ISSN 2278-3091
Volume 14, No.3, May - June 2025
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse061432025.pdf
https://doi.org/10.30534/ijatcse/2025/061432025

System Log Parameter Attributes for Predicting Software Failures: Systematic Literature Review

Juliet Gathoni Muchori[1], Gabriel Ndung'u Kamau[2], Rachael Ndung'u[3]
Department of Information Technology, Murang'a University of Technology, Kenya
jmuchori@mut.ac.ke, gkamau@mut.ac.ke, rndung'u@@mut.ac.ke

Received Date: April 18, 2025 Accepted Date: May 24, 2025 Published Date: June 06, 2025

ABSTRACT

The early prediction of software failure is important in the field of software engineering since it leads to the development of better quality software, along with a reduction in maintenance cost and effort. However, even though there is growing interest in early prediction of software failure, the existing literature shows some gaps. While many studies are quite reliant on static code metrics or test case execution data, they tend to miss out on vital dynamic and contextual information which can be obtained by analyzing software system logs. Log data is regularly created by computing systems during their runtime and contains rich information including event sequences, timestamps, error messages, and system states that can potentially being utilized in the identification of anomalies and predictions of failures on real-time. The objective of this work is to categorize the existing literature on the use of system logs for predicting software, through systematic literature review, with the help of the guidelines from Barbara Kitchenham. The review categorizes system logs into four primary parameters: resource/hardware logs, workload/performance logs, network logs, and security logs. It also highlights the machine learning models. The findings reveal that log attributes such as CPU usage, memory utilization, disk space, transaction processing, and network errors are consistently identified as key predictors of software failure. This finding aligns with expert opinions, demonstrating strong agreement on the relevance of these attributes for predicting software failure. This study contributes to the growing body of knowledge on software failure prediction, emphasizing the importance of integrating machine learning with systematic log monitoring to enhance proactive system failure management. Future work should focus on developing real-time monitoring tools that leverage machine learning models to automate failure detection and prediction across various system components.

Key words : Machine learning, Log parameters, System logs, software failure

1. INTRODUCTION

In today's world, software breakdown is very costly in large and medium critical software systems. The size and complexity of software's make runtime errors unavoidable. Companies that vend software's such as Enterprise Resource Planning (ERPs) or High-Performance Computing (HPCs) solutions could benefit from accurate forecasting of software failure. Timely prediction of impeding software failure allows the root cause of the error to be corrected before it impacts the software. Software failure describes the inability of a program to function correctly due to erroneous logic. It's also referred to as a crash or software breakdown [1]. Software failure is the instant in time where a software projects no longer meets its expectations [2]. Software failures caused by hardware or software errors often result in task and job failures. Such failures can severely reduce the reliability of software, loss of customers, fatalities in critical systems and could demand huge number of resources to recover the service from failures [3].

Software defect prediction aids software engineers to detect faulty constructions, such as modules or classes, early to avoid losses that may result from a failed software and assist to identify security, hardware and network problems. Software fault prediction increases software quality and minimizes maintenance effort [4]. It helps in obtaining the preferred software quality with improved cost and effort [4]. System logs are used to identify runtime errors, diagnose the root cause of software failure, troubleshoot runtime errors and monitor software behaviors [5]. On some occasions, log parameters can provide rich data for business decision making. The parameters may be of different types such as: security parameter logs which are used to keep logs related to unauthorized access to the software [6], [7].The resource parameter logs which keep track of how the devices such as memory, random access memory (RAM), had disk and input and/or output I/O devices are servicing the software. In case they are almost full it detects and registers in the log or in

case the device fails it is recorded [8]. The other type of log is the workload parameter logs which keep track of the events and process coordinated in the software. It can detect database connection failure, application failure, and components communication failure, among others [9]. Finally is the hardware parameter logs which indicates failure of peripherals serving the software's and lastly the Network parameter which shows failures to communicate with the nodes in a network. The models using one parameter are not able to detect failures caused by other parameters

Several researchers have developed models for software failure predictions For instance, recent works in this area have focused mainly on machine learning techniques such as RNN [7], LSTM [10], [11], [12] [13] and CNN [9]. The researchers have majored in single parameters in each model. By identifying the attributes of system log parameters for predicting software failures, this study analyzes and synthesizes existing literature to describe models that have used system logs for software failure predictions, highlighting their weaknesses, and categorize key parameters that contribute to successful predictions.

2.RELATED WORKS

Systems Log Parameters

Extant literature classifies system log parameters into four main categories namely: security, workload, network and resources or hardware [14] [15] [16] [17] [18]. [19] [20]. Security log parameters include details like login timestamps, user IDs, IP addresses, and whether the login attempt was successful or failed. They can provide information about denied or allowed connections, blocked IP addresses, and attempted attacks. For example, unauthorized access attempts, port scanning attempts. To mitigate against failures due to errors in security log parameters, intrusion detection/prevention System (IDS/IPS) Logs capture information about potential threats detected by intrusion detection or prevention systems [18] They help identify and block malicious activities, such as unauthorized network access, malware, or other attacks that include malware detection, abnormal network traffic, intrusion attempts [15] [16] [17].

Network log parameters capture information about network traffic, including details such as source and destination IP addresses, port numbers, protocols used (e.g., TCP, UDP), packet sizes, and transmission timestamps. This data provides visibility into the volume, type, and patterns of network communications occurring within the network [20]. Network logs are structured records that document events and activities related to network communications within a computer network. They play a critical role in monitoring traffic patterns, detecting security incidents, evaluating performance, and tracking network device activities. These logs capture detailed data such as IP addresses, port numbers, protocols, packet sizes, and timestamps, offering insight into the nature and volume of network interactions. They also record security events like intrusion attempts, malware infections, and unauthorized access, along with activities from devices such as routers and switches [20]. Additionally, network logs support protocol analysis and categorize events by severity to aid in prioritizing responses.

Hardware logs are essential records that capture runtime information related to system hardware components [19]. These records contain events and activities generated by the hardware components. These components include servers, routers, switches, and physical devices within a computing environment such as peripherals, chipsets, CPU (Central Processing Unit), memory modules (RAM), storage devices (hard drives, SSDs), motherboard, power supply unit (PSU), and peripheral devices. All physical issues like component failure, overheating and power issues can be termed hardware problems. A typical hardware log usually contains the following details: Timestamp describing when the event occurred, event security level that categorizes the log as either just information, warning or an error and it also includes event details that describe the event that occurred in detail. [18] [21] [22] [23]. Overall, hardware logs serve as valuable diagnostic tools for identifying hardware failures, troubleshooting issues, monitoring system performance, and predicting potential hardware-related failures. By analyzing hardware logs, system administrators can take proactive measures to maintain system reliability and minimize downtime [24]. Hardware logs also include storage logs which structure records of events and activities associated with storage devices and file system operations in a computer system. These logs provide insights into the performance, health, and usage of storage resources, facilitating troubleshooting, monitoring, and optimization of storage infrastructure [14]. These logs contain information about data reads, writes, access permissions, and storage health. Usually, a storage log record contains Timestamp that indicates the day and the time the storage event occurred, data access patterns that contain details about the read and write operations while storage health metric field contains data about the disk space, I/O latency and error rates. Another important feature contained in storage logs is the security event that provides record of access attempts and change of permissions [14].

Machine Learning Models for Prediction Software Failures

Several machine learning models have been developed to predict software failures. For instance, Banjongkan and others [25] developed a job failure prediction model in HPC systems using the decision tree algorithm. Job failure was predicted at two distinct states namely job submit, and job start states. Workload logs from HPC systems were used to train the decision tree. Ali and others [26] proposed a machine learning model to classify and predict software incidents. In their work, they considered an abnormal process that disrupted operational procedures. Their technique used an active learning approach to label data that was appropriate for building the model. K Means clustering was used to do unsupervised labelling of the data through clustering [26] The labelled logs were then fed to the SVM to predict the failure. It classified results into groups It used workload logs only which had attributes which consisted of date and time, source, EventID and Task category. The model had some weaknesses such as did not consider scalability issues, used only one type of logs that is workload logs which gotten from ECLIPSE software repository and never considered scalability in future. A study done by Mohamed and others [27] used machine learning to predict failure in HPC systems and their applications. Their model considered system components failure datasets which are under workload dataset where he considered the use of time series and ML algorithms to predict failure of all applications and system components. It was noted that a process could have failed due to several reasons such as hardware, software, human error, network and undermined.

Das et al., [10] [12] implemented the LSTM combined with RNN and LSTM models respectively to detect node failure. Node failure is termed as any abnormal shutdown that may occur on a node due to software or hardware problems. It would identify log events that would lead to failure, predict which specific node would fail and for how many minutes, retrain chain recognition of events augmented with expected lead time to failure. The study focused on resources logs due to hardware. Gao and others [11] developed a model to predict task failure in cloud data centers. The model was based on bi-directional LSTM to identify job failures and forecast their occurrence. Bi-LSTM model contains one input layer, Bi-LSTM layer and logistic regression layer for classification. The jobs termination status of the tasks is classified based on the task attributes and performance data. Their model used data center logs from Google cluster trace. It displayed whether a task or a job has failed or completed. It used workload log type of data.

Convolutional Neural Networks (CNN) was used to make a model predict and alert failure risk failure of virtual machines using log analysis. Nam [9] developed the model to preprocess the failure using constant time gap. The model learns to calculate the probability that a failure will occur after gap minutes based on the input window size of the input logs. The model preprocesses the failure messages using word embeddings, then the embedded words were fed as input to train the CNN model. The input had constant time gaps that were used to ensure the CNN can calculate the probability of failure after some gap minutes. Google word2vec was used for word embeddings to form a log corpus. The dataset and failure injections were simulated using OpenStack. It uses workload logs. The challenge it faces is failure to use different logs to allow future growth in failure prediction.

Lin et al [28] proposed a hybrid system that combined LSTM and Random Forest Classifier to forecast the failure of a node in the cloud service system using a variety of attributes such as disk sector error logs, service error, rack location, load balancer group, IO response and the OSBuildGroup. The dataset was borrowed from Microsoft Cloud System logs from a live production environment. LSTM model handles the temporal and spatial features of the data to provide the predicted sequence. The sequence was ranked using Random Forest and the cost-sensitive function was used to make classifications [28].Its weakness was that it focused on resources logs only. Another model that combines several algorithms is Jingweng Wang et al. [8]. This technique utilized several machine learning algorithms to predict anomalies in financial Information Technology systems. It had KPI (Key performance indicators) which would record the usage of the CPU, disks, and the memory for each of the servers at all-time thus producing time series data of KPIs. Incase an anomaly would occur it would be recorded. Therefore, the system log parameter used to detect software failure was resource parameter. A hierarchy of classifiers was used with Random Forest classifier at the data prediction phase while the anomaly detection module used Decision Tree (DT), RF Classifier, KNN, Gradient Boosting Decision Tree (GBDT) and Logistic Regression (LR). Outputs of DT, RF, KNN, GBDT were fed to the LR classifier that determined the severity of the failure while the KNN model was used at the end of the model to classify anomalies with different severity [4]. The main weakness of the model was that it would only detect software failure using resource parameters.

Also, Brown et. al [6] developed a log anomaly detection technique that leveraged the combined strengths of Recurrent Neural Networks and Attention Based Mechanisms. Their solution was unsupervised since the logs were not labelled. Language modelling was applied to the logs to assign probabilities to sequences and tokenization was used to break

down the log lines into words and characters. The LSTM model was fed the preprocessed logs, and it predicted the sequences. LSTM model was optimized by dot product attention to be selective to the relevant tokens. This feature allowed the model to access the relevant information with ease while making predictions. Their model used a public dataset drawn from the Los Almos National Laboratory (LANL), of network traffic, authentication info and DNS logs from a vast 58 billion logs [6]. It used authentication logs which consisted of attributes like Source user, Destination user, Source pc, Destination pc, Authentication type, Logon type, Authentication orientation, Success/failure. Its main weakness is that it focused on a single parameter which limits future growth of the model.

Hybrid models combine the strength of several learning techniques through a combination of several ML algorithms or DL models or both. For instance, Savaranan and Sangeetha [29] combined the power of Adaptive Dimensional Search based Particle Swarm Optimization (ADS-PSO) with the Hyper Basis Function Neural Network to improve its prediction accuracy while predicting software failure. It used the events logs of a task to determine the degree of accuracy and classified the result into two. The Saravanan hybrid model [29] used event logs to predict software failure. The initial stage of their model [29], used the Hyper Basis Function Neural Network (HBFNN), which is a three-layered feed-forward neural network. The classification weights drawn from the neural network were optimized further using the ADS-PSO algorithm to minimize the cost of misclassification. The dataset used was made from workload event logs from the Blue Gene/p intrepid system. The challenge was lack of scalability.

3.RESEARCH METHOD

This research employs a systematic literature review (SLR) methodology based on Barbara Kitchenham's original guidelines (2007) to comprehensively analyze and synthesize the existing literature on importances of system logs in predicting software failure. These guidelines were chosen for their relevance to system logs, providing domain-specific guidance for analyzing technical papers, software failure prediction studies, and IT initiatives. A scientific literature review aims to identify, evaluate, and interpret relevant research on a specific question, topic, or phenomenon. This study qualifies as a tertiary review, focusing on systematic reviews, which are secondary studies. The approach is divided into three phases:

A. Planning Review Phase

The planning phase ensures the SLR is thorough, objective, and methodical. It includes identifying the need for the review, defining study subjects, and developing a robust review methodology.

Objectives of the Systematic Literature Review

This literature study seeks to compile and evaluate the vast body of knowledge on system logs by understanding their importance in scientific research, learning how to prepare the logs before using them in any research, understanding the different machine learning models used in software failure prediction and their weaknesses, and exploring the different attributes that a log can have, along with methods for interpreting its actual message to classify the log. Lastly, it aims to identify the different types of system logs that can be used for log classification. This work will contribute to improving software failure prediction in case the researcher wants a better understanding of logs. It will foster innovation and future research and can also guide researchers in determining which types of logs to focus on when conducting research. According to Barbara Kitchenham's original guidelines (2007), Figure 1 shows the phases that have been followed in conducting the systematic literature review.

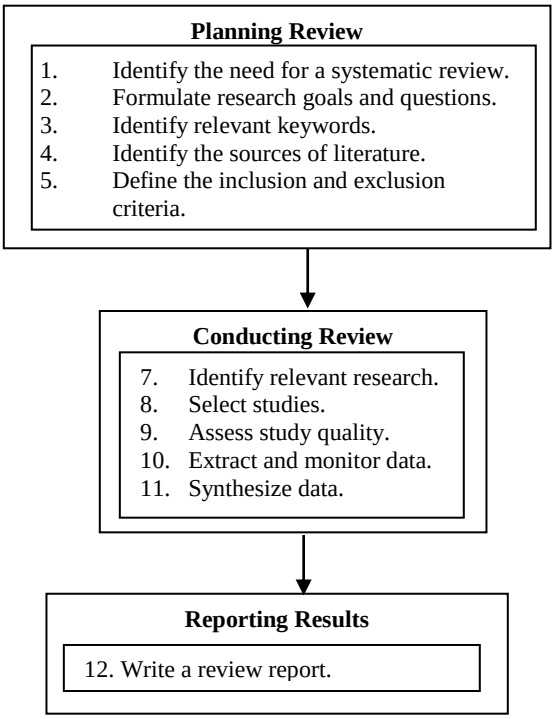


Figure 1: Barbara Kitchenham's original guidelines (2007)

Research Questions

To achieve the objectives of this research, several key research questions were formulated to guide the investigation and ensure comprehensive coverage of the topic. The following primary research questions served as the basis for this review paper:

RQ1: Which machine learning models have used system logs to detect software failures, and what are their weaknesses

RQ2: Which log parameter attributes can be used to predict software failures?

Search Phrases and Sources

A stepwise analysis of system logs was conducted using well-known paper indexing engines. Some of the search strings used include "system logs," "log collection," "log parsing," "log anomaly," "feature extraction," "machine learning," and "software failure." The search engines used in this paper are listed in Table 1:

Table 1: Search engines

Finding Engine	Source Address
Semantic Scholar	https://www.semanticscholar.org/
Scopus	https://www.scopus.com
Science Direct	https://www.sciencedirect.com/
ACM Digital Library	https://dl.acm.org
IEEE Xplore	https://ieeexplore.ieee.org/
SpringerLink	https://link.springer.com
Research Gate	https://www.researchgate.net/
Google Scholar	https://scholar.google.com

B. Conducting Research

The following processes were undertaken throughout the review: eligibility criteria, the search process, study selection, data collection process, inclusion and exclusion criteria, and synthesis methods.

Eligibility Criteria

This section defines the inclusion and exclusion criteria for the literature review and outlines how studies were categorized for synthesis. These criteria establish the scope of the review, ensuring that only the most relevant studies are included in the data analysis. The selection process for articles in the Systematic Literature Review was conducted in three stages: initial selection based on the title, second selection through abstract reading, and final selection after reviewing the full paper. The inclusion and exclusion criteria used in this research are as follows:

a) Articles on existing software failure detection.
b) Articles using system logs and relevant to the machine learning field.
c) Only the most recent versions of articles (if available) are considered.
d) Articles published between 2016 and 2024 are included.
e) Only peer-reviewed journal publications or conference proceedings are considered.
f) Articles, papers, and journals that are not peer-reviewed journal or conference publications are excluded.

Search Process

"Search criteria" refers to the specific terms, parameters, or conditions used to define and narrow down search results, whether in a search engine or database. Well-defined criteria ensure that only the most relevant information is retrieved. For instance, keywords were used to focus the search. These included terms such as "software failure," "failure prediction," "system logs," "log parsing," "feature extraction," "logs," "anomaly detection," and "machine learning." Filters were also applied to refine the results by specifying the publication timeframe. Only journal papers published between 2016 and 2024 were considered, as the researcher was interested in the most recent advancements in the field. The papers were then screened for relevance, focusing on their titles, abstracts, and keywords. Following this, a snowballing method was employed to expand the pool of relevant studies. This included both backward snowballing which involved examining the reference lists of selected studies and forward snowballing which involves analyzing citations to these papers to uncover additional relevant publications.

Data Extraction Synthesis

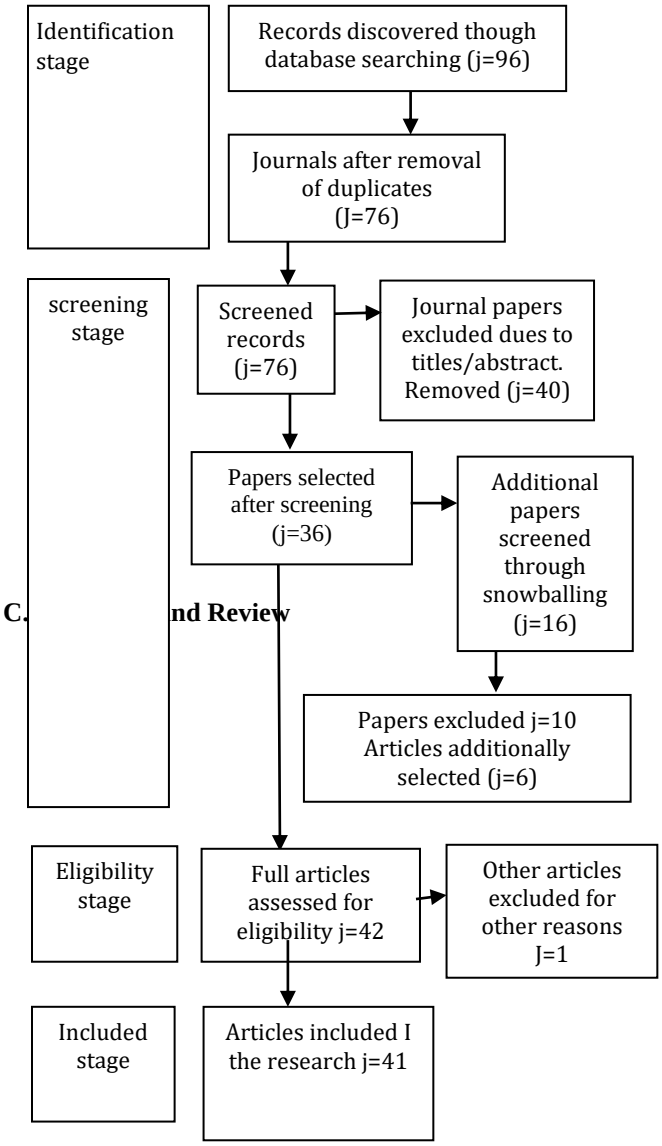
To identify the most recent models, journal papers published between years 2016–2024, were reviewed. Table 2 and the pie chart present the number of papers selected from each academic database during this period.

Table 2: No. of papers selected

Academic Databases	No of Journal Papers Reviewed
IEEE	55
Elsevier	15
SPRINGER	16
ACM Digital library	14
Citeseer Library	7
arXiv.	19
Wiley	6
Total	122

The article's abstract was gathered as a key component in the first phase of the analysis. To address the research questions, specific sections of the selected papers were considered, including the abstract, introduction, literature review, and results sections. The process for deciding whether to include an article in the final review was carried out in two stages. First, each abstract was read and classified as either relevant or not, based on its alignment with the search phrases. The relevance of each abstract was evaluated by comparing it to predefined search criteria.

The review also considered factors such as the authors, publication date, article type (e.g., journal or conference proceeding), technique-based taxonomy, and datasets. These details were crucial for answering the research questions. Figure 2 presents a flowchart illustrating the article and paper selection process.



C. nd Review

Figure 2: Flowchart for selection of journal papers

This section discusses the responses to our research questions based on the papers reviewed. Specifically, it examines the, models used in software failure prediction, and classification of logs. The research findings are structured as follows in alignment with the research questions

4.RESULTS

RQ1: What are machine learning models for predicting software failures?

The first objective of research was to analyze the existing machine learning (ML) models that have been utilized to detect software failures using system logs. Table 3 provides a comparative summary of these models, highlighting the ML or deep learning (DL) algorithms employed, the types of log mining techniques used, datasets utilized, evaluation metrics applied, and notably, the specific weaknesses of each approach.

Table 3: Machine Learning Models for Predicting Software Failures

Publication	Underlying DL & ML models	Log Mining Technique	Datasets	Metrics	Weakness
Banjongkan et al. [25]	Decision Tree: C5.0, CART & CHAID	N/A	Los Alamos National Laboratory (LANL) & National Electronic and Computer Technology Center (NECTEC)	Prediction Acc. Recall, Precision, F1 Score	- It focused focusing on point of failure and scalability issues. - increase generalizability of the model, network logs -
Ali et al. [30]	SVM, *k*-Means Clustering	Waikato Environment for Knowledge Analysis (WEKA)	WEKA log files	False Positive Rate (FPR), True Positive Rate, Precision, Recall, F-	- Did not consider scalability issues - It used only one type of logs that is workload logs.Therefore should work on the ability to use different logs parameters to detect

				Measure, RoC Curve	failures
B. Mohammed et al. [31]	SVM, RF, Linear Discriminant Analysis (LDA), CART, KNN	N/A	NERSC I/O failure data.	Prediction Acc. Sensitivity ROC	- The model relied on one type of log that is system component failure logs and also did not consider time parameter in its feature selection despite it being tested with different machine learning algorithms. - It mostly considered the accuracy of the algorithms.
Lu et al. [32]	k-Means	Word count	Spark Log Files	Prediction Acc.	- Data used was too small and caused overfitting. - Used one type of log parameters which is resource log. Therefore should work on the ability to use different logs parameters to detect failures - Did not consider scalability of the model which can lead to future growth.
Savaranan and Sangeetha [29]	Hyper Basis Function Neural Network (HBFNN	N/A	Blue Gene/p intrepid system	False Positive Rate (FPR), Time Complexity	- The dataset used was made from workload event logs from the Blue Gene/p intrepid system. - lacked scalability issues.
Lin et al. [28]	LSTM & Random Classifier (RF)	N/A	Microsoft cloud system logs	Precision, recall, F1	- Its weakness was that it focused on resources logs only, therefore should wotk on the ability to use different logs parameters to detect failures
Wang et al. . [33]	PCA-Q, Logistic Regression, SVM	N/A	HDFS logs	Prediction Acc. Recall, F1 Score	- Scalability issues - Generalizability issues
Gao and others [11]	Bi-LSTM, LR	N/A	Google Cluster Trace Logs	F1 Score, Receiver operating Characteristic (RoC) Curve, Time-cost overhead, Prediction Acc.	- It used workload log type of data only so the scope was limited to one type of failure in the clouds - Never considered future growth in terms of wider logs coverage.
Aarohi Framework [34]	LSTM	Regular expression, Parser	HPC logs: Cray XK & BlueGene/P	Prediction Acc.	- Scalability issues - Generalizability issues
Jingweng et al.[16]	Decision Tree (DT), RF, kNN GBDT, LR	N/A	System logs of IT financial system server clusters	F-Score, Precision, Recall	- The main weakness of the model was that it would only detect software failure using resource parameter.
Brown et al. [6]	RNN, Attention mechanism	Language Modelling-Tokenization	Los Alamos National Laboratory (LANL) -Network, DNS and Authentication logs	AUC-RoC Curve Score	- Its main weakness is that it focused on a single parameter which limits future growth of the model.
Nam [9]	CNN	Word2vec	Openstack logs simulation	Prediction Acc.	- It uses workload logs only so failure to use different logs limits future growth of the model in relation failure prediction.
Mantyla et al. [13]	LSTM, N-GRAM	N/A	HDFS, Profilence Dataset	F-Score & Prediction Acc.	- Focused on comparing the performance of N-Grams and Deep leaning to figure out which one uses lesser time to predict errors and which is more accurate. It gave less concentration on

					future growth which can be shown through scalability - Used only one log parameter in the comparison in system logs. - Focused more on the performance of different machine learning algorithms rather than failure prediction
Benaddy at al. [7]	RNN	N/A	Failure data from commercial, word processing apps	Prediction Acc.	- It used security logs only. - Was limited to numerical events only and it did not use text-based logs.
Das et al., [10] [12]	the LSTM combined with RNN	of events augmented with expected lead time to failure.	Controller (bcsysd), Boot-logs, SEDC differ from XE	Recall Precision	- Challenges is that it focused node failure using resources logs only - There is need to work on the continuous learning ability of a model

As indicated in Table 3 a consistent pattern observed is the reliance on a single type of log parameter—such as workload, resource, hardware, or network logs without integrating multiple log types for more comprehensive failure prediction. For instance, models like those proposed by Ali et al. [37], Lu et al. [39], and Gao et al. [11] predominantly rely on workload logs, limiting their scope and potentially reducing the robustness of failure detection when dealing with diverse system environments. Similarly, other studies such as those by B. Mohammed et al. [38] and Lin et al. [25] utilize only resource logs or system component failure logs, often without incorporating time-related parameters or other contextual information, which may affect prediction accuracy and limit scalability.

In addition, several approaches such as those presented by Jingweng et al. [16] and Brown et al. [15] focus on specific parameter types (e.g., resource logs or network logs), which hampers generalizability and adaptability to broader software environments. The models developed by Savaranan and Sangeetha [24] and Aarohi Framework [40] also suffer from scalability challenges, suggesting a need for solutions that can scale effectively in high-performance computing (HPC) or cloud environments. Overall, the analysis reveals that while there has been substantial progress in the use of ML for log-based software failure detection, most existing models suffer from limitations related to generalizability, scalability, and limited use of log parameter diversity. Future research should aim to develop classifiers for instance a multiple log parameter classifier using a Random Forest algorithm that can integrate diverse log types to improve the robustness, scalability, and accuracy of software failure detection systems.

RQ 2: Which are the Log Parameter Attributes to Predict Software Failures?

The second objective of this study the key system log parameters and their attributes that can be used to predict software failure. The classification and understanding of log parameters are crucial in the development of reliable machine learning models for software failure detection. Table 4 outlines four primary types of log parameters commonly found in system logs: security logs, workload logs, network logs, and resource or hardware logs. Each log type provides unique and complementary insights into the software environment, and collectively, they can greatly enhance the ability of a classifier to detect and predict failures accurately.

Table 4: Types of System Log Parameters and Key Attributes

Log Type	Description	Key Attributes	No. of References in the Literature
Security logs	Handle data that is related to the security of the software. This data shows irregularities that may appear if unauthorized personnel access the software. It holds the time of access, device mac addresses, Source user, Destination user, Source pc, Destination pc, Authentication type, Logon type, Authentication orientation, Success/failure.	Unauthorized Access Logs [8], [11], [18], [19], [41]	5
		Intrusion Detection Logs [18], 11], [18], [19]	4
		Malware Activity Logs [8], [11], [18], [19], [41]	5
		Authentication Failure Logs [8], [11], [18], [19], [41]	5
		Security Patch Logs [11], [18], [19], [41]	4
Workload logs	Dataset was as a result of classes, functions and other parts of programming weakness that caused the failure of the software. These failures arise because of poor programming and are mostly found during testing. When the software is not well tested, then the	Traffic Load Logs [2], [11], [15], [35]	4
		Transaction Processing Logs [2], [5], [29], [30],[35}	5
		Response Time Logs [2], [5], [12], 30]	4
		System Overload Logs 5], [12], [35]	3

	failure occurs.	Workload Balancing Logs[2], [5], [38]	3
Network log	Network logs are structured records of events and activities associated with network communications within a computer network. These logs provide insights into network traffic patterns, security incidents, performance metrics, and network device activities, facilitating network monitoring, troubleshooting, and security analysis	Latency Logs [2], [5], [14], [36]	4
		Bandwidth Utilization Logs [5], [14], [36]	3
		Packet Loss Logs [2], 5], [14], [36]	4
		Connectivity Logs [2], 5], [14], [36]	4
		Network Error Logs [2], 5], [14], [36]	4
Resources or Hardware' s logs	Are datasets that were as a result of failure in devices needed for the running of the software. This device may include the CPU, Memory, Network and Disk I/O devices.	CPU Usage Logs [5], [10], [11], [13], [28], [36]	3
		Memory Utilization Logs [5], [10], [11], [13], [36]	5
		Disk Space Logs [5], [10], [13],[36]	4
		Power Supply Logs [5]	1
		Process execution logs. 5], [10], [13]	3

As indicated in Table 4, security logs play a vital role in monitoring access-related anomalies. They include details such as the time of access, MAC addresses of devices, authentication types, and success or failure of login attempts. This log type is instrumental in identifying security breaches or unauthorized access, which could potentially lead to system compromises or software failures. Despite their value, many machine learning models have yet to fully incorporate security logs in failure prediction, thus missing out on detecting failures caused by external intrusions or internal misconfigurations. Workload logs, on the other hand, capture the software's operational behavior, especially during testing or runtime. These logs highlight functional weaknesses, such as programming errors or faulty logic, which may not be discovered until later stages of development. They are particularly important for models focused on detecting software bugs or malfunctions that stem from poor coding practices or insufficient testing. Network logs contribute to a different dimension, offering visibility into the interactions and communications within a system's network. They record data related to IP addresses, protocol usage, packet transmission, and device activity. Analyzing these logs helps identify failures arising from network disruptions, latency issues, or unauthorized network activity. Yet, models that focus only on network logs tend to overlook software faults not related to connectivity or communication errors. Lastly, resource or hardware logs monitor the performance and health of physical and virtual components, such as the CPU, memory, and disk I/O. These logs are essential for detecting hardware-induced failures that could severely impact software performance. However, relying solely on these logs could result in overlooking software-related anomalies unrelated to hardware performance.

Empirical Validation

To empirically validate the findings, a survey was conducted using a questionnaire to gather expert opinions on systems and their attributes, to determine whether the information from literature aligned with expert perspectives. The questionnaire attracted 55 more respondents from different disciplines, such as Technologists, Network Engineers, Security Analysts, Data Scientists, Data Analysts, Software Developers, Cloud Engineers, IT Consultants, and others, with varying levels of experience ranging from 1-5 years, 6-10 years, 11-15 years, and over 16 years. To ensure more reliable results, the researcher chose to eliminate responses from interns and attachment people reducing the number to 52 respondents whose responses were used to validate the literature. Figure 3 illustrates the years of experience of the 52 experts while figure 4 indicates their areas of specialization.

Figure 3: Experts years of experience

Figure 4: Experts area of specialization

Following the reliability test, as shown in Figure 5.the Cronbach's Alpha was found to be 0.842, which was regarded as acceptable. Generally, a value above 0.7 is considered adequate for research, while higher values, particularly between 0.8 and 0.9, signal stronger reliability. With a value of 0.842, the result indicates high reliability, suggesting that the 20 items on the scale are consistently measuring the same construct with great effectiveness.

Reliability

Scale: ALL VARIABLES

Case Processing Summary

		N	%
Cases	Valid	10	100.0
	Excluded[a]	0	.0
	Total	10	100.0

a. Listwise deletion based on all variables in the procedure.

Reliability Statistics

Cronbach's Alpha	N of Items
.842	20

Figure 5: Reliability test for the Questionnaire

System Log Parameter Attributes to Predict Software Failures

Table 5 results summarized the relevance of various system log parameter attributes in predicting software failures. Respondents rated each parameter attributes based on their perceived effectiveness in comparison with no. of references in the literature

Table 5: Relevance of various log parameters attributes in predicting software failures

	Respondents	Expert Mean	Expert Std. Deviation
"CPU Usage Logs High CPU usage logs indicate potential system failure risks."	52	4.77	.807
"Memory Utilization Logs Memory exhaustion logs are a key predictor of software crashes."	52	4.75	.682
"Disk Space Logs Logs showing low disk space often correlate with system instability."	52	4.65	.789
"Power Supply Logs Resource logs capturing power fluctuations are useful in predicting system failures."	52	4.12	.855
"Process Execution Logs Logs indicating excessive process execution time are a sign of potential system failure."	52	4.65	.905
"Traffic Load Logs Logs showing peak user loads correlate with software performance degradation."	52	4.73	.689
"Transaction Processing Logs System failures increase when workload logs indicate high transaction volumes."	52	4.65	.789
"Response Time Logs Logs showing prolonged response times are a sign of potential system crashes."	52	4.54	.896
"System Overload Logs Excessive workload logs often precede software failures."	52	4.67	.785

"Workload Balancing Logs Logs indicating poor workload balancing can predict software breakdowns."	52	4.52	.874
"Latency Logs High latency in network logs is a strong predictor of software failure."	52	4.31	1.164
"Bandwidth Utilization Logs Insufficient bandwidth logs often indicate impending system failure."	52	4.58	.997
"Packet Loss Logs Frequent packet loss logs correlate with degraded software performance."	52	4.63	.971
"Connectivity Logs Logs showing frequent connection failures often precede software crashes."	52	4.44	.978
"Network Error Logs High occurrences of network errors in logs are a sign of potential system failure."	52	4.63	.768
"Unauthorized Access Logs Logs showing multiple unauthorized access attempts are linked to system vulnerabilities."	52	4.56	.850
"Intrusion Detection Logs Security logs detecting frequent intrusion attempts are a precursor to software failure."	52	4.60	.934
"Malware Activity Logs Logs indicating malware infections are a major predictor of software crashes."	52	4.69	.673
"Authentication Failure Logs A high number of failed logins in security logs is associated with system risks."	52	4.06	.958
"Security Patch Logs Infrequent security updates in logs increase software failure risks."	52	4.56	.938
Valid N (listwise)	52		

As exhibited in Table 5, CPU usage logs (4.77), memory utilization Logs (4.75), traffic load logs (4.73), malware activity logs (4.69), and system overload logs (4.67) received the highest ratings, consequently, these logs are considered as strong predictors of software performance failure due to their direct link to system resource exhaustion and abnormal usage patterns. Response time logs (4.54), connectivity logs (4.44), and power supply logs (4.12) were rated slightly lower but still indicate meaningful contributions to early failure detection. Authentication failure logs (4.06) was perceived as less impactful in failure prediction, potentially due to its more specific relevance to security incidents than to system stability.

Pearson Correlation Test Between Literature and Expert Opinions

To further assess the alignment between expert opinion and literature review, a correlation analysis was conducted as indicated in Table 6.

Table 6: Pearson Correlation Test Between Literature and Expert Opinions

		Literature	*Expert*
Literature	*Pearson Correlation*	*1*	*.479**
	Sig. (2-tailed)		*.032*
	N	*20*	*20*
Expert	*Pearson Correlation*	*.479**	*1*
	Sig. (2-tailed)	*.032*	
	N	*20*	*20*

**. Correlation is significant at the 0.05 level (2-tailed).*

As indicated in Table 6 the correlation coefficient is 0.479 with a p-value of 0.032, indicating a moderate but statistically significant positive relationship. This means that attributes commonly cited in academic and industry literature also tend to be rated highly by experts, suggesting consistency and validation between theoretical frameworks and practical expertise.

5.CONCLUSIONS AND FUTURE WORKS

The study was designed around two primary research questions. The first aimed to investigate the machine learning algorithms and datasets employed in software

failure prediction, as well as to identify the limitations of these models. The second sought to examine the attributes of various log parameters identified in the literature. The study finding revealed that various models ranging from traditional algorithms like SVM and Decision Trees to deep learning approaches such as LSTM, RNN, CNN, and Bi-LST have been applied to different types of logs. However, many of these models suffer from limitations such as reliance on a single type of log, scalability issues, lack of generalizability, and neglect of critical parameters like time and context. Through an empirical validation involving ICT professionals from diverse specializations, the study confirmed the significance of twenty log attributes in predicting software failures. The analysis highlights that log types such as security, workload, resource, and network logs all play vital roles. Parameters like CPU and memory usage, transaction volumes, malware activity, and unauthorized access attempts were consistently rated highly, indicating their practical utility in early failure detection. Also, different types of system logs require different analytical models depending on their specific characteristics. For instance, workload logs may require models focused on anomaly detection, while security logs may be better suited for classification models. This theory contributes to the analytical perspective by highlighting the importance of context-based model selection, guiding practitioners to choose the most appropriate machine learning models for the type of data they are analyzing. This ensures that predictive models are tailored to the specific context of the logs, optimizing their performance.

In conclusion, the diversity of system log parameters provides a strong foundation for developing robust machine learning classifiers and models. The major limitation in many existing models is their reliance on a single log type, which constrains their effectiveness and scalability. To overcome this, integrating multiple log parameters—as suggested in the proposed multiple log parameter classifier using a Random Forest algorithm—could significantly improve the accuracy, generalizability, and resilience of failure detection systems across different environments and use cases.

Building on these findings, several directions for future work are proposed: Future models should integrate multiple types of log-resource, workload, network, and security, rather than relying on a single source. This will enhance the generalizability and accuracy of failure predictions. Upcoming research should focus on incorporating time-based parameters and contextual relationships within logs to improve the interpretability and precision of ML models. By addressing these areas, future research will not only improve software reliability and reduce downtime but also contribute to more intelligent and self-healing computing systems.

ACKNOWLEDGEMENT

The researcher wish would acknowledge my Ph.D. supervisors and the entire faculty members for their tireless guidance in writing this research work, my spouse, children, and my family at large for their financial and moral support.

DECLARATIONS

Conflict of Interest

The researchers declared that they have no conflicts of interest

Informed Consent

This may not be applicable because this is a review article, and respondents are not involved.

Ethics Approval

It is not applicable because this is a review article, and no respondents are required.

Funding

The study did not receive funding from any institution

REFERENCES

[1] K. Zhang, J. Xu, M. R. Min, G. Jiang, K. Pelechrinis and ". H. Zhang, "Automated IT System Failure Prediction: A Deep Learning Approach," *IEEE International Conference on Big Data (Big Data),* no. 7840733, 2016.

[2] D. A. Tamburri, F. Palomba and R. Kazman, "Success and failure in software engineering: A followup systematic literature review.," *IEEE Transactions on Engineering ManagemenT,* vol. 68, no. 2, pp. 599-611, 2020.

[3] J. Gao, H. Wang and H. Shen, " Task Failure Prediction in Cloud Data Centers Using Deep Learning," *in IEEE Transactions on Services Computing,* vol. 15 , no. 3, pp. 1411-1422,, May-June 2022.

[4] Rathore, S. S and S. Kumar, " A study on software fault prediction techniques," *Springer,* no. 51, pp. 255-327, 2019.

[5] A. Miranskyy, A. Hamou-Lhadj, E. Cialini and A. Larsson, "Operational Log Analysis for Big Data Systems: Challenges and Solutions," *IEEE,* vol. 33, no. 1, pp. 1-1, 2016.

[6] A. Brown, A. Tuor, B. Hutchinson and N. Nichols, " Recurrent neural network attention mechanisms for interpretable system log anomaly detection.," *ACM,* pp. 1-8, 2018, June..

[7] M. Benaddy, B. El Habil, O. El Meslouhi and S. Krit, "Recurrent neural network for software failure prediction," in *in Proceedings of the Fourth International Conference on Engineering & MIS* , Istanbul, Turkey, , 2018.

[8] J. Wang, J. Liu, J. Pu, Q. Yang, Z. Miao, J. Gao and Y. Song, "An anomaly prediction framework for financial IT systems using hybrid machine learning methods," *Journal of Ambient Intelligence and Humanized Computing, Springer,* pp. 1-10, 2019.

[9] S. Nam, J. Hong, J. Yoo and J. Hong, "Virtual machine failure prediction using log analysis.," *IEEE,* pp. 279-285, 2021 September.

[10] A. Das, F. Mueller, C. Siegel and A. Vishnu, " Desh: deep learning for system health prediction of lead times to failure in hpc.," *ACM,* p. 4, 2018 June.

[11] J. Gao, H. Wang and H. Shen, "Task failure prediction in cloud data centers using deep learning," *IEEE,* vol. 15, no. 3, pp. 1411-1422, 2020.

[12] A. Das, F. Mueller and B. Rountree, "Aarohi: Making real-time node failure prediction feasible.," *IEEE,* pp. 1092-1101, 2020, May.

[13] M. Mäntylä, M. Varela and S. Hashemi, "Pinpointing anomaly events in logs from stability testing–n-grams vs. deep-learning.," *International Conference on Software Testing, IEEE,* pp. 285-292, 2022 April.

[14] X. J. D. A. S. E. Alter J, "SSD failures in the field: symptoms, causes, and prediction models. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis," *dl.acm.org,* pp. 1-14, 2019.

[15] L. Max, F. Skopik, M. Wurzenberger and A. Rauber, ""System log clustering approaches for cyber security applications: A survey." Computers & Security ," *Elsevier,* 2020.

[16] R. T. Reshmi, "Information security breaches due to ransomware attacks-a systematic literature review.," *International Journal of Information Management Data Insights, Elsevier,* Vols. 1(2),, p. 100013., 2021.

[17] M. R. Kumar and R. Kumar., "Utilizing windows event logs for malware detection using machine learning." In 2nd International Conference on Computer Vision and Internet of Things .," *Science direct,* pp. 19-27, 2024.

[18] A. vinaypamnani, V. Pamnani and V. paolomatarazzo, "learn.microsoft.com," microsoft, 22 12 2022. [Online]. Available: https://learn.microsoft.com/en-us/windows/security/threat-protection/auditing/basic-security-audit-policy-settings. [Accessed 23 2 2024].

[19] W. P. Gholamian S, " A comprehensive survey of logging in software: From logging statements automation to log mining and analysis.," *arxiv.org,* no. 2110.12489, 2021.

[20] D. S. C. R. W. S. L. Q. Ji W, "A CNN-based network failure prediction method with logs. In 2018 Chinese Control And Decision Conference (CCDC)," *IEEE,* pp. 4087-4090, 2018.

[21] "documentation.suse.com," SUSE, [Online]. Available: https://documentation.suse.com/sles/12-SP5/html/SLES-all/cha-audit-comp.html#:~:text=Linux%20audit%20provides%20tools%20that,them%20into%20human%20readable%20format.&text=Audit%20provides%20a%20utility%20that,User.

[22] A. Jeyashankar, "socinvestigation.com," socinvestigation, 21 June 2021. [Online]. Available: https://www.socinvestigation.com/linux-audit-logs-cheatsheet-detect-respond-faster/. [Accessed 23 2 2024].

[23] A. Sharif, "crowdstrike.com," crowdstrike.com, 13 February 2023. [Online]. Available: https://www.crowdstrike.com/guides/apache-logging/. [Accessed 2 23 2024].

[24] D. S. M. B. E. F. C. T. B. M. F. K. S. D. T. P. Das, "Failure prediction by utilizing log analysis: A systematic mapping study. In Proceedings of the International Conference on Research n Adaptive and Convergent Systems," *dl.acm.org,* pp. 188-195, 2020.

[25] A. Banjongkan, W. Pongsena, N. Kerdprasop and K. Kerdprasop, "A Study of Job Failure Prediction at Job Submit-State and Job Start-State in High-Performance Computing System: Using Decision Tree Algorithms," *Advances in Information Technology, Research Gate,* 2021.

[26] S. Ali, M. Adeel, S. Johar, M. Zeeshan, S. Baseer and A. Irshad, "Classification and prediction of software incidents using machine learning techniques.," *Security and Communication Networks, ,* pp. 1-16, 2021.

[27] B. Mohammed, I. Awan, H. Ugail and M. Younas, "Failure prediction using machine learning in a virtualised HPC system and application. Cluster Computing," *Cluster Computing,* no. 22, pp. 471-485., 2019.

[28] Q. Lin, K. Hsieh, Y. Dang, H. Zhang, K. Sui, Y. Xu, J. Lou, C. Li, Y. Wu, R. Yao and M. Chintalapati, "Predicting node failure in cloud service systems," *ACM,* pp. 480-490, 2018, October.

[29] M. Saravanan And D. Sangeetha, "Adaptive Dimensional Particle Swarm Optimization Based Hyper Basis Function Neural Network Classification For Software Failure Cause Prediction.," *European Journal Of Molecular & Clinical Medicine,* Vol. 7, No. 9, Pp. 956-969, 2020.

[30] S. Ali, M. Adeel, S. Johar, 3. M. Zeeshan, S. Baseer and A. Irshad, "Classification and Prediction of Software Incidents UsingMachine Learning Techniques," *Hindawi Security and Communication Networks,* vol. 9609823, no. I, p. 16, 2021.

[31] M. Bashir, I. U. Awan, H. Ugail and Muhammad, "Failure Prediction using Machine Learning in a Virtualised HPC System and application," *Cluster Computing,* vol. 22, no. 2, pp. 471-485, 2019.

[32] S. Lu, B. Rao, X. Wei, B. Tak, L. Wang and L. Wang, "Log-based Abnormal Task Detection and Root Cause Analysis for Spark," *IEEE International Conference on Web Services (ICWS),* vol. 135, no. 1, pp. 389-396, 2017.

[33] B. Wang, Q. Hua, H. Zhang, X. Tan, Y. Nan, R. Chen and X. Shu, "Research on anomaly detection and real-time reliability evaluation with the log of cloud platform.," *Alexandria Engineering Journal,* vol. 61, no. 9, pp. 7183-7193., 2022.

[34] A. Das, F. Mueller and B. Rountree, "Aarohi: Making Real-Time Node Failure Prediction Feasible," in *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, New Orleans, LA, USA, 2020.

[35] Z. Chen, J. Liu, W. Gu, Y. Su and M. and Lyu, "Experience report: Deep learning-based system log analysis for anomaly detection," *arxiv,* 2021.

[36] R. ranjan, "medium.com," medium.com, 30 10 2022. [Online]. Available: https://medium.com/@rajeevranjancom/windows-event-log-analysis-incident-response-guide-739af79b518b. [Accessed 23 2 2024].

[37] A. Das, A. Vishnu, C. Siegel and F. Mueller, "Desh: Deep Learning for System Health Prediction of Lead Times to Failure in HPC," in *HPDC '18: International Symposium on High-Performance Paralleland Distributed Computing*, Tempe, AZ, USA, 2017.

[38] J. Gao, H. Wang and H. Shen, "Task Failure Prediction in Cloud Data Centers Using Deep Learning," *IEEE,* pp. 1111-1116, 2019.

[39] M. Mäntylä, M. Varela and S. Hashemi, "Pinpointing Anomaly Events in Logs from Stability Testing – N-Grams vs. Deep-Learning," *CoRR,* vol. arXiv:2202.09214v2, no. 2, 2022.

[40] M. Benaddy, B. E. Habil, O. E. Meslouhi and S.-D. Krit, "Recurrent neural network for software failure prediction," in *Proceedings of the Fourth International Conference on Engineering & MIS 2018*, Istanbul, Turkey, 2018.

[41] W. Yi-chen and Y.-l. Chang., ""Ransomware detection on linux using machine learning with random forest algorithm." .," *preprints posted on arXiv.,* 2024.