ISSN (e): 2250-3005 || Volume, 16 || Issue, 11|| November – 2026 ||
International Journal of Computational Engineering Research (IJCER)

Artificial Intelligence for Log Mining-Based Anomaly and Threat Detection: A Systematic Review

Mr. Deepak C Dhanwani[a], Dr. Rajeshri R. Shelke[b]
[a.] *Department of Artificial Intelligence and Data Science, Sipna College of Engineering and Technology Amravati,*
[b.] *Department of Computer Science and Engineering, HVPM College of Engineering and Technology Amravati.*
[a,b] *Sant Gadge Baba Amravati University, Amravati*

Abstract: *Computer systems produce a large number of logs every day. Cyberattacks often leave some signs in these logs. Traditional tools use fixed rules to find attacks, but they may miss new attacks and cannot easily handle large amounts of data. AI uses a different approach. It learns what normal log activity looks like and detects unusual activity. This paper discusses five main AI approaches. Machine learning methods such as Decision Trees and XGBoost are fast and easy to understand. Deep learning methods such as LSTM and Autoencoders learn patterns from log sequences. Transformers and LLMs can understand complex log patterns. Graph Neural Networks connect related events to detect multi-step attacks. Explainable and Federated Learning help make AI results easier to understand and support secure learning.The paper explains the strengths and limitations of these methods and compares them based on accuracy, false alarms, speed, cost, and understandability. It also discusses common problems, such as limited attack data and changing log formats, and suggests areas for future research.*

Keywords: *Log Analysis; Anomaly Detection; Artificial Intelligence; Machine Learning; Deep Learning; Cybersecurity; Intrusion Detection; Explainable AI; Log Mining*

Date of Submission: 01-11-2026 Date of acceptance: 10-11-2026

I. Introduction

Cloud platforms, microservices, and data centers produce huge streams of logs that record logins, process activity, network connections, and errors. Every stage of a cyberattack leaves some trace in this data, which makes logs one of the richest sources of evidence for security monitoring. A mid-sized distributed application can generate millions of log lines a day.The problem is not a shortage of evidence but the difficulty of finding it. Over 99% of log traffic reflects routine activity, and attackers often spread their actions across long time windows so each step blends into normal behavior. For about two decades, log-based threat detection relied on rule-based correlation inside Security Information and Event Management (SIEM) platforms, where analysts write rules that match known attack signatures. This approach works for previously catalogued threats, but surveys of machine learning for intrusion detection point to its main weaknesses: it reacts only after a signature exists, it needs constant manual updates as systems change, and it produces many false alarms [37]. Sommer and Paxson argued that the cost of investigating false positives, the shortage of labeled attack data, and the difficulty of realistic evaluation have kept machine learning out of most operational intrusion detection systems [38]. As the rest of this paper shows, these concerns still apply to the field today.

Artificial Intelligence takes a different approach: instead of encoding what an attack looks like, a model learns what normal behavior looks like from historical logs and flags anything that deviates from it. This makes it possible to detect attacks that have not been seen before, adapt automatically as systems change, and analyze logs at a scale no analyst could match by hand. Over the past decade, this idea has driven work in classical machine learning, deep sequence models, Transformer and Large Language Model architectures, graph-based methods, and privacy-preserving federated learning.AI uses a different approach. Instead of looking only for known attacks, it learns normal behavior from past logs and identifies unusual activity. This helps detect new attacks, adapt to system changes, and handle large amounts of log data. Recent studies use machine learning, deep learning, Transformers, Large Language Models, graph-based methods, and federated learning.This review focuses on log-based attack detection rather than network-traffic detection. Related studies on IoT and DDoS detection can be found in [30], [31], [32], [34].The review covers the main steps of log analysis: collection,

parsing, detection, explanation, and response. It also compares different techniques based on accuracy, false positives, scalability, cost, adaptability, interpretability, and real-time performance.

II. Log Data and Preprocessing

2.1 What Logs Capture and Where Attacks Appear

A log is a timestamped record, usually with a severity level, a source component, and a short message, that a system writes automatically as it runs. Different sources reveal different things: operating-system logs (Windows Event Log, Linux Syslog) show privilege escalation and malware execution; application logs show web attacks such as SQL injection; network and firewall logs show intrusion attempts; cloud audit logs (AWS, Azure) show misconfigurations and compromised credentials; and distributed-system logs (HDFS, OpenStack, Spark) are widely used as research benchmarks for large-scale anomaly and failure detection.

A multi-stage attack leaves different evidence at each point in its lifecycle: repeated failed logins during initial access, unusual process creation during execution, new scheduled tasks during persistence, unexpected permission changes during privilege escalation, and unusually large outbound transfers during data exfiltration. Recognizing these patterns across many log sources, rather than in any single line, is what makes log-based detection both useful and difficult.

2.2 The Log Mining Pipeline

Most AI-based log mining systems follow the same overall pipeline, shown in Figure 1. Logs are collected from many sources, aggregated and stored or streamed through a message queue, parsed to separate the fixed part of a message from its variable details, converted into numeric features or learned embeddings, and used to train a model that recognizes normal or normal-versus-abnormal patterns. The output is a score; high-scoring anomalies are sent to analysts through a SIEM or SOAR platform, sometimes triggering an automated response.

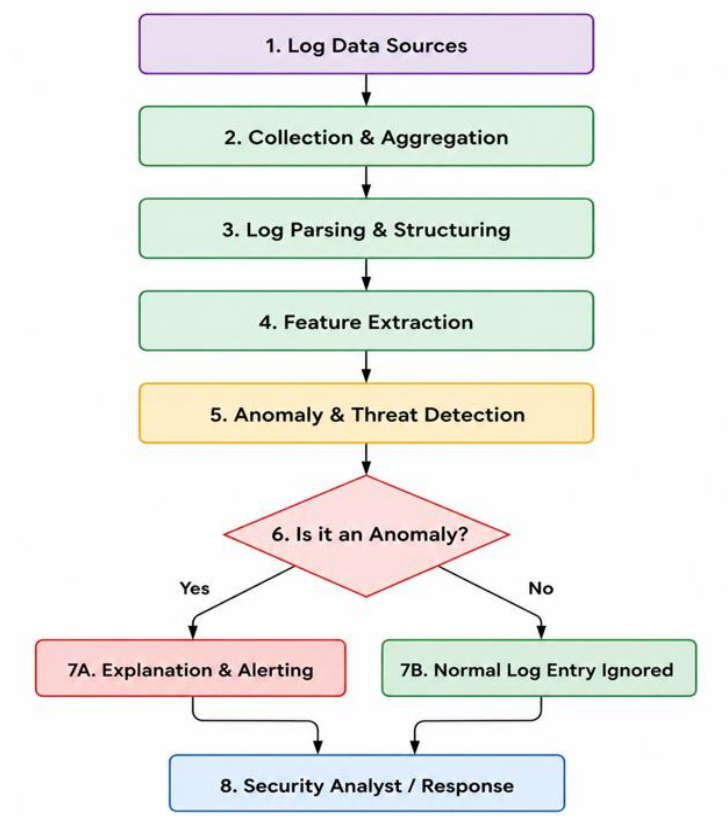


Figure 1. End-to-end AI-based log mining pipeline, from log collection to analyst response.

2.3 Log Parsing: From Raw Text to Structured Events

Raw log messages are free text, so nearly every downstream detector depends on parsing to turn them into structured event templates. Xu et al. mined console logs with Principal Component Analysis to detect large-scale system problems [29], building on Oliner and Stearley's earlier statistical study of production supercomputer logs, which described how failures and anomalies are distributed across large-scale log data and helped motivate later benchmark datasets [28]. Makanju et al. proposed IPLoM, an iterative-partitioning method for clustering log events [50], and Du and Li proposed Spell, a streaming parser based on longest-common-

subsequence matching [49]. He et al. proposed Drain, an online, fixed-depth tree parser that is still one of the most widely used tools in later deep-learning pipelines [11], and separately published an evaluation of how different parsers affect downstream log mining [51]. Earlier work by Bertero et al. showed that natural-language-processing methods, treating log messages as short pieces of text, can support anomaly detection without a dedicated structural parser [59].

More recent work measures how parsing quality affects deep-learning-based detectors directly. Khan et al. found that parsing errors reduce detection accuracy [24], and Le and Zhang's NeuralLog showed that removing the parsing step altogether, and working directly on semantically embedded raw text, can improve detection instead of hurting it [8]. Together, these results show that parsing is a design choice with measurable consequences, not a solved preliminary step.

2.4 Benchmark Datasets

Table 1 lists the datasets used most often in this literature. BGL, one of the most common benchmarks, comes from an early study that used IBM BlueGene/L supercomputer event logs to predict hardware failures [52]. A limitation that runs through the field, discussed later under Research Gaps, is that HDFS, BGL, Thunderbird, and OpenStack were built to study system failures and reliability, not adversarial security threats [28]. Loghub and Loghub 2.0 improve reproducibility by collecting many systems' logs into one benchmark set, but they do not close this security-labeling gap [12].

Dataset	Type	Approx. Scale	Security-Labeled?	Best Use
HDFS	Distributed file-system logs	~11 million lines	No (failure-labeled)	Sequence-model benchmarking
BGL	Supercomputer logs	~4.7 million lines	No (alert-labeled)	Stability testing
Thunderbird	Supercomputer logs	~211 million lines	No	Scalability testing
OpenStack	Cloud platform logs	~1.3 million lines	Injected faults only	Cloud anomaly research
Loghub / Loghub 2.0 [12]	Aggregated multi-system collection	19+ datasets	Varies	Reproducible cross-study benchmarking
AWS CloudTrail	Cloud API audit logs	User-defined	User-defined	Cloud intrusion / misconfiguration detection
Windows Event Log / Linux Syslog	OS-level logs	Varies	Partial	Endpoint / host intrusion detection

Table 1. Public log and security datasets used across the reviewed literature.

The tools used to build these pipelines in practice include ingestion and visualization platforms such as the ELK stack, SIEM and HIDS tools such as Splunk and Wazuh, stream-processing systems such as Kafka and Spark, and machine learning libraries such as scikit-learn for classical baselines and TensorFlow or PyTorch for deep architectures. The AI techniques covered next are usually added on top of this existing infrastructure rather than replacing it.

III. AI Detection Techniques

This part goes through the literature by technique family, describing for each one the working idea, strengths, limitations, and typical applications. Figure 2 summarizes the resulting taxonomy and its rough order of appearance over time.

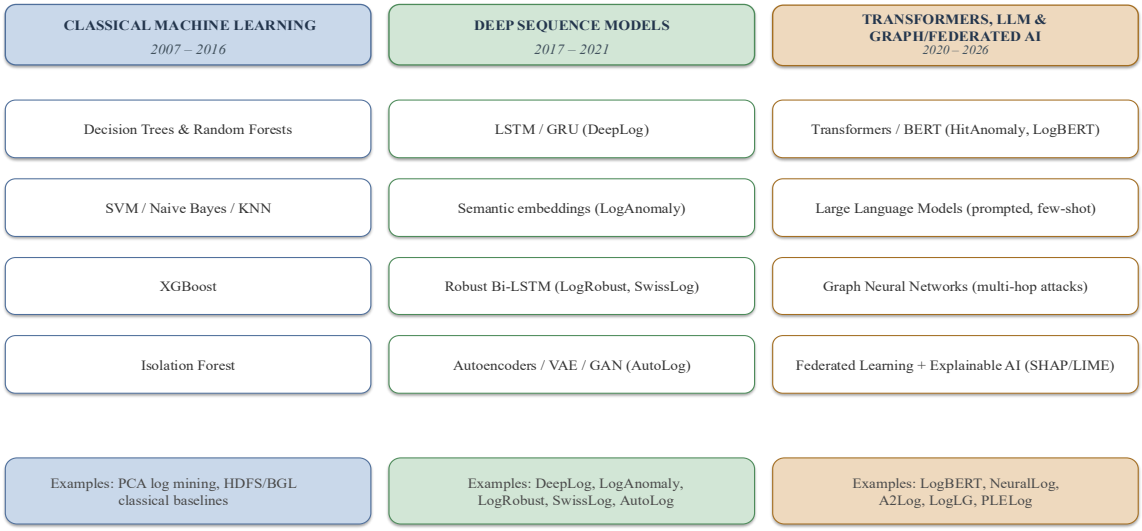


Figure 2. Taxonomy and chronological development of AI techniques applied to log mining.

3.1 Classical Machine Learning

Decision Trees and Random Forests split data using a sequence of feature-value questions, and Random Forests combine many trees to reduce overfitting. Both remain useful for tabular, log-derived features because they train quickly, need only modest amounts of data, and show which features drove a decision, though they perform worse on raw sequential or unstructured text. Support Vector Machines and Naive Bayes give lightweight baselines, though Naive Bayes' assumption that features are independent is often violated by correlated log events. XGBoost-style gradient boosting is consistently reported as one of the strongest classical baselines on structured log features [44], and Isolation Forest, which isolates anomalies because they are rare and different from normal points [43], is a common unsupervised choice when labeled anomalies are not available. He et al. set standard classical baselines (Decision Tree, Logistic Regression, SVM, clustering) on HDFS and BGL [10], and the broader anomaly-detection survey by Chandola et al. places these methods within a wider set of classification-, clustering-, and statistics-based approaches [27]. Ali et al. compared classical methods against deep architectures and found that well-tuned classical baselines stay competitive on several public benchmarks [13], which suggests that method choice should depend on the dataset and deployment setting rather than an assumption that deep learning always wins. Chen et al.'s comparative study reached a similar conclusion and pointed to a gap between results reported in research and what holds up in industrial practice [9].

3.2 Deep Sequence Models

Recurrent Neural Networks process sequences step by step, carrying forward a memory of earlier inputs, which fits logs well because they are ordered in time. Long Short-Term Memory (LSTM) networks [45] and Gated Recurrent Units improve on plain RNNs with gating mechanisms that keep relevant information over longer sequences, which is why they became the standard choice in early deep-learning log anomaly detection. Du et al.'s DeepLog was the first widely used method to treat log-key sequences as a language-modeling problem: an LSTM learns normal execution patterns and flags low-probability next events as anomalies, with a workflow-reconstruction step added for root-cause diagnosis [1]. Meng et al.'s LogAnomaly added template2vec semantic embeddings to capture both sequential and quantitative anomalies and to generalize across large template vocabularies [2]. Because software updates constantly change log templates, Zhang et al.'s LogRobust used attention-based Bi-LSTM modeling built specifically to handle unstable, changing templates [16], and Li et al.'s SwissLog combined semantic and time-interval embeddings within an attention Bi-LSTM to detect different fault types under changing conditions [17].

Autoencoders learn to compress and reconstruct their input; a log sequence that the model cannot reconstruct well is flagged as unusual. Catillo et al.'s AutoLog applied this idea to system logs without needing labeled data [20], and Variational Autoencoders and GANs [46], [47] extend the same idea to learn a generative model of normal behavior. Han and Yuan looked at a related practical problem, a model trained on one system needing to work on another, through unsupervised domain adaptation for cross-system anomaly detection [21].

A methodological warning comes from Le and Zhang's evaluation study, which found that reported deep-learning performance drops sharply once realistic chronological or cross-system data splits replace the random splits used in much earlier work [7]. This matches the broader survey of sequence-based anomaly detection by Chandola et al. [57] and stands out as one of the more important findings in the field, since it means many earlier results looked better on paper than they would in real deployment. Landauer et al.'s wider survey of deep learning for log-based anomaly detection reaches a similar conclusion, noting that gains measured on benchmark logs tend to shrink once models face noisier, less curated production data [14].

3.3 Transformers and Large Language Models

The Transformer architecture [41] processes a whole sequence at once through self-attention, weighing how relevant each part of the sequence is to every other part, instead of moving through it step by step as an RNN does. Huang et al.'s HitAnomaly uses hierarchical Transformer encoders to model log templates and their parameter values together [3], while Nedelkoski et al.'s Logsy applies self-attention directly to raw tokenized logs, removing the need for a separate parser [5]. BERT [42] is pretrained on large text corpora with masked-token prediction, and Guo et al.'s LogBERT adapts this with masked log-key prediction, which allows training without labeled anomalies [4]. Ott et al. found that detectors built on pretrained language models carry over better across different log datasets than models trained from scratch on a single dataset [53], and Huang et al. later showed that pretraining hierarchical Transformers on large amounts of unlabeled log data before fine-tuning improves accuracy further [22], following the same pretrain-then-fine-tune pattern used in natural language processing. Wittkopp et al.'s A2Log combines unsupervised candidate selection with attentive classification in a two-step pipeline [19].

Large Language Models are trained on even larger text corpora and can be applied with little or no task-specific training, and they can produce natural-language explanations for their decisions. Liu et al. tested prompting LLMs directly for interpretable, in-context log analysis [25], and Xu and Ding's survey describes both the strong zero- and few-shot generalization of LLMs for anomaly and out-of-distribution detection and their sensitivity to prompt design and inference cost [26].

3.4 Graph Neural Networks

Graph Neural Networks (GNNs) represent log events, hosts, or processes as nodes and learn from the connections between them, building on earlier work that established how well such networks can tell graph structures apart [58]. This lets GNNs pick up multi-hop attack patterns, such as an attacker moving from one host to another, that models reading a single log stream cannot see. Li et al. proposed a graph-level GNN method that produces an explanation for each anomaly it flags [18], and Guo et al.'s LogLG deals with the shortage of labeled data directly by building a log-event graph and using weak supervision to approach fully supervised accuracy with far fewer labels [54]. A related semi-supervised method, Yang et al.'s PLELog, uses probabilistic label estimation with an attention-based GRU to reduce the need for labeled data without building a graph at all [6]. Bilot et al.'s survey places these efforts within a wider set of GNN applications to intrusion detection and points to scalability and explainability as the main open problems for this family [15].

3.5 Explainability and Federated Learning

SHAP [40] and LIME [48] were built for general machine learning, but they are now used inside log-anomaly pipelines and explainable intrusion detection system (X-IDS) designs to trace a flagged anomaly back to specific input features. Neupane et al.'s survey covers this area and proposes a general X-IDS architecture [36]. Federated Averaging, the original federated-learning algorithm [39], underlies recent proposals for training intrusion-detection models across organizations without sharing raw log data, though Nguyen et al.'s survey lists uneven data distributions across organizations and the risk of malicious participants as open problems [35].

3.6 Recent Developments (2023-2026)

Recent work moves in two directions at once: better semantic accuracy through larger pretrained models and improved log representations, and more practical deployment through parser-free pipelines, prompt-based LLM analysis, and large-scale real-time testing. Wu et al. found that the choice of log representation affects downstream accuracy about as much as the choice of model [23]. Aziz and Munir compared deep-learning approaches for log anomaly detection and looked at how parsing choices interact with accuracy [55], and Mathe et al. tested several anomaly-detection algorithms specifically for real-time, large-scale log analysis, confirming that the accuracy-speed trade-off described throughout this paper remains unresolved [56].

IV. Comparative Analysis

Table 2 compares the technique families from the previous part against the measures that matter for a deployment decision: typical accuracy or F1 on public benchmarks, false-positive rate, scalability to high-volume streams, computational cost, adaptability to log-format drift, interpretability, and real-time capability.

Technique Family	Accuracy / F1	False-Positive Rate	Scalability	Computational Cost	Adaptability	Interpretability	Real-Time
Decision Tree / Random Forest	Moderate-High	Moderate	High	Low	Low-Moderate	High	High
SVM / Naive Bayes / KNN	Moderate	Moderate-High	Moderate	Low	Low	Moderate-High	Low-Moderate
XGBoost	High	Low	High	Low	Low-Moderate	High	Moderate
Isolation Forest (unsup.)	Moderate-High	Low	High	Low	Moderate	High	Moderate
CNN / RNN	High	Moderate	Moderate	Moderate	Low-Moderate	Moderate	Low
LSTM / GRU	High	Moderate-High	Moderate	Moderate	Low	Moderate	Low
Autoencoder / VAE / GAN	Moderate-High	Low-Moderate	Moderate	Moderate	Moderate	Moderate	Low
Transformer / BERT-based	Very High	Low-Moderate	Low-Moderate	High	Moderate-High	Low-Moderate	Low-Moderate
GPT / LLM (zero/few-shot)	Very High (variable)	Low-Moderate	Low	Very High	High	High	Low
Graph Neural Networks	Very High (relational)	Low	Low	High	Moderate	Low-Moderate	Low-Moderate

Table 2. Comparative analysis of AI technique families for log-based anomaly and threat detection.

No single family wins on every measure. Classical machine learning, especially Random Forest, XGBoost, and Isolation Forest, gives the best balance of accuracy and cost and is the easiest to interpret, which makes it a good fit for high-throughput, real-time first-pass screening [13], [44]. Deep sequence models pick up temporal patterns well but lose accuracy once log formats drift [7], [16]. Transformer- and BERT-based models reach the highest raw accuracy on public benchmarks but at a much higher computing cost [3], [4]. Large Language Models generalize best to situations not seen before and can explain their findings in plain language, at the highest cost and slowest speed [25], [26]. Graph Neural Networks are the only family that captures relationships across multiple connected events, which matters for multi-step attacks, but they scale poorly and are harder to interpret on very large, constantly changing graphs [15], [18].

V. Key Literature Findings

Table 3 groups the main technique papers, comparative studies, and surveys covered earlier by technique family and year, with the full citation for each.

Authors (Year)	Technique / Focus	Key Contribution
Xu et al. (2009) [29]	PCA-based console log mining	Early PCA method for detecting large-scale system problems from console logs.
Makanju et al. (2009) [50]	IPLoM log parser	Iterative partitioning approach to clustering raw log messages into templates.
Du & Li (2016) [49]	Spell (streaming parser)	Longest-common-subsequence-based streaming log parser.
He et al. (2017) [11]	Drain (log parser)	Fixed-depth tree parser; among the most widely used log-parsing tools.
He et al. (2016) [51]	Log-parsing evaluation study	Empirically evaluates parsing methods and their downstream effect on accuracy.
Liang et al. (2007) [52]	BlueGene/L failure prediction	Source of the widely used BGL benchmark dataset.
He et al. (2016) [10]	Classical ML benchmark	Establishes Decision Tree, Logistic Regression, SVM, clustering baselines on HDFS/BGL.
Ali et al. (2023) [13]	ML/DL comparative study	Shows well-tuned classical baselines stay competitive with deep learning.
Du et al. (2017) [1]	DeepLog (LSTM)	First to treat log-key sequences as language modeling; flags low-probability events.
Meng et al. (2019) [2]	LogAnomaly (template2vec+LSTM)	Semantic embeddings for joint sequential and quantitative anomaly detection.
Zhang et al. (2019) [16]	LogRobust (Attention Bi-LSTM)	Designed to stay accurate despite unstable, evolving log templates.
Li et al. (2020) [17]	SwissLog (semantic+temporal Bi-LSTM)	Stays accurate across diverse fault types and timing changes.
Le & Zhang (2022) [7]	Reproducibility / evaluation study	Deep-learning performance drops sharply under realistic chronological splits.
Catillo et al. (2022) [20]	AutoLog (Autoencoder)	Reconstruction-error-based anomaly flagging without labeled data.
Han & Yuan (2021) [21]	Domain-adaptive model	Unsupervised domain adaptation enables cross-system anomaly detection.
Huang et al. (2020) [3]	HitAnomaly (Hierarchical Transformer)	Jointly models log template sequences and parameter values.
Nedelkoski et al. (2020) [5]	Logsy (self-attention Transformer)	Applies self-attention directly to raw tokenized logs; removes parser dependency.
Guo et al. (2021) [4]	LogBERT	BERT-style masked log-key prediction; trains without labeled anomalies.
Le & Zhang (2021) [8]	NeuralLog (parser-free Transformer)	Removes parsing step; works on semantic embeddings of raw text.
Wittkopp et al. (2022) [19]	A2Log	Two-step unsupervised pipeline combining candidate selection with attentive classification.
Ott et al. (2021) [53]	Pretrained-language-model detector	Carries over better across log datasets than dataset-specific models.
Huang et al. (2023) [22]	Pretrained hierarchical Transformer	Large-scale unlabeled pretraining before fine-tuning improves accuracy.
Liu et al. (2024)	Prompted LLM (in-context	In-context log analysis producing human-readable

Authors (Year)	Technique / Focus	Key Contribution
[25]	analysis)	explanations.
Yang et al. (2021) [6]	PLELog (semi-supervised GRU)	Probabilistic label estimation reduces reliance on fully labeled data.
Guo et al. (2022) [54]	LogLG (weak supervision + graph)	Log-event graph with weak supervision approaches fully supervised accuracy.
Li et al. (2023) [18]	Graph-level GNN (OCDiGCN)	Captures multi-hop relational anomalies with a built-in explanation.
Chen et al. (2021) [9]	Comparative DL study	Highlights the gap between reported research results and industrial practice.
Nguyen et al. (2023) [35]	Survey (Federated Learning for IDS)	Non-IID data and communication overhead identified as key challenges.
Wu et al. (2023) [23]	Log representation study	Representation choice affects accuracy as much as model choice.
Khan et al. (2024) [24]	Log-parsing sensitivity study	Quantifies how parsing errors propagate into and reduce detection accuracy.
Aziz & Munir (2024) [55]	Comparative DL evaluation	Compares DL approaches; examines parsing-accuracy interactions.
Mathe et al. (2025) [56]	Real-time benchmarking study	Benchmarks algorithms for real-time, large-scale log analysis.
Landauer et al. (2023) [14]	Survey (DL for logs)	Surveys deep learning methods for log-based anomaly detection.
Neupane et al. (2022) [36]	Survey (Explainable IDS)	Reviews explainable intrusion detection methods; proposes a generic X-IDS architecture.
Bilot et al. (2023) [15]	Survey (Graph Neural Networks)	Systematic taxonomy of GNN applications; scalability and explainability as open problems.
Xu & Ding (2024) [26]	Survey (LLMs for anomaly detection)	Strong zero/few-shot generalization; sensitive to prompting and inference cost.

Table 3. Structured summary of the reviewed AI-based log mining literature, grouped by technique family.

Read across rather than paper-by-paper, this literature supports several concrete conclusions:

- Algorithm choice should match the deployment setting; deep learning does not automatically win. Well-tuned classical baselines stay competitive with deep architectures on several public benchmarks [13], and a comparative study found a gap between reported research results and what holds up in industrial practice [9].
- Reported deep-learning accuracy depends heavily on how the data is split. It drops under realistic chronological or cross-system evaluation compared with the random splits common in earlier work [7], and a related study found that data-preprocessing leakage separately inflates the reported reliability of ML-based intrusion detection models [33].
- Log-format instability is a major cause of poor generalization, and it is the direct reason behind designs such as LogRobust [16] and SwissLog [17].
- Parsing quality is a hidden but measurable source of downstream error [24]. Parser-free approaches that work directly on raw or semantically embedded text are one way to reduce this dependency [8].
- The most accurate architectures, Transformers, LLMs, and GNNs, are also the hardest to interpret and the most expensive to run [3], [25], [15]. This is why explainability methods such as SHAP are increasingly built into detection pipelines from the start rather than added afterward [40], [36].
- Public benchmark datasets are built around reliability, not security [28], [12], so strong benchmark performance does not guarantee real attack-detection performance. This matches earlier, broader warnings about evaluating machine learning for intrusion detection outside controlled conditions [38].

- Privacy-preserving collaboration is promising but still early-stage. Federated learning allows training across organizations without centralizing raw log data, but uneven data distributions and the risk of malicious participants remain open problems [35].

VI. Research Gaps

6.1 Security-Label Scarcity

The most widely used public benchmarks, HDFS, BGL, Thunderbird, and OpenStack, were built to study system reliability rather than adversarial threats [28]. Datasets with genuine attacker activity exist but are usually kept private for security or legal reasons, so the field mostly tests against the wrong kind of ground truth.

6.2 Log Format Instability and Poor Generalization

Models trained on one software version or system often lose accuracy quickly once logging statements change or the underlying system changes [7]. This is the direct reason LogRobust [16] and SwissLog [17] were designed the way they were.

6.3 The Accuracy-Speed-Explainability Trade-off

The most accurate architectures covered here, Transformers, BERT variants, GNNs, and LLMs, are also the most expensive to run and the hardest to interpret. Explainable AI methods [40], [36] help with this only partly, since producing explanations in real time for high-volume streaming logs is still largely unsolved.

6.4 Inconsistent, Leakage-Prone Evaluation

Studies split training and test data randomly, chronologically, or across systems, and this inconsistency inflates reported performance compared with realistic deployment [7]. A related empirical study found that preprocessing pattern leakage separately and measurably distorts the reported reliability of ML-based intrusion detection [33], which makes fair comparison across the literature difficult.

6.5 Log Parsing as a Hidden Source of Error

A model can be well designed and still inherit errors from an upstream parser, which measurably reduces detection accuracy [24]. Parsing quality is often treated as a solved, secondary step rather than something to report and control for directly.

6.6 Immature Privacy-Preserving Collaboration

Federated learning allows model training across organizations without sharing raw log data, but uneven data distributions and vulnerability to malicious participants are still unresolved [35], [39].

VII. Future Research Directions

7.1 Realistic, Security-Labeled Benchmarks

The field needs large-scale, multi-source log datasets that build on the Loghub model [12] with genuine attacker-labeled ground truth instead of purely synthetic labels.

7.2 Latency-Aware Model Compression

Compressing Transformer-, BERT-, and LLM-based detectors would make them practical for real-time use in security operations centers, where speed matters as much as accuracy [56].

7.3 Standardized, Leakage-Free Evaluation

Chronological and cross-system train/test splits, along with explicit checks for preprocessing leakage, should become a standard reporting requirement [7], [33].

7.4 Hybrid, Layered Architectures

Instead of relying on one algorithm, a practical system could combine a fast, low-cost classical model, such as Isolation Forest, XGBoost, or a small LSTM, for high-throughput first-pass screening, and send only high-suspicion events to a more accurate Transformer, BERT, or GNN model for closer analysis. This directly addresses the accuracy-speed trade-off described earlier. Attention-weight visualization, SHAP-style attribution, and LLM-generated natural-language explanations could sit on top of this for analyst-facing interpretability, and an optional federated-learning layer [39] could periodically update the shared model using privacy-preserving contributions from multiple organizations.

7.5 Continual and Lifelong Learning

Models that adapt to changing log formats and system behavior over time, without retraining from scratch or losing previously learned attack patterns, would directly address the generalization problem described earlier.

7.6 Grounded, Trustworthy LLM Explanations

Explanation methods that point to specific log evidence for each detection, instead of producing plausible-sounding but ungrounded text, would improve analyst trust beyond current prompt-based approaches [25].

VIII. Conclusion

Log-based threat detection has gradually moved from simple rule-based methods to machine learning, deep learning, Transformers, LLMs, and Graph Neural Networks. Each new approach has improved the ability to detect complex and unknown threats, but it also requires more computing resources.Overall, traditional

methods are fast and cost-effective, while deep learning methods are better at finding complex patterns in logs. Advanced models such as Transformers, LLMs, and graph-based methods can improve detection performance, but they require more resources and can be harder to explain.

The literature review paper shows that no single method is suitable for every situation. The choice depends on the system requirements and the balance needed between accuracy, speed, cost, adaptability, and explainability. Future research should focus on developing models that are accurate, practical, adaptable, and easier to understand for real-time threat detection.

References

[1] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," in Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS), 2017, pp. 1285–1298, doi: 10.1145/3133956.3134015.
[2] W. Meng et al., "LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs," in Proc. 28th Int. Joint Conf. Artif. Intell. (IJCAI), 2019, pp. 4739–4745, doi: 10.24963/ijcai.2019/658.
[3] S. Huang, Y. Liu, C. Fung, R. He, Y. Zhao, H. Yang, and Z. Luan, "HitAnomaly: Hierarchical transformers for anomaly detection in system log," IEEE Trans. Netw. Service Manage., vol. 17, no. 4, pp. 2064–2076, 2020, doi: 10.1109/TNSM.2020.3034647.
[4] H. Guo, S. Yuan, and X. Wu, "LogBERT: Log anomaly detection via BERT," in Proc. Int. Joint Conf. Neural Netw. (IJCNN), 2021, pp. 1–8.
[5] S. Nedelkoski, J. Bogatinovski, A. Acker, J. Cardoso, and O. Kao, "Self-attentive classification-based anomaly detection in unstructured logs," in Proc. IEEE Int. Conf. Data Mining (ICDM), 2020, pp. 1196–1201.
[6] L. Yang, J. Chen, Z. Wang, W. Wang, J. Jiang, X. Dong, and W. Zhang, "Semi-supervised log-based anomaly detection via probabilistic label estimation," in Proc. IEEE/ACM 43rd Int. Conf. Softw. Eng. (ICSE), 2021, pp. 1448–1460, doi: 10.1109/ICSE43902.2021.00130.
[7] V.-H. Le and H. Zhang, "Log-based anomaly detection with deep learning: How far are we?" in Proc. 44th Int. Conf. Softw. Eng. (ICSE), 2022, pp. 1356–1367.
[8] V.-H. Le and H. Zhang, "Log-based anomaly detection without log parsing," in Proc. 36th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE), 2021, pp. 492–504, doi: 10.1109/ASE51524.2021.9678773.
[9] Z. Chen, J. Liu, W. Gu, Y. Su, and M. R. Lyu, "Experience report: Deep learning-based system log analysis for anomaly detection," arXiv:2107.05908, 2021.
[10] S. He, J. Zhu, P. He, and M. R. Lyu, "Experience report: System log analysis for anomaly detection," in Proc. IEEE 27th Int. Symp. Softw. Rel. Eng. (ISSRE), 2016, pp. 207–218.
[11] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An online log parsing approach with fixed depth tree," in Proc. IEEE Int. Conf. Web Services (ICWS), 2017, pp. 33–40, doi: 10.1109/ICWS.2017.13.
[12] J. Zhu, S. He, P. He, J. Liu, and M. R. Lyu, "Loghub: A large collection of system log datasets for AI-driven log analytics," in Proc. IEEE 34th Int. Symp. Softw. Rel. Eng. (ISSRE), 2023, pp. 355–366, doi: 10.1109/ISSRE59848.2023.00071.
[13] S. Ali, C. Boufaied, D. Bianculli, P. Branco, and L. Briand, "A comprehensive study of machine learning techniques for log-based anomaly detection," arXiv:2307.16714, 2023.
[14] M. Landauer, S. Onder, F. Skopik, and M. Wurzenberger, "Deep learning for anomaly detection in log data: A survey," Mach. Learn. Appl., vol. 12, art. 100470, 2023, doi: 10.1016/j.mlwa.2023.100470.
[15] T. Bilot, N. El Madhoun, K. Al Agha, and A. Zouaoui, "Graph neural networks for intrusion detection: A survey," IEEE Access, vol. 11, pp. 49114–49139, 2023, doi: 10.1109/ACCESS.2023.3275789.
[16] X. Zhang et al., "Robust log-based anomaly detection on unstable log data," in Proc. ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng. (ESEC/FSE), 2019, pp. 807–817, doi: 10.1145/3338906.3338931.
[17] X. Li, P. Chen, L. Jing, Z. He, and G. Yu, "SwissLog: Robust and unified deep learning based log anomaly detection for diverse faults," in Proc. IEEE 31st Int. Symp. Softw. Rel. Eng. (ISSRE), 2020, pp. 92–103, doi: 10.1109/ISSRE5003.2020.00018.
[18] Z. Li, J. Shi, and M. van Leeuwen, "Graph neural network based log anomaly detection and explanation," arXiv:2307.00527, 2023.
[19] T. Wittkopp, A. Acker, S. Nedelkoski, J. Bogatinovski, D. Scheinert et al., "A2Log: Attentive augmented log anomaly detection," in Proc. 55th Hawaii Int. Conf. Syst. Sci. (HICSS), 2022, pp. 1853–1862.
[20] M. Catillo, A. Pecchia, and U. Villano, "AutoLog: Anomaly detection by deep autoencoding of system logs," Expert Syst. Appl., vol. 191, art. 116263, 2022.
[21] X. Han and S. Yuan, "Unsupervised cross-system log anomaly detection via domain adaptation," in Proc. 30th ACM Int. Conf. Inf. Knowl. Manage. (CIKM), 2021, pp. 3068–3072.
[22] S. Huang, Y. Liu, C. J. Fung, H. Wang, H. Yang, and Z. Luan, "Improving log-based anomaly detection by pre-training hierarchical transformers," IEEE Trans. Comput., vol. 72, no. 9, pp. 2656–2667, 2023.
[23] X. Wu, H. Li, and F. Khomh, "On the effectiveness of log representation for log-based anomaly detection," Empir. Softw. Eng., vol. 28, no. 6, 2023.
[24] Z. A. Khan, D. Shin, D. Bianculli, and L. C. Briand, "Impact of log parsing on deep learning-based anomaly detection," Empir. Softw. Eng., vol. 29, no. 6, pp. 1–33, 2024.
[25] Y. Liu, S. Tao, W. Meng, J. Wang, W. Ma, Y. Chen, Y. Zhao, H. Yang, and Y. Jiang, "Interpretable online log analysis using large language models with prompt strategies," in Proc. IEEE/ACM 32nd Int. Conf. Program Comprehension (ICPC), 2024, pp. 35–46.
[26] R. Xu and K. Ding, "Large language models for anomaly and out-of-distribution detection: A survey," arXiv:2409.01980, 2024.
[27] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," ACM Comput. Surv., vol. 41, no. 3, pp. 1–58, 2009.
[28] A. Oliner and J. Stearley, "What supercomputers say: A study of five system logs," in Proc. 37th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN), 2007, pp. 575–584.
[29] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in Proc. ACM SIGOPS 22nd Symp. Oper. Syst. Princ. (SOSP), 2009, pp. 117–132, doi: 10.1145/1629575.1629587.
[30] J. Lansky, S. Ali, M. Mohammadi et al., "Deep learning-based intrusion detection systems: A systematic review," IEEE Access, vol. 9, pp. 101574–101599, 2021, doi: 10.1109/ACCESS.2021.3097247.
[31] Z. Ahmad, A. S. Khan, C. W. Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," Trans. Emerg. Telecommun. Technol., vol. 32, no. 1, e4150, 2021, doi: 10.1002/ett.4150.
[32] S. V. N. Santhosh Kumar, M. Selvi, and A. Kannan, "A comprehensive survey on machine learning-based intrusion detection systems for secure communication in Internet of Things," Comput. Intell. Neurosci., vol. 2023, art. 8981988, 2023, doi: 10.1155/2023/8981988.
[33] M. A. Bouke and A. Abdullah, "An empirical study of pattern leakage impact during data preprocessing on machine learning-based intrusion detection models reliability," Expert Syst. Appl., vol. 230, art. 120715, 2023, doi: 10.1016/j.eswa.2023.120715.

[34] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Deep learning-based intrusion detection for distributed denial of service attacks: A comprehensive review," IEEE Access, vol. 11, pp. 4401–4426, 2023.
[35] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "Federated learning for intrusion detection systems: A survey," ACM Comput. Surv., vol. 55, no. 13s, pp. 1–35, 2023.
[36] S. Neupane, J. Ables, W. Anderson, S. Mittal, S. Rahimi, I. Banicescu, and M. Seale, "Explainable intrusion detection systems (X-IDS): A survey of current methods, challenges, and opportunities," IEEE Access, vol. 10, pp. 112392–112415, 2022, doi: 10.1109/ACCESS.2022.3216617.
[37] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," IEEE Commun. Surv. Tutorials, vol. 18, no. 2, pp. 1153–1176, 2016.
[38] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in Proc. IEEE Symp. Secur. Privacy, 2010, pp. 305–316.
[39] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in Proc. 20th Int. Conf. Artif. Intell. Stat. (AISTATS), 2017, pp. 1273–1282.
[40] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2017, pp. 4765–4774.
[41] A. Vaswani et al., "Attention is all you need," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2017, pp. 5998–6008.
[42] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proc. NAACL-HLT, 2019, pp. 4171–4186, doi: 10.18653/v1/N19-1423.
[43] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in Proc. 8th IEEE Int. Conf. Data Mining (ICDM), 2008, pp. 413–422, doi: 10.1109/ICDM.2008.17.
[44] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2016, pp. 785–794, doi: 10.1145/2939672.2939785.
[45] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.
[46] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2014.
[47] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," arXiv:1312.6114, 2014.
[48] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?': Explaining the predictions of any classifier," in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2016, pp. 1135–1144, doi: 10.1145/2939672.2939778.
[49] M. Du and F. Li, "Spell: Streaming parsing of system event logs," in Proc. IEEE 16th Int. Conf. Data Mining (ICDM), 2016, pp. 859–864, doi: 10.1109/ICDM.2016.0103.
[50] A. A. Makanju, A. N. Zincir-Heywood, and E. E. Milios, "Clustering event logs using iterative partitioning," in Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2009, pp. 1255–1264, doi: 10.1145/1557019.1557154.
[51] P. He, J. Zhu, S. He, J. Li, and M. R. Lyu, "An evaluation study on log parsing and its use in log mining," in Proc. 46th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN), 2016, pp. 654–661.
[52] Y. Liang, Y. Zhang, H. Xiong, and R. Sahoo, "Failure prediction in IBM BlueGene/L event logs," in Proc. IEEE 7th Int. Conf. Data Mining (ICDM), 2007, pp. 583–588, doi: 10.1109/ICDM.2007.46.
[53] H. Ott, J. Bogatinovski, A. Acker, S. Nedelkoski, and O. Kao, "Robust and transferable anomaly detection in log data using pre-trained language models," in Proc. IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS) Workshops, 2021, pp. 1–7.
[54] H. Guo, Y. Guo, R. Chen, J. Yang, and T. Zhang, "LogLG: Weakly supervised log anomaly detection via log-event graph construction," in Proc. Int. Conf. Database Syst. Adv. Appl. (DASFAA), 2022, pp. 490–501.
[55] A. Aziz and K. Munir, "Anomaly detection in logs using deep learning," IEEE Access, vol. 12, pp. 176124–176135, 2024, doi: 10.1109/ACCESS.2024.3506332.
[56] Z. Máthé, A. Nagy, and T. Kovács, "Anomaly detection algorithms for real-time log data analysis at scale," IEEE Access, vol. 13, 2025, doi: 10.1109/ACCESS.2025.3594469.
[57] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection for discrete sequences: A survey," IEEE Trans. Knowl. Data Eng., vol. 24, no. 5, pp. 823–839, 2012, doi: 10.1109/TKDE.2010.235.
[58] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in Proc. Int. Conf. Learn. Represent. (ICLR), 2019.
[59] C. Bertero, M. Roy, C. Sauvanaud, and G. Trédan, "Experience report: Log mining using natural language processing and application to anomaly detection," in Proc. IEEE 28th Int. Symp. Softw. Rel. Eng. (ISSRE), 2017, pp. 351–360.