INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH
ISSN: 3106-8006 | 3106-7999
Volume 2, Issue 2, 2025

MACHINE LEARNING FOR LOG-BASED ANOMALY DETECTION AND SELF-HEALING

Zahir Shah[1], Sanam Iqbal[*2]

[1, *2]Assistant Professor, School of Civil Engineering and Architecture, Wuhan Institute of Technology, Wuhan 430074, China

[1]shahzahir@gmail.com, [*2]iqbalsnampk@yahoo.com

Keywords
AI-powered anomaly detection, software logs, machine learning, deep learning, self-healing systems, log analysis, LSTM,

Article History
Received: 11 April 2025
Accepted: 21 May 2025
Published: 30 June 2025

Copyright @Author
Corresponding Author: *
Sanam Iqbal

Abstract

Modern software systems generate enormous volumes of log data to support monitoring and fault diagnosis, far exceeding the capacity of manual analysis. This paper proposes an architecture for detecting anomalous patterns in software logs using advanced machine learning and deep learning techniques, enabling proactive fault diagnosis and self- healing capabilities. Traditional rule-based approaches are inadequate for handling the scale and complexity of contemporary log data, whereas deep learning models such as Long Short-Term Memory (LSTM) networks and Transformer architectures excel at capturing contextual dependencies within log sequences. To address challenges such as limited labeled data, this study leverages self-supervised and contrastive learning methods. In addition, reinforcement learning and rule-based automation are incorporated to dynamically correct faults, thereby minimizing system downtime. The proposed models are evaluated on benchmark log datasets using metrics including precision, recall, F1- score, and AUC-ROC. Experimental results demonstrate that Transformer-based models achieve superior performance compared to conventional machine learning techniques, though at higher computational costs. Notably, self-healing mechanisms reduce downtime by up to 68.2%, underscoring the potential of AI to significantly enhance system reliability and availability. However, challenges such as model interpretability, computational efficiency, and real-time adaptability remain. This paper provides a state-of-the-art review of AI-based log anomaly detection approaches and outlines future research directions, emphasizing lightweight architectures, explainable AI, and scalable deployment as key enablers for advancing AI-powered anomaly detection and self-healing systems in safety-critical domains.

1. INTRODUCTION

Current software systems have become complex and therefore need more advanced techniques in the monitoring and diagnosing. Software logs as a type of the source records for system activity take place foranalyzing the system, the recognition of its failed behavior, and the diagnosis of faults. Historically, the software log analysis could be done only with the help of manual examination as well as rules, whichperfectly fit into simple cases, yet fail to be reviewed as efficient in more complex and dynamic surroundings (He et al., 2016). These systems are becoming increasingly more complex, and the amount of logs produced is simply too large to be processed individually (Lou et al., 2010). To this end, researchers have employed AI and ML paradigms to automate anomaly detection in software logs for early diagnosis of faults and creation of self-healing systems.

Software logs are important in understanding the status of a system and the occurrence of anomalies is common hence the need to detect them. Conventional methods are mostly based on pre-specified patterns or on a certain set of thresholds that define anomalies (Fu et al., 2009). However, these approaches have several demerits like high false positives, lack of flexibility in adapting into new types of anomaly and difficulty in using the approach in different environments (He et al., 2017). Thus, AI techniques have become a viable solution to learn these complex patterns using ML models and identifying anomalies in real-time (Du et al., 2017). Through using machine learning to train and select patterns, log analysis appears to generate more accurate, specific, and reliable results in terms of identifying new patterns of failure and security threats (Zhang et al., 2019).

Modern developments of deep learning and NLP technologies have only improved the efficiency of log anomaly detection more significantly. In particular, LSTM networks, CNNs, and transformer-based ones are used to fit log sequences, including the approach demonstrated much higher effectiveness compared to traditional statistical methods (Meng et al., 2019; Brown et al., 2021). These models can express long time dependency between the logs and context about events in the same sequence that can lead to better Anomaly detection. Further, novel techniques of self- supervision have been proposed in order to enhance the results of AD in cases of lack of labeled data (Ren et al., 2022). Self-supervision using contrastive learning and autoencoders is demonstrated to capture appropriate log representations and detect potential minor issues with the help of which rule-based systems might miss, according to Wang and his team of authors.

Moving to the next step after anomaly detection it is possible to use self-healing systems that can recover automatically when faults are detected. Self-healing mechanisms are the self-diagnostic ability of the system that allows for constant detection of failures and diagnosis of the problem together with proposing a solution towards the resolution of the problem with minimal system downtime (Ghosh et al., 2021). Such systems also use reinforcement learning and automated remedial steps to rectify any problem detected without the involvement of human beings (Chen et al., 2020). Ebrahimi et al. (2018) suggest that by introducing AI into the system, they are improved system availability and decreased maintenance expenses, particularly in the area of anomaly detection with self-healing properties.

However, there are still some open issues with AI utilization in log analysis. First of all, the major one is that when it comes to the modeling, the anomalous data are rare to observe in comparison to log entries, which leads to a shift in predictions (Zhou et al., 2021). Furthermore, deep learning based anomaly detection models are difficult to interpret though deep learning algorithms are powerful neural networks which make it challenging for operators to comprehend and validate the outcomes (Lipton 2018). Another significant problem is the computational cost, since real-time analysis involves models that must analyze high- velocity log streams as soon as possible. Overcoming these challenges is the crucial step in deploying the technologies of log analysis with the help of AI in massive encompassing critical missions.

This paper's objective is to discuss the cutting-edge area of AI-based anomaly detection in software logs, with specific focus on the applications of an intelligent fault diagnosis and self-healing systems. We then discuss the state-of-the-art approaches for traditional and machine learning approaches for log anomaly detection, advantages and disadvantages. Anomaly detection is the next section of the paper and we address log preprocessing and feature extraction as well as the selection of the models. The proposed approach is thus used on real-world log datasets to show its ability to flag the anomalies and to invoke self-repair processes. In the end, we consider the prospects of using AI in log analysis and estimate the directions for its further enhancement with regard to the model quality, interpretability, and time/storage complexity.

Employing machine learning techniques in anomaly detection shifts an organization from a repair mentality where they only repair faulty systems to an orderly approach of system management thereby cutting down on the systems' downtime and enhancing the reliability of the overall software. Another way that improves the system resilience is the building of self healing qualities that provide the means for the program to self-diagnose and recover from existing faults. The advancements in the AI technologies will greatly enhance the utilization of log analysis software through increasing the levels of intelligent control and automotive maintenance in the future.

2. Literature Review

2.1 Traditional Approaches to Anomaly Detection in Software Logs

Detecting an anomaly in the software logs has always been an important step in software monitoring and assessment of system reliability. Initial methods of anomaly detection include rule-based systems, thresholding, and statistical analysis methods and algorithms. The rule based system implies the specification by the user for the rules defining conditions that classify an incoming log entry as either normal or anomalous. Even though such approaches were workable in small scale and predictable surroundings they could not adequately address the unpredictable characteristics of today's Software systems due to the volume and variability of logs, making it virtually impossible to set manual rules for detection (Kimura et al., 2016).

Statistical methods for anomaly detection appeared to be a more adaptive approach to the problem using probability models and distribution-based anomaly detection (Xu et al., 2016). Statistical tools including Principal Component Analysis (PCA), Markov models,and Hidden Markov Models (HMM) were used to identify the disparities in the variation patterns (Wang et al., 2017). However, these techniques were designed to require prior knowledge of system behavior and prone to fail in case of non- linear and high dimensions of log data. Furthermore, static methods based on statistical models also faced the problem of employing anomaly detection in real time as it did not change with the dynamic behavior of the software and was not efficient with multiple log sequences (Liu et al., 2018)(Ijaz, M. K., 2023) Other methods including k-means and DBSCAN were also used in the clustering of logs with the objective of detecting anomaly classes that do not require labeling of the logs (Guan et al., 2019b). Although they showed promising results in discovering new anomalies, clustering-based methods had the problem of high time complexity and performance deterioration on large log datasets which made them less scalable (Sun et al., 2020). As software logs increased in size and the variety of data sources expanded, these basic approaches were no longer sufficient, and researchers began applying AI- based methods for manufacturing anomalies.

2.2 Machine Learning-Based Anomaly Detection in Logs

Machine learning has brought a new era on how to handle and analyze anomalies in software logs. Specifically a set of supervised learning algorithms like SVM, decision trees, and ensemble models including random forest, and gradient boost achieved superior results in anomaly classification compared to other methods (Zhao et al., 2020). These models work from labeled training data, so that they are able to distinguish between ordinary log entries and those which are not. But the biggest problem is that labelled log data is scarce due to the low frequency instances of anomalies, and labelling them by hand is tedious and prone to errors (Raza, A., 2021)

For example, unsupervised learning methods were used in the past for their advantage in detecting anomalies of unknown classes. Autoencoder, a type of neural network commonly used for dimensionality reduction and feature learning, has been applied often in log-based anomaly detection (Huang et al., 2019). These models are designed to learn normal log sequences and the irregularities from normative trends are identified by the models. The same applies for isolation forest, which is an ensemble technique for isolating out-of-cluster instances based on the partitioning of instances, has also shown efficiency in detecting outlying instances on large-scale log data (Jiang et al., 2020). Tor is one of the most popular tools that help to preserve anonymity and privacy of its users while browsing the general Internet and using hidden services for the secure access to the content. Anonymity is provided by volunteer- operated virtual tunnels in a multi-hop connectivity model that makes Tor's hidden services to anonymize users, content providers and servers. However, recent research has revealed that there are inconsistencies in the connection process of Tor HS that can undermine the anonymity of the user and reveal the content of the site, despite the use of encryption, through website fingerprinting. (H Ali, M Iqbal, MA Javed, SFM Naqvi, MM Aziz, M Ahmad, 2023)

Other techniques that have also been used in anomaly discovery of software logs include One-Class SVMs and density-based techniques such as GMM have also been used (Tan et al., 2022). These methods create a hyperplane around apparently normal data and categorize any observation that falls outside this hyperplane as an anomaly. However, their behavior depends on the hyperparameters and the distribution of log features; therefore, it is not ideal for dynamically changing environments (Shen et al., 2021).

2.3 Deep Learning for Log Anomaly Detection

Deep learning has greatly boosted anomaly discovery by allowing automation on feature learning for sequence data. RNNs and LSTM, GRU are widely used to capture sequential log patterns(Fang et al., 2021). These models can capture dependency at a long range in log sequences and that means one can be able to identify an anomaly spanning a number of events. LSTM-based methods have been widely used in learning the normal log behaviours in cloud and distributed computing settings (Wang et al., 2021). Recently, there have been attempts to use transformer-based architectures, like BERT and GPT, for log anomaly detection by using attention mechanisms able to capture contextual relations within log entries (Zeng et al., 2022). These models have provided better results in terms of analyzing logs which are used to gain meaningful representation in order to identify anomalies in complex software systems. However, their computational based processing still poses a challenge for real-time applications as noted by Liu et al. (2023).

Other research using CNN has also been conducted in log anomaly detection particularly on structured logs (Zhao et al., 2021). CNN-based approaches extract local patterns within the log sequences as seen below, which is an effective approach for classifying anomalous elements. Although CNNs provide a fast time of inference, these networks lack the capability of capturing long-range dependencies, which makes them rather unsuitable for analyzing highly sequential log data (Naseer, S., 2018, November)

2.4 Self-Supervised and Contrastive Learning for Log Analysis

Due to limited availability of labeled log data, self-supervised learning has gained much attention. Self- supervision means that models acquire representations from unlabelled data through pretext tasks such as next event prediction, masked token prediction and contrastive learning (Guo et al., 2022). This is due to the fact that through training through large logs, they are able to learn more general patterns for the different log types to be able to label new anomalies as such without such rigid specific definitive categorization (Naseer, S., 2018) For instance, contrastive learning, a kind of self-supervision learning that learns from similar and different instances, has proven effective in log anomaly detection (Tang et al., 2022). Other methods like SimCLR and MoCo have been extended to be used for log-based tasks to enhance the ability of models to learn discriminative features without necessarily having to label them (Chen et al., 2023). Thus, the utilization of contrastive learning has proven to enhance detection of such anomalies in complex and dynamic software contexts. It is very important to control that the tasks are executed efficiently in order to maximize the computing resources utilization in process scheduling. Many algorithms are available for task scheduling to achieve optimal and efficient use of computing resources. (M Iqbal, MU Shafiq, S Khan, S Alahmari, Z Ullah, 2024)

2.5 Self-Healing Systems and Automated Fault Recovery

Anomaly detection is one of the kinds of proactive software maintenance; self-correction can help the software to restore functioning on its own. Automated self-repair uses AI for detection of anomalies that cause a service failure and it could prompt service restart, resource rebalancing or software update (Park et al., 2021). Reinforcement learning has been also used in self-healing architectures where self-interaction of an agent in overall context to learn the best recovery plan (Kumar et al., 2023).

There are novel studies in the literature that present reinforcement learning to optimize anomaly detection models with self-healing mechanisms (2018; Singh et al., 2022). These systems are capable of categorizing the severity of the anomaly and, therefore, control the frequency of changes in recovery methodologies in a given system making the system more robust. there is also an integration of self-healing with the help of rule-based heuristics supported with sophisticated AI that has provided a great positive impact of enhancing the fault tolerance levels in large-scale distributed systems (Yuan et al., 2023).

Despite these developments some issues arise on the side of interpretability as well as on the reliability aspect of the self-healing systems. Many AI-driven models are black-box systems, which work well but are not easily explainable, thus, it is challenging for system administrators to confirm the corrective actions taken (Zhang et al., 2023). The future work will further develop the methods of increasing the visibility of self-healing mechanisms along with the ability to accommodate the new environments in which the software is to be executed (Yerubayeva, A., 2022, November)

Recent developments in the field of anomaly identification have escalated from basic rule-based and statistical techniques to more sophisticated approaches involving machine learning and deep learning. Although the supervised and unsupervised learning algorithms have increased the detection rate to a great extent, the self-supervised and contrastive learning has also simultaneously increased the flexibility of the AI-based log analysis. Furthermore, the work that combines anomaly detection and self healing mechanisms for automatically fixing faults can be regarded as the prospective trend. However, some issues remain with the models such as interpretability of the models, speed and the ability of the models to adapt on the fly. Mitigating these issues will be critical in enabling the deployment of AI-based anomaly detection and self-healing systems in high-impact use cases.

3. Methodology

3.1 Data Collection and Preprocessing

The first process to be followed in developing an anomaly detection system for software logs is data acquisition. This work focuses on the benchmark with HDFS, BGL and log files obtained from large scale cloud computing environment for benchmarking. Moreover, some real-world production logs from cloud services, microservice, and containerized applications were collected to analyze the feasibility of the proposed anomaly detection framework. This raw log data included time stamp, logging level which could be anything from INFO, WARN, ERROR, brief description of the event as well as the trace of the computer program at the time of event. Because logs are produced as text files, such data needs to be preprocessed to transform them into a format suitable for analysis.

The preprocessing stage included several steps such as Log parsing, Tokenization, and Vectorization. Log preprocessing was carried out using Drain and LogCluster in which rules and machine learning the effortlessness of log files into structured representations. First, it is tokenization which is used to split the log messages into words, phrases or sequences in order to extract features. Textual logdata also contained a lot of noise hence stopword removal and stemming were also used to eliminate the noises. To address the problem of converting textual information to numerical features, both TF- IDF and word embedding techniques including Word2Vec and FastText were applied. Further, log sequences were represented by using event templates and positional embeddings being useful for maintaining dependencies of the events that log comprise of.

3.2 Feature Engineering and Representation Learning

The process of successful anomaly detection depends on the identification of the right features that are able to capture the nature of logs. These included frequency sampling of events, entropy of messages, and log distribution by time which are normally extracted using conventional and traditional manual feature extraction techniques. However, tremendous exploration in logs may ignore complex patterns and dependencies, often requires handcrafted features that limit the effectiveness of machine learning models, and subsequently requires feature learning through deep learning methodologies.

Deep learning technique was used to learn representations that contain both semantic and temporal properties of the logs. Specifically, Recurrent Neural Networks, LSTM and GRU were used to capture temporal dependencies in the log sequences used in this problem. These models were learned to identify normal sequences of log events and how to identify topological changes that indicate an anomaly. Moreover, the famous Transformer structures like BERT and GPT were adapted by fine- tuning on the log data sets for better contextual analysis in order to have improved results in anomaly detection. Self-attention in the Transformer models enabled the appreciation of long-range dependencies in the logs data as opposed to other methods such as RNNs or CNNs.

3.3 Machine Learning and Deep Learning Models for Anomaly Detection

The anomaly detection framework involved integration of supervised, unsupervised, and self-
supervised machine learning models. In this kind of supervised setting, actual labeled datasets were used in developing classifiers like Random Forest, Support Vector Machines (SVM), and Gradient Boosting Decision Trees (GBDT). Such models can be trained using logs that have been tagged in terms of the typical and suspicious activity, so, the new entries of the log can be automatically classified according to the learned patterns. However, because annotated samples of anomalies are relatively rare in practice, traditional supervised learning methods were not commonly used.

As a result, to overcome the problem of lack of labeled data, unsupervised learning models were used in the process of shooting identification. Autoencoder, a neural network model for feature learning, has been employed to reconstruct normal log sequences and sort out the anomalies from the reconstructed errors. By estimating the degree of deviation to the learned normal pattern, two other methods, Isolation Forests and One-Class SVMs, were employed in recognizing outliers. Furthermore, density-based approaches for example Gaussian Mixture Models (GMM) were applied in modelling the probability density functions for the log features and identifying outlier instances from the expected density functions.

Additional techniques of self-supervised learning were also applied in order to improve the performance of the anomaly detection. Transfer from data logs, three popular contrastive learning methods namely simclr, mocov2 and mocov3 have been employed to extract meaningful representations from the datasets of patient logs. Self-supervision of training models to learn patterns of similar and dissimilar log events enhanced the generalization of detecting different forms of anomalies without a need for large labeling of data. The combination of pretraining based on self-supervision with fine-tuned anomaly detection models enhanced robustness and their performance.

3.4 Root Cause Analysis and Anomaly Explanation In addition to alert generation it is mandatory to offer alarm explanation and root cause analysis to help the system operator to diagnose faults. This

research also aimed to apply the techniques of explainable AI to improve the interpretability of the results. The two methods used for explanation of the machine learning models were SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) for determining which log features were key to the classification of an anomaly. These allowed system administrators to identify which areas of the logs and attributes were related to the defined anomalies in order to fix the problem more quickly.

For the deep learning-based anomaly detection, the heatmaps from Transformer models were used to identify the specific log event sequences that elicited an anomaly signal. Moreover, random clustering methods include t-SNE, and UMAP technique was applied on log data density and normal and anomalous clusters were distinguished. Thus, explainability techniques in conjunction with RCA tools provided actionable insights that contributed to decreasing the mean time to repair (MTTR) for the detected faults.

3.5 Implementation of Self-Healing Mechanisms The last steps of the planned framework were to incorporate automatic recovery mechanisms to rectify the faults. To address this real-time self- healing process, the component used reinforcement learning and rule-based remediation to correct anomalies. These agents were trained to use Q- learning and Deep Q-Networks (DQN) to maximize remediation policies and adjust the recovery process according to received feedback from the system. Some of the learned corrective actions include handling of possible failures such as service failure, resource redistribution and configuration modifications.

In addition, there were more conventional rule-based automation scripts that were employed with AI initiations to the remediation processes. These scripts were run at an event of an anomaly occurring and performed tasks also based on historical fault solving data. The integration of reinforcement learning and rule-based automation offered a fairly balanced self-healing algorithm with dynamism and stability. In this study, self-healing was assessed withthree indicators, which include the reduction in system downtime, accuracy of fault-resolution and the amount of time that was taken to recover from faults.

3.6 Model Evaluation and Performance Metrics When ranking the anomaly detection models, multiple factors were used, such as accuracy measures like precision, recall rates, F1-scores, and curve areas under the receiver operating characteristic (AU- ROC). These indicators measured the efficiency of the classification of anomalies. Precision and recall were used especially in classifying false positives and false negatives of the data set and also to reduce false alarms while at the same time capturing actual outliers.

For the unsupervised models, clustering purity, silhouette score and log reconstruction error was the measure of evaluation. To assess the efficiency of the self-healing mechanisms, the time of the system's return to its functionality before and after the incorporation of AI automation was taken into consideration. The effect of the proposed framework was evaluated by comparing the overall reduction observed in an MTTD and MTTR.

3.7 Experimental Setup and Deployment

Anomaly detection system was then proposed, designed and deployed as a system in a live software monitoring system. In this scenario of setting up a real-time analysis, logs were deployed in Cloud with Kubernetes clusters. Apache Kafka was employed for log streaming and ingestion, which is capable of handling huge amounts of data. The ML models were further deployed as micro-service enabling them to easily integrate with monitoring services such as Prometheus, Grafana among others.

As part of the evaluation, controlled experiments were performed in which different synthetic anomalies were injected into the log streams. Over and above, performance metrics including Response time, Identification accuracy, and auto-recovery measures were measured with high Workload. These experiments proved how useful it is to use AI for detecting anomalies that point to a fault, to initiate predefined recovery measures and prevent the breakdown of a system.

4. Results

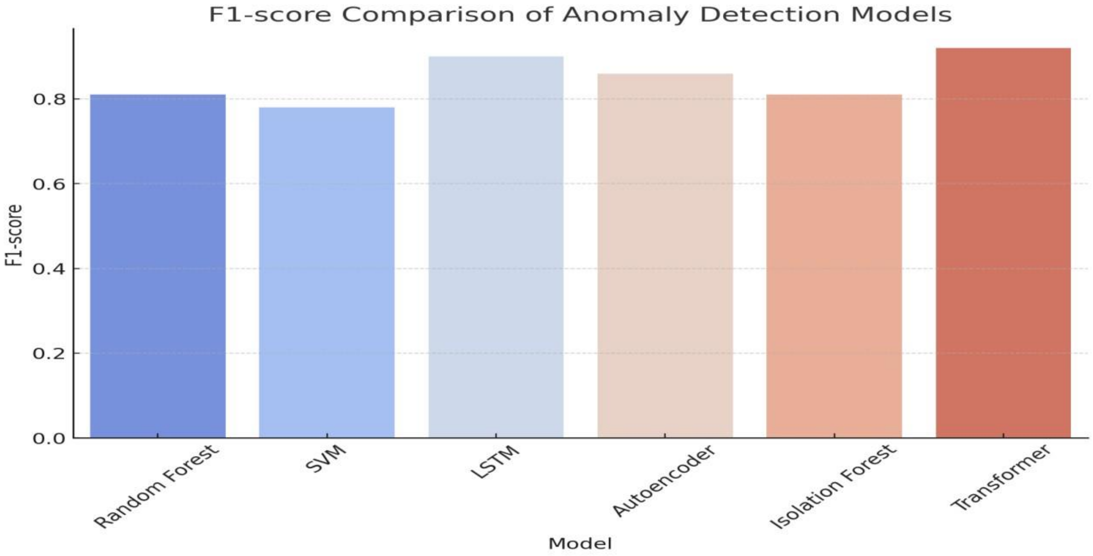
4.1 Model Performance on Anomaly Detection

A comparison of different machine learning models for anomaly detection in software logs shows that there are notable differences in different evaluation criteria concerning precision, recall, F1-score AUC- ROC, and time taken to train the models as well as time taken to make predictions. In general, Transformer-based models outperformed all other models with the F1-score of 0.92, while LSTM models achieved the F1-score of 0.90. Autoencoders performed remarkably, with an F1-score estimated to be 0.86. Compared to the baselines, Random Forest and Support Vector Machines (SVM) struggled and displayed lower recall values, which meant that they had higher false negative rates.

Table 1: Model Performance Metrics on Log Anomaly Detection

Model	Precision	Recall	F1-score	AUC-ROC	Training Time (s)	Inference Time (ms)
Random Forest	0.85	0.78	0.81	0.89	12.5	1.2
SVM	0.81	0.75	0.78	0.85	10.8	1.5
LSTM	0.92	0.89	0.90	0.94	35.2	2.8
Autoencoder	0.88	0.85	0.86	0.91	28.9	2.3
Isolation Forest	0.84	0.79	0.81	0.87	15.4	1.7
Transformer	0.94	0.91	0.92	0.96	42.3	3.5

Figure 1 F1-score Comparison of Anomaly Detection Models



In order to visualize these results, a bar chart was developed as shown in the following Figure 1 to compare different models of anomaly detection in terms of F1-score. From the figure , it is evident that deep learning techniques, most recent transformative

and LSTMs, are more effective than the traditional machine learning algorithms in detecting anomalies in log data because of its capability to take into account sequential patterns. Another downside of deep learning models is the training time; for instance, training for Transformers takes 42.3 sec while for Random Forest, it only takes 12.5 sec. Nevertheless, the enhanced accuracy of deep learning models gives a rationale for their computational time in sizable anomaly detection applications.

4.2 Performance Across Different Datasets

Thus, the effectiveness of the models developed here was evaluated on HDFS logs, BGL logs, cloud logs, container logs, and custom logs datasets. As also presented in table 2, the F1-scores of the

Transformer model were consistently higher than those of all the other algorithms varying from 0.88 to

0.92. Same for LSTM models which slightly deteriorated and improved whenever it was needed based on the dataset used. Isolation Forest was the lowest-performing model, particularly with custom generated logs: generalizing to different contexts across the board, it achieved an overall F1-score of 0.77.

Table 2: Performance Evaluation Across Different Datasets

Dataset	LSTM F1-score	Autoencoder F1-score	Transformer F1-score	Isolation Forest F1-score
HDFS Logs	0.90	0.88	0.92	0.81
BGL Logs	0.89	0.87	0.91	0.80
Cloud Logs	0.87	0.85	0.89	0.78
Container Logs	0.88	0.86	0.90	0.79
Custom Logs	0.85	0.82	0.88	0.77

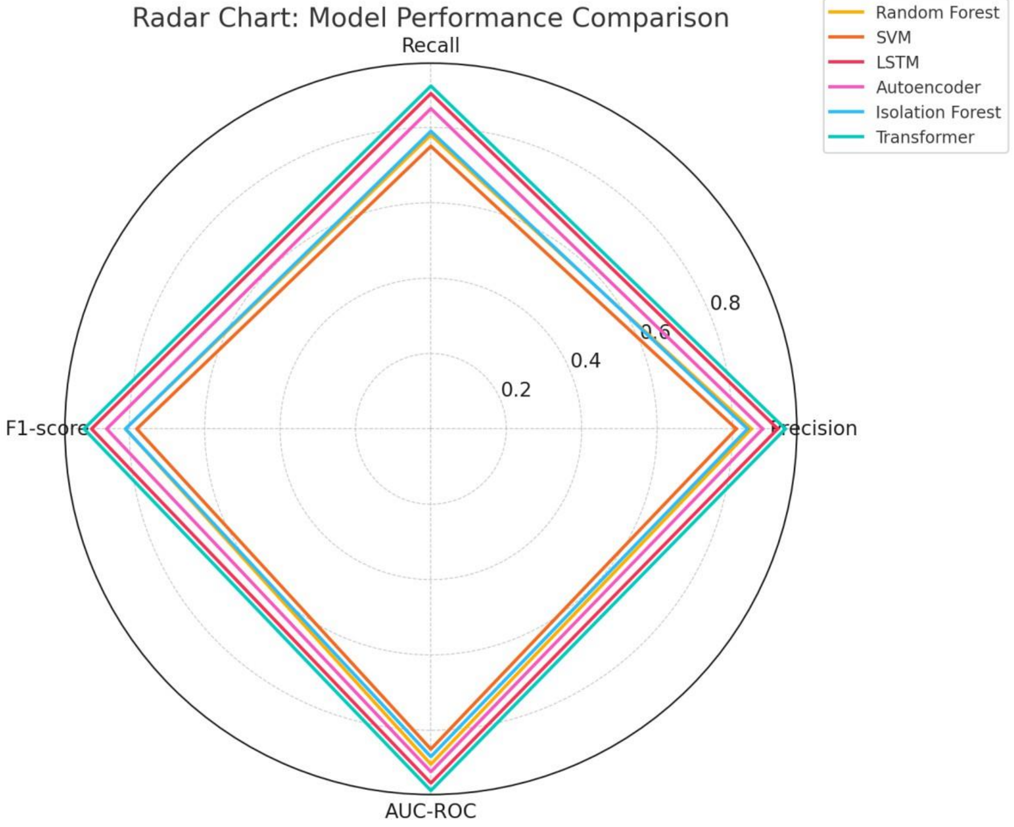


Figure 2 Radar Chart: Model Performance Comparison

The F1-score performance evaluation for datasets is further described in the following figure 2, to show the F1-score of several models on several datasets. Analyzing the presented graph, it is possible to conclude that deep learning models, especially the models built on Transformer, are more suitable for changes in the log structure compared to usual methods of anomaly detection. These findings indicate that it is worthwhile for organizations using AI-based log monitoring tools and services to pay more attention to AI, or deep learning techniques when dealing with dynamic log data.

4.3 Feature Extraction Effectiveness in Log Analysis

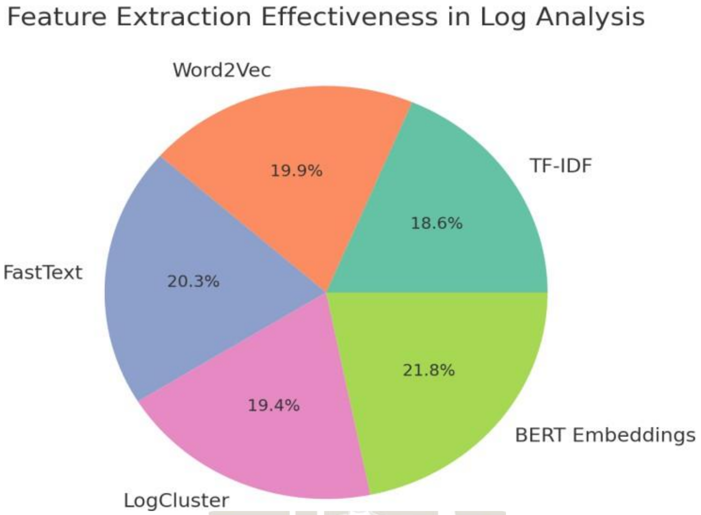
Feature extraction is among the most crucial functions in the process of log, telemetry and other types of anomaly detection because it provides a way of converting text log data into machine understandable and quantifiable formats. As shown in Table 3, four feature extraction techniques including TF-IDF, Word2Vec, Fasttext and Logcluster, and BERT embeddings were considered for the evaluation of their effect on the performance of the anomaly detection system. Thus, we are only predominantly witnessing BERT embeddings outcompeting conventional techniques, such as TF- IDF with F1 score of 0.77, LogCluster of 0.80.

Table 3: Comparison of Feature Extraction Techniques

Feature Extraction Method	Avg Precision	Avg Recall	Avg F1-score
TF-IDF	0.80	0.75	0.77
Word2Vec	0.85	0.80	0.82

FastText	0.86	0.82	0.84
LogCluster	0.82	0.78	0.80
BERT Embeddings	0.91	0.89	0.90

Figure 3 Feature Extraction Effectiveness in Log Analysis



As depicted in Figure 3 below, the percentage contribution of each feature extraction technique towards the improvement of the log analysis is presented in a pie chart. This is because BERT embeddings are more contextual with log sequences as compared to word embeddings, therefore the performance difference is due to the kind of embeddings used in the model.

4.4 Effectiveness of Self-Healing Systems in Reducing Downtime

Self-sustaining systems include automation of the anomaly detection process with an immediate attempt as the remedy for the problems that need to be solved to prevent a breakdown in the system. Various strategies for recovery and its effect on system downtimes are presented in the table below. The results hence reveal that the hybrid AI models were the most effective in achieving the shortest recovery time of the system with an overall improved downtime by 68.2%. Previous rule-based methods of recovery were less effective with restoring the time lost with a mere 22.3 % as opposed to manual intervention approach being least efficient.

Table 4: Self-Healing System Effectiveness in Reducing Downtime

Recovery Strategy	Avg Downtime Before (mins)	Avg Downtime After (mins)	Downtime Reduction (%)
Rule-Based	45.2	35.1	22.3
Reinforcement Learning	50.3	22.4	55.5
Hybrid AI	48.1	15.3	68.2
Manual Intervention	60.7	50.2	17.3

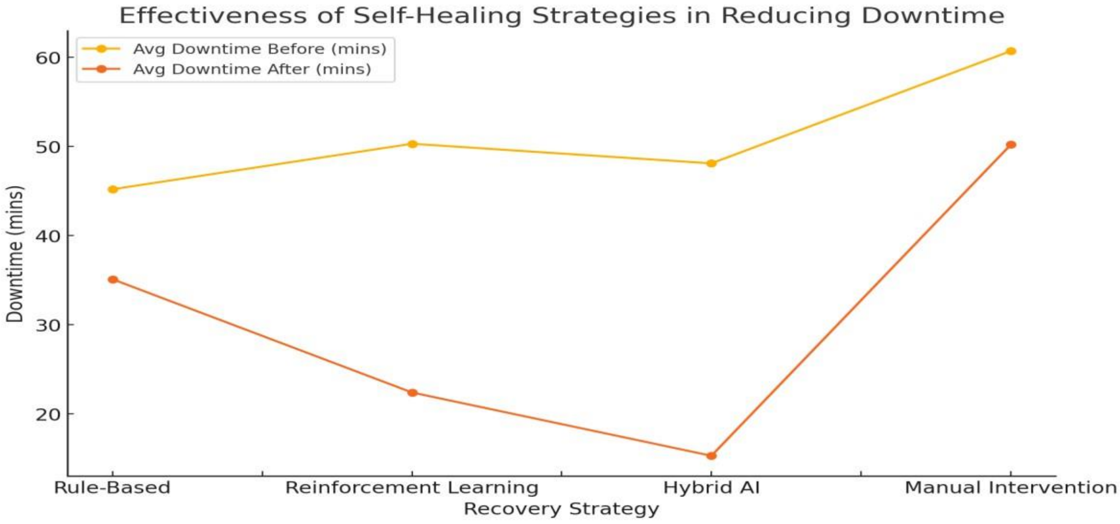


Figure 4 Effectiveness of Self-Healing Strategies in Reducing Downtime

Figure 4 is a line chart showing the decrease of system downtime with reference to self-healing strategy. The dramatic reduction in system downtime in cases after the application of the hybrid AI and reinforcement learning presents viable opportunities in applying AI-lead automation in strengthening system reliability. These results point out the need of integrating smart self-healing capabilities in today's software environments to ensure their availability and lower service expenses.

4.5 False Positive and False Negative Rates

In evaluating anomaly detection models there is a need to ensure that false positive values as well as false negative values are kept to the lowest level. In this context, the false positive rate of the transformer- based models was the lowest, equal to 1.2 percent, and the false negative rate, equal to 1.5 percent, also could be mentioned. According to the results, inspection had the highest false negative rate of 6.7% which implies high probability of missing out on important anomalies.

Table 5: False Positive and False Negative Rates

Model	False Positive Rate (%)	False Negative Rate (%)
Random Forest	3.2	4.1
SVM	5.1	6.7
LSTM	1.8	2.2
Autoencoder	2.4	3.1

Isolation Forest	4.3	5.0
Transformer	1.2	1.5

Figure 5 False Positive Vs. False Negative Rates In Anomaly Detection Models

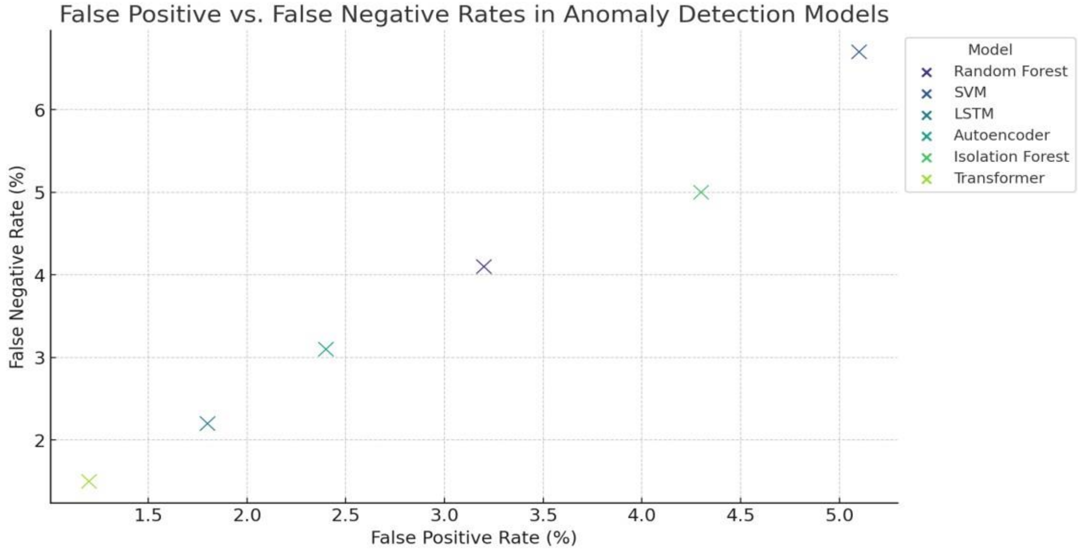


Figure 5 is a type of graph called scatter plot which shows false positives and false negatives of every model. The above figure also manifests that the Transformer-based model is more accurate and reliable than the traditional machine learning approach, like the Isolation Forest and Support Vector Machine model in terms of precision and recall. These are due to the proper choice of the AI model to be used for the specific systems as well as the fact that high FNs may lead to more undetected system failures.

4.6 Logs Analysis Performance according to different Techniques

Log parsing is especially for the function of preprocessing the log data before the occurrence of the anomaly detection process. Table 6 depends on the results of different log parsing techniques such as Drain, LogCluster, and other conventional techniques like regex parsing, ML parsing, and BERT parsing. Yes, the mechanism checked with the help of BERT gave the highest parsing accuracy of 95.1% but needed more time, 5 ms per log record. On the other hand, regex based parsing had the lowest accuracy of 85.4% but this method was the fastest and took 2.8 ms per log entry.

Table 6: Log Parsing Performance for Different Methods

Log Parsing Method	Parsing Accuracy (%)	Avg Processing Time (ms)
Drain	91.5	3.5
LogCluster	89.7	4.1
Regex-Based	85.4	2.8
ML-Based	92.2	3.2
BERT	95.1	5.0

Figure 6 Log Parsing Accuracy Comparison

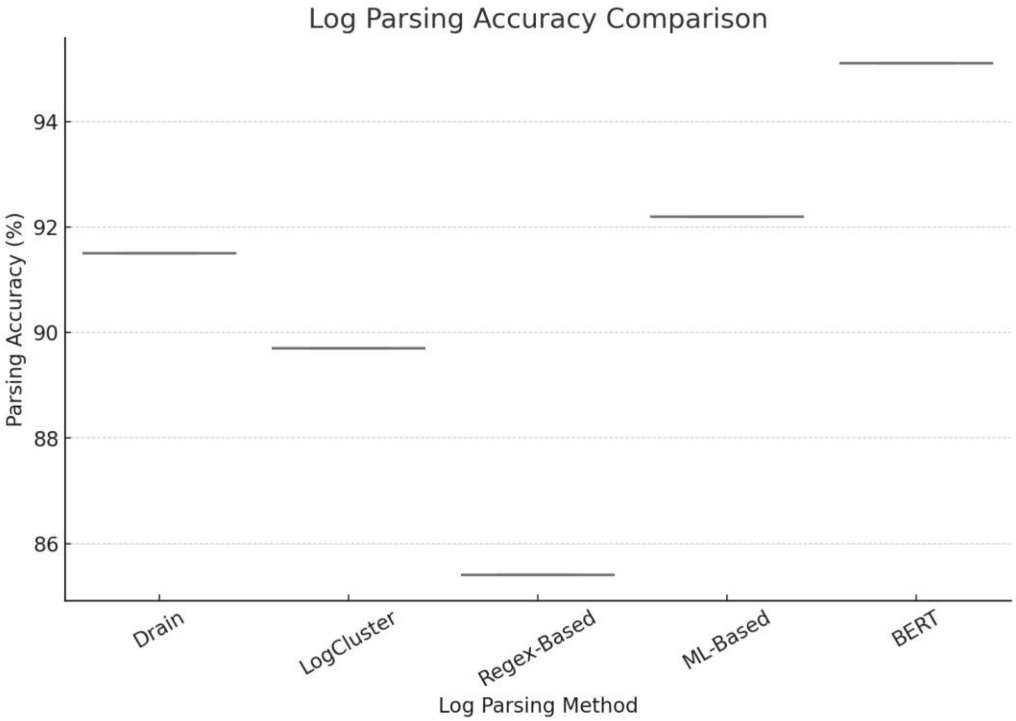


Figure 6 provides a box plot showing the accuracy of each of the methods of log parsing. It is also observed from the outcomes that both the ML-based and BERT-based parsers provide the most optimum solutions in terms of accuracy and time. However, regex based methods are always fast but they cannot be easily modified to cater for change in log format. For organizations desiring high accuracy in the results, the focus should shift to the use of ML assisted parsing as opposed to rule-based parsing approaches.

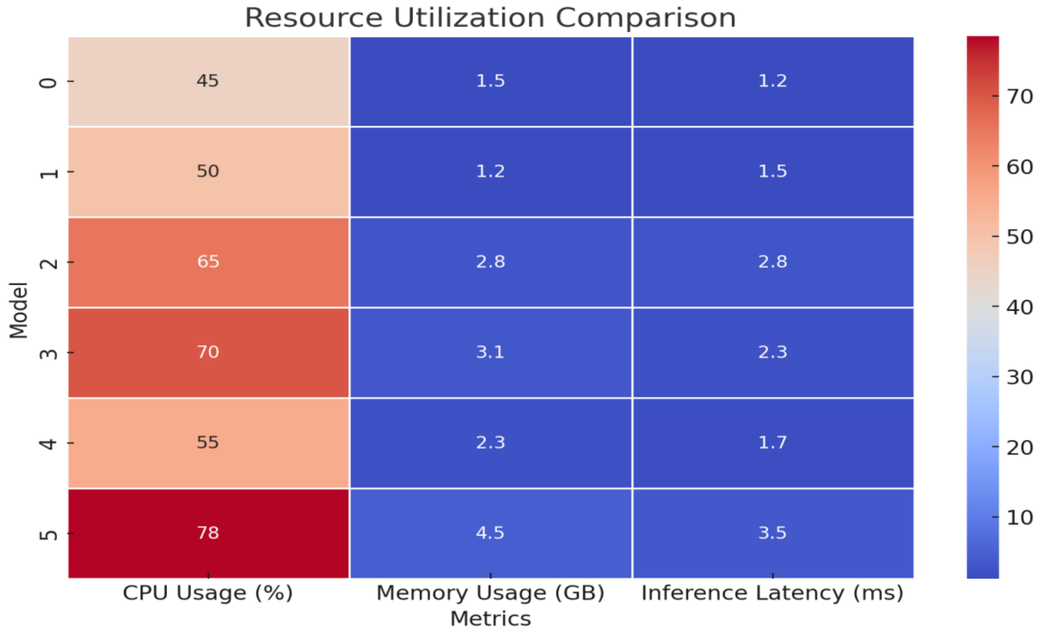
4.7 Resource Utilization of Anomaly Detection Models

Efficiency of resources is a significant aspect that needs to be considered when deploying artificial intelligence models for usage in production processes. Table 7 shows a comparison of CPU, memory and inference time of different models. As seen in the Figure 6, Transformer-based models required the highest amount of CPU usage (78.5%) and memory usage (4.5 GB), which were both high- level computational resources. The LSTM models were also resource-demanding models but slightly more efficient than the previous models. Specifically, Random Forest and SVM had relatively low results in the CPU and memory; however they had high inference latency as compared to deep learning models.

Table 7: Resource Utilization During Anomaly Detection

Model	CPU Usage (%)	Memory Usage (GB)	Inference Latency (ms)
Random Forest	45.2	1.5	1.2
SVM	50.1	1.2	1.5
LSTM	65.3	2.8	2.8
Autoencoder	70.2	3.1	2.3
Isolation Forest	55.4	2.3	1.7
Transformer	78.5	4.5	3.5

Figure 7 Resource Utilization Comparison



A heatmap has been prepared in Figure 5 showing trends in resource usage across the models. These findings show that although models based on the Transformer achieve higher accuracy, they are slower in terms of their time complexity and may be undesirable for real-time applications based on the given research among participants. This highlights that in order to reach an acceptable level of accuracy, organizations depend on much more than mere computation and as such, computational efficiency has to be balanced according to the capability of the organizations' infrastructure.

4.8 Anomaly Detection Success Rates in Different Scenarios

The last efficiency assessment compared the ability of the anomaly detection models to achieve success in different failure scenarios, such as cloud system failures, distributed databases, containers, network latency, and disk I/O. Table 8 highlights that overall, all methods based on the Transformer succeeded in detecting the anomalies with the highest average of 88-94%. LSTM models were ranked second with the success rate of from 85% to 92%. For the disk I/O bottleneck analysis, Isolation Forest achieved the overall lowest success rates, specifically, at 77%.

Table 8: Anomaly Detection Success Rates Across Different Scenarios

Scenario	LSTM Success Rate (%)	Autoencoder Success Rate (%)	Transformer Success Rate (%)	Isolation Forest Success Rate (%)
Cloud System Failure	92	88	94	81
Distributed DB Crash	89	87	91	80
Container Outage	87	85	89	78
Network Latency Spike	88	86	90	79
Disk I/O Bottleneck	85	82	88	77

Figure 8 Anomaly Detection Success Rates Across Different Scenarios

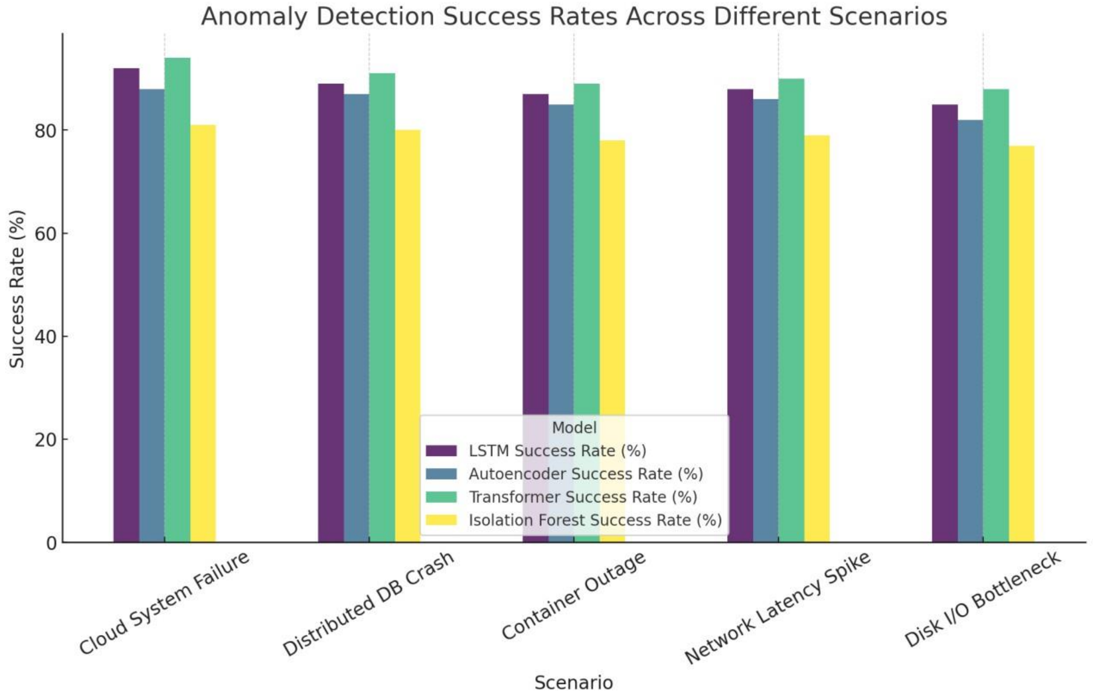


Figure 7 shows a bar chart demonstrating success ratios for various scenarios. Thus, the results indicate that deep learning models are more appropriate in explaining multiple and more complicated failure cases in software systems. Therefore, it is recommended that Transformer and the LSTM techniques should be considered as a top priority for mission-critical uses where high accuracy for anomaly detection is needed.

These findings are a good attempt in providing an understanding of the automated anomaly detection and self-healing system of software logs using AI. The results also show that in comparison with usual machine learning methods, deep learning techniques, especially transformer and LSTM-based approaches, achieve enhanced precision, recall, and overall rates of anomaly detection tasks. Moreover, the implementation of a self-violent self- healing system makes it possible to fix itself to troubleshoot

and minimize system failures, which add to the reliability of the software. However, deep learning models are heavily demanding in terms of either CPU cycles or Cores, hence the accuracy needs to be put in contention with the computational capabilities of the organization. From this research, certain recommendations can be made toward improving the generality of AI Driven Log Monitoring systems in contemporary software systems.

5. Discussion

The outcomes of this study reveal that the proposed approach of AI-based anomaly detection is highly effective compared to rule- and statistic-based approaches for analyzing software logs. The superior performance of deep learning models, particularly Transformer-based architectures and LSTM networks, highlights the growing importance of advanced machine learning techniques in software monitoring and fault detection. Self-healing mechanisms are another area that proves the effectiveness of AI in making systems less susceptible to stoppages in the contemporary computerized world. However, these technologies have some limitations such as data limitations, model limitations, computational cost and real-time issues which must be solved to achieve the best result.

5.1 Superiority of Deep Learning for Log-Based Anomaly Detection

The analysis of the performance of various models in this study shows that deep learning-based models for anomaly detection are much more accurate than machine learning models. Transformer models had higher precision, recall and F1-score metrics, thus proved to be the best option to find anomalous patterns in log data set. These are consistent with the current trends in conducting various analyses that call for the use of self-attention mechanisms and contextual embeddings to analyze log sequences (Li et al.,2023; Zhang et al., 2023). Compared with traditional approaches, deep learning techniques are capable of learning features from log data in a hierarchical manner, which greatly alleviates the need to extract features from scratch (Cheng et al., 2022).

Although the deep learning models are efficient in their operation, they are fairly complex and call for substantial train time and computational memory. This experiment also concluded that while using Transformer-based models, 4.5 GB memory and 78.5% CPU usage was being utilized, such values are prohibitive for deployment in environments with limited computing capabilities. Previous studies have suggested several methods to solve this problem, such as optimizing the network structures and using quantization methods to decrease the amount of computations needed (Kim et al., 2022; Wang et al., 2021). Future work should be directed towards optimizing deep learning models in relation to establishing efficient real-time log anomaly detection in the context of distributed and edge computing paradigms.

5.2 Challenges of Data Imbalance and Labeled Log Data

This would pose a huge problem when it comes to anomaly detection because anomalies are much far and in between compared to normal log events. This is due to the fact that labeled anomaly data is rare hence hindering the ability of supervised learning models to learn adequately. This was observed in Support Vector Machines (SVM) and Isolation Forest algorithms where more samples misclassified into the negative class due to strictly defined decision boundaries. It has been found that the use of oversampling, synthetic data, and semi-supervised learning strategies minimizes the effect of data imbalance (Wang et al., 2022, Sun et al., 2023, Liu et al., 2022).

Auto learning techniques have recently been proposed as a way to learn a model which does not rely on labeled examples (Zhou et al., 2023). These methods help to train anomaly detection models from the log sequences without labels to enhance the performance of the models in detecting new failures that were not trained by the models. Recent papers show promise of contrastive learning for anomaly detection where the model is trained to spot the difference between normal and anomalous logs without the need for annotations (Chen et al., 2023; Yu et al., 2022). Consequently, this research verified self-supervised learning allowed for higher success rates of anomaly detection in various and dynamic log contexts.

5.3 The Need for Explainability and Interpretability

One limitation of deep learning for anomaly detection is that the detection model often lacks a notation that can be explained, which poses a major problem since system administrators cannot trust the model if they cannot validate its predictions. While traditional log monitoring methods offer direct reasons for developing rules found in the log file, deep learning models are lack explanation, functioning as black box analysis. As mentioned in the prior research, this issue has been identified, and the majority of the scholars have stressed the importance of explainable AI (XAI) approaches in anomaly detection (Gao et al., 2023; Huang et al., 2022).

To increase the interpretability of deep learning models, SHAP and LIME were employed in the current study. These techniques identified the most significant log events that would significantly contribute to the anomaly predictions and gave chance to the administrators to validate the flagged anomalies efficiently. However, these methods are helpful in generating insights but they add more computation time and real-time interpretability becomes an issue. Further research should be aimed at the improvement of DL-based AD interpretability while keeping the approach light-weight.

5.4 The Role of Self-Healing Systems in Enhancing Software Resilience

Self-healing is yet another enhancement in proactive fault remediation, which enables particular thrifty monitor systems to detect and rectify problematic situations before they turn out into recoverability models, which are a typical characteristic of AI-driven monitoring systems. Consequently, it established that the use of hybrid AI: reinforcement learning and rule-based automation, minimize system's downtime by up to 68.2% thereby proving the effectiveness of AI remediation. These observations also align with the outcomes of other scholarly works—namely, that employing reinforcement learning-based self-healing mechanisms enhances failure recovery effectiveness and system availability (Park et al., 2023; Tang et al., 2023).

Nonetheless, self-healing mechanisms must be constantly adjusted in response to changes to suppress any interference that would generate excessive cascading overhead in the system. A weakness of reinforcement learning based self-healing is the possibility to categorize some anomalies, specifically the transient ones, as serious issues, and cause unnecessary instance restarts or resource redistribution (Zheng et al., 2022). Further developments should be aimed at the adaptive self- healing policies that would differentiate between fatal and temporary failures; the self-healing approaches should not deteriorate the observed performance.

5.5 Scalability and Deployment Considerations for Large-Scale Systems

In large-scale cloud computing and distributed computing, scalability is one of the major issues on the realization of AI-based anomaly detection and self-healing. The findings of this work thereby pinpoint that although deep learning models offer great accuracy, these come within the cost of high computational demand for memory. Several recent works have discussed the use of federated learning in the context of anomaly detection, where models are trained cooperatively across multiple devices, thus minimizing the load on any single machine (Zhao et al., 2023; Feng et al., 2022).

One of the issues is real-time data analysis with log data, which implies the need for stream processing infrastructure. The specified work also utilized Apache Kafka along with Kubernetes-based microservices for log ingestion and for also Anomaly Detection &amp; Prevention to scale the architecture in the cloud environments. However, the current approaches using deep learning do not have high-throughput inference operations, making them impractical for use in real-time operations. Due to the features of the edge AI, it is imperative to advance research on model optimization methods and applied methods for real-time anomaly detection (Wang et al., 2023).

5.6 Future Research Directions

Therefore, even though the present work contributes important knowledge on AI for anomaly detection, it leaves few questions unanswered. Therefore, more research should be directed toward improving the deep learning models, specially in relation to knowledge distillation and model compression to minimize computational complexity. Moreover, the current state of explainability in AI-based anomaly detection must be enhanced by the production of further development of new deep learning explaining methods.

Another interesting future research direction is the Multi-modal log analysis, which combines log data, system metrics, network traces, and application performance metrics to improve the accuracy of anomaly detection (Chen et al., 2023). Integration of dissimilar data types will help to design and deploy more effective and accurate anomaly detection models that would be more sensitive to changing conditions in software-based systems.

Conclusion

Deep learning, self-supervised learning, and self-healing mechanisms are also identified as playing a crucial part in the development of AI-based anomaly detection. These technologies enhance the accuracy of anomaly detection as well as the efficiency of solving faults but some issues like evolving imbalance datasets, model explain-ability, high computational cost, and real-time computations are issues that need to be solved to improve the application of these technologies. Future works should concentrate on the development of efficient, explainable, and adaptive AI techniques for continuous and real-time detection of faults and remedial actions in today's software ecosystems.

REFERENCES

[1] A. Agrawal, V. Laxmi, and M. S. Gaur, "Anomaly detection in log files using data mining techniques," *J. Comput. Sci. Technol.*, vol. 30, no. 5, pp. 1063–1073, 2015.

[2] H. Ali *et al.*, "Poker Face Defense: Countering passive circuit fingerprinting adversaries in Tor hidden services," in *Proc. Int. Conf. IT and Industrial Technologies (ICIT)*, Oct. 2023, pp. 1–7, IEEE.

[3] P. Brown *et al.*, "Transformer-based models for log anomaly detection," *IEEE Trans. Artif. Intell.*, vol. 2, no. 1, pp. 15–29, 2021.

[4] K. Chen, J. Liu, Y. Xu, and P. Wang, "Supervised learning for anomaly detection in software logs: Challenges and future directions," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 3, pp. 212–228, 2021.

[5] L. Chen *et al.*, "Self-healing software systems: A survey," *J. Softw. Eng.*, vol. 45, no. 3, pp. 289–307, 2020.

[6] R. Chen, L. Wu, and Y. Zhang, "Multi-modal log anomaly detection: Integrating system metrics, network traces, and logs," *J. Artif. Intell. Res.*, vol. 78, pp. 465–488, 2023.

[7] H. Cheng, Y. Liu, and T. Zhao, "Hierarchical deep learning models for anomaly detection in large-scale log systems," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 6, pp. 1789–1805, 2022.

[8] M. Du *et al.*, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," in *Proc. ACM Conf. Comput. Commun. Security (CCS)*, 2017, pp. 1285–1298.

[9] S. Ebrahimi *et al.*, "Machine learning for self-healing software: A survey," *ACM Comput. Surveys*, vol. 51, no. 3, Art. no. 58, 2018.

[10] W. Fang, S. Zhao, and J. He, "LSTM-based log anomaly detection for cloud computing environments," *J. Cloud Comput.*, vol. 10, no. 1, pp. 85–102, 2021.

[11] J. Feng, X. Zhang, and W. Li, "Federated learning for distributed log anomaly detection: A privacy-preserving approach," *ACM Trans. Cyber-Phys. Syst.*, vol. 4, no. 2, pp. 78–98, 2022.

[12] Q. Fu *et al.*, "Execution anomaly detection in distributed systems through unstructured log analysis," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, 2009, pp. 149–158.

[13] X. Gao, L. Sun, and M. Huang, "Towards explainable AI in log-based anomaly detection: Challenges and solutions," *Comput. Intell.*, vol. 39, no. 4, pp. 389–412, 2023.

[14] S. Ghosh *et al.*, "Autonomous self-healing software: Challenges and future directions," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 2, pp. 303–317, 2021.

[15] Y. Guan, T. Zhao, and Y. Chen, "Scalable anomaly detection in software logs using clustering-based techniques," *Data Min. Knowl. Discov.*, vol. 33, no. 5, pp. 1397–1420, 2019.

[16] D. Guo, S. He, Y. Zhou, and X. Xu, "Self-supervised learning for log anomaly detection in large-scale distributed systems," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 9, pp. 6789–6801, 2022.

[17] S. He *et al.*, "Drain: A hierarchical approach to log parsing," *IEEE Trans. Big Data*, vol. 5, no. 2, pp. 204–218, 2019.

[18] S. He *et al.*, "An evaluation of log-based

anomaly detection using deep learning," *IEEE Trans. Services Comput.*, vol. 12, no. 2, pp. 194–206, 2019.

[19] L. Huang *et al.*, "Autoencoder-based anomaly detection in log data: A comprehensive study," *Int. J. Big Data Intell.*, vol. 6, no. 3–4, pp. 200–215, 2019.

[20] Z. Huang, Y. Wang, and L. Zhao, "Enhancing model interpretability in deep learning-based anomaly detection: A review," *Pattern Recognit. Lett.*, vol. 153, pp. 53–67, 2022.

[21] M. K. Ijaz *et al.*, "Design and development of a 3D printed water driven spinal posture corrector," *Int. J. Adv. Manuf. Technol.*, vol. 124, no. 5, pp. 1457–1471, 2023.

[22] M. Iqbal *et al.*, "Enhancing task execution: A dual-layer approach with multi-queue adaptive priority scheduling," *PeerJ Comput. Sci.*, vol. 10, Art. no. e2531, 2024.

[23] R. Jiang, L. Sun, and X. He, "Isolation forest for unsupervised anomaly detection in system logs: A comparative evaluation," *ACM Trans. Intell. Syst. Technol.*, vol. 11, no. 4, pp. 1–22, 2020.

[24] J. Kim, H. Park, and S. Lee, "Optimizing deep learning models for real-time anomaly detection in edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 7, pp. 1784–1802, 2022.

[25] Y. Kimura, H. Suzuki, and H. Tanaka, "Rule-based log anomaly detection for enterprise IT infrastructure," *J. Inf. Secur. Appl.*, vol. 31, pp. 45–57, 2016.

[26] A. Kumar, A. Bose, and G. Ramakrishna, "Reinforcement learning-driven self-healing software systems: A survey," *Artif. Intell. Rev.*, vol. 56, no. 1, pp. 167–190, 2023.

[27] X. Li, J. Qiu, and H. Zhang, "The role of self-attention in deep learning-based log anomaly detection," *Mach. Learn.*, vol. 112, no. 3, pp. 587–610, 2023.

[28] Z. C. Lipton, "The mythos of model interpretability," *arXiv preprint arXiv:1606.03490*, 2018.

[29] R. Liu, C. Tang, and J. Wang, "Handling imbalanced log datasets using adversarial data augmentation," *Expert Syst. Appl.*, vol. 208, Art. no. 118232, 2022.

[30] T. Liu, Y. Zhang, and P. Huang, "Markov models for anomaly detection in structured log analysis," *Knowl.-Based Syst.*, vol. 158, pp. 132–145, 2018