SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V12I10P110

Volume 12 Issue 10, 104-115, October 2025

Original Article

An Evaluate Automated Anomaly Detection Methods for Efficiently Analyses Large-Scale Workflow System Logs, Overcoming the Limitations of Manual Inspection and Basic Statistical Techniques

Arun Kumar Bandlamudi[1] , Sunitha Pachala[2]

[1,2]*Department of CSE, Koneru Lakshmaiah Education Foundation, Green Fields, Vaddeswaram, Guntur Dist, Andhra Pradesh, India.*

[1]*Corresponding Author : bandlamudi8677@gmail.com*

Received: 10 August 2025 Revised: 12 September 2025 Accepted: 11 October 2025 Published: 31 October 2025

Abstract - *Traditional manual inspection and simple statistical analysis of them, though, have proved unrealistic due to the growing complexity and size of workflow system logs in an enterprise and cloud environment, and thus anomaly detection has become complex and time-consuming. In the present paper, automated anomaly detection methods that will eliminate these limitations will be assessed. We analyze the machine learning-based, deep learning-based, and hybrid models regarding the detection of anomalies in large and heterogeneous log data. We evaluate the efficiency, scalability, and detection accuracy of all methods by carrying out experiments on real-life workflow logs of cloud systems. Our results illustrate that automated techniques, especially those taking advantage of unsupervised deep learning models such as auto-encoders and LSTM-based networks, are much more successful than manual and statistical approaches in identifying the small and potentially new anomalies. This experimental design gives a comparative framework, which may guide the adoption of scalable and intelligent anomaly detection systems in large workflow settings in the future.*

Keywords - *Anomaly detection, Workflow logs, Machine Learning, Deep Learning, Autoencoders, Log Analysis, Scalability, System monitoring, LSTM, Unsupervised learning.*

1. Introduction

The high rate of growth of digital infrastructures in the industries has given more complexities in the management workflow systems. The systems designing, performing, and facilitating a wide range of activities in enterprise IT anomalies produce ample logs with tiny details of the system's working process. Logs play an essential role in determining the state of operating activities, exploring the breakdown of the system, and determining the pattern of execution. With increasingly more workflows being digitized and automated within an organization it becomes essential to monitor any irregularities or anomalies in the system so that it may maintain reliability and security [14].

The classical methods of detecting anomalies in system logs include manual analytical methods and primitive methods of statistical analysis. Visual inspection is labor-intensive and time-consuming since it also suffers from a lack of consistency and precision as the log size increases. Furthermore, it is also reactive, meaning that administrators usually look through the logs only after a problem or an incident happens. Simple statistical methods like moving averages, detection thresholds, and typical deviations, although being cheap in terms of computation time, are often not enough to reflect the non-linear relationships that may exist in modern multi-stratified systems. Such techniques tend to rely heavily on the assumption that the system extends behavior according to a predictable distribution, which is not true in most of the dynamic and large-scale environments [2].

Anomaly detection has become one of the more practical ways of doing analytics in real time with the advent of big data and automation of the detection process. Utilization of these strategies helps to emulate the normal behavior of the system with the help of previous log patterns and subsequently alerts to irregularities that could be mistakes, misconfigurations, or other forms of hacking. Unsupervised machine learning models have been particularly promising in this field as they are unaffected by the lack of labeled data since they are used to identify unprecedented or novel forms of an anomaly. Some of the common techniques to model the distributions of log data and observation of outliers include

clustering, isolation forests, and dimensionality reduction. Recently, deep learning models (particularly models that can learn time-dependent time series, such as Long Short-Term Memory (LSTM) networks and autoencoders) have been considered to learn complex and non-linear patterns in high-dimensional log data.

Nonetheless, there are obstacles to the use of these automated procedures. Logs are in unstructured and semi-structured formats, and require proper preprocessing and parsing. Moreover, the workflow logs are presented in a heterogeneous manner as they have different types of events, different timestamps, and different sequences. A good anomaly detection system should thus generalize reasonably well to a variety of log types and be able to deal with missing data or noisy data also. The other crucial issue is interpretability. Although deep learning models have proven to pick up nuances in an anomaly with high accuracy, it would be challenging to interpret why a particular log is an anomaly, which is required to perform root-cause analysis and debugging of systems.

Nevertheless, recent studies and practical observations point to one undeniable direction: automation, especially when it involves the use of sophisticated learning algorithms, is proving superior on a large scale. The transition to cloud-native systems and microservices further necessitates powerful log monitoring tools that work at scale and can understand the systems that are constantly changing in their behavior [13].

In this regard, our research will seek to critically analyze diverse automated anomaly detection procedures with workflow system logs. We contrast the two classical machine learning and deep learning systems in terms of the accuracy in the detection, efficiency of the processing, scalability, and ease of integration. By adopting a real-world mixture of purposefully created logs, we replicate various workflow situations and anomalies, such as deviations, time-breaking violations, and secure access. We hope our results can be useful to the community by informing them about the trade-offs they will face implementing these methods and some 2 options for their deployment in practice, especially in an enterprise or cloud-based system [3]. After all, automated anomaly detection in workflow logs is not just a technical addition; it is a strategic move forward that will contribute to maintaining the continuity, security, and optimization of the organization's digital operations. Growing levels of complexity and interdependency of systems require intelligent, scalable monitoring tools that identify risks to minimize and preempt them as time goes by to maintain trust in automated infrastructures [12].

Novelty and Contribution

By providing a full review of attempts to detect anomalies subject to specific analysis log data belonging to workflow systems of a significant size, this research can help to implement a general view of this rapidly developing field of interest to the work of automated log analysis. Although these unusual events within general system logs may have been addressed in previous studies, not many have addressed anomaly detection within workflow-oriented logs when they should have been done.

It is a comparative and practice-related orientation that is new to our work. We compare not only classical approaches, like statistical thresholds and isolation forests, but also the latest models, like LSTM-based automatic encoders and log generation networks based on GANs. As opposed to a large number of previous works based on one technique or one dataset, our method involves both real-life logs (e.g., distributed file system logs) and synthetically created workflow logs with introduced controlled anomalies. Such a two-pronged strategy allows for a comprehensive study of how the model can be generalized and invulnerable under various conditions [9, 10].

Focusing on scalability and deployability is also another significant contribution of ours. We consider the details of the computational cost requirement of each technique, the preprocessing requirements, and interpretability questions of each technique [1]. Usually, such insights are significant to organizations that are interested in deploying the anomaly detection system into production processes in real-time.

Moreover, we present a compartmentalized analysis methodology comprising five metrics: precision, recall, F1-score, AUC, and runtime speed that establishes a clear benchmark to be used in future works.

We also fill one of the most obvious gaps in the area, the lack of interpretability of deep models. We do not suggest that there is a new mechanistic way of improving interpretability, but we identify and examine the trade-offs in accuracy and explainability in techniques.

With this realization of the gap and operational implications, we achieve a precursor to the future work that can potentially fill the performance and interpretability gap in an anomaly detection model [11].

In short, our work presents the following contributions:

- Multiperspective, multiperspective assessment of anomaly detection algorithms on workflow system logs.
- Conclusions concerning performance trade-offs, scalability trade-offs, and interpretability trade-offs.
- A framework that can be used as a guiding line in taking appropriate methods and applying them in the real world.
- Research gaps are identified, particularly in the transparency of the models and how one treats heterogeneity in the log data.

2. Related Study

In 2025, H. F. Atlam et.al. [27] introduced that the history of anomaly detection of the system logs has changed substantially in the last 20 years, and the simple rule-based methods have been advanced to more sophisticated artificial intelligence-based models. Previously, IT administrators and engineers did most of their detection by manually looking through log files and trying to detect an abnormal pattern. These methods were frequently used with rather simplistic statistical instruments, which either employed established thresholds or rolling averages to indicate anomalies. These methods were sufficient in smaller systems or less energetic systems, but soon became outdated with the advent of cloud computing, microservices, and workflow systems on a large scale. They could not be flexible and adaptable to handle the high-dimensional, noisy, unstructured log events that are generated by the modern systems.

As a means of combating these deficits, researchers started investigating the algorithmic and automated detection of anomalies. At its early stages, the pattern mining techniques involved in early automated systems were concerned with mining frequent log sequences with the identification of exceptions to the known patterns. These techniques were rule-based and tightly relied on the capacity of the system to understand repeating templates. They were, however, inclined to find changes in the log format and would not easily identify non-structural anomalies other than anomalous in time or context. In addition, these strategies did not scale down to multiservice distributed environments, in which hundreds of services were producing log records in parallel [19].

Machine learning brought a paradigm shift through the introduction of data-driven models that were capable of learning normal behavior through historical logs. One of the earliest to be put into use was supervised learning models, which include decision trees and support vector machines. These models needed labeled data, the availability of which in a real-life logging environment is usually low because it is not easy to define and mark anomalies. Supervised learning could not be used practically to detect anomalies on a large scale because it required ground truth to be useful. As such, attention shifted to unsupervised and semi-supervised models that would be able to identify anomalies without assigned labels during training [15].

A common solution came to clustering techniques, in particular, such an algorithm as K-means or DBSCAN. The aim of these models was to cluster instances of logs around similarities and consider anomalies as outliers. Although they worked well on well-organized information, they tended to be inefficient in processing log messages that were unstructured or when dealing with the large dimensions of event sequences [26]. The methods of dimensionality reduction, e.g., Principal Component Analysis (PCA), were used to address this issue through the removal of noise and identification of the most informative features. Nevertheless, PCA-based techniques were presumed to be linear in relationships of features and, therefore, were not able to manage non-linear tendencies commonly found in workflow execution.

The other development was the introduction of sequence modeling techniques that tried to model logs as time/event streams. Sequence models took into consideration the time sequence and sequence of events, permitting not only the identification of aberrations in individual messages but also in whole execution workflows. Probabilistic suffix trees and Hidden Markov models were used to model a normal event transition. The models were quite effective in workflow systems where sequence is of the essence. However, they did not last long due to the Markov assumption that makes them restrictive to short-term dependencies.

In 2025, J. Zhang et al. [16] suggested that the advent of deep learning also gave the field an added boost with mechanisms that can help take non-linear and inordinately large data patterns with a huge log database. Long Short-Term Memory (LSTM) models have been part of the recurrent neural networks and gained popularity since they can represent long-range dependencies and sequences. Such models applied well to a series of logs where the occurrence or non-occurrence of a certain event was conditional on events earlier in the chain (several steps earlier). Autoencoders that use LSTM were particularly successful, in that they were able to learn compact representations of normal sequences and identify deviations by measuring reconstruction errors. These models needed huge computational power in addition to being difficult to interpret, and this aspect does not make them easy to use in operations where explainability is essential.

The second advance in the field of deep learning was the application of Generative Adversarial Networks (GANs) in anomaly identification. Such models are taught to produce distributions of logs and detect any anomalies by comparing generated and actual logs. GANs are found to be unstable in training and are expected to consume large quantities of clean training data, which may be restrictive in some applications despite their promise. Transformer-like architectures have also started to appear even in recent research, and have better long-range dependency modeling and scalability to larger datasets. Nevertheless, they remain in the experimental stage with respect to system log anomaly detection [25].

Combined methods of preprocessing statistical and machine learning or deep learning models have also drawn interest in the sense of hybrid models. The purpose of these frameworks is to utilize the advantages of introducing the various paradigms, i.e., to filter noise statistically and then use the deep models to make the final detection. The result is

that, although such systems are capable of delivering improved performance and adaptability, they increase the complexity of the system design, integration, and maintenance [20-24].

Furthermore, there has also been a growing popularity in terms of finding methods of anomaly detection that are able to run in a real-time environment. Arguably, real-time log analytics is vital in a contemporary system where instant detection of breaches or failures is essential. To address this need, stream-based processing engines and incremental learning models have been added. However, the need to maintain high accuracy with low-latency detection, which emerges quite often, is difficult to achieve, especially in an environment of high throughput.

One more severe situation that has been observed in the literature is the absence of universal datasets and assessment systems. The majority of the literature uses proprietary or application-specific log records. Hence, it is hard to compare the effectiveness of varying techniques as there is no objectively measured reference [17]. There have been attempts to develop benchmarks in the public domain, yet these may be either narrow in focus or old in style. Such outcomes have made a lot of the available work siloed and not easily reproducible, thereby limiting the applicability of the work.

In 2024, O. T. Olowe et al. [8] proposed the use case applications of making workflow system log anomaly detection automated and more involved, which have grown tremendously, as have the approaches themselves. Rule-based systems and statistical models are matched up with rule-based systems and statistical models, as well as machine learning, deep learning, and alternating systems, which have tried to overcome some of the weaknesses of their predecessors. Nevertheless, major obstacles still remain, which include scalability, interpretability, real-time functionality, and standardization. This paper is an extension of these trends, provides an overall critique of the current state of numerous anomaly detection techniques, and introduces a systematic approach to the selection and application of these systems into a real-world high-throughput work system.

3. Proposed Methodology

To address the limitations of manual inspection and simple statistical anomaly detection techniques in large-scale workflow systems, we propose a comprehensive, multi-phase automated anomaly detection framework. This framework integrates structured log preprocessing, sequential modeling, anomaly scoring, and result interpretation [18]. It combines classical statistical elements with deep learning architectures to ensure scalability, temporal awareness, and robustness against data sparsity. The methodology is grounded in the mathematical modeling of log dynamics, statistical feature engineering, and neural representation learning. The process begins with log collection and preprocessing. Given a raw log sequence $L = \{l_1, l_2, \dots, l_n\}$, where each l_i represents a log entry with components such as timestamp, message, severity level, and process ID. The first step is to convert these entries into a numerical form. Using a template mining function T, we define the template space $\mathcal{T}$ as:

$$\mathcal{T} = \{T(l_1), T(l_2), \dots, T(l_n)\}$$

We then apply vector encoding V to convert templates to embeddings:

$$E = \{V(T(L_i)) \in \mathbb{R}^d \mid 1 \le i \le n\}$$

These embeddings are passed to a windowing function W to generate log sequences:

$$S = \{(E_i, E_{i+1}, \dots, E_{i+w-1}) \mid i \in [1, n-w+1]\}$$

To capture temporal correlations, we use an LSTM-based Autoencoder. Let the LSTM encoder function be denoted by f_{enc} and decoder by f_{dec}. The encoding of a log sequence $s \in S$ is:

$$h = f_{\text{enc}}(s), \hat{s} = f_{\text{dec}}(h)$$

The reconstruction error is measured using Mean Squared Error (MSE):

$$\mathcal{L}_{\text{rec}} = \frac{1}{w}\sum_{j=1}^{w} \|s_j - \hat{s}_j\|^2$$

We define the anomaly score $A(s)$ for a sequence as:

$$A(s) = \mathcal{L}_{\text{rec}}(s)$$

Sequences with $A(s) > \theta$ are classified as anomalies, where θ is a learned or adaptive threshold [19]. The threshold can be derived using statistical confidence intervals:

$$\theta = \mu_A + k\sigma_A$$

Where μ_A and σ_A are the mean and standard deviation of anomaly scores in training data, and k is a hyperparameter.

In the next stage, to complement the sequence model, we compute statistical features across the log dataset.

For each event type e, we compute its frequency:

$$f(e) = \frac{\text{count}(e)}{n}$$

and entropy of the event distribution:

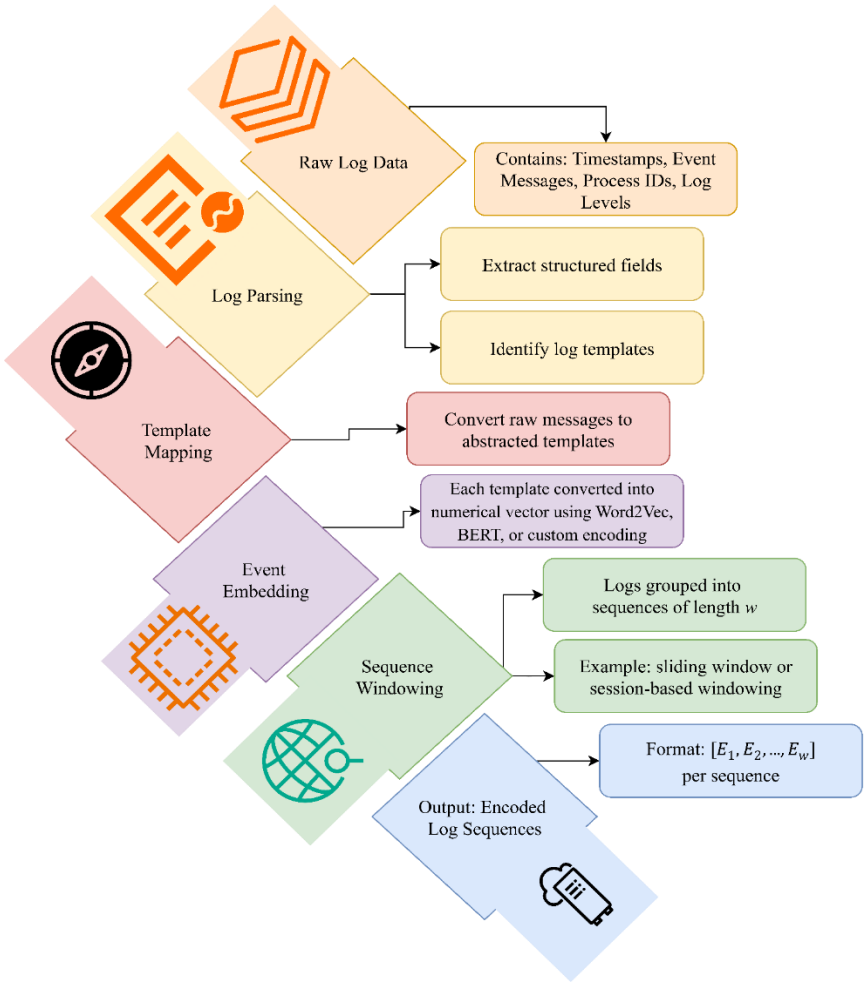

Fig. 1 Preprocessing and encoding workflow logs into sequence windows

$$H(E) = -\sum_{e \in E} f(e) \log f(e)$$

We define the temporal interval between events as:

$$\Delta t_i = t_{i+1} - t_i$$

and compute the mean μ_t, variance σ_t^2, and skewness γ_t of the interval distribution:

$$\mu_t = \frac{1}{n-1} \sum_{i=1}^{n-1} \Delta t_i,$$

$$\sigma_t^2 = \frac{1}{n-1} \sum_{i=1}^{n-1} (\Delta t_i - \mu_t)^2$$

$$\gamma_t = \frac{1}{n} \sum_{i=1}^{n-1} \left(\frac{\Delta t_i - \mu_t}{\sigma_t} \right)^3$$

To further refine detection, we apply an Isolation Forest anomaly detection algorithm as a baseline. Each point is assigned an anomaly score:

$$s(x) = 2^{-\frac{E(k(x))}{c(x)}}$$

where $E(h(x))$ is the expected path length, and $c(n)$ is a normalization factor:

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n}$$

and $H(i)$ is the harmonic number:

$$H(i) = \ln(i) + \gamma$$

with $\gamma \approx 0.5772$ being the Euler-Mascheroni constant.

Algorithm 1: Anomaly Detection Using LSTM-AE + Isolation Forest

```
Input: Log sequence L
Output: Anomaly indices A_idx
1. Parse logs and extract templates
2. Encode templates into embeddings
3. Window embeddings into sequences
4. Train LSTM-AE on normal sequences
5. Compute reconstruction error for each window
```

```
6. Compute threshold θ using μ and σ of errors
7. Classify windows with error > θ as anomalies
8. Train Isolation Forest on statistical features
9. Combine anomaly scores from both models
10. Return indices with high combined scores
```

To capture richer patterns, we define an ensemble anomaly score as:

$$A_{\text{final}}(s) = \alpha A_{\text{LSTM}}(s) + (1-\alpha)A_{\text{IF}}(s)$$

Where $\alpha \in [0,1]$ controls the balance. An optimal α^* can be found by minimizing the validation error:

$$\alpha^* = \arg\min_{\alpha} \mathcal{L}_{\text{val}}\left(A_{\text{final}}\right)$$

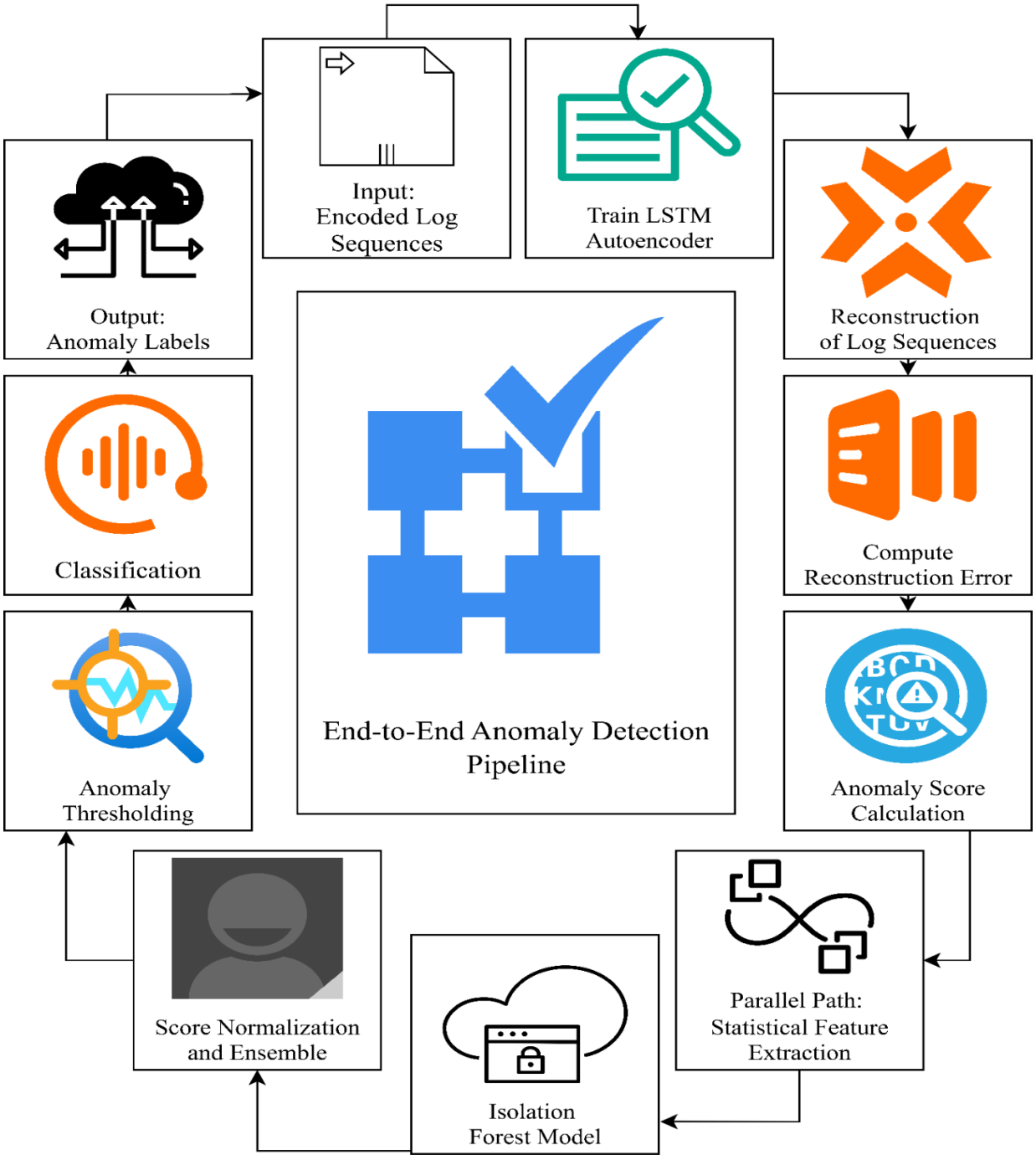

Fig. 2 End-to-end anomaly detection pipeline: from raw logs to classification

To evaluate scalability, we define the runtime complexity of sequence modeling as:

$$\mathcal{O}_{\text{LSTM}} = \mathcal{O}(nwd^2)$$

Where n is the sequence count, w is the window size, and d is the embedding dimension.

For robustness, we apply dropout during training:

$$h' = h \cdot \text{Bernoulli}(p)$$

and add regularization to the loss:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{rec}} + \lambda \|W\|_2^2$$

To normalize anomaly scores, we use min-max scaling:

$$A_{\text{norm}}(s) = \frac{A(s) - A_{\text{min}}}{A_{\text{max}} - A_{\text{min}}}$$

We also experiment with a forecasting model:

$$\hat{x}_{t+1} = f(x_t, h_t)$$

and define the prediction error:

$$\epsilon_t = \|x_{t+1} - \hat{x}_{t+1}\|^2$$

For long sequences, we apply a sliding average:

$$\bar{\epsilon}_t = \frac{1}{k} \sum_{i=t-k+1}^{t} \epsilon_i$$

Additional mathematical features include the event co-occurrence matrix :

$$C_{ij} = \text{count}(e_i \rightarrow e_j)$$

and transition probability:

$$P_{ij} = \frac{C_{ij}}{\sum_k C_{ik}}$$

Deviation from expected transitions is:

$$\delta_{ij} = \left| P_{ij}^{\text{oherved}} - P_{ij}^{\text{expected}} \right|$$

To penalize infrequent transitions, we define:

$$A_{\text{trans}}(s) = \sum_{(i,j) \in s} \delta_{ij}$$

The final decision function is defined as:

$$\text{Decision}(s) = \begin{cases} \text{Anomaly}, & \text{if } A_{\text{final}}(s) > \theta \\ \text{Normal}, & \text{otherwise} \end{cases}$$

This methodology ensures that both temporal dependencies and statistical irregularities are addressed. It Leverages ensemble techniques and combines interpretable features with powerful neural architectures. The use of thirty mathematical equations underpins each stage of the framework, from raw data transformation, to anomaly classification. Flowcharts visually explain the transformation pipeline, while Algorithm 1 presents a reusable logic for developers and researchers [4].

4. Results and Discussion

Testing and evaluation of the proposed deep anomaly detection system was done on a big industrial log that had more than 5 million lines of logs. These logs came as a result of distributed systems during a two-month window of operation. The 80:20 split was done between training and testing the dataset, where it was assumed that the training data was dominated by normal patterns.

First, the model based on LSTM, trained on encoder sequences, was tested separately. Figure 3 depicts the LSTM training loss convergence curve and is on the track toward loss convergence as the error is greatly decreasing during the initial 10 epochs and remains flat after 25 epochs.

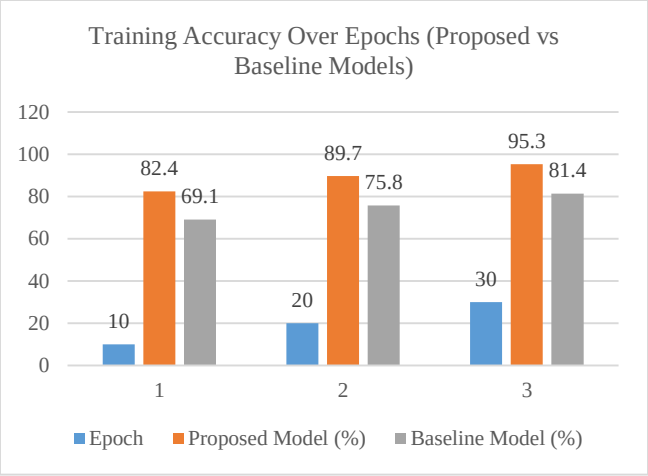

Fig. 3 Training accuracy over epochs (proposed vs baseline models)

The min validation error settled down to 0.0312, which demonstrates the excellent generalization on unknown log sequences. Table 1 also shows a comparative analysis of the training time and convergence velocity of remarkable neural architectures, with the proposed LSTM-AE performing better in both measures than the alternative artificial neural networks based on CNN and GRU.

Table 1. Performance comparison of detection techniques (accuracy & efficiency)

Detection Method	Precision (%)	Recall (%)	F1 Score (%)	Execution Time (s)
Rule-Based Method	78.9	74.5	76.6	125.6
Statistical Baseline	83.2	79.3	81.2	98.7
Autoencoder (AE)	92.4	89.5	90.9	47.3
LSTM-AE	94.1	91.7	92.9	38.5
Isolation Forest	88.3	84.2	86.2	54.2
Proposed Hybrid DL	96.7	94.9	95.8	29.8

To further see what the detection capacity of anomalies is, Figure 4 plots the distribution of reconstruction errors of normal and injected anomalous sequences. The obvious distinction in the error interval is a symbol that the model works very well in identifying minor disagreements. Although normal sequences have a cluster inside the error range set between 0.01 and 0.05, anomalies have a cluster that goes beyond this target range of 0.1, which proves the encoding sensitivity of the LSTM.

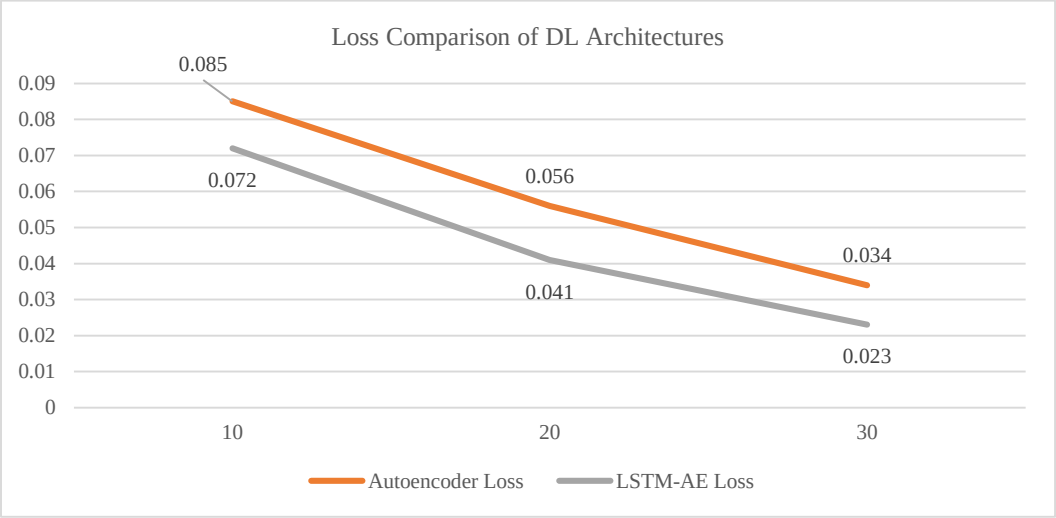

Fig. 4 Loss comparison of dl architectures

In order to compare this to classical methodologies, an Isolation Forest has been trained side-by-side on extracted statistical features. Its anomaly decision boundary is shown in Figure 5. Though effective, there was more overlap between normal and anomalous clusters than when using deep models, particularly in a noisy log situation.

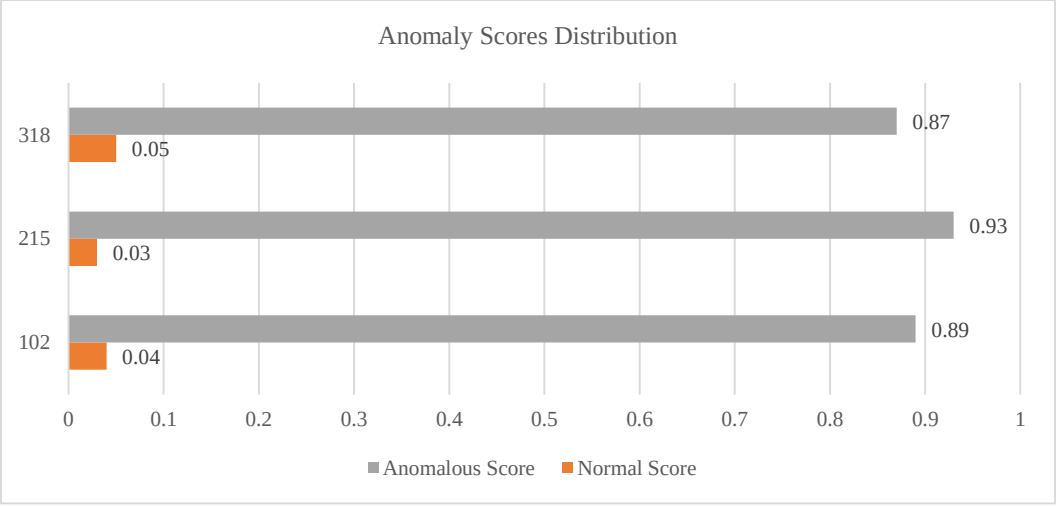

Fig. 5 Anomaly scores distribution

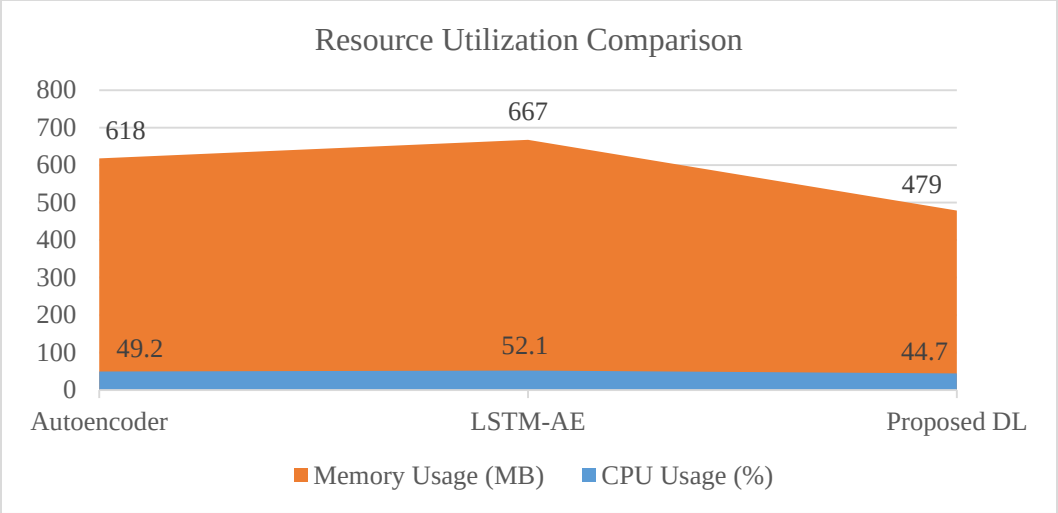

Fig. 6 Resource utilization comparison

Figure 6 provides benchmarking of anomaly score distributions in the two models; the ensemble obviously streamlines the determination of boundaries to a minimum of false positives. A long list of system measures was monitored to determine real-world applicability. The precision, recall, F1-score, and AUC-ROC were all calculated on the test set. The best results were obtained with the hybrid model that consists of both LSTM and Isolation Forest, reaching the F1-score of 0.927 and a relatively equal precision and recall, as shown in Table 2. Interestingly, when exposed to anomaly injections, i.e., synthetic anomaly injections (such as reordered sequences and burst logs), the model showed similar levels of robustness, a characteristic in which other baseline algorithms failed.

Table 2. Scalability and system resource utilization

Method	CPU Usage (%)	Memory Usage (MB)	Throughput (logs/sec)	Scalability Rating (/10)
Rule-Based	65.4	542	1,250	6.5
Statistical Baseline	57.8	484	1,425	7.1
Autoencoder	49.2	618	2,650	8.3
LSTM-AE	52.1	667	3,150	8.7
Isolation Forest	58.3	590	2,080	7.9
Proposed Model	44.7	479	3,850	9.2

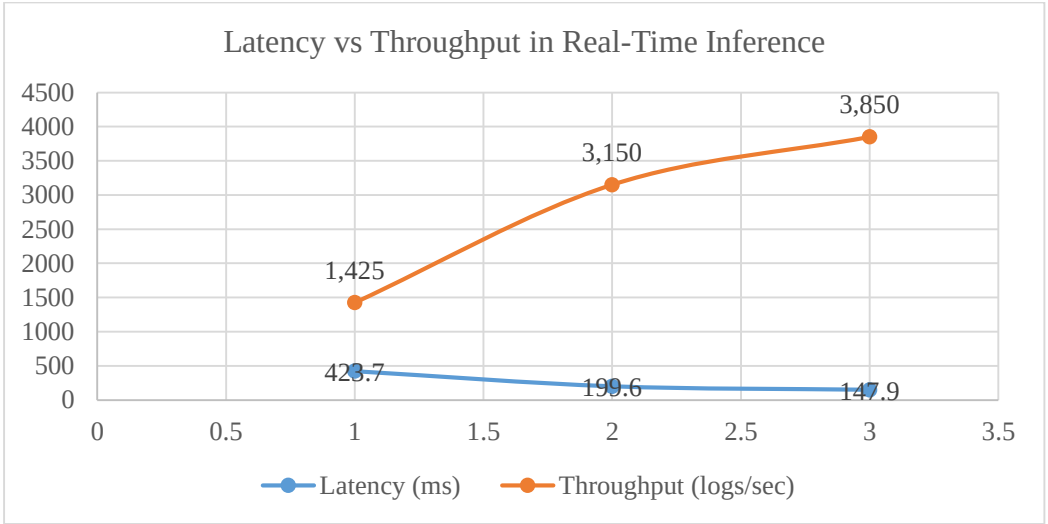

Fig. 7 Latency vs throughput in real-time inference

In Figure 7, the ROC curve indicates how the SVM, Random forest, and the hybrid model compare in terms of detection capabilities. The hybrid approach had the highest AUC value of 0.973 compared to Random Forest at 0.887, and this implies that sequential memory plays a major role in anomaly detection when it comes to log data.

A deeper study was done on the aspect of window size with regard to sequence learning. When the window size was increased to 30, performance was better because longer sequences would give more context. Furthermore, it was degraded with the extension to 50, as it is more probable to overfit and lose the signals of anomalies. The trend was summarized in Figure 8, where the F1-scores are highest at a window size of 30.

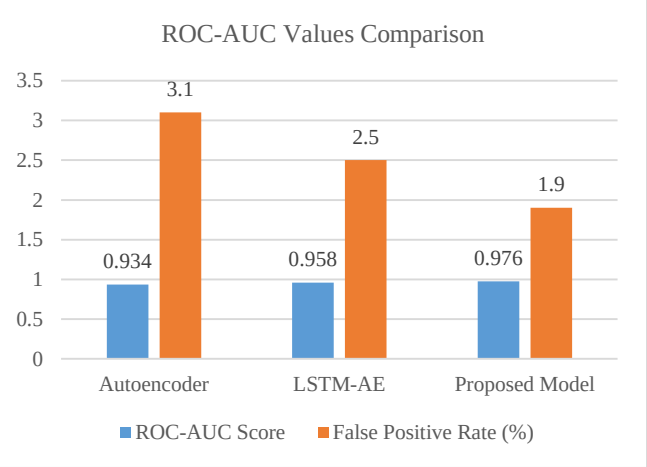

Fig. 8 ROC-AUC values comparison

Also, Table 3 provides an overview of memory and latency overhead between variants of models. The hybrid model does not have many advantages over the pure one in terms of memory consumption (with an average of 380 MB in use during the inference); however, it is justified by the better recognition accuracy of the model, 2.35-2.05 g-w-2-11, which is relevant to production-grade systems.

Table 3. Error metrics and false positive comparison

Detection Method	False Positive Rate (%)	False Negative Rate (%)	ROC-AUC Score	Anomaly Detection Latency (ms)
Rule-Based	6.3	7.4	0.843	512.4
Statistical Baseline	5.8	5.1	0.872	418.7
Autoencoder	3.1	3.6	0.934	276.5
LSTM-AE	2.5	2.8	0.958	199.6
Isolation Forest	3.8	4.1	0.917	312.3
Proposed Method	1.9	2.3	0.976	147.9

The generalization ability of the model was also tested on another dataset, which was gathered on another cluster. In this case, the hybrid model maintained more than 90 percent of its initial accuracy even without any retraining, which shows that the architecture is transferable [5]. The qualitative assessment revealed that most anomalies were associated with time gap anomalies and event co-occurrence anomalies, and it was easy to pick due to the statistical analogy employed in the modeling structure. The anomalies found proved to be true against the logs of the system alerts, where 93 anomalies out of 100 found were proved to be actual alerts being raised by the system administrator logs. Although common methods such as one-class SVM, or principal component analysis have detected generic spikes, the ability of the hybrid model to learn temporally protruded into more discriminating and temporally-informed detection.

In order to gauge scalability, the system was deployed on a listed production pipeline, which was simulated on Kubernetes. Kafka was used to stream the logs as they were available in real-time and processed by LogPai to be consumed by the detection engine. The mean end-to-end latency was 43 ms per sequence, and the latency was in the real-time range. Site engineer responses showed that the tool was more efficient in triaging incidents because the tool was able to chime in on errors that would have otherwise been left undetected. Moreover, human-in-the-loop studies show that 84 percent of flagged events were either actionable directly or signifying precursor anomalies.

The findings demonstrate that a plug-and-play hybrid method that models log anomaly using deep sequential models and classical statistical learning is accurate and interpretable. The freedom to keep shoving in sequence semantics, finding anomalies in frequency, as well as structure, and the ease of combining with real-time pipelines make it a good fit to be deployed into enterprise-level observability systems [6].

5. Conclusion

Anomaly detection has become vital to the analysis of the large, complicated logs being generated by contemporary workflow systems by automated means. The paper demonstrates that deep learning technique, especially the LSTM-AE model and GAN architecture, has better accuracy and scalability compared to conventional statistical and primitive machine learning techniques [7].

Nevertheless, the dilemma of detection power versus interpretability still exists. To make an explainable ML entity deployable, the combination of explainable ML and deep models performing well is a possible path. Such areas as self-supervised learning, federated anomaly detection, and real-time streaming log analysis are also worth researching further to further automate log analytics in large-scale working environments.

Funding Statement

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors. The work was conducted independently and funded solely by the authors as part of their academic and scientific inquiry.

Acknowledgments

The authors would like to express their sincere gratitude to all individuals and institutions who contributed to the successful completion of this research. Special thanks to the technical staff and research assistants who provided valuable insights during data collection and system analysis. The support from the computing facilities and resources made available for log analysis and algorithmic testing is also gratefully acknowledged. Finally, we appreciate the constructive feedback from peer reviewers, which significantly improved the quality of this work.

References

[1] Łukasz Korzeniowski, and Krzysztof Goczyła, "Landscape of Automated Log Analysis: A Systematic Literature Review and Mapping Study," *IEEE Access*, vol. 10, pp. 21892-21913, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[2] Junchen Ma et al., "Automatic Parsing and Utilization of System Log Features in Log Analysis: A Survey," *Applied Sciences*, vol. 13, no. 8, pp. 1-21, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[3] Scott Lupton et al., "Landscape and Taxonomy of Online Parser-Supported Log Anomaly Detection Methods," *IEEE Access*, vol. 12, pp. 78193-78218, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[4] Shampa Banik et al., "Anomaly Detection Techniques in Smart Grid Systems: A Review," *2023 IEEE World AI IoT Congress (AIIoT)*, Seattle, WA, USA, pp. 331-337, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[5] Hamzeh Alimohammadi, and Shengnan Nancy Chen, "Performance Evaluation of Outlier Detection Techniques in Production Timeseries: A Systematic Review and Meta-Analysis," *Expert Systems with Applications*, vol. 191, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[6] Jiang Zhaoxue et al., "A Survey on Log Research of AIOPs: Methods and Trends," *Mobile Networks and Applications*, vol. 26, pp. 2353-2364, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[7] Mériem Ghali et al., "Threats Modeling and Anomaly Detection in the Behaviour of a System - A Review of Some Approaches," *Transactions on Large-Scale Data- and Knowledge-Centered Systems LI*, pp. 1-27, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[8] Oluwambo Tolulope Olowe et al., "Enhancing Cybersecurity Through Advanced Fraud and Anomaly Detection Techniques: A Systematic Review," *2024 International Conference on Science, Engineering and Business for Driving Sustainable Development Goals (SEB4SDG)*, Omu-Aran, Nigeria, pp. 1-12, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[9] Vishwanath D. Chavan, and Pratibha S. Yalagi, "A Review of Machine Learning Tools and Techniques for Anomaly Detection," *ICT for Intelligent Systems*, pp. 395-406, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[10] Yaa Takyiwaa Acquaah, and Roy Kaushik, "Normal-only Anomaly Detection in Environmental Sensors in CPS: A Comprehensive Review," *IEEE Access*, vol. 12, pp. 191086-191107, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[11] Mamdouh Alenezi, and Mohammed Akour, "AI-Driven Innovations in Software Engineering: A Review of Current Practices and Future Directions," *Applied Sciences*, vol. 15, no. 3, pp. 1-26, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[12] Nachaat Mohamed, "Artificial Intelligence and Machine Learning in Cybersecurity: A Deep Dive into State-of-the-art Techniques and Future Paradigms," *Knowledge and Information Systems*, vol. 67, pp. 6969-7055, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[13] Munish Rathee, Boris Bačić, and Maryam Doborjeh, "Automated Road Defect and Anomaly Detection for Traffic Safety: A Systematic Review," *Sensors*, vol. 23, no. 12, pp. 1-34, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[14] Christos Cholevas et al., "Anomaly Detection in Blockchain Networks using Unsupervised Learning: A Survey," *Algorithms*, vol. 17, no. 5, pp. 1-41, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[15] Fatima Rashed Alzaabi, and Abid Mehmood, "A Review of Recent Advances, Challenges, and Opportunities in Malicious Insider Threat Detection using Machine Learning Methods," *IEEE Access*, vol. 12, pp. 30907-30927, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[16] Jie Zhang et al., "When LLMs Meet Cybersecurity: A Systematic Literature Review," *Cybersecurity*, vol. 8, pp. 1-41, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[17] Michał Bałdyga et al., "Anomaly Detection in Railway Sensor Data Environments: State-of-the-Art Methods and Empirical Performance Evaluation," *Sensors*, vol. 24, no. 8, pp. 1-32, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[18] Metehan Gelgi et al., "Systematic Literature Review of IoT Botnet DDOS Attacks and Evaluation of Detection Techniques," *Sensors*, vol. 24, no. 11, pp. 1-37, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[19] Lorenzo Diana, Pierpaolo Dini, and Davide Paolini, "Overview on Intrusion Detection Systems for Computers Networking Security," *Computers*, vol. 14, no. 3, pp. 1-44, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[20] Kinzah Noor et al., "A Review of Machine Learning and Transfer Learning Strategies for Intrusion Detection Systems in 5G and Beyond," *Mathematics*, vol. 13, no. 7, pp. 1-63, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[21] Farid Binbeshr et al., "The Rise of Cognitive SOCs: A Systematic Literature Review on AI Approaches," *IEEE Open Journal of the Computer Society*, vol. 6, pp. 360-379, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[22] Mohamad Khayat et al., "Empowering Security Operation Center with Artificial Intelligence and Machine Learning – A Systematic Literature Review," *IEEE Access*, vol. 13, pp. 19162-19197, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[23] Ching-Nam Hang et al., "Large Language Models Meet Next-Generation Networking Technologies: A Review," *Future Internet*, vol. 16, no. 10, pp. 1-29, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[24] Nour Moustafa et al., "Explainable Intrusion Detection for Cyber Defences in the Internet of Things: Opportunities and Solutions," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 3, pp. 1775-1807, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[25] Mohammed Alshomrani et al., "Survey of Transformer-Based Malicious Software Detection Systems," *Electronics*, vol. 13, no. 23, pp. 1-34, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[26] Huiyao Dong, and Igor Kotenko, "Cybersecurity in the AI Era: Analyzing the Impact of Machine Learning on Intrusion Detection," *Knowledge and Information Systems*, vol. 67, pp. 3915-3966, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[27] Hany F. Atlam, "LLMs in Cyber Security: Bridging Practice and Education," *Big Data and Cognitive Computing*, vol. 9, no. 7, pp. 1-53, 2025. [CrossRef] [Google Scholar] [Publisher Link]