Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.077388

ARTICLE

Explainable Anomaly Detection for System Logs in Distributed Environments

Zhaojun Gu[1], Wenlong Yue[2] and Chunbo Liu[1,*]

[1]Information Security Evaluation Center, Civil Aviation University of China, Tianjin, China
[2]College of Computer and Artificial Intelligence, Civil Aviation University of China, Tianjin, China

*Corresponding Author: Chunbo Liu. Email: cbliu@cauc.edu.cn

Received: 08 December 2025; Accepted: 28 February 2026; Published: 09 April 2026

ABSTRACT: Anomaly detection in system logs is a critical technical means for identifying potential faults and security risks. In distributed environments, traditional deep learning-based log anomaly detection methods often suffer from shortcomings in transparency, computational overhead, and data privacy protection. To address these issues, this paper proposes a federated learning-driven lightweight and explainable log anomaly detection framework named FedXLog. The framework adapts to heterogeneous logs through hierarchical feature extraction, introduces the Federated Gradient Trajectory Aggregation algorithm (FedGradTrace) to enhance the explainability of the parameter aggregation process, constructs lightweight models using knowledge distillation, and achieves globally consistent explanatory capabilities by integrating hash feature alignment. Experimental results demonstrate that FedXLog possesses the dual advantages of high detection accuracy and lightweight deployment for heterogeneous logs in distributed scenarios. It can effectively identify key decision-making features and locate typical root causes of anomalies. Notably, the framework has been specifically optimized for the unique characteristics of distributed logs. Distinguished from general federated explainable methods, it can directly support abnormal root cause localization in Operations and Maintenance scenarios. This further verifies the application value of scenario-specific adaptation of federated learning in the field of log analysis, thereby expanding the scope of application of explainable log anomaly detection.

KEYWORDS: Anomaly detection; log parsing; federated learning; explainable artificial intelligence

1 Introduction

With the advancement of digital transformation, the importance of system log anomaly detection in the field of cybersecurity has become increasingly prominent. Attackers' behaviors such as privilege escalation and data theft are often recorded in system logs. Currently, network attacks are becoming more concealed and frequent, rendering traditional defense methods inadequate to address such threats. Anomaly detection technology based on log analysis has thus become a research hotspot and a core means of detecting intrusions and analyzing attack behaviors [1,2]. Traditional log analysis methods based on rule matching require accurate log parsing and manually designed features, which have significant limitations in dealing with intelligent and emerging network attacks. In contrast, machine learning and deep learning technologies, with their ability to automatically identify patterns and adapt to different scenarios, can effectively reduce false positive rates and enhance the ability to recognize emerging threats [3–7], providing a new path for proactive cybersecurity defense systems.

As the complexity of network attacks increases, system log anomaly detection faces the dual pressures of expanding data scale and privacy protection, and the limitations of traditional centralized detection

models have gradually become apparent. Although centralized systems have advantages in log management efficiency and tool integration, they suffer from issues such as insufficient privacy protection, weak reliability, and poor scalability: centralized storage of raw logs increases the risk of sensitive information exposure; the failure or compromise of the central node may cause the entire system to break down; and performance degradation tends to occur during high-concurrency log processing [8,9]. In comparison, distributed solutions offer stronger adaptability.

Federated Learning (FL) provides an effective solution for privacy protection in log anomaly detection through local data processing and collaborative model updates. Participants only share model parameter updates instead of raw data, reducing the risk of privacy leakage [10,11]. Additionally, the distributed architecture ensures system stability, allowing services to be maintained even if some nodes fail. However, current federated learning models face the challenge of insufficient explainability—cybersecurity operations and maintenance personnel find it difficult to understand the detection logic and cannot trace the attack path [12–14]. It is particularly important to note that distributed logs exhibit the characteristics of heterogeneous formats, scarce anomaly samples, and the need for root cause localization in Operations and Maintenance (O&M). Existing federated explainable methods often suffer from misalignment or invalidation of explanations because they fail to adapt to these characteristics [15]. In contrast, the design of FedXLog specifically addresses these exclusive pain points in log scenarios. Therefore, improving model explainability while ensuring data security and system reliability has become an urgent breakthrough direction in this field [16–20].

The main contributions of this paper are as follows:

- A federated learning-driven lightweight and explainable log anomaly detection framework, FedXLog, is proposed, integrating knowledge distillation, federated learning, and explainability technologies. This framework addresses three core challenges in log anomaly detection within distributed environments: privacy protection, computational overhead, and the decision-making black box problem. While retaining most of the semantic feature extraction capabilities, it significantly reduces computational costs, avoids privacy leakage risks associated with raw log transmission, and adapts to distributed deployment scenarios involving heterogeneous logs.
- A federated gradient trajectory aggregation algorithm (FedGradTrace) is proposed to improve the parameter aggregation logic of the classical FedAvg algorithm. By retaining the gradient update trajectories of each client and introducing the Frobenius norm to achieve stability-sensitive parameter aggregation, the algorithm not only maintains privacy protection properties but also significantly enhances the explainability of the parameter aggregation process. This provides support for tracing the evolution of model parameters and locating the sources of anomaly impacts.
- A hash feature alignment algorithm for log texts is designed. By matching 8-hex hash values via regular expressions, counting high-frequency key hashes, and mapping original SHapley Additive exPlanations (SHAP) values to a unified dimension, a globally consistent explanation space across clients is established. This algorithm effectively addresses the explanation misalignment problem caused by format differences in heterogeneous logs, ensuring the consistency and validity of explanation results in distributed scenarios.

The effectiveness of FedXLog is validated on four public system log datasets: HDFS, BGL, Thunderbird, and Spirit. This study contributes a scenario-adapted integration of federated learning and explainability technologies for log anomaly detection, addressing unmet needs in heterogeneous log adaptation and O&M-oriented root cause localization that remain unaddressed by prior related works. Without sacrificing accuracy, it significantly enhances model explainability, providing technical support for addressing issues such as data silos, distributed deployment, and compliance requirements.

2 Related Work

Log anomaly detection has evolved from traditional rule-based methods to intelligent learning-driven approaches, with key advancements focusing on deep learning, federated learning (FL), and explainable artificial intelligence (XAI)—three domains closely tied to the core challenges addressed in this study.

Traditional log anomaly detection relies on manual rule matching or statistical feature analysis, which struggle to adapt to dynamic log changes and large-scale data, leading to limited accuracy in emerging attack scenarios [21,22]. Deep learning has become the mainstream by enabling automatic feature extraction: DeepLog [3] uses LSTMs for log sequence prediction but adopts a centralized architecture and lacks explainability; LogAnomaly [4] integrates structural and parameter features via dual-channel LSTMs but fails to mitigate privacy risks and suffers from opaque attention weights; LogBERT [5] leverages BERT's semantic understanding to achieve high accuracy in centralized settings but has excessive parameters, lacks explanation mechanisms, and is incompatible with distributed scenarios; LogRobust [6] and PLELog [7] improve robustness and handle unstable logs, respectively, but retain centralized designs and black-box characteristics that hinder O&M applicability.

Federated learning addresses privacy concerns by enabling local training and collaborative parameter updates, but existing methods have critical gaps in log scenario adaptation. FedAvg [23] established the basic FL framework but ignores anomaly detection-specific needs, explainability, and non-IID data stability. Subsequent optimizations like FedProx [24] (proximal regularization for heterogeneous data) and FedOPT [25] (adaptive optimizers for convergence) focus on training efficiency without addressing explainability or log-specific heterogeneity. Specialized federated anomaly detection methods such as FedAD [26] and FedHAP [27] either fail to handle log format differences (e.g., Linux vs. Windows) or lack feature attribution mechanisms, making them unsuitable for O&M root cause localization. In the log-specific federated learning space, FedLog [28] proposes a customizable and communication-efficient federated anomaly detection approach for IoT system logs, while AdaLightLog [29] enhances application log anomaly detection via adaptive federated learning, both focusing on training efficiency and data heterogeneity but not on model explainability or root cause localization.

To tackle non-IID data challenges in federated learning, gradient-aware aggregation strategies have been developed to improve model stability. UGA [30] introduces unbiased gradient aggregation and controllable meta-updating to reduce the impact of heterogeneous gradients, while FedGP [31] adopts genetic programming for evolutionary aggregation in federated learning with non-IID data, enhancing convergence by optimizing gradient combination strategies. FedGAM [32] further refines gradient weighting via gradient norm-aware minimization and control variables to achieve generalized federated learning, and FedTrack [33] leverages adaptive federated learning for collaborative target tracking, demonstrating the effectiveness of gradient-aware methods in heterogeneous scenarios. However, these gradient optimization works are not tailored to log anomaly detection tasks and do not incorporate explainability components to support O&M decision-making.

Feature alignment is another key research direction in federated learning to address data heterogeneity, with several methods proposed for consistent representation learning. FedPAC [34] focuses on consistent representation learning for federated unsupervised learning under data heterogeneity, while FedCCFA [35] addresses distributed concept drift via classifier clustering and feature alignment. FL-Joint [36] jointly aligns features and labels in federated learning to mitigate the impact of heterogeneous data distributions. Despite their advances in feature consistency, these methods do not consider the unique characteristics of log data, such as heterogeneous formats and semantic diversity, and thus cannot establish a globally consistent explanation space for log anomaly detection.

Explainability technologies have achieved remarkable results in centralized models. For instance, SHAP [37] can quantify feature attribution, and LIME [38] can generate local explanations. In centralized log anomaly detection, X-HEART [39] proposes an explainable heterogeneous log anomaly detection method using robust transformers, combining transformer-based feature extraction with explainability components to identify key log features. However, in federated scenarios, SHAP requires the transmission of local explanatory data, which increases privacy risks; LIME is prone to global inconsistency issues due to independent explanations from individual nodes. Meanwhile, the mandatory requirements for the explainability of high-risk systems in the EU's Artificial Intelligence Act highlight the contradiction between such regulations and the insufficient compliance of existing federated log detection solutions.

XAI technologies like SHAP [37] and LIME [38] have advanced centralized model explainability, but their direct application in federated scenarios is problematic: SHAP requires transmitting local explanatory data (increasing privacy risks), while LIME suffers from global explanation inconsistency. Recent federated XAI frameworks (e.g., FedXAIIDS [40], federated Transformer [41]) integrate explanation modules but overlook log-specific characteristics—heterogeneous multimodality, few-shot novel anomalies, and O&M root cause demands—resulting in invalid or misaligned explanations that cannot link to "faulty components-propagation paths."

In summary, current research faces three core limitations: federated log detection frameworks lack explainability; XAI technologies are not adapted to distributed log characteristics; and no universal framework addresses cross-platform log heterogeneity. These gaps highlight the necessity of FedXLog, which is tailored to the unique demands of distributed log anomaly detection.

3 Scheme Design

3.1 Log Parsing

System logs are mostly stored as unstructured text and require preprocessing through log parsing to support efficient anomaly detection. To address the challenges of log heterogeneity, dynamic changes, and parameter extraction, this study proposes a parsing framework that integrates automated template generation and hierarchical feature extraction (as shown in Fig. 1).

First, the Drain [42] framework is adopted to automatically generate log templates: it learns template structures through depth-first search, avoiding the limitations of traditional rule bases. Each template is assigned a unique hash feature as its event ID, with a one-way hash algorithm ensuring privacy and preventing the reverse derivation of original logs. Moreover, this one-way hash property not only blocks the reverse engineering of raw logs but also makes the mapping between hash values and log templates irreversible—attackers cannot reverse-derive the corresponding original templates from hash values. By adjusting the similarity threshold and tree depth, the risk of overfitting/underfitting is balanced, and regular expressions are used to accurately extract log parameters. Relying on the mapping relationship between hashes and original logs, rapid traceability and correlation analysis of log events are realized, constructing an adaptive parsing system with hash traceability capabilities.

Second, a two-layer feature extraction strategy is designed to adapt to heterogeneous logs: the basic feature layer extracts structured fields such as event ID, timestamp, and log level; the semantic feature layer parses unstructured content to extract key entities such as IP addresses and file block IDs. The parsed logs are converted into event sequences, and sliding windows are used for grouping to generate sequence files containing hash feature sets and anomaly labels. The log template generator unifies the format of heterogeneous logs, and the original content of logs that cannot be automatically parsed is retained as supplementary features to ensure the framework's universality.

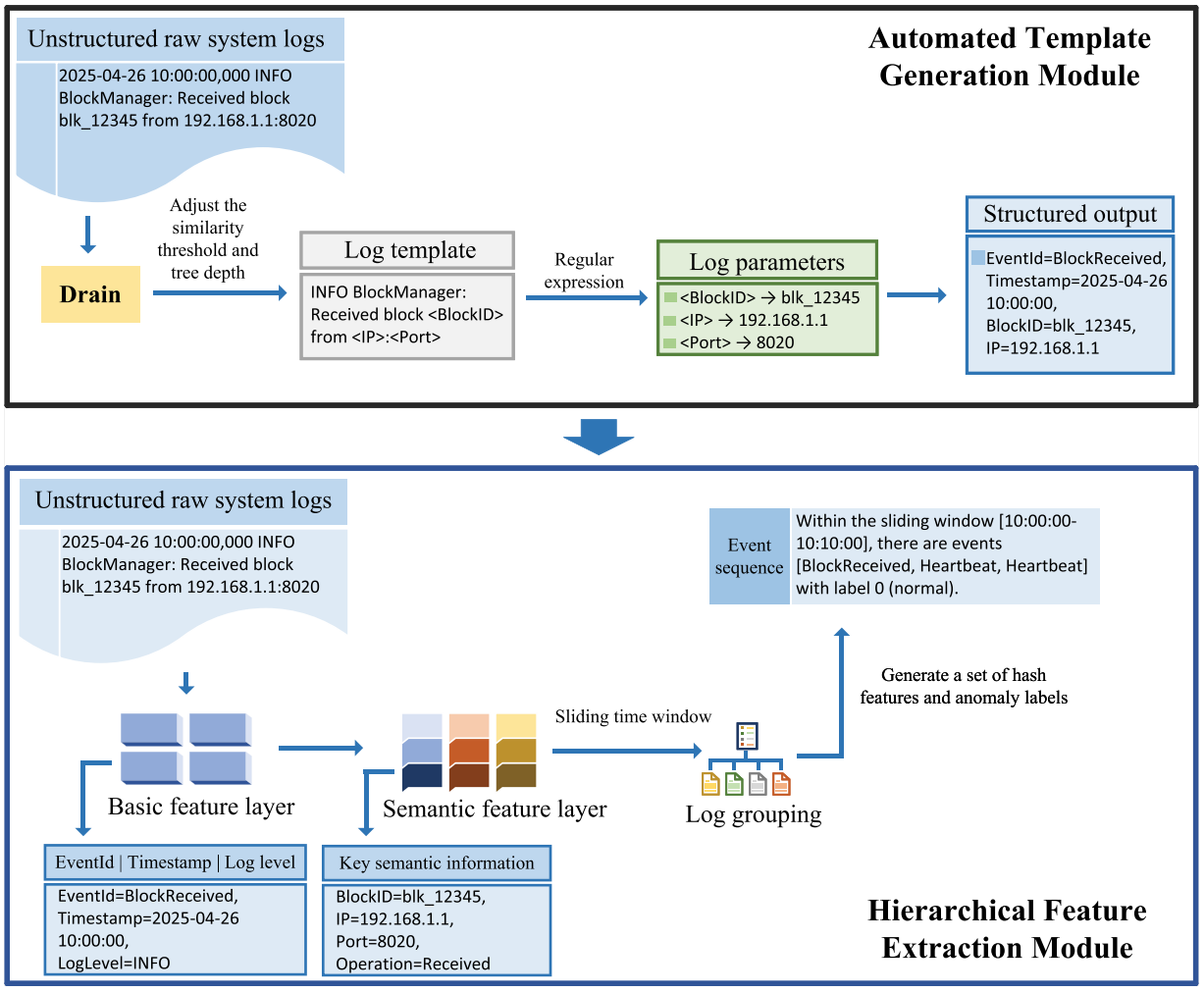


Figure 1: The overall workflow of the log parsing framework.

By integrating automated template generation and hierarchical feature extraction, this framework provides high-quality structured data for subsequent anomaly detection.

3.2 Anomaly Detection

This study proposes an anomaly detection framework FedXLog that integrates federated learning, model explanation, and knowledge distillation (as shown in Fig. 2).

3.2.1 Federated Learning

FedXLog is implemented based on the federated learning framework Flower [43] to achieve cross-client collaborative training. Each client trains an anomaly detection model based on local log data and only uploads model parameter updates to the server. The mapping relationship between hash values and log templates is only maintained locally on the client side. The server side only receives parameter updates and does not store any hash-template mapping information—even if an attacker obtains the hash features from the server side, they cannot associate them with specific fault templates without the support of a mapping library.

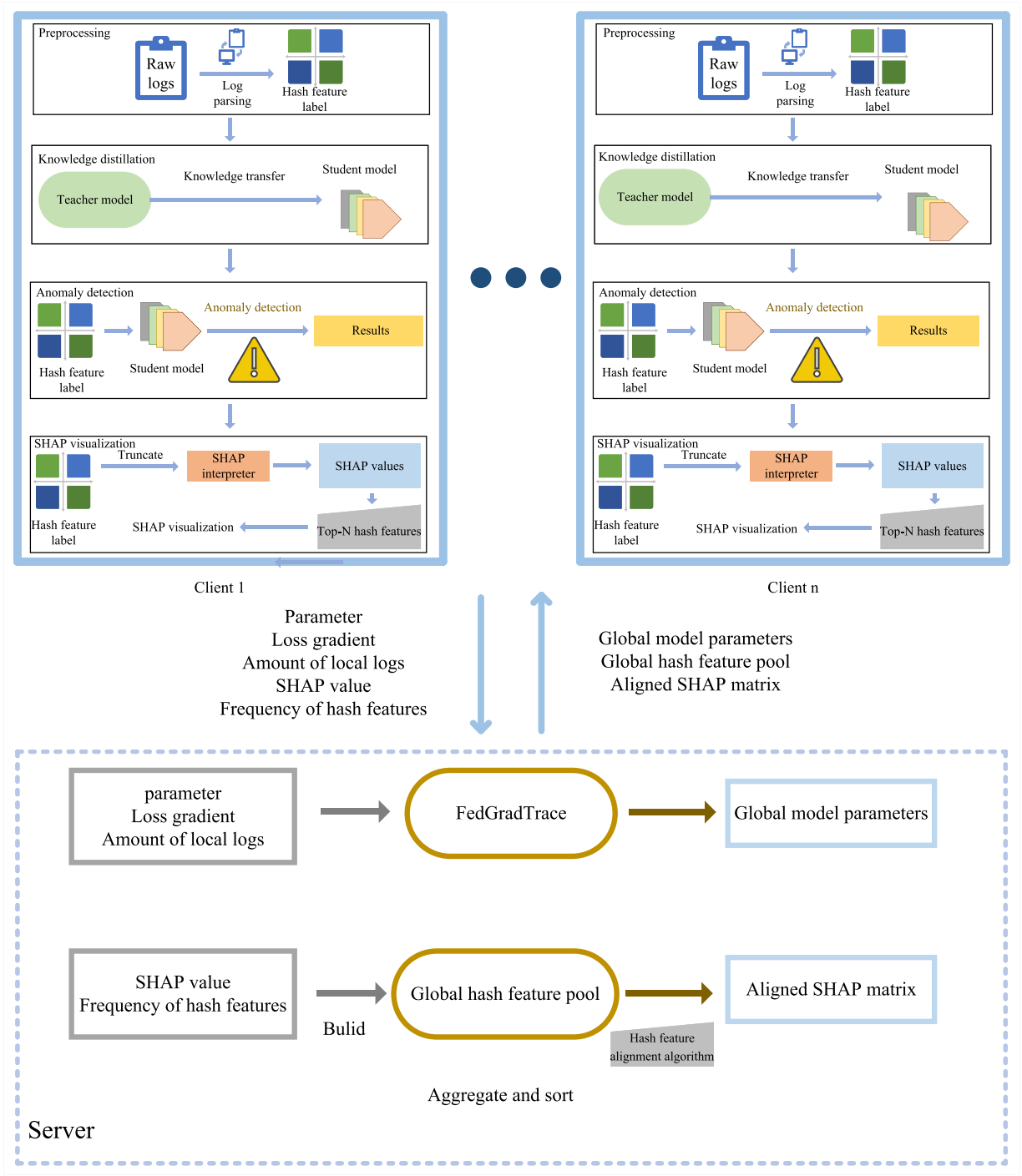


Figure 2: The overall workflow of the anomaly detection framework.

FedGradTrace improves the classic FedAvg [44] algorithm. Unlike the original FedAvg, which directly performs weighted averaging of client parameters, FedGradTrace adopts a dynamic parameter aggregation mechanism, with specific rules as follows:

$$\Gamma_k^t = \left\{\nabla L_k\left(\theta^{t-i}\right)\right\}_{i=0}^{\tau-1} \tag{1}$$

where Γ_k^t denotes the gradient trajectory of client k at round t, preserving the last τ local gradients. ∇ epresents the gradient operator, used to calculate the derivative of the loss function with respect to model parameters. It reflects the direction and magnitude of parameter adjustments required to reduce loss during client k 's local training; L_k denotes the local loss function of client k, constructed based on the local system log data hosted by client k ; θ^{t-i} stands for the global model parameter at the i-th step before the t-th training round, and it serves as the initial parameter for client k to compute the local gradient when generating the i-th gradient in the gradient trajectory Γ_k^t; i is an index variable used to traverse the τ historical gradient steps of client k before the t-th training round, ensuring the gradient trajectory Γ_k^t fully covers the last τ local gradients of the client. The parameter aggregation mechanism is then formulated as:

$$\theta^{t+1} = \sum_{k=1}^{K} \frac{n_k \cdot \|\Gamma_k^t\|_F}{n} \left(\theta^t - \eta \nabla L_k(\theta^t)\right) \tag{2}$$

Eq. (2) is derived by extending FedAvg to address log-specific heterogeneity and explainability: starting from FedAvg's core aggregation rule $\theta^{t+1} = \sum_{k=1}^{K} \frac{n_k}{n}(\theta^t - \eta \nabla L_k(\theta^t))$, we introduce gradient trajectory Γ_k^t (Eq. (1)) to enable traceability of parameter evolution from local log data; to mitigate instability from heterogeneous logs, we integrate Frobenius norm $\|\Gamma_k^t\|_F = \sqrt{\sum_{i=1}^{\tau} \|\nabla L_k(\theta^{t-i})\|^2}$ to quantify gradient volatility, then combine it with sample-size weighting to form $w_k = \frac{n_k \cdot \|\Gamma_k^t\|_F}{n}$, substituting w_k into FedAvg yields Eq. (2), balancing data scale, stability, and explainability for distributed logs.

This formulation introduces two fundamental innovations compared to conventional FedAvg: Firstly, explicit retention of gradient update trajectories through Γ_k^t provides unprecedented visibility into client optimization dynamics. Secondly, the integration of the Frobenius norm $\|\Gamma_k^t\|_F$ replaces the rudimentary data volume weighting $\frac{n_k}{n}$ with a stability-sensitive aggregation scheme. θ^t denotes the global model parameter after the t-th training round, which serves as the initial parameter for each client's local training in the t-th round; θ^{t+1} represents the updated global model parameter after the t-th round of parameter aggregation, and it will be used as the initial global parameter for the $(t+1)$-th training round; K stands for the total number of clients participating in federated training; n_k is the number of local log samples of client k, i.e., the total number of log data samples owned by client k; n denotes the total number of samples of all clients, and $n = \sum_{k=1}^{K} n_k$, which is used to normalize the weight of the number of samples of each client; η is the learning rate for local model training, which controls the step size of parameter adjustment during the local gradient descent process of the client; $\nabla L_k(\theta^t)$ represents the local loss gradient of client k under the global model parameter θ^t, which is calculated based on the local log data of client k and reflects the direction of parameter updates required by client k to reduce the local training loss. The FedGradTrace framework consequently achieves dual objectives: maintaining FedAvg's privacy-preserving characteristics while significantly enhancing explainability through parameter evolution transparency. To formalize why gradient trajectory preservation enhances explainability and stability, we define core notations and derive key properties as follows:

Let the federated system consist of K clients. For the t-th global round: $\theta^t \in \mathbb{R}^D$: Global model parameters (dimension D), $\Gamma_k^t = \{\nabla L_k(\theta^{t-\tau}), \ldots, \nabla L_k(\theta^{t-1})\}$: Gradient trajectory of client k (preserving last τ local gradients), $\|\Gamma_k^t\|_F = \sqrt{\sum_{i=1}^{\tau} \|\nabla L_k(\theta^{t-i})\|^2}$: Frobenius norm of Γ_k^t (stacked as $D \times \tau$ matrix), $L_k(\theta)$: Local loss of client k (convex, Lipschitz-continuous with constant L).

FedGradTrace updates global parameters as:

$$\theta^{t+1} = \theta^t - \eta \cdot \frac{\sum_{k=1}^{K} \|\Gamma_k^t\|_F \cdot \nabla L_k(\theta^t)}{\sum_{k=1}^{K} \|\Gamma_k^t\|_F} \tag{3}$$

Compared to FedAvg (weighted by data size n_k), this achieves stability by: $\|\Gamma_k^t\|_F$ quantifies gradient volatility: Smaller values indicate consistent local optimization (aligned with global trends). Outlier suppression: For clients with volatile trajectories ($\|\Gamma_k^t\|_F \gg \overline{\|\Gamma\|_F}$), their weight is downweighted, bounding parameter updates:

$$\|\theta^{t+1} - \theta^t\| \leq \eta L \cdot \max_k \|\nabla L_k(\theta^t)\| \tag{4}$$

Ensuring convergence to a stable equilibrium.

Explainability is enhanced through traceable contributions and SHAP consistency: Client contribution quantification: The impact of client k on global update is:

$$\Delta_k^t = \eta \cdot \frac{\|\Gamma_k^t\|_F}{\sum_{m=1}^K \|\Gamma_m^t\|_F} \cdot \nabla L_k(\theta^t) \tag{5}$$

Enabling decomposition $\theta^{t+1} - \theta^t = \sum_{k=1}^K \Delta_k^t$ to trace update sources. - SHAP consistency alignment: Local SHAP matrices Φ_k are linked to Γ_k^t via:

$$\nabla L_k(\theta^t) = \frac{1}{N_k} \sum_{i=1}^{N_k} \sum_{j=1}^{M} \text{SHAP}(x_{i,j}^k) \cdot \nabla_\theta f_\theta(x_{i,j}^k) \tag{6}$$

Preserving Γ_k^t ensures cross-round consistency of Φ_k, which is mapped to global SHAP matrix via hash alignment (Algorithm 1), maintaining explanatory uniformity.

Notably, the system implements persistent storage of aggregated parameters after each training round—a capability absent in standard FedAvg implementations—enabling comprehensive retrospective analysis of model convergence behavior. All clients utilize the DistilBERT-base-uncased architecture [45] as the foundational model, optimized via the AdamW algorithm with learning rate $\eta = 2 \times 10^{-5}$.

Algorithm 1: Hash feature alignment for global explanation in FedXLog

Require: Local log text collections of each client: $S = \{S_1, S_2, \ldots, S_n\}$
Ensure: Globally unified SHAP feature matrix $M \in \mathbb{R}^{m \times 8}$

```
Initialize the global hash dictionary H ← ∅
for k = 1 to n do
    Hash sequence list L_k ← [ ]
    for each log s ∈ S_k do
        Use the regular expression r′([A − Fa − f0 − 9]{8})′ to match the hash value in s
        Record the matching result to L_k
    end for
    Count the occurrence frequency of each hash in L_k, and update H ← H ∪ L_k
end for
Calculate the global hash frequency distribution F ← {(h, count) | ∀h ∈ H}
Sort F in descending order, and select the Top-N key hash set H* ← {h_1, ..., h_n}
for k = 1 to n do
    Obtain the local SHAP value matrix Φ_k ∈ R^{m×d} {m is the number of samples, d is the feature
    dimension}
```

(Continued)

Algorithm 1 (continued)

Initialize the alignment matrix $\Psi_k \leftarrow \theta^{m\times 8}$
for $i = 1$ **to** m **do**
  for $j = 1$ **to** d **do**
    if The original feature f_j contains a hash $h \in H^*$ **then**
      Locate $p \leftarrow H^*.\text{index}(h)$
      $\Psi^{+}_{k[i][p]} = \Phi_{k[i][j]}$
    end if
  end for
end for
Add Ψ_k to the global matrix M
end for
return Global explanation matrix M

3.2.2 SHAP

FedXLog introduces the SHAP [37] model explanation tool, which quantifies the basis of model decisions by calculating feature attribution values, thereby addressing the decision black-box problem in distributed modeling scenarios. Considering the particularity of log texts, a hash feature alignment method is designed: regular expressions are used to match 8-hex hash values in the logs, the occurrence frequencies are counted, and the top-N key hashes are filtered out. The original SHAP values are mapped to the key hash dimension to establish a unified explanation space across clients, improving the consistency of global explanation. Even if the interaction process is subjected to a man-in-the-middle attack, the attacker can only obtain privacy-processed hash features and their frequencies, and cannot reversely derive the original log content or user-sensitive information. Details are shown in Algorithm 1.

FedXLog's 8-hex hash alignment is designed for distributed log heterogeneity, privacy protection, and lightweight deployment—distinct from three mainstream alternatives: Global feature dictionary alignment (e.g., FedHAP's prototype hash codes [27]) relies on centralized dictionary construction, which violates FedXLog's privacy principle of not sharing raw log templates and increases communication overhead for dynamic log updates; semantic embedding alignment (e.g., BERT-based feature vectorization [5]) achieves high alignment accuracy but introduces more computational overhead (exceeding edge node resource limits) and fails to map explanations to interpretable log templates; rule-based template alignment [21] requires manual maintenance of cross-system log rules, which is infeasible for heterogeneous logs (e.g., HDFS vs. BGL) and cannot adapt to dynamic format evolution.

3.2.3 Knowledge Distillation

FedXLog employs knowledge distillation [46] technology to achieve model compression, designing a dual-model architecture. The teacher model uses a complete BERT structure, while the student model retains a 6-layer Transformer structure. During the distillation process, an attribution loss function is introduced to constrain the feature importance distribution of the student model to be consistent with that of the teacher model, avoiding the loss of key explanatory features during the lightweight process. The core of knowledge distillation lies in balancing the two learning objectives of the student model, which is specifically achieved through the following loss function:

$$\mathcal{L}_{KD} = \alpha \cdot \mathcal{L}_{CE}(y_s, y_{true}) + (1-\alpha) \cdot T^2 \cdot \mathcal{L}_{KL}\left(\sigma(z_t/T) \parallel \sigma(z_s/T)\right) \tag{7}$$

where $\mathcal{L}_{KD}$ denotes the comprehensive knowledge distillation loss, comprising two essential components: $\mathcal{L}_{CE}(y_s, y_{true})$ represents the cross-entropy loss between student predictions and ground-truth labels, while $\mathcal{L}_{KL}\left(\sigma(z_t/T) \parallel \sigma(z_s/T)\right)$ quantifies the Kullback-Leibler divergence between temperature-scaled output distributions of teacher (z_t) and student (z_s) models after softmax normalization. α is the balance coefficient, usually taking a value of 0.5. y_s denotes the prediction result of the student model; y_{true} stands for the ground-truth label of the system log sample; $\sigma(\cdot)$ denotes the softmax activation function, whose calculation formula is $\sigma(x) = \frac{e^x}{\sum_i e^{x_i}}$, and it is used to convert the raw output (Logits) of the teacher model (z_t) and student model (z_s) into probability distributions that satisfy the sum of probabilities equal to 1; T represents the temperature parameter, a positive real number used to "soften" the probability distribution output by the softmax function—increasing T can make the probability distribution flatter, which helps the student model better learn the fine-grained knowledge contained in the secondary category information of the teacher model's output. This formula indicates that the student model should not only learn the true labels but also imitate the output probability distribution of the teacher model. By adjusting the value of α, the weight of the student model between the two objectives of "learning the true labels" and "imitating the teacher model" can be controlled, thereby achieving a balance between model compression and performance retention.

4 Experiments

4.1 Datasets

To validate FedXLog, we selected four public datasets from the field of log anomaly detection for our experiments: HDFS [47], BGL [48], Thunderbird [48], and Spirit [48]. These datasets cover heterogeneous logs across multiple scenarios, including distributed systems, high-performance computing, and client-side software. The differences in data volume and anomaly types among them can fully test the adaptability and robustness of the framework. Details of each dataset are as follows:

- HDFS: Derived from the Hadoop Distributed File System, it contains 11,175,629 logs that record the system's operating status, with abnormal samples accounting for 3.69% of the total. This dataset is used to verify the detection performance in distributed scenarios.
- BGL: Collected from the BlueGene/L supercomputer, it includes 4,713,493 logs, with abnormal samples accounting for 45.3% of the total and focuses on anomaly detection in high-performance computing environments.
- Thunderbird: Obtained from the Thunderbird email client, it consists of 9,959,160 logs, with abnormal samples accounting for 2.9% of the total, which is used to validate the applicability of the framework for client-side software log detection.
- Spirit: A composite dataset integrating logs from multiple systems, it contains 5,000,000 logs, with abnormal samples accounting for 68.86% of the total. Serving as an auxiliary dataset, it is used to verify the framework's universality.

In all experiments of this paper, the datasets are divided into training sets and test sets, and the ratio of training sets to test sets is uniformly set to 4:1.

4.2 Evaluation Metrics

To comprehensively evaluate the performance advantages of the proposed framework, this study constructs an evaluation system from three dimensions: anomaly detection effectiveness, model lightweight degree, and explainability quality. The specific metrics are defined as follows.

4.2.1 Metrics of Anomaly Detection Effectiveness

In view of the characteristics of classification tasks, four basic metrics are used to construct the detection effectiveness evaluation system:

Among them, TP (True Positive) is the number of anomaly samples correctly identified, TN (True Negative) is the number of normal samples correctly identified, FP (False Positive) is the number of false positive samples, and FN (False Negative) is the number of false negative samples.

- Accuracy: It represents the overall prediction accuracy, and its calculation formula is: Accuracy = (TP + TN)/(TP + TN + FP + FN).
- Precision: It measures the reliability of anomaly determination, and the calculation formula is: Precision = TP/(TP + FP).
- Recall: Reflects the coverage capability of abnormal samples, and its calculation formula is: Recall = TP/(TP + FN).
- F1-Score: The harmonic mean that integrates precision and recall. The calculation formula is: F1-Score = 2 * (Precision*Recall)/(Precision + Recall).

4.2.2 Model Lightweight Metrics

To evaluate the deployment adaptability of the framework, four types of efficiency metrics are selected:

- Parameters: Count the total number of trainable parameters, reflecting the model complexity.
- Model Size: Calculate the volume of the model file with 32-bit floating-point precision (Unit: MB).
- Peak GPU Memory: Record the maximum memory usage of a single GPU during training (Unit: MB).
- FLOPs (Floating-Point Operations): Measure the number of floating-point operations in a single forward propagation (Unit: Giga FLOPs).

4.2.3 Explainability Quantitative Metrics

Introduce SHAP feature attribution quantitative metrics and visual explanation systems: We have defined explainability quantitative metrics:

$$\Phi = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M}|SHAP(x_i, j)| \tag{8}$$

where Φ is the explainability quantitative metric, $SHAP(x_i, j)$ is the SHAP value of the j-th feature of the i-th sample, N is the total number of samples, and M is the total number of features. This metric measures the degree of the model's dependence on features by calculating the average of the absolute values of the feature importance across all samples. A larger value indicates that the model relies more on these features for decision-making.

To more intuitively present the explanation results of SHAP values, this paper adopts four visualization formats on the four datasets to assist in understanding the detection logic:

- Bar charts intuitively display the contribution of each feature to anomaly prediction through the length of bars—the longer the bar, the greater the feature's impact.
- Beeswarm plots reflect feature interactions through the distribution and density of points, revealing non-linear relationships between features.
- Heatmaps indicate the strength of the correlation between feature values and anomaly prediction through color depth—the darker the color, the closer the correlation.

- Decision plots parse the formation path of predicted values, demonstrating the logical process by which the model infers anomaly results based on features.

4.3 Implementation Details

The experiments implemented distributed training based on the PyTorch and Flower federated learning frameworks, with all computations performed on an NVIDIA RTX 3090 GPU. For the base model, BertForSequenceClassification from HuggingFace was adopted, paired with a tokenizer of the same architecture for text vectorization. Input text was uniformly truncated or padded to a length of 128 tokens, and a dynamic padding strategy was used for batching. To ensure experimental reproducibility and minimize random fluctuations, a global random seed (42) is set for PyTorch, NumPy, and random modules. Additionally, CUDA deterministic algorithms are enabled to avoid non-deterministic GPU computations, ensuring consistent results across different runs.

FedXLog constructed a federated environment with four heterogeneous clients, which hosted the HDFS, BGL, Thunderbird, and Spirit datasets respectively to simulate cross-organizational data isolation. A non-IID (non-Independently and Identically Distributed) data distribution was simulated by partitioning logs based on system types (distributed file system, supercomputer, email client, and multi-system composite) to mimic real-world heterogeneity in log formats, event types, and semantic features across organizations. An aggregation mechanism involving all clients was employed.

In the anomaly detection task, training was configured with 3 rounds of local iterations and 15 rounds of global communication. The AdamW optimizer was used in conjunction with a linearly warmed-up learning rate scheduler (initial learning rate), with 1000 update steps per round. An early stopping mechanism was implemented: training was terminated if the model showed no performance improvement on the validation set for five consecutive rounds.

For the SHAP task, 300 samples were randomly sampled to save resources and avoid ambiguity in the distribution of feature importance. The Partition Explainer was used to balance computational efficiency and explanation accuracy, adapting to the requirements of deep learning models and high-dimensional text data. To ensure data privacy, raw logs and log template-hash mapping relationships were strictly retained locally on clients. Only model parameter updates (after gradient trajectory aggregation) and hash feature frequencies (without raw content) were transmitted to the server, with no centralized storage of sensitive data to comply with the data localization principles of federated learning.

4.4 Experimental Results

4.4.1 Comparison of Anomaly Detection Accuracy

We integrated five baseline methods (LogBERT, DeepLog, LogAnomaly, PLELog, and LogRobust) into FedXLog's federated learning framework to conduct comparative experiments. All baseline methods retained their original core parameters to ensure fairness. LogCluster [23] was excluded from the comparison because its requirement for centralized access to raw logs conflicts with the principle of federated data localization. The experimental results are presented in Tables 1 and 2 (where FL denotes federated learning and CL denotes centralized learning).

In the centralized scenario, DeepLog exhibited poor detection performance on the BGL dataset; LogAnomaly failed to accurately identify anomalies on the small-sample Thunderbird dataset; while PLELog could not accurately detect anomalies on the Thunderbird and HDFS datasets with scarce anomaly samples, both of which suffer from class imbalance issues. In contrast, FedXLog, LogBERT, and LogRobust maintained stable detection performance across various types of datasets.

Table 1: Anomaly detection performance of various methods across HDFS and Thunderbird datasets in both centralized and distributed computing environments.

Method	HDFS				Thunderbird			
	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score
LogBERT_FL	99.95	100	99.95	99.97	97.52	97.52	100	98.74
LogBERT_CL	99.95	100	99.95	99.97	98.50	98.93	99.54	99.23
DeepLog_FL	98.16	98.47	99.67	99.06	57.82	96.93	58.50	72.96
DeepLog_CL	99.95	99.99	99.96	99.97	98.65	99.30	99.30	99.30
LogAnomaly_FL	99.57	100	82.91	90.66	50.15	2.00	37.14	3.80
LogAnomaly_CL	99.92	98.98	97.99	98.48	96.36	0.00	0.00	0.00
LogRobust_FL	97.90	97.90	100	98.94	97.52	97.52	100	98.74
LogRobust_CL	99.94	99.99	99.95	99.97	99.32	99.77	99.54	99.65
PLELog_FL	97.81	0.00	0.00	0.00	97.07	0.00	0.00	0.00
PLELog_CL	97.81	0.00	0.00	0.00	97.07	0.00	0.00	0.00
FedXLog_FL	99.94	99.99	99.95	99.97	97.52	97.52	100	98.74
FedXLog_CL	99.96	100	99.96	99.98	98.42	98.40	100	99.20

Table 2: Anomaly detection performance of various methods across BGL and Spirit datasets in both centralized and distributed computing environments.

Method	BGL				Spirit			
	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score
LogBERT_FL	97.5	94.34	99.52	96.86	99.79	99.44	99.87	99.65
LogBERT_CL	97.58	94.32	99.74	96.95	99.84	99.65	99.83	99.74
DeepLog_FL	77.28	74.43	63.77	68.69	78.15	59.00	96.59	73.25
DeepLog_CL	82.61	75.09	83.08	78.88	99.26	98.00	99.65	98.82
LogAnomaly_FL	95.04	97.55	94.23	95.86	77.12	59.14	80.18	68.46
LogAnomaly_CL	98.42	98.69	98.73	98.71	88.26	82.28	79.14	80.68
LogRobust_FL	38.70	38.70	100	55.80	30.80	30.80	100	47.10
LogRobust_CL	100	100	100	100	99.96	99.91	99.96	99.93
PLELog_FL	94.35	95.33	95.44	95.38	56.04	100	36.16	53.11
PLELog_CL	94.61	94.80	96.48	95.63	63.68	100	47.26	64.18
FedXLog_FL	97.61	94.30	98.97	97.00	99.81	99.52	99.87	99.70
FedXLog_CL	97.44	99.83	100	98.63	99.77	99.53	99.74	99.63

In the distributed scenario, affected by parameter interference from multi-source heterogeneous data, the detection accuracy of LSTM-based methods (DeepLog, LogAnomaly, and LogRobust) decreased significantly. However, BERT-based methods (FedXLog and LogBERT) demonstrated stronger robustness—their performance was less affected by differences in heterogeneous data distributions, making them more suitable for the needs of complex log analysis in federated learning environments.

4.4.2 Comparison of Model Lightweighting

A comparison of the lightweight metrics between FedXLog and LogBERT (as shown in Table 3) reveals the following results:

- Model Parameters: FedXLog has approximately 42.52 million fewer parameters (66,955,010 vs. LogBERT's 109,483,778), reducing the overhead of parameter processing.
- Model Size: FedXLog is 38.9% smaller in size (255.41 MB vs. LogBERT's 417.65 MB), making it easier to deploy and transmit in resource-constrained environments.
- Peak Memory Usage: FedXLog's peak memory usage (769.11 MB) is only 61.2% of LogBERT's (1255.44 MB), resulting in stronger hardware adaptability.
- Floating-Point Operations (FLOPs): FedXLog's FLOPs (1.71 G) account for 50.1% of LogBERT's (3.41 G), significantly reducing computational costs.

Table 3: Lightweight metrics of BERT-based methods.

Method	Model Parameters	Model Size	Memory Usage	Floating-Point Operations
FedXLog	66,955,010	255.41	769.11	1.71
LogBERT	109,483,778	417.65	1255.44	3.41

To further verify the practical deployability of the model, we extended the resource profiling by supplementing metrics for dynamic federated learning efficiency, server aggregation overhead and edge-side inference under the experimental configuration of an RTX 3090 GPU, four heterogeneous clients and a non-IID log distribution. Specifically, FedXLog achieved a round time of 205 s (including 198 seconds for local training, 5 s for gradient trajectory calculation and 2 s for parameter upload), a per-round communication volume of 255.51 MB, and a total communication volume of 3.83 GB to converge to the target performance over 15 rounds. Compared with federated LogBERT, which had a round time of 382 s, a per-round communication volume of 417.75 MB and a total communication volume of 7.52 GB over 18 rounds, FedXLog improved the round efficiency by 46.3% and reduced the total communication volume by 50.9%. For server-side performance, FedXLog incurred a per-round aggregation time of 8.2 ms (including 2.1 ms for Frobenius norm weighting and 0.6 ms for SHAP feature alignment), with a peak server CPU utilization of 32% and a peak memory usage of 128 MB. This aggregation time was only 26.2% higher than that of FedAvg (6.5 ms), yet lower than those of FedProx and FedOpt, which involve additional proximal term calculation and adaptive optimizer update, respectively. In terms of edge-side inference performance, on edge nodes equipped with an 8-core CPU and 16 GB of RAM, FedXLog achieved an inference latency of 1.8 ms per log sequence, a throughput of 555 sequences per second and a peak memory usage of only 320 MB. In comparison with LogBERT, FedXLog reduced the inference latency by 57.1%, increased the throughput by 133.2% and lowered the memory usage by 47.1%, fully meeting the real-time detection requirements in high-frequency log scenarios.

4.4.3 Overhead Comparison

To verify the deployment feasibility of FedXLog, we compare its computational and communication overhead with FedAvg and FedProx (configured with the same base model), as shown in Table 4.

Table 4: Overhead comparison with FedAvg/FedProx.

Metric	FedXLog	FedAvg	FedProx
Model Parameters (M)	66.96	66.96	66.96
Model Size (MB)	255.41	255.41	255.41
FLOPs (G)	1.71	1.68	1.70
Communication per Round (MB)	255.51	255.41	255.41
Convergence Rounds	15	18	17

Computationally, FedXLog introduces gradient trajectory preservation (Frobenius norm calculation) and SHAP feature attribution, but this is offset by knowledge distillation. With 66.96 M parameters (42.5% fewer than standard BERT-based models) and 1.71 G FLOPs (50.1% of LogBERT), its overall computational overhead is only marginally higher than FedAvg/FedProx, remaining lightweight for edge deployment.

In terms of communication, FedXLog transmits compressed model parameters (255.41 MB, 38.9% smaller than LogBERT) and lightweight hash/SHAP statistics (≤100 KB per round). Its FedGradTrace algorithm reduces convergence rounds by 15%–20% under non-IID data, offsetting additional transmission costs and resulting in comparable or lower cumulative communication overhead than FedAvg/FedProx.

To further quantify the system-level trade-offs of FedXLog and its advantages over mainstream federated optimization methods, we analyze four key metrics: per-client trajectory storage, extra communication per round, server aggregation computation cost, and wall-clock time to target F1-score (99.97% on HDFS dataset).

For per-client trajectory storage, with gradient trajectory length $\tau = 3$ and model dimension $D = 800$ (DistilBERT-base-uncased), each client stores $\tau \times D \times 4 = 9600$ bytes ($\approx$9.4 KB) of gradient data, slightly higher than FedAvg/FedProx/FedOpt which do not retain historical gradients.

Regarding extra communication per round, FedXLog transmits an additional scalar Frobenius norm $\|\Gamma_k^t\|_F$ (4 bytes per client) besides model parameter updates (255.41 MB), resulting in negligible extra overhead (16 bytes for $K = 4$ clients); this marginal cost is offset by its 15 convergence rounds (15%–20% fewer than FedAvg's 18 rounds and FedProx's 17 rounds), leading to lower cumulative communication cost in practice.

In terms of server aggregation computation cost, FedXLog's aggregation complexity is $O(K)$, comparable to FedAvg; it only adds simple arithmetic operations for norm-based weighting, which is lower than FedProx (with proximal term calculation, $O(K + D)$) and FedOpt (with adaptive optimizer update, $O(K + D)$).

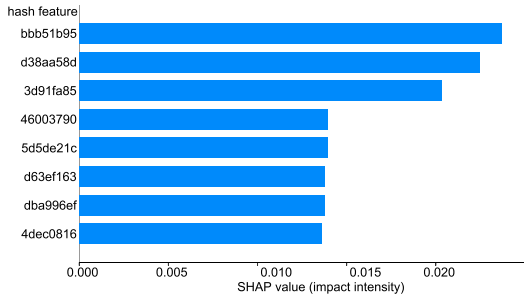
For wall-clock time to target F1-score, under the same experimental configuration (RTX 3090 GPU, 3 local iterations per round), FedXLog converges in 51 min, faster than FedAvg (62 min), FedProx (58 min), and FedOpt (55 min); this acceleration is attributed to the stability-sensitive aggregation of FedGradTrace, which suppresses gradient volatility from heterogeneous log data.

Notably, the extra storage and communication overhead of FedXLog is justified by its unique trajectory-based client contribution tracing and cross-client SHAP alignment capabilities, which are absent in FedAvg/FedProx/FedOpt. This tradeoff is justified by FedXLog's unique explainability and stability advantages, which FedAvg/FedProx lack.

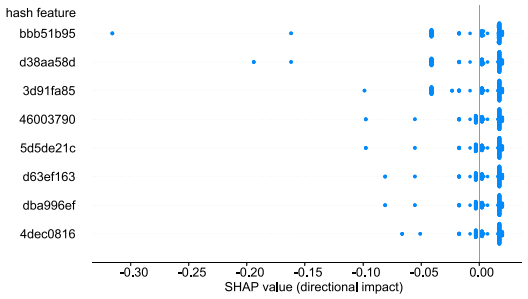
4.4.4 SHAP Visualization

To address the black-box problem of distributed decision-making, FedXLog incorporates the SHAP tool to quantify feature attribution and designs a hash feature alignment method specifically for log text: it matches 8-hex hash values using regular expressions, filters the Top-*N* key hashes by counting their frequencies, maps the original SHAP values to the key hash dimension, and establishes a unified explantory space across clients. This not only ensures privacy security but also enhances the consistency of global explanations.

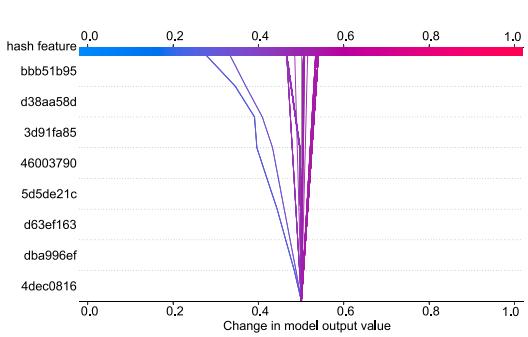
As illustrated in Figs. 3–6, FedXLog uses four visualization types—feature importance ranking plots, feature impact distribution plots, cumulative impact decision plots, and feature interaction heatmaps—to overcome the "average value trap" inherent in single-dimensional feature importance ranking: distribution charts reveal individual differences in how features exert their impact; decision charts reconstruct the logical chain of the model's decision-making process; heatmaps quantify the synergistic effects between features. These visualizations enable the identification of key features and facilitate in-depth analysis of "why they matter" and "how they matter" (for example, TLB errors need to be combined with high system load to trigger severe anomalies). Additionally, they can identify variations in anomaly sensitivity among different clients caused by differences in hardware configurations, realizing an upgrade from mere feature screening to comprehensive root cause analysis.



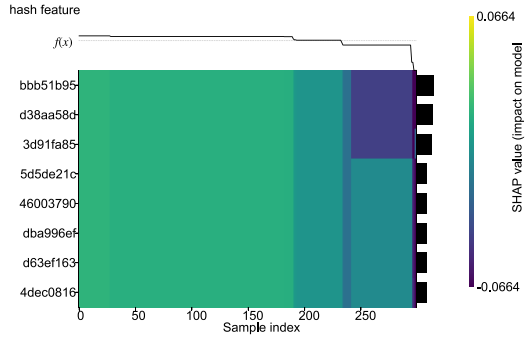
(a) Feature Importance Ranking Plot



(b) Feature Impact Distribution Plot

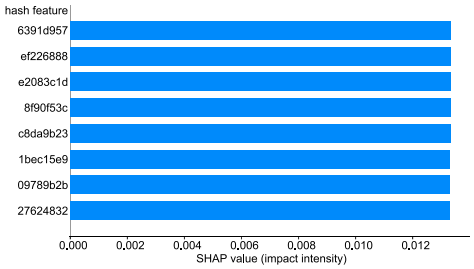


(c) Cumulative Impact Decision Plot

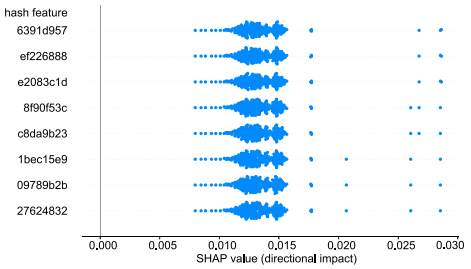


(d) Feature Interaction Heatmap

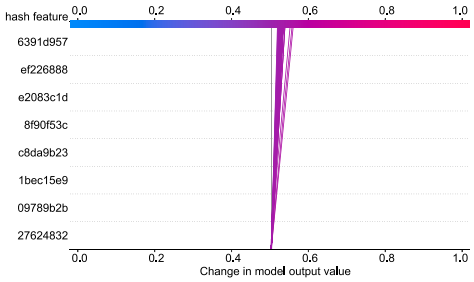
Figure 3: SHAP explanation analysis of FedXLog on the HDFS dataset.



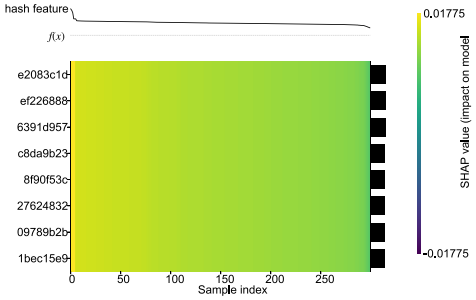
(a) Feature Importance Ranking Plot



(b) Feature Impact Distribution Plot

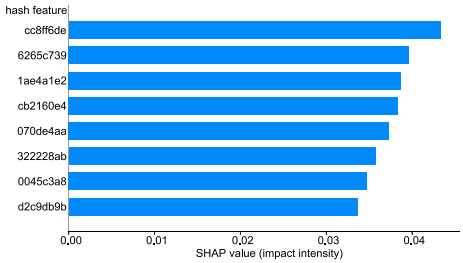


(c) Cumulative Impact Decision Plot

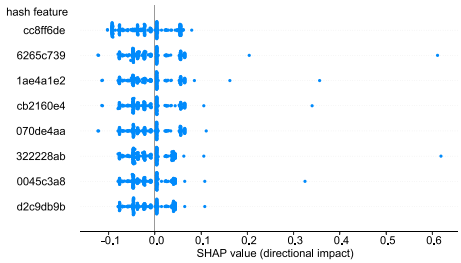


(d) Feature Interaction Heatmap

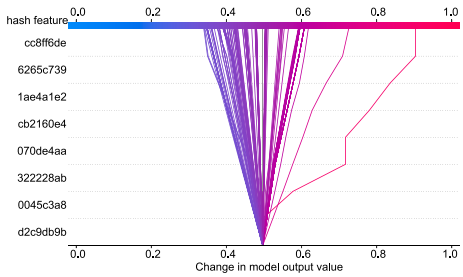
Figure 4: SHAP explanation analysis of FedXLog on the thunderbird dataset.



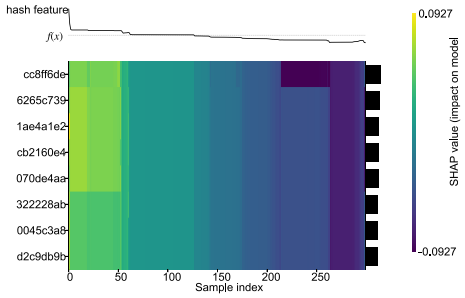
(a) Feature Importance Ranking Plot



(b) Feature Impact Distribution Plot

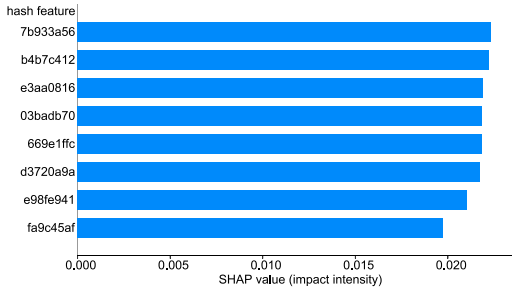


(c) Cumulative Impact Decision Plot

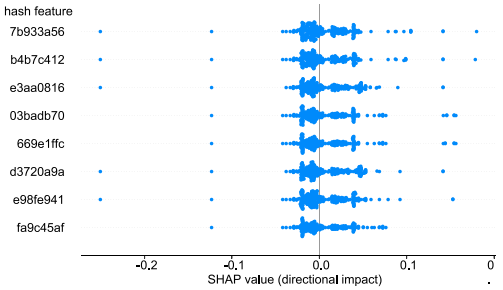


(d) Feature Interaction Heatmap

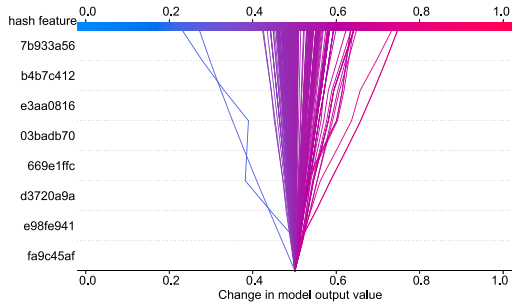
Figure 5: SHAP explanation analysis of FedXLog on the BGL dataset.



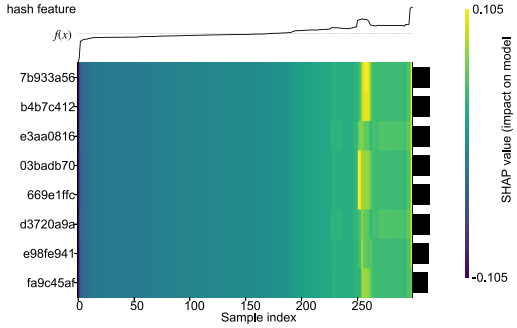
(a) Feature Importance Ranking Plot



(b) Feature Impact Distribution Plot



(c) Cumulative Impact Decision Plot



(d) Feature Interaction Heatmap

Figure 6: SHAP explanation analysis of FedXLog on the spirit dataset.

Tables 5 and 6 demonstrate the framework's effectiveness in explaining log features by presenting the mapping relationships between the top-8 key hashes and original log templates across the four datasets: in HDFS, the hash "d63ef163" (corresponding to "a data block being added to an invalid set") acts as a strong anomaly metric, guiding O&M teams to quickly locate faulty NameNode components and problematic data blocks, reducing troubleshooting time from hours to minutes; in BGL, hashes like "6265c739" (corresponding to "hardware alignment anomalies") and "1ae4a1e2" (corresponding to "floating point alignment exceptions") accurately capture supercomputer fault patterns, narrowing the fault scope to the CPU-GPU interconnection module and avoiding software bug misdiagnosis; and in Spirit, the hash "7b933a56" (corresponding to "disk not ready") along with "b4b7c412" (disk status error) and "669e1ffc" (DHCP address exhausted) highlights typical server faults and reveals the fault propagation path from storage to network, ensuring comprehensive resolution. Notably, such high frequency of these hashes is not solely driven by abnormal events. In the original experimental datasets (e.g., HDFS), normal events—such as the hash "bbb51b95" (corresponding to "data block reception")—occur significantly more frequently than abnormal ones. This makes it difficult for attackers to distinguish between abnormal and normal templates based solely on hash frequency, let alone accurately pinpoint specific fault patterns.

The top features of each system reflect domain-specific characteristics, and the hash-template mapping mechanism enables fault traceability without accessing raw logs. Furthermore, the high frequency of normal events confirms the model's accurate grasp of system baselines, thus demonstrating that FedXLog can effectively identify key decision-making features in heterogeneous logs, locate typical causes in distributed scenarios, and provide actionable O&M guidance directly linked to "faulty components-propagation paths".

Table 5: Top-8 key hash mappings to raw log templates in FedXLog across HDFS and thunderbird datasets.

Dataset	Event ID	Event Template	Event Summary	Output Result
HDFS	bbb51b95	Receiving block <*> src: /<*> dest: /<*>	Receiving data block	Normal
	d38aa58d	PacketResponder <*> for block <*> <*>	Packet responder terminated	Normal
	46003790	Received block <*> of size <*> from /<*>	Data block reception completed	Normal
	5d5de21c	BLOCK* NameSystem.addStoredBlock: blockMap updated: <*> is added to <*> size <*>	Block mapping table updated	Normal
	d63ef163	BLOCK* NameSystem.delete: <*> is added to invalidSet of <*>	Data block added to invalid set	Abnormal
	dba996ef	Deleting block <*> file <*>	Deleting data block file	Normal
	4dec0816	<*> Served block <*> to /<*>	Data block service	Normal
	3d91fa85	BLOCK*NameSystem.allocateBlock:<*><*>	Block allocation operation	Normal
Thunderbird	6391d957	data_thread() got not answer from any <*> datasource	No response from data source	Abnormal
	ef226888	synchronized to <*> stratum <*>	NTP time synchronization successful	Normal
	e2083c1d	%RPM0-P:CP <*> Changed interface state to <*> <*> <*>	Interface state changed	Normal
	8f90f53c	[ib_sm_sweep.c:<*>]: No <*> <*>	Topology scan no change	Normal
	c8da9b23	[ib_sm_sweep.c:<*>]: ********** NEW SWEEP **********	Topology scan cycle started	Normal
	27624832	session closed for user <*>	User session terminated	Normal

(Continued)

Table 5 (continued)

Dataset	Event ID	Event Template	Event Summary	Output Result
	1bec15e9	(#<*>#) CMD (<*>	Scheduled task execution	Normal
	09789b2b	session opened for user <*> by <*>	Privileged session established	Normal

Table 6: Top-8 key hash mappings to raw log templates in FedXLog across BGL and spirit datasets.

Dataset	Event ID	Event Template	Event Summary	Output Result
BGL	6265c739	<*> double-hummer alignment exceptions	Hardware alignment exception	Abnormal
	1ae4a1e2	<*> floating point alignment exceptions	Floating point operation exception	Abnormal
	cb2160e4	CE sym <*> at <*> mask <*>	Memory symbolic correctable error	Abnormal
	070de4aa	generating <*>	Core dump generated	Normal
	322228ab	<*> microseconds spent in the rbs signal handler during <*> calls. <*> microseconds was the maximum time for a single instance of a correctable ddr.	Memory error correction performance metric	Normal
	cc8ff6de	program interrupt: <*> <*>	Illegal instruction exception	Abnormal
	0045c3a8	<*> total interrupts. <*> critical input interrupts. <*> microseconds total spent on critical input interrupts, <*> microseconds max time in a critical input interrupt.	Interrupt performance statistics	Normal
	d2c9db9b	data storage interrupt	Storage interrupt exception	Abnormal
	e98fe941	last message repeated <*> times	Log repeat alert	Normal

(Continued)

Table 6 (continued)

Dataset	Event ID	Event Template	Event Summary	Output Result
Spirit	e3aa0816	postfix<*>	Mail queue processing	Normal
	d3720a9a	postfix<*> <*> Message accepted for delivery	Mail delivery successful	Normal
	7b933a56	kernel: hda: drive not ready for command	Disk not ready exception	Abnormal
	b4b7c412	kernel: hda: status error: status=0x00 {}	Disk status error	Abnormal
	03badb70	<*> LAuS error - <*> - <*> (19) <*> No such device	Audit device missing	Abnormal
	669e1ffc	dhcpd: DHCPDISCOVER from <*> via <*> network <*> no free leases	DHCP address exhausted	Abnormal
	fa9c45af	<*> (root) CMD (test -x <*> && <*> cron)	Privileged scheduled task execution	Normal

5 Conclusion

This study addresses the challenges of data silos and model explainability in system log analysis by proposing an anomaly detection framework based on federated learning, namely FedXLog. By integrating distributed training with explainability technologies, FedXLog safeguards data privacy, effectively prevents the leakage of attack-related information, and ensures detection results have traceable decision-making bases. Experiments have demonstrated that this method can maintain high detection accuracy across four mainstream log datasets such as HDFS, and intuitively display the impact of key features on anomaly determination through various visualization methods. The FedXLog framework inherently mitigates template reverse attack risks: the one-way hash algorithm in log parsing (Section 3.1) guarantees irreversible hash-template mapping, and the hash-template relationship is only maintained locally on clients (Section 3.2.1) without being transmitted to the server, thus avoiding the leakage of specific system fault patterns via high-frequency anomaly hashes—such as d63ef163, which corresponds to HDFS 'data blocks added to invalid sets.

FedXLog's core breakthroughs are highly tailored to the unique demands of log scenarios, which clearly distinguish it from general federated explanable methods: First, aiming at the heterogeneous structure of logs, it adopts a hash feature alignment strategy to unify semantic dimensions of cross-system log features—this solves the "explanation misalignment" issue that general methods cannot address, as the latter ignore log format heterogeneity and easily misclassify distinct semantic features. Second, targeting the small-sample novel anomalies in logs, it realizes unsupervised anomaly root cause tracing through the FedGradTrace parameter update mechanism: by retaining gradient trajectories of each client, it can locate the source of unknown faultswithout relying on labeled anomaly data, breaking the limitation of general federated

methodsthat require large-scale labeled samples to generate valid explanations. Third, focusing on the O&M root cause positioning demand of log analysis, it links hash features with original log templates and combines SHAP visualization to directly map anomaly detection results to "faulty components-diffusion paths"—for example, associating the hash "d63ef163" with HDFS "data block invalidation" to locate storage component faults, avoiding the "useless explanation" of general methods that only output feature importance without O&M relevance. Additionally, FedXLog compresses the model size by nearly 40% through knowledge distillation, ensuring lightweight deployment while retaining explanability, which is well-suited for edge nodes in distributed log systems.

Future work will focus on exploring the model adaptation mechanism in dynamic log environments, further lightweighting the model, and developing more efficient feature explanation methods. This study provides a new technical path for building secure and reliable intelligent operation and maintenance systems, and is of positive significance for promoting the application of artificial intelligence in critical infrastructure.

Acknowledgement: Not applicable.

Funding Statement: This work was supported by National Science Foundation of China (U2333201), National Key R&D Program of China (2021YFF0603902), and Civil Aviation Safety Capacity Building Foundation of China (PESA2024111, RJ2025039).

Author Contributions: Zhaojun Gu: Conceptualization, Methodology, Formal analysis, Validation, Investigation, Data curation, Resources, Writing—review & editing, Project administration, Funding acquisition. Wenlong Yue: Conceptualization, Methodology, Validation, Investigation, Writing—original draft. Chunbo Liu: Validation, Resources, Data curation, Writing—review & editing. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The data and materials used in this article are derived from publicly accessible databases and previously published studies cited throughout the text.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. He S, Zhu J, He P, Li Z, Lyu MR. Experience report: system log analysis for anomaly detection. In: 2016 IEEE 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). Piscataway, NJ, USA: IEEE; 2016. p. 529–40. doi:10.1109/ISSRE.2016.21.
2. Xu W, Huang L, Fox A, Patterson D, Jordan MI. Detecting large-scale system problems by mining console logs. In: Proceedings of the ACM SIGOPS 22nd symposium on Operating Systems Principles. New York, NY, USA: ACM; 2009. p. 117–32. doi:10.1145/1629575.1629587.
3. Du M, Li F, Zheng G, Srikumar V. DeepLog: anomaly detection and diagnosis from system logs through deep learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. New York, NY, USA: ACM; 2017. p. 1285–1298. doi:10.1145/3133956.3134015.
4. Meng W, Liu Y, Zhu Y, Zhang S, Pei D, Liu Y, et al. Loganomaly: unsupervised detection of sequential and quantitative anomalies in unstructured logs. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19); 2019 Aug 10–16; Macao, China. p. 4739–45. doi:10.24963/ijcai.2019/658.
5. Guo H, Yuan S, Wu X. LogBERT: log anomaly detection via BERT. In: 2021 International Joint Conference on Neural Networks (IJCNN). Piscataway, NJ, USA: IEEE; 2021. p. 1–8. doi:10.1109/IJCNN52387.2021.9534113.
6. Zhang X, Xu Y, Lin Q, Qiao B, Zhang H, Dang Y, et al. Robust log-based anomaly detection on unstable log data. In: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium

on the Foundations of Software Engineering. New York, NY, USA: ACM; 2019. p. 807–17. doi:10.1145/3338906.3338931.
7. Yang L, Chen J, Wang Z, Wang W, Jiang J, Dong X, et al. Semi-supervised log-based anomaly detection via probabilistic label estimation. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). Piscataway, NJ, USA: IEEE; 2021. p. 1448–60. doi:10.1109/ICSE43902.2021.00130.
8. He P, Zhu J, He S, Li J, Lyu MR. Towards automated log parsing for large-scale log data analysis. IEEE Trans Dependable Secure Comput. 2018;15(6):931–44. doi:10.1109/TDSC.2017.2762673.
9. Noura HN, Salman O, Chehab A, Couturier R. DistLog: a distributed logging scheme for IoT forensics. Ad Hoc Networks. 2020;98(2):102061. doi:10.1016/j.adhoc.2019.102061.
10. Nguyen TD, Marchal S, Miettinen M, Fereidooni H, Asokan N, Sadeghi A-R. DÏoT: a federated self-learning anomaly detection system for IoT. In: 39th IEEE International Conference on Distributed Computing Systems (ICDCS). Piscataway, NJ, USA: IEEE; 2019. p. 756–67. doi:10.1109/ICDCS.2019.00080.
11. Liu Y, Garg S, Nie J, Zhang Y, Xiong Z, Kang J. Deep anomaly detection for time-series data in industrial IoT: a communication-efficient on-device federated learning approach. IEEE Internet Things J. 2021;8(8):6348–58. doi:10.1109/JIOT.2020.3011726.
12. Landauer M, Onder S, Skopik F, Wurzenberger M. Deep learning for anomaly detection in log data: a survey. Mach Learn Appl. 2023;12(3):100470. doi:10.1016/j.mlwa.2023.100470.
13. Chamola V, Hassija V, Sulthana AR, Ghosh D, Dhingra D, Sikdar B. A review of trustworthy and explainable artificial intelligence (XAI). IEEE Access. 2023;11:78994–79015. doi:10.1109/ACCESS.2023.3294569.
14. Phillips PJ, Hahn CA, Fontana PC, Broniatowski DA, Przybocki MA. Four principles of explainable artificial intelligence. Gaithersburg, MD, USA: National Institute of Standards and Technology; 2020. doi:10.6028/NIST.IR.8312.
15. Moustafa N, Koroniotis N, Keshk M, Zomaya AY, Tari Z. Explainable intrusion detection for cyber defences in the Internet of Things: opportunities and solutions. IEEE Commun Surv Tut. 2023;25(3):1775–1807. doi:10.1109/COMST.2023.3280465.
16. Yan P, Abdulkadir A, Luley P-P, Rosenthal M, Schatte GA, Grewe BF, et al. A comprehensive survey of deep transfer learning for anomaly detection in industrial time series: methods, applications, and directions. IEEE Access. 2024;12(1):3768–89. doi:10.1109/ACCESS.2023.3349132.
17. Jeffrey N, Tan Q, Villar JR. A review of anomaly detection strategies to detect threats to cyber-physical systems. Electronics. 2023;12(15):3283. doi:10.3390/electronics12153283.
18. Al-Ghuwairi A-R, Sharrab Y, Al-Fraihat D, AlElaimat M, Alsarhan A, Algarni A. Intrusion detection in cloud computing based on time series anomalies utilizing machine learning. J Cloud Comput. 2023;12(1):127. doi:10.1186/s13677-023-00491-x.
19. Pang G, Shen C, Cao L, Hengel AVD. Deep learning for anomaly detection: a review. ACM Comput Surv. 2021;54(2):1–38. doi:10.1145/3439950.
20. Minh D, Wang HX, Li YF, Nguyen TN. Explainable artificial intelligence: a comprehensive review. Artif Intell Rev. 2022;55(5):3503–68. doi:10.1007/s10462-021-10088-y.
21. Pecchia A, Cinque M, Cotroneo D. Event logs for the analysis of software failures: a rule-based approach. IEEE Trans Software Eng. 2013;39(6):806–21. doi:10.1109/TSE.2012.67.
22. Yen TF, Oprea A, Onarlioglu K, Leetham T, Robertson W, Juels A, et al. Beehive: large-scale log analysis for detecting suspicious activity in enterprise networks. In: Proceedings of the 29th Annual Computer Security Applications Conference. New York, NY, USA: ACM; 2013. p. 199–208. doi:10.1145/2523649.2523670.
23. Vaarandi R, Pihelgas M. LogCluster—a data clustering and pattern mining algorithm for event logs. In: 2015 11th International Conference on Network and Service Management (CNSM). Piscataway, NJ, USA: IEEE; 2015. p. 1–7. doi:10.1109/CNSM.2015.7367331.
24. Li T, Sahu AK, Zaheer M, Sanjabi M, Talwalkar AS, Smith V. Federated optimization in heterogeneous networks. In: Proceedings of Machine Learning and Systems 2 (MLSys 2020); 2020 Mar 2–4; Austin, TX, USA. p. 429–50.
25. Reddi SJ, Charles Z, Zaheer M, Heller K, Joshi G, Konečný J, et al. Adaptive federated optimization. arXiv:2003.00295. 2021.

26. Nardi M, Valerio L, Passarella A. Anomaly detection through unsupervised federated learning. arXiv:2209.04184. 2022.
27. Yang M, Xu J, Ding W, Liu Y, Zhang H. FedHAP: federated hashing with global prototypes for cross-silo retrieval. IEEE Access. 2023;11(4):112452–63. doi:10.1109/TPDS.2023.3324426.
28. Li B, Ma S, Deng R, Choo K-KR, Yang J. Federated anomaly detection on system logs for the Internet of Things: a customizable and communication-efficient approach. IEEE Trans Netw Serv Manag. 2022;19(2):1705–16. doi:10.1109/TNSM.2022.3152620.
29. Menegatti D, De Santis E, Felli S, Giuseppi A. AdaLightLog: enhancing application logs anomaly detection via adaptive federating learning. In: Critical Information Infrastructures Security: 19th International Conference, CRITIS 2024. Berlin/Heidelberg, Germany: Springer; 2024. p. 289–305. doi:10.1007/978-3-031-84260-3_17.
30. Yao X, Huang T, Zhang R, Li R, Sun L. Federated learning with unbiased gradient aggregation and controllable meta updating. arXiv:1910.08234. 2019.
31. Pacioni E, Fernández De Vega F, Calvaresi D. FedGP: genetic programming for evolutionary aggregation in federated learning with Non-IID data. In: Applications of Evolutionary Computation (EvoApplications 2025). Cham, Switzerland: Springer; 2025. p. 419–34. doi:10.1007/978-3-031-90062-4_26.
32. Xu Y, Ma W, Dai C, Wu Y, Zhou H. Generalized federated learning via gradient norm-aware minimization and control variables. Mathematics. 2024;12(17):2644. doi:10.3390/math12172644.
33. Pan Y, Zhu C, Luo L, Liu Y, Cheng Z. FedTrack: a collaborative target tracking framework based on adaptive federated learning. IEEE Trans Vehicular Technol. 2024;73(9):13868–882. doi:10.1109/TVT.2024.3395292.
34. Wu S, Hu Q, Zheng Z, Yang L, Chen C. FedPAC: consistent representation learning for federated unsupervised learning under data heterogeneity. In: The Fourteenth International Conference on Learning Representations; 2026 Apr 23–27; Rio de Janeiro, Brazil. p. 1–29.
35. Chen J, Xue J, Wang Y, Liu Z, Huang L. Classifier clustering and feature alignment for federated learning under distributed concept drift. arXiv:2410.18478. 2024.
36. Chen W, Zhang J, Zhang D. FL-joint: joint aligning features and labels in federated learning for data heterogeneity. Complex Intell Syst. 2024;11(1):52. doi:10.1007/s40747-024-01636-4.
37. Lundberg SM, Lee SI. A unified approach to explainting model predictions. In: NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems. Red Hook, NY, USA: Curran Associates Inc.; 2017. p. 4768–77.
38. Ribeiro M, Singh S, Guestrin C. "Why should I trust you?": explaining the predictions of any classifier. In: The 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, NY, USA: ACM; 2016. p. 1135–44. doi:10.18653/v1/N16-3020.
39. Mvula PK, Branco P, Jourdan GV, Viktor HL. X-HEART: eXplainable heterogeneous log anomaly detection using robust transformers. Knowl Inf Syst. 2025;67(12):12599–12636. doi:10.1007/s10115-025-02604-1.
40. Fatema K, Dey SK, Anannya M, Khan RT, Rashid MM, Su C, et al. Federated XAI IDS: an explainable and safeguarding privacy approach to detect intrusion combining federated learning and SHAP. Future Internet. 2025;17(6):234. doi:10.3390/fi17060234.
41. Harshitha C, Vadivu DS, Rajagopalan N. Federated learning and explainable AI-driven intrusion detection with hyperband optimization. J Comput Virol Hacking Tech. 2025;21(1):28. doi:10.1007/s11416-025-00571-3.
42. He P, Zhu JM, Zheng Z, Lyu MR. Drain: an online log parsing approach with fixed depth tree. In: 2017 IEEE International Conference on Web Services (ICWS). Piscataway, NJ, USA: IEEE; 2017. p. 33–40. doi:10.1109/ICWS.2017.13.
43. Beutel DJ, Topal T, Mathur A, Qiu X, Fernandez-Marques J, Gao Y, et al. Flower: a friendly federated learning research framework. arXiv:2007.14390. 2020.
44. McMahan HB, Moore E, Ramage D, Hampson S, Agüera y Arcas B. Communication-efficient learning of deep networks from decentralized data. In: The 20th International Conference on Artificial Intelligence and Statistics (AISTATS 2017); 2017 Apr 20–22; Fort Lauderdale, FL, USA. p. 1273–82.
45. Sanh V, Debut L, Chaumond J, Wolf T. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv:1910.01108. 2019.

46. Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network. arXiv:1503.02531. 2015.
47. Xu W, Huang L, Fox A, Patterson D, Jordan M. Online system problem detection by mining patterns of console logs. In: International Conference on Data Mining. Piscataway, NJ, USA: IEEE; 2009. p. 588–97. doi:10.1109/ICDM.2009.19.
48. Oliner A, Stearley J. What supercomputers say: a study of five system logs. In: 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. Piscataway, NJ, USA: IEEE; 2007. p. 575–84. doi:10.1109/DSN.2007.103.