[84, 91, 92]	3
High performance systems	[55, 82]	2
Large-scale industrial systems	[18, 42]	2
Datacenters	[14, 87]	2
Clusters	[29, 34]	2
Others	[1, 2, 21, 47, 48, 55, 73, 79, 86, 90, 94]	11

Similarly, these techniques are highly effective in detecting performance problems by identifying systemic degradation, resource bottlenecks, and faults associated with software aging. As shown in the **Performance** row of the table, which is among the most heavily populated categories, the most common approach involves monitoring OS Data (16 studies). Several studies [9, 14, 32, 82, 85] utilize metrics such as CPU and memory usage to detect performance-related faults.

5.3 RQ3. Which data mining-based anomaly detection algorithms and techniques are applied for fault monitoring?

Anomaly detection techniques can be classified in different ways. Grohmann et al. [31] classify anomaly detection techniques into two types: *time series forecasting-based detection* and *normal-state modeling-based detection*. Both models expect system behavior and identify anomalies by deviations, but differ in implementation. *Time series forecasting* uses historical data to predict future behavior, flagging anomalies when real data significantly diverges. *Normal-state modeling* employs explicit models, clustering, and adaptive thresholds to detect deviations from predefined standard patterns, without relying on future data prediction, though time series may aid model creation.

Within the *normal-state modeling* category, techniques can be further divided into supervised methods (*e.g.,* K-nearest neighbors, decision trees) and unsupervised methods (*e.g.,* clustering). For clarity and analytical purposes, this study organizes fault detection techniques based on anomaly detection into three categories:

(1) time series forecasting methods
(2) supervised classification approaches
(3) unsupervised normal state modeling techniques

Table 3 summarizes the selected studies according to the type of approach used for their solution. We observed that **supervised classification techniques** using machine learning and **techniques based on normal-state modeling** of a system are well represented, with 37 approaches using supervised classification and 48 based on unsupervised normal-state modeling. However, only nine of the selected studies employ time series forecasting as a method for anomaly detection.

The following discussion further examines each of the previously introduced categories, highlighting their main approaches and representative techniques.

5.3.1 *Supervised Classifier-based Techniques.*

Within the studies that use classifier-based techniques, different machine learning classifiers were applied. Four algorithm families were frequently found: deep learning models (13 studies), Support Vector Machines (SVMs) (12 studies), tree-based models (9 studies), and Bayesian models (6 studies). Additionally, a significant number of papers (12 studies) applied other, less common classifier models. Table 4 organizes these studies according to the machine learning classifiers they used.

Decision tree-based algorithms operate through the recursive partitioning of data to create a hierarchical model. Four studies demonstrated their effective application. Catovic et al. [18], for example, introduced the *Linnaeus* technology, a log classification pipeline that integrates the Random Forest algorithm. Tuncer et al. [82] also used decision trees, Random Forest, and AdaBoost for fault detection in high-performance systems, highlighting that these models have advantages such as high explainability and a natural capacity for multiclass classification.

Bayesian methods were employed in six of the selected studies. Its main advantage is computational efficiency, which makes it suitable for processing large volumes of data. Of particular interest is the study by Shafiq et al. [75], which successfully implemented a parallelized version of Naive Bayes in a Big Data environment, allowing not only for fault detection but also their classification into multiple categories. This efficiency is possible due to the assumption of independence between attributes, and as mentioned by uk Kim and Hariri [84], the algorithm demonstrates good performance even when this premise is not strictly met.

Deep learning models stood out for their great flexibility, being applied in various architectures to solve the problem of anomaly detection. For instance, in the study by Elahi et al. [23], logs were preprocessed, transformed into two-dimensional attributes, and treated as images, then classified by a single-layer neural network. In a more complex approach, the study by Le and Zhang [52]

Table 2: Map of Fault Types to Monitored System Artifacts

Fault Type	Monitored Artifact						
	General	Application Logs	OS data	Application Data	User feedback	Network Traffic	Unspecified
General	[23, 59]	[18, 22, 35, 39, 40, 52, 54, 55, 57, 60, 66, 83, 87, 92, 94]	[28, 34, 42, 47, 56, 67–69, 73]	[28, 56]		[20, 28]	[41, 71]
Network		[17]	[21, 64, 78]	[7]		[10, 13, 21, 24–27, 37, 48, 49, 70, 74, 77, 78, 80, 84, 88, 89]	
Information Security		[16, 17]	[21, 58, 78, 84, 93]	[7]		[13, 21, 24, 26, 27, 37, 48, 49, 70, 74, 77, 78, 88, 89]	[11, 90]
Performance		[8, 61, 79]	[1–3, 5, 6, 9, 14, 32, 33, 53, 62, 63, 79, 82, 85, 91]	[33, 53]		[62]	[4, 43, 65]
Tensor shapes				[86]			
User interface					[95]		
System components		[75]					
I/O		[29]					

Table 3: Anomaly detection approaches

Approach	Study	Quantity
Supervised machine learning-based classification	[8, 9, 14, 18, 28, 29, 34, 36, 37, 40, 42, 47–50, 52–56, 58, 60, 66, 69, 70, 73, 75, 77, 79, 82–84, 86, 88, 89, 94, 95]	37
Unsupervised normal state modeling	[1, 4–7, 10, 11, 13, 14, 16, 17, 21–24, 26–28, 32, 35, 36, 39–41, 43, 47, 49, 50, 57, 58, 61–66, 69, 71, 74, 77, 78, 80, 85, 87, 90–93]	48
Time series prediction	[1–4, 25, 33, 38, 59, 67]	9
Other	[68]	1
Unspecified	[20, 81]	2

Table 4: Machine learning algorithms

Algorithm	Study	Quantity
Deep learning models	[8, 9, 36, 37, 40, 42, 52, 54, 56, 58, 60, 83, 94]	13
SVMs	[18, 28, 29, 37, 47, 55, 69, 70, 77, 79, 84, 88]	12
Tree-Based models	[18, 36, 37, 48, 58, 70, 73, 82, 89]	9
Bayesian models	[18, 34, 70, 75, 84, 89]	6
Other	[14, 18, 36, 47, 49, 50, 53, 55, 66, 70, 86, 95]	12

used a Transformer-based model to analyze the semantics of logs without the need for traditional parsing. Many studies, including [22, 42, 54, 56, 94], applied a recurrent neural network model with long short-term memory (LSTM) to classify data in an industrial environment, an architecture especially effective for handling sequential data. This variety of applications demonstrates the ability of neural networks to adapt to different data types and levels of complexity.

From the selected studies, 12 used **Support Vector Machine (SVM)** related approaches which demonstrated effectiveness in handling both single-class and multi-class anomaly detection scenarios. A notable contribution is the study by Fu et al. [28], which proposed a hybrid anomaly detection framework that adaptively combines one-class SVM and binary SVM through a sliding scale weighting strategy. This approach addresses a common challenge in cloud computing environments where datasets are initially composed entirely of normal data but gradually incorporate anomalous records over time. The main advantages of using SVM include its ability to handle high-dimensional data through kernel functions, robust performance with unbalanced datasets, and the capacity to provide both binary and multi-class classification for anomaly type identification.

5.3.2 *Time Series-based Techniques.*

Anomaly detection techniques based on time series forecasting use models that learn the system's normal behavior from historical data to then predict its future state. A fault is identified when the actual observed state significantly diverges from what was predicted by the model. Although it is a powerful approach, only nine of the selected studies used it.

For instance, in the study by Liu et al. [59], pre-processed data is fed into a "collaborative machine," an architecture proposed by the authors that captures relationships between monitored artifacts across both attribute and temporal dimensions. Following this feature extraction stage, the system employs a recurrent neural network with long short-term memory (LSTM) to predict future system behavior. Similarly, Le and Zhang [25] demonstrated the use of time-series forecasting for anomaly detection by implementing an enhanced version of the Holt-Winters statistical method, a triple exponential smoothing technique that captures trends in time series data, to predict normal system behavior, where deviations from these predictions serve as indicators of potential faults.

5.3.3 *Unsupervised Normal State Modeling Techniques.*

Unsupervised normal state modeling techniques consist of creating a model that represents the expected behavior of the system under normal operating conditions. Anomaly detection occurs when the actual observed state significantly deviates from this established model. Among the selected studies, 48 have used this approach.

The approaches range from applying clustering algorithms and neural networks to using concepts from physics. For example, Flanagan et al. [27] performed clustering on network traffic data and treats them as two-dimensional images so that a convolutional neural network (CNN) can detect changes. In a different manner, Monni et al. [64] proposed the use of free energy calculation, a concept originating from physics, through a stochastic neural network known as a Restricted Boltzmann Machine (RBM) to model the system's normal state and identify anomalies based on energy variations.

Other studies adapted or applied established anomaly detection techniques. Gross et al. [32], for instance, applied the Multivariate State Estimation Technique (MSET) to identify symptoms of software aging, while Wang et al. [85] proposed a new algorithm based on Permutation Entropy to analyze the complexity of time series. Recurrent neural networks were also used for this purpose, as in [17], which feeds an LSTM with log messages to model the system's expected behavior.

5.4 RQ4. Which data preprocessing techniques are employed to enhance the effectiveness of data mining-based fault monitoring?

Data preprocessing is a fundamental step for the effectiveness of anomaly detection approaches, transforming collected artifacts into formats suitable for analysis. The techniques employed vary significantly according to the nature of the artifact, with a clear distinction between the handling of logs and metric data.

For system logs, which are textual and unstructured data, preprocessing is more extensive. The most common technique is parsing, which consists of separating messages into a fixed template and dynamic variables, as was done by Uchida et al. [83]. After parsing, steps such as tokenization, filtering of irrelevant words, and vectorization are applied to convert the text into a numerical format. However, more recent studies show a trend towards automation and sophistication in this stage. Le and Zhang [52], for example, dispensed traditional parsing and uses the *BERT* model to extract semantics directly from the raw text. In contrast, Nedelkoski et al. [66] employed a self-supervised approach for the model to learn the log structure autonomously.

For other artifacts, such as system metrics and network traffic, preprocessing focuses on other tasks. Dimensionality reduction is frequently applied to decrease noise and computational cost, using techniques like Principal Component Analysis (PCA), as in the study by Wang et al. [85], or intelligent feature selection, as in the studies by Yahyaoui et al. [89] and Jones et al. [48]. Data normalization, seen in the studies by Monni et al. [64] and Tuncer et al. [82], is another common step in ensuring that different metrics are on a consistent scale. Additionally, data cleaning and balancing techniques were identified, such as replacing missing values and undersampling classes to correct imbalances in the training set as used in the study by Huch et al. [42].

5.5 RQ5. How are the data mining-based fault monitoring approaches evaluated?

The evaluation of fault monitoring approaches predominantly relies on two categories of metrics: classification and performance measures. Classification metrics dominate the landscape, reflecting the common treatment of fault detection as a binary classification problem.

Table 5 gives an overview of the most frequently used metrics in the selected studies.

As shown in Table 5, **Precision** and **Recall** emerged as the most popular metrics, appearing in 47 studies, followed by detailed classification components (FPR, TPR, FNR, TNR) employed in 39 papers, and **F1-Score** used in 32 studies. Notably, Accuracy, while appearing in 17 studies, was less prevalent than the other primary metrics. This preference for Precision, Recall, and F1-Score over Accuracy is well-justified in the fault detection domain, where faults represent rare events and a model could achieve misleadingly high Accuracy simply by classifying all instances as "normal". The substantial adoption of granular classification measures (FPR, TPR, FNR, TNR) in 39 studies indicates a commitment to comprehensive evaluation beyond aggregate metrics, enabling detailed analysis of model behavior across different error types.

Recognizing the inherent class imbalance challenge in fault detection, two papers employed the Matthews Correlation Coefficient (MCC), a metric specifically designed to provide robust evaluation in imbalanced scenarios. This relatively low adoption suggests an opportunity for broader implementation of imbalance-aware metrics in future fault detection research.

The evaluation process presented particular challenges for supervised approaches that required labeled data. To address this limitation, the studies adopted several innovative labeling strategies. These included the injection of synthetic anomalies into controlled environments (*e.g.,* [8, 82, 84]), retrospective labeling of events preceding known failures (*e.g.,* [18, 42]), and the use of user feedback to identify problematic instances (*e.g.,* [95]). These approaches, among

Table 5: Evaluation metrics used in the selected studies

Metric	Studies	Quantity
Accuracy	[5, 6, 14, 21, 25, 29, 37, 64–67, 79, 80, 84–87, 89]	18
Precision / Recall	[8, 13, 20–22, 24, 25, 28, 29, 35–42, 47, 52, 54–59, 61, 63–67, 69, 71, 73, 77, 79, 80, 82–86, 91–95]	47
F1-score	[8–10, 13, 14, 20, 22, 35–40, 52, 54–56, 59, 63, 64, 66, 71, 73, 82–86, 88, 91, 92, 94]	32
FPR, TPR, FNR or TNR	[4–7, 13, 16, 17, 20–22, 24, 25, 29, 34, 36, 38–40, 43, 49, 54, 55, 58, 60, 61, 64, 65, 70, 73, 77, 80, 82–84, 86–88, 90, 93]	39
MCC	[83, 92]	1

others, employed across the reviewed studies, enabled rigorous evaluation while working within the constraints of limited labeled fault data.

Beyond classification metrics, computational performance evaluation emerged as a critical consideration for assessing the real-world viability of fault monitoring solutions, focusing on two primary dimensions: temporal behavior and resource utilization. Temporal performance analysis encompassed various time-related metrics, with studies adopting different perspectives based on their specific requirements. Yahyaoui et al. [89] focused on throughput for Big Data processing efficiency, while studies [34, 59, 70, 82, 93] conducted comprehensive temporal analysis measuring execution time across both training and prediction phases. Detection latency represented another crucial temporal dimension, where Tuncer et al. [82] calculated the time between fault injection and effective identification, with studies [14, 21, 78] similarly measuring time-to-identification metrics. Complementing these assessments, resource utilization evaluation addressed system overhead concerns, as studies such as [21, 23, 64, 67] measured the impact of their fault monitoring approaches on CPU or memory consumption to ensure the monitoring solution did not impose excessive computational burden on the target system.

5.6 RQ6. What are the primary challenges and limitations encountered when implementing data mining-based anomaly detection techniques for fault monitoring?

The challenges in implementing fault monitoring permeate different stages of the process, from data collection to model evaluation. This section analyzes four identified challenges and the proposed strategies to overcome them.

Data Collection and Labeling. In the data collection stage, two main approaches are identified: the use of existing datasets in repositories, as in the study by Monni et al. [23] and Elahi et al. [23], or the direct monitoring of the target system. The second approach is generally preferable, as the effectiveness of detection techniques when applied to systems different from those used in training may lack adequate validation. A significant challenge is that most of the approaches identified in the studies are supervised, requiring labeled data for training. To circumvent the high cost of manual labeling, different strategies have been proposed: uk Kim and Hariri [84] used synthetic anomaly injection, while Catovic et al. [18] defined records that precede failures as anomalous. Although these alternatives reduce costs, they may be less efficient than precise manual labeling.

Data Preprocessing. The preprocessing of collected data presents specific challenges depending on the chosen detection model. While decision tree-based algorithms are relatively robust to different attribute scales, others, like SVMs, may require normalization to ensure adequate performance. Liu et al. [59] emphasized this challenge, noting that input data types can exhibit substantial variation and require normalization to establish a unified scale range for effective model training. Furthermore, dimensionality reduction techniques, such as PCA and feature selection, may be necessary to improve efficiency and reduce noise in the data. The practical importance of such techniques is exemplified by Yahyaoui et al. [89], who evaluated their proposed solution using a dataset containing over 100 features. Their work underscored the need for feature selection to identify the most relevant attributes and improve overall detection performance.

The predominance of logs as a data source requires specific preprocessing techniques to convert unstructured text into formats suitable for analysis. This includes tokenization for segmenting words or symbols, parsing for extracting relevant patterns, and filtering irrelevant information. Depending on the model, it may also be necessary to apply vectorization techniques, such as TF-IDF, to represent logs efficiently. Handling incomplete records, removing duplicates, and grouping similar events are additional steps that can significantly impact the quality of the results obtained in fault detection.

Processing Large Volumes of Data. Performance in systems that generate large volumes of data represents another significant challenge, as demonstrated in [23, 75] with Big Data systems. The high flow of information requires efficient solutions for storage, processing, and analysis.

Strategies to deal with this challenge include preprocessing techniques, such as dividing data into blocks or clusters for efficient processing distribution, as well as dimensionality reduction through algorithms like PCA and feature selection techniques. Furthermore, the use of parallelizable algorithms is fundamental to improve scalability and response time. Models based on KNN and Naive Bayes can be optimized for distributed execution.

Detection Model Selection. In selecting the detection model, it is crucial to consider multiple requirements beyond classification performance, such as interpretability, scalability, computational cost, and control of false positives and negatives. Neural network-based models, although powerful and versatile, generally offer lower interpretability than tree-based models. Deep neural networks can identify complex patterns and make highly accurate predictions, but their internal workings are often considered a "black box," making it difficult to explain the results. This can be a problem in domains where transparency is essential, such as cybersecurity. In contrast, decision tree-based models offer greater interpretability, allowing one to visualize the logic behind the decisions and identify which attributes most influence the results.

Furthermore, it is crucial to balance the model's sensitivity and specificity to reduce false positives and false negatives, considering the severity of the type of fault to be detected. In this context, data balancing techniques and the optimization of specific metrics, such as precision, recall, or F1-score, are frequently used to enhance the model's performance according to the system's requirements.

6 DISCUSSION AND THREATS TO VALIDITY

This systematic mapping has highlighted the significant advantages that real-time monitoring techniques present in comparison to traditional testing methods. The analysis of the selected studies allowed for the identification of the necessary steps for implementing anomaly detection techniques and the inherent challenges of this process.

This work complements previous contributions, such as [31, 72], by providing an updated overview of the area, considering that many of the analyzed works are recent. Furthermore, this research addresses essential issues for the practical implementation of data mining-based fault monitoring solutions, including the analysis of systems that benefit from this strategy, the types of faults that can be monitored, and the fundamental preprocessing techniques for the effectiveness of these solutions.

Among the main findings, which directly answer the proposed research questions, the following stand out:

(1) the identification of large-scale and high-complexity systems, such as Big Data, cloud, and IoT, as those that benefit most from these approaches;
(2) the categorization of fault types, such as performance and Information Security problems;
(3) the identification of monitored artifacts, such as logs and network traffic;
(4) the description of data mining-based anomaly detection techniques and algorithms, such as decision trees and neural networks;
(5) the analysis of the most used preprocessing techniques, such as tokenization and normalization;
(6) the identification of evaluation metrics for anomaly detection models, such as precision and execution time; and
(7) an understanding of the challenges in implementing data mining-based fault monitoring systems.

6.1 Threats to Validity

This systematic mapping has some limitations. Regarding threats to internal validity, one key limitation is the selection of a single database. However, this threat was mitigated by applying backward snowballing, which enabled the identification of an additional 60 relevant studies, increasing the total to 84 studies. Nonetheless, the scope could be further enhanced by applying forward snowballing to analyze citing papers and expanding the search to include other academic databases.

Another threat to external validity is the terminology. We observed a variability of definitions and terminologies in the literature. As highlighted by Grohmann et al. [31], conducting an exhaustive review using search strings, as proposed by Kitchenham and Charters [51], proved to be challenging due to the expressive volume of results and the diversity of definitions used.

An example of this terminological complexity is the concept of fault, which, according to IEEE Std 1044:2009 [44], refers to the manifestation of an error during the execution of the software. However, many studies interpret fault detection more broadly, applying it outside of runtime contexts, for instance, in static code analysis. To mitigate this problem, we made adjustments and included other terms, such as "runtime" and "real-time", even though [44] defines a fault as inherently occurring at runtime. This adaptation was necessary because the absence of these terms resulted in searches that returned many studies related to fault prevention or prediction.

7 CONCLUSION AND FUTURE WORK

This systematic mapping provided a comprehensive overview of the field of data mining-based fault monitoring, following the rigorous guidelines proposed by Kitchenham and Charters [51]. The process began with a planning phase, where we defined our research questions, inclusion/exclusion criteria, and a precise search string. In the conduction phase, this search was executed on the Scopus database, initially yielding 90 studies. After a multi-stage screening of titles, abstracts, and full texts, 24 of these met our criteria. We then applied backward snowballing to this set, which identified an additional 60 relevant works, resulting in a final selection of 84 primary studies for analysis. In the final synthesis phase, data was extracted from these studies to systematically answer our research questions and map the current state of the field.

Thus, systems that benefit from data mining-based fault monitoring techniques were identified, especially large-scale and high-complexity systems such as Big Data, distributed computing, cloud, and IoT. These systems exhibit dynamic interactions between heterogeneous components, which makes traditional testing methods less effective. The types of detectable faults include performance issues, information security problems, and network traffic anomalies, using artifacts such as logs, resource usage, and network traffic.

The analysis classified anomaly detection techniques into three main categories: supervised classification, unsupervised modeling of normal states, and time series forecasting. In supervised learning, algorithms based on decision trees, SVMs, and Bayesian methods stand out. Yet, neural networks, including LSTMs, emerged as a prevailing technology, demonstrating their dominance across both supervised and unsupervised methods.

Regarding evaluation metrics, the most common were classification metrics, such as precision, recall, and F1-score. However, it would be interesting for more studies to incorporate operational performance metrics, such as throughput, CPU usage, and memory consumption, to evaluate the computational efficiency of the proposed techniques.

Among the main challenges identified are the difficulty in labeling data for training supervised models, the complexity in interpreting the models, and the need for computational efficiency in real-time monitoring.

Therefore, the main contribution of this systematic mapping lies in the comprehensive detailing of how data mining-based fault monitoring systems are implemented in practice. The study structurally categorized the different aspects of these implementations, identifying patterns among the types of systems monitored, the categories of faults detected, the artifacts used as data sources, the mining algorithms employed, and the related challenges. This systematic categorization provides a solid foundation for researchers and practitioners to understand the current state of the art, as well as providing practical guidelines for the implementation of new fault monitoring systems in different application domains.

This work contributes to an updated understanding of the field of data mining-based fault monitoring, providing a basis for future research and the development of more effective and robust solutions. By identifying approaches, challenges, and opportunities, this mapping directs future efforts to enhance the reliability and maintenance of increasingly complex software systems.

For future work, we will explore the feasibility of transfer learning to simplify data collection and model training. Moreover, establishing standardized benchmarks for comparative evaluation of fault monitoring techniques across different application scenarios would significantly enhance research and development in this domain, although the heterogeneity of the target systems presents substantial challenges. Furthermore, more comprehensive mappings, with multiple databases and *Forward snowballing* could provide a more complete view of the area.

REFERENCES

[1] [n. d.]. ARF-Predictor: Effective Prediction of Aging-Related Failure Using Entropy | Request PDF. ([n. d.]). https://doi.org/10.1109/TDSC.2016.2604381

[2] [n. d.]. A Methodology for Detection and Estimation of Software Aging. In *ResearchGate*. https://doi.org/10.1109/ISSRE.1998.730892

[3] [n. d.]. Modeling of Software Aging Based on Non-stationary Time Series. In *ResearchGate*. https://doi.org/10.1109/ISAI.2016.0046

[4] [n. d.]. Real-time Detection of Performance Anomalies for Cloud Services. In *ResearchGate*. https://doi.org/10.1109/IWQoS.2016.7590412

[5] [n. d.]. Statistical Technique for Online Anomaly Detection Using Spark over Heterogeneous Data from Multi-Source VMware Performance Data. In *ResearchGate*. https://doi.org/10.1109/BigData.2014.7004343

[6] ResearchGate [n. d.]. *UBL: Unsupervised Behavior Learning for Predicting Performance Anomalies in Virtualized Cloud Systems*. ResearchGate. https://doi.org/10.1145/2371536.2371572

[7] Amr S. Abed, Charles Clancy, and David S. Levy. [n. d.]. Intrusion Detection System for Applications Using Linux Containers. In *Security and Trust Management* (Cham, 2015), Sara Foresti (Ed.). Springer International Publishing, 123–135. https://doi.org/10.1007/978-3-319-24858-5_8

[8] A. Alnafessah and G. Casale. [n. d.]. TRACK-Plus: Optimizing Artificial Neural Networks for Hybrid Anomaly Detection in Data Streaming Systems. 8 ([n. d.]), 146613–146626. https://doi.org/10.1109/ACCESS.2020.3015346

[9] Ahmad Alnafessah and Giuliano Casale. 2020. Artificial Neural Networks Based Techniques for Anomaly Detection in Apache Spark. 23, 2 (2020), 1345–1360. https://doi.org/10.1007/s10586-019-02998-y

[10] Julija Asmuss and Gunars Lauks. 2015. Network Traffic Classification for Anomaly Detection Fuzzy Clustering Based Approach. In *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*. 313–318. https://doi.org/10.1109/FSKD.2015.7381960

[11] Matthew Bailey, Connor Collins, Matthew Sinda, and Gongzhu Hu. 2017. Intrusion Detection Using Clustering of Network Traffic Flows. In *2017 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. 615–620. https://doi.org/10.1109/SNPD.2017.8022786

[12] Kirti Bhandari, Kuldeep Kumar, and Amrit Lal Sangal. 2022. Data quality issues in software fault prediction: a systematic literature review. *Artif. Intell. Rev.* 56, 8 (Dec. 2022), 7839–7908. https://doi.org/10.1007/s10462-022-10371-6

[13] Monowar H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita. 2011. NADO: Network Anomaly Detection Using Outlier Approach. In *Proceedings of the 2011 International Conference on Communication, Computing & Security* (New York, NY, USA) *(ICCCS '11)*. Association for Computing Machinery, 531–536. https://doi.org/10.1145/1947940.1948050

[14] Peter Bodik, Moises Goldszmidt, Armando Fox, Dawn B. Woodard, and Hans Andersen. 2010. Fingerprinting the Datacenter: Automated Classification of Performance Crises. In *Proceedings of the 5th European Conference on Computer Systems (EuroSys '10)*. Association for Computing Machinery, New York, NY, USA, 111–124. https://doi.org/10.1145/1755913.1755926

[15] Karina Castelo Branco, Valéria Dantas, and Liana Carvalho. 2024. Interoperability Testing Guide for the Internet of Things. In *Proceedings of the 30th Brazilian Symposium on Multimedia and the Web* (Juiz de Fora/MG). SBC, Porto Alegre, RS, Brasil, 188–196. https://doi.org/10.5753/webmedia.2024.242058

[16] Jakub Breier and Jana Branišová. 2015. Anomaly Detection from Log Files Using Data Mining Techniques. In *Information Science and Applications* (Berlin, Heidelberg), Kuinam J. Kim (Ed.). Springer, 449–457. https://doi.org/10.1007/978-3-662-46578-3_53

[17] Andy Brown, Aaron Tuor, Brian Hutchinson, and Nicole Nichols. [n. d.]. Recurrent Neural Network Attention Mechanisms for Interpretable System Log Anomaly Detection. In *Proceedings of the First Workshop on Machine Learning for Computing Systems* (New York, NY, USA, 2018-06-12) *(MLCS'18)*. Association for Computing Machinery, 1–8. https://doi.org/10.1145/3217871.3217872

[18] Armin Catovic, Carolyn Cartwright, Yasmin Tesfaldet Gebreyesus, and Simone Ferlin. [n. d.]. 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)* (2021-05). 11–18. https://doi.org/10.1109/WAIN52551.2021.00008

[19] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM Comput. Surv.* 41, 3, Article 15 (July 2009), 58 pages. https://doi.org/10.1145/1541880.1541882

[20] M.Y. Chen, E. Kiciman, E. Fratkin, A. Fox, and E. Brewer. 2002. Pinpoint: Problem Determination in Large, Dynamic Internet Services. In *Proceedings International Conference on Dependable Systems and Networks*. 595–604. https://doi.org/10.1109/DSN.2002.1029005

[21] Boxiang Dong, Zhengzhang Chen, Hui (Wendy) Wang, Lu-An Tang, Kai Zhang, Ying Lin, Zhichun Li, and Haifeng Chen. 2017. Efficient Discovery of Abnormal Event Sequences in Enterprise Security Systems. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (New York, NY, USA) *(CIKM '17)*. Association for Computing Machinery, 707–715. https://doi.org/10.1145/3132847.3132854

[22] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017. DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA) *(CCS '17)*. Association for Computing Machinery, 1285–1298. https://doi.org/10.1145/3133956.3134015

[23] Manzoor Elahi, Xinjie Lv, Wasif Nisar, Imran Ali Khan, Ying Qiao, and Hongan Wang. 2008. DB-Outlier Detection Algorithm Using Divide and Conquer Approach over Dynamic DataStream. In *2008 International Conference on Computer Science and Software Engineering*, Vol. 4. 438–443. https://doi.org/10.1109/CSSE.2008.1053

[24] Eleazar Eskin, Andrew Arnold, Michael Prerau, Leonid Portnoy, and Sal Stolfo. 2002. A Geometric Framework for Unsupervised Anomaly Detection. In *Applications of Data Mining in Computer Security*, Daniel Barbará and Sushil Jajodia (Eds.). Springer US, 77–101. https://doi.org/10.1007/978-1-4615-0953-0_4

[25] prefix=de useprefix=true family=Assis, given=Marcos V. O., Joel J. P. C. Rodrigues, and Mario Lemes Proença. 2013. A Novel Anomaly Detection System Based on Seven-Dimensional Flow Analysis. In *2013 IEEE Global Communications Conference (GLOBECOM)*. 735–740. https://doi.org/10.1109/GLOCOM.2013.6831160

[26] Kieran Flanagan, Enda Fallon, Abir Awad, and Paul Connolly. 2017. Self-Configuring NetFlow Anomaly Detection Using Cluster Density Analysis. In *2017 19th International Conference on Advanced Communication Technology (ICACT)*. 421–427. https://doi.org/10.23919/ICACT.2017.7890124

[27] Kieran Flanagan, Enda Fallon, Paul Jacob, Abir Awad, and Paul Connolly. [n. d.]. 2D2N: A Dynamic Degenerative Neural Network for Classification of Images of Live Network Data. In *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)* (2019-01). 1–7. https://doi.org/10.1109/CCNC.2019.8651695

[28] Song Fu, Jianguo Liu, and Husanbir Pannu. 2012. A Hybrid Anomaly Detection Framework in Cloud Computing Using One-Class and Two-Class Support Vector Machines. In *Advanced Data Mining and Applications* (Berlin, Heidelberg), Shuigeng Zhou, Songmao Zhang, and George Karypis (Eds.). Springer, 726–738. https://doi.org/10.1007/978-3-642-35527-1_60

[29] Errin W Fulp, Glenn A Fink, and Jereme N Haack. [n. d.]. Predicting Computer System Failures Using Support Vector Machines. ([n. d.]).

[30] Matthias Galster, Uwe Zdun, Danny Weyns, Rick Rabiser, Bo Zhang, Michael Goedicke, and Gilles Perrouin. 2017. Variability and Complexity in Software Design: Towards a Research Agenda. *SIGSOFT Softw. Eng. Notes* 41, 6 (Jan. 2017), 27–30. https://doi.org/10.1145/3011286.3011291

[31] Johannes Grohmann, Nikolas Herbst, Avi Chalbani, Yair Arian, Noam Peretz, and Samuel Kounev. 2020. A Taxonomy of Techniques for SLO Failure Prediction in Software Systems. *Computers* 9, 1 (2020). https://doi.org/10.3390/computers9010010

[32] K.C. Gross, V. Bhardwaj, and R. Bickford. [n. d.]. Proactive Detection of Software Aging Mechanisms in Performance Critical Computers. In *27th Annual NASA Goddard/IEEE Software Engineering Workshop, 2002. Proceedings.* (2002-12). 17–23. https://doi.org/10.1109/SEW.2002.1199445

[33] M. Grottke, L. Li, K. Vaidyanathan, and K.S. Trivedi. 2006. Analysis of Software Aging in a Web Server. 55, 3 (2006), 411–420. https://doi.org/10.1109/TR.2006.879609

[34] Xiaohui Gu and Haixun Wang. 2009. Online Anomaly Prediction for Robust Cluster Systems. In *2009 IEEE 25th International Conference on Data Engineering*. 1000–1011. https://doi.org/10.1109/ICDE.2009.128

[35] Haixuan Guo, Shuhan Yuan, and Xintao Wu. 2021. LogBERT: Log Anomaly Detection via BERT. In *2021 International Joint Conference on Neural Networks (IJCNN)*. 1–8. https://doi.org/10.1109/IJCNN52387.2021.9534113

[36] Md Sadun Haq, Thien Duc Nguyen, Ali Şaman Tosun, Franziska Vollmer, Turgay Korkmaz, and Ahmad-Reza Sadeghi. 2024. SoK: A Comprehensive Analysis and Evaluation of Docker Container Attack and Defense Mechanisms. In *2024 IEEE Symposium on Security and Privacy (SP)*. 4573–4590. https://doi.org/10.1109/SP54263.2024.00268

[37] Mahmudul Hasan, Md. Milon Islam, Md Ishrak Islam Zarif, and M. M. A. Hashem. 2019. Attack and Anomaly Detection in IoT Sensors in IoT Sites Using Machine Learning Approaches. 7 (2019), 100059. https://doi.org/10.1016/j.iot.2019.100059

[38] Rin Hirakawa, Keitaro Tominaga, and Yoshihisa Nakatoh. 2020. Study on Software Log Anomaly Detection System with Unsupervised Learning Algorithm. *Advances in Intelligent Systems and Computing* 1152 AISC (2020), 122 – 128. https://doi.org/10.1007/978-3-030-44267-5_18 Cited by: 1.

[39] Rin Hirakawa, Hironori Uchida, Asato Nakano, Keitaro Tominaga, and Yoshihisa Nakatoh. 2021. Anomaly Detection on Software Log Based on Temporal Memory. 95 (2021), 107433. https://doi.org/10.1016/j.compeleceng.2021.107433

[40] Rin Hirakawa, Hironori Uchida, Asato Nakano, Keitaro Tominaga, and Yoshihisa Nakatoh. 2021. Large Scale Log Anomaly Detection via Spatial Pooling. 1 (2021), 188–196. https://doi.org/10.1016/j.cogr.2021.10.001

[41] Chengqiang Huang, Geyong Min, Yulei Wu, Yiming Ying, Ke Pei, and Zuochang Xiang. 2022. Time Series Anomaly Detection for Trustworthy Services in Cloud Computing Systems. 8, 1 (2022), 60–72. https://doi.org/10.1109/TBDATA.2017.2711039

[42] Fabian Huch, Mojdeh Golagha, Ana Petrovska, and Alexander Krauss. [n. d.]. Machine Learning-Based Run-Time Anomaly Detection in Software Systems: An Industrial Evaluation. In *2018 IEEE Workshop on Machine Learning Techniques for Software Quality Evaluation (MaLTeSQuE)* (2018-03). 13–18. https://doi.org/10.1109/MALTESQUE.2018.8368453

[43] Olumuyiwa Ibidunmoye, Thijs Metsch, and Erik Elmroth. 2016. Real-Time Detection of Performance Anomalies for Cloud Services. In *2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*. 1–2. https://doi.org/10.1109/IWQoS.2016.7590412

[44] IEEE. 2010. IEEE Standard Classification for Software Anomalies. *IEEE Std 1044-2009 (Revision of IEEE Std 1044-1993)* (2010), 1–23. https://doi.org/10.1109/IEEESTD.2010.5399061

[45] Tong Jia, Ying Li, Chengbo Zhang, Wensheng Xia, Jie Jiang, and Yuhong Liu. 2018. Machine Deserves Better Logging: A Log Enhancement Approach for Automatic Fault Diagnosis. In *2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. 106–111. https://doi.org/10.1109/ISSREW.2018.00-22

[46] Peng Jieyang, Andreas Kimmig, Wang Dongkun, Zhibin Niu, Fan Zhi, Wang Jiahai, Xiufeng Liu, and Jivka Ovtcharova. 2022. A systematic review of data-driven approaches to fault diagnosis and early warning. *J. Intell. Manuf.* 34, 8 (Sept. 2022), 3277–3304. https://doi.org/10.1007/s10845-022-02020-0

[47] Shi Jin, Zhaobo Zhang, Krishnendu Chakrabarty, and Xinli Gu. 2016. Accurate Anomaly Detection Using Correlation-Based Time-Series Analysis in a Core Router System. In *2016 IEEE International Test Conference (ITC)*. 1–10. https://doi.org/10.1109/TEST.2016.7805836

[48] Mike Jones, Masrufa Bayesh, and Sharmin Jahan. [n. d.]. Unlocking Deeper Understanding: Leveraging Explainable AI for API Anomaly Detection Insights. In *Proceedings of the 2024 16th International Conference on Machine Learning and Computing* (New York, NY, USA, 2024-06-07) *(ICMLC '24)*. Association for Computing Machinery, 211–217. https://doi.org/10.1145/3651671.3651738

[49] Georgios Kathareios, Andreea Anghel, Akos Mate, Rolf Clauberg, and Mitch Gusat. 2017. Catch It If You Can: Real-Time Network Anomaly Detection with Low False Alarm Rates. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. 924–929. https://doi.org/10.1109/ICMLA.2017.00-36

[50] Cherry Khosla and Baljit Singh Saini. 2023. Proposed Framework for Performance Tuning in Enterprise Applications using Machine Learning. In *2023 7th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*. 800–806. https://doi.org/10.1109/ICECA58529.2023.10394833

[51] Barbara Kitchenham and Stuart Charters. 2007. Guidelines for performing Systematic Literature Reviews in Software Engineering. 2 (01 2007).

[52] Van-Hoang Le and Hongyu Zhang. 2021. Log-Based Anomaly Detection Without Log Parsing. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 492–504. https://doi.org/10.1109/ASE51524.2021.9678773

[53] Jingwei Li, Yong Qi, and Lin Cai. 2018. A Hybrid Approach for Predicting Aging-Related Failures of Software Systems. In *2018 IEEE Symposium on Service-Oriented System Engineering (SOSE)*. 96–105. https://doi.org/10.1109/SOSE.2018.00021

[54] Xiaoyun Li, Pengfei Chen, Linxiao Jing, Zilong He, and Guangba Yu. 2020. SwissLog: Robust and Unified Deep Learning Based Log Anomaly Detection for Diverse Faults. In *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*. 92–103. https://doi.org/10.1109/ISSRE5003.2020.00018

[55] Yinglung Liang, Yanyong Zhang, Hui Xiong, and Ramendra Sahoo. 2007. Failure Prediction in IBM BlueGene/L Event Logs. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*. 583–588. https://doi.org/10.1109/ICDM.2007.46

[56] Qingwei Lin, Ken Hsieh, Yingnong Dang, Hongyu Zhang, Kaixin Sui, Yong Xu, Jian-Guang Lou, Chenggang Li, Youjiang Wu, Randolph Yao, Murali Chintalapati, and Dongmei Zhang. 2018. Predicting Node Failure in Cloud Service Systems. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (New York, NY, USA) *(ESEC/FSE 2018)*. Association for Computing Machinery, 480–490. https://doi.org/10.1145/3236024.3236060

[57] Qingwei Lin, Hongyu Zhang, Jian-Guang Lou, Yu Zhang, and Xuewei Chen. 2016. Log Clustering Based Problem Identification for Online Service Systems. In *Proceedings of the 38th International Conference on Software Engineering Companion* (New York, NY, USA) *(ICSE '16)*. Association for Computing Machinery, 102–111. https://doi.org/10.1145/2889160.2889232

[58] Yuhang Lin, Olufogorehan Tunde-Onadele, Xiaohui Gu, Jingzhu He, and Hugo Latapie. 2022. SHIL: Self-Supervised Hybrid Learning for Security Attack Detection in Containerized Applications. In *2022 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*. 41–50. https://doi.org/10.1109/ACSOS55765.2022.00022

[59] Jinyang Liu, Tianyi Yang, Zhuangbin Chen, Yuxin Su, Cong Feng, Zengyin Yang, and Michael R. Lyu. 2023. Practical Anomaly Detection over Multivariate Monitoring Metrics for Online Services. In *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*. 36–45. https://doi.org/10.1109/ISSRE59848.2023.00045

[60] Siyang Lu, Xiang Wei, Yandong Li, and Liqiang Wang. 2018. Detecting Anomaly in Big Data System Logs Using Convolutional Neural Network. In *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*. 151–158. https://doi.org/10.1109/DASC/PiCom/DataCom/CyberSciTec.2018.00037

[61] Siyang Lu, Xiang Wei, Bingbing Rao, Byungchul Tak, Long Wang, and Liqiang Wang. 2019. LADRA: Log-based Abnormal Task Detection and Root-Cause Analysis in Big Data Processing with Spark. 95 (2019), 392–403. https://doi.org/10.1016/j.future.2018.12.002

[62] Joao Paulo Magalhaes and Luis Moura Silva. 2010. Detection of Performance Anomalies in Web-Based Applications. In *2010 Ninth IEEE International Symposium on Network Computing and Applications*. 60–67. https://doi.org/10.1109/NCA.2010.15

[63] Leonardo Mariani, Cristina Monni, Mauro Pezzé, Oliviero Riganelli, and Rui Xin. 2018. Localizing Faults in Cloud Systems. In *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*. 262–273. https://doi.org/10.1109/ICST.2018.00034

[64] Cristina Monni, Mauro Pezzè, and Gaetano Prisco. [n. d.]. An RBM Anomaly Detector for the Cloud. In *2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST)* (2019-04). 148–159. https://doi.org/10.1109/ICST.2019.00024

[65] Kamini Nalavade and B. B. Meshram. 2014. Evaluation of K-Means Clustering for Effective Intrusion Detection and Prevention in Massive Network Traffic Data. 96, 7 (2014), 9–14. https://doi.org/10.5120/16804-6526

[66] Sasho Nedelkoski, Jasmin Bogatinovski, Alexander Acker, Jorge Cardoso, and Odej Kao. [n. d.]. Self-Supervised Log Parsing. In *Machine Learning and Knowledge Discovery in Databases: Applied Data Science Track* (Cham, 2021), Yuxiao Dong, Dunja Mladenić, and Craig Saunders (Eds.). Springer International Publishing, 122–138. https://doi.org/10.1007/978-3-030-67667-4_8

[67] Hiep Nguyen, Zhiming Shen, Yongmin Tan, and Xiaohui Gu. 2013. FChain: Toward Black-Box Online Fault Localization for Cloud Systems. In *2013 IEEE 33rd International Conference on Distributed Computing Systems*. 21–30. https://doi.org/10.1109/ICDCS.2013.26

[68] Hiroyuki Okamura, Junjun Zheng, and Tadashi Dohi. 2017. A Statistical Framework on Software Aging Modeling with Continuous-Time Hidden Markov Model. In *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*. 114–123. https://doi.org/10.1109/SRDS.2017.24

[69] Husanbir S. Pannu, Jianguo Liu, and Song Fu. 2012. A Self-Evolving Anomaly Detection Framework for Developing Highly Dependable Utility Clouds. In

2012 IEEE Global Communications Conference (GLOBECOM). 1605–1610. https://doi.org/10.1109/GLOCOM.2012.6503343

[70] M. Mazhar Rathore, Awais Ahmad, and Anand Paul. 2016. Real Time Intrusion Detection System for Ultra-High-Speed Big Data Environments. 72, 9 (2016), 3489–3510. https://doi.org/10.1007/s11227-015-1615-5

[71] Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou, Tony Xing, Mao Yang, Jie Tong, and Qi Zhang. 2019. Time-Series Anomaly Detection Service at Microsoft. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (New York, NY, USA) *(KDD '19)*. Association for Computing Machinery, 3009–3017. https://doi.org/10.1145/3292500.3330680

[72] Felix Salfner, Maren Lenk, and Miroslaw Malek. 2010. A survey of online failure prediction methods. *ACM Comput. Surv.* 42, 3, Article 10 (March 2010), 42 pages. https://doi.org/10.1145/1670679.1670680

[73] Carla Sauvanaud, Kahina Lazri, Mohamed Kaâniche, and Karama Kanoun. 2016. Anomaly Detection and Root Cause Localization in Virtual Network Functions. In *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*. 196–206. https://doi.org/10.1109/ISSRE.2016.32

[74] Christopher Schon, Niall Adams, and Marina Evangelou. 2017. Clustering and Monitoring Edge Behaviour in Enterprise Network Traffic. In *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*. 31–36. https://doi.org/10.1109/ISI.2017.8004870

[75] M. Omair Shafiq, Maryam Fekri, and Rami Ibrahim. [n. d.]. MapReduce Based Classification for Fault Detection in Big Data Applications. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)* (2017-12). 637–642. https://doi.org/10.1109/ICMLA.2017.00-89

[76] Seyed Reza Shahamiri, Wan Mohd Nasir Wan Kadir, and Siti Zaiton Mohd-Hashim. 2009. A Comparative Study on Automated Software Test Oracle Methods. In *2009 Fourth International Conference on Software Engineering Advances*. 140–145. https://doi.org/10.1109/ICSEA.2009.29

[77] Taeshik Shon and Jongsub Moon. 2007. A Hybrid Machine Learning Approach to Network Anomaly Detection. 177, 18 (2007), 3799–3821. https://doi.org/10.1016/j.ins.2007.03.025

[78] Md Amran Siddiqui, Alan Fern, Thomas G. Dietterich, Ryan Wright, Alec Theriault, and David W. Archer. 2018. Feedback-Guided Anomaly Discovery via Online Optimization. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (New York, NY, USA) *(KDD '18)*. Association for Computing Machinery, 2200–2209. https://doi.org/10.1145/3219819.3220083

[79] Daniel Sun, Min Fu, Liming Zhu, Guoqiang Li, and Qinghua Lu. 2016. Non-Intrusive Anomaly Detection With Streaming Performance Metrics and Logs for DevOps in Public Clouds: A Case Study in AWS. 4, 2 (2016), 278–289. https://doi.org/10.1109/TETC.2016.2520883

[80] Tong Sun, Yan Liu, and Jing Chen. 2017. A Dynamic Network Anomaly Detection Method Based on Trend Analysis. In *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*. 405–411. https://doi.org/10.1109/CompComm.2017.8322580

[81] Pratik Thantharate. 2023. IntelligentMonitor: Empowering DevOps Environments with Advanced Monitoring and Observability. In *2023 International Conference on Information Technology (ICIT)*. 800–805. https://doi.org/10.1109/ICIT58056.2023.10226123

[82] Ozan Tuncer, Emre Ates, Yijia Zhang, Ata Turk, Jim Brandt, Vitus J. Leung, Manuel Egele, and Ayse K. Coskun. [n. d.]. Online Diagnosis of Performance Variation in HPC Systems Using Machine Learning. 30, 4 ([n. d.]), 883–896. https://doi.org/10.1109/TPDS.2018.2870403

[83] Hironori Uchida, Keitaro Tominaga, Hideki Itai, Yujie Li, and Yoshihisa Nakatoh. 2024. Improving Log Anomaly Detection via Spatial Pooling: Combining SPClassifier with Ensemble Method. *Cognitive Robotics* 4 (Jan. 2024), 217–227. https://doi.org/10.1016/j.cogr.2024.10.001

[84] Byoung uk Kim and Salim Hariri. 2007. Anomaly-Based Fault Detection System in Distributed System. In *5th ACIS International Conference on Software Engineering Research, Management & Applications (SERA 2007)*. 782–789. https://doi.org/10.1109/SERA.2007.55

[85] S. Wang, M. Lu, S. Kong, and J. Ai. [n. d.]. A Dynamic Anomaly Detection Approach Based on Permutation Entropy for Predicting Aging-Related Failures. 22, 11 ([n. d.]), 1–18. https://doi.org/10.3390/e22111225

[86] Dangwei Wu, Beijun Shen, Yuting Chen, He Jiang, and Lei Qiao. 2022. Automatically Repairing Tensor Shape Faults in Deep Learning Programs. *Information and Software Technology* 151 (Nov. 2022), 107027. https://doi.org/10.1016/j.infsof.2022.107027

[87] Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael I. Jordan. 2009. Detecting Large-Scale System Problems by Mining Console Logs. In *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles* (New York, NY, USA) *(SOSP '09)*. Association for Computing Machinery, 117–132. https://doi.org/10.1145/1629575.1629587

[88] Aymen Yahyaoui, Takoua Abdellatif, and Rabah Attia. 2019. Hierarchical Anomaly Based Intrusion Detection and Localization in IoT. In *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*. 108–113. https://doi.org/10.1109/IWCMC.2019.8766574

[89] Aymen Yahyaoui, Haithem Lakhdhar, Takoua Abdellatif, and Rabah Attia. [n. d.]. Machine Learning Based Network Intrusion Detection for Data Streaming IoT Applications. In *2021 21st ACIS International Winter Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD-Winter)* (2021-01). 51–56. https://doi.org/10.1109/SNPDWinter52325.2021.00019

[90] Nong Ye and Qiang Chen. 2001. An Anomaly Detection Technique Based on a Chi-Square Statistic for Detecting Intrusions into Information Systems. 17, 2 (2001), 105–112. https://doi.org/10.1002/qre.392

[91] Li Yu and Zhiling Lan. [n. d.]. A Scalable, Non-Parametric Method for Detecting Performance Anomaly in Large Scale Computing. 27, 7 ([n. d.]), 1902–1914. https://doi.org/10.1109/TPDS.2015.2475741

[92] Bo Zhang, Hongyu Zhang, Pablo Moscato, and Aozhong Zhang. 2020. Anomaly Detection via Mining Numerical Workflow Relations from Logs. In *2020 International Symposium on Reliable Distributed Systems (SRDS)*. 195–204. https://doi.org/10.1109/SRDS51746.2020.00027

[93] Lu Zhang, Reginald Cushing, prefix=de useprefix=false family=Laat, given=Cees, and Paola Grosso. 2021. A Real-Time Intrusion Detection System Based on OC-SVM for Containerized Applications. In *2021 IEEE 24th International Conference on Computational Science and Engineering (CSE)*. 138–145. https://doi.org/10.1109/CSE53436.2021.00029

[94] Xu Zhang, Yong Xu, Qingwei Lin, Bo Qiao, Hongyu Zhang, Yingnong Dang, Chunyu Xie, Xinsheng Yang, Qian Cheng, Ze Li, Junjie Chen, Xiaoting He, Randolph Yao, Jian-Guang Lou, Murali Chintalapati, Furao Shen, and Dongmei Zhang. 2019. 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* (New York, NY, USA) *(ESEC/FSE 2019)*. Association for Computing Machinery, 807–817. https://doi.org/10.1145/3338906.3338931

[95] Wujie Zheng, Haochuan Lu, Yangfan Zhou, Jianming Liang, Haibing Zheng, and Yuetang Deng. [n. d.]. iFeedback: Exploiting User Feedback for Real-Time Issue Detection in Large-Scale Online Service Systems. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)* (2019-11). 352–363. https://doi.org/10.1109/ASE.2019.00041