The Journal of Systems and Software 235 (2026) 112781

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

LogMeta: A few-shot model-agnostic meta-learning framework for robust and adaptive log anomaly detection

Yicheng Sun *, Jacky Wai Keung, Hi Kuen Yu, Wenqiang Luo

Department of Computer Science, City University of Hong Kong, Hong Kong, China

ARTICLE INFO

Editor: Dr. Gabriele Bavota

Keywords:
Empirical software engineering
Software log analysis
Log anomaly detection
Model-agnostic meta-Learning
Hybrid language model

ABSTRACT

Context: Log anomaly detection is critical for maintaining the security, stability, and operational efficiency of modern software systems, especially as they generate vast and diverse log data. However, existing deep learning models struggle with the challenges of heterogeneous log formats across systems and the scarcity of labeled anomaly logs, limiting their real-world deployment and generalization capabilities.

Objective: To address these challenges, we propose LogMeta, a novel semi-supervised framework designed for adaptive and efficient log anomaly detection in diverse and low-resource environments.

Method: LogMeta integrates Model-Agnostic Meta-Learning (MAML) with a hybrid language model to address key challenges. MAML enables LogMeta to rapidly adapt to unseen log systems using few-shot samples, while the hybrid model combines RoBERTa for extracting semantic representations with Bi-LSTM and attention mechanisms to capture sequential dependencies and critical features within log sequences. This design reduces reliance on large-scale labeled datasets and enhances adaptability in heterogeneous environments.

Results: Experimental evaluations on multiple benchmark datasets demonstrate that LogMeta consistently outperforms state-of-the-art supervised and unsupervised methods, achieving up to a 28.3% improvement in F1-scores under low-resource scenarios compared to other models. Furthermore, LogMeta exhibits exceptional domain transfer capabilities, maintaining robust performance across diverse log datasets with minimal fine-tuning. In terms of efficiency, LogMeta achieves competitive training and inference times, making it suitable for real-time anomaly detection in large-scale systems.

Conclusion: LogMeta provides a scalable and practical solution for real-world log anomaly detection, overcoming challenges related to data heterogeneity and label scarcity. Its strong generalization capabilities, minimal supervision requirements, and adaptability to new log systems make it a promising tool for enhancing software system reliability and security.

1. Introduction

In modern computing environments, logs serve as critical artifacts for system monitoring, debugging, and performance optimization (Pirelli, 2024; He et al., 2016; Lou et al., 2010). These logs are typically generated by a wide range of software and hardware systems, capturing key operational details and events (Le and Zhang, 2022). However, given the increasing complexity and scale of systems, log anomaly detection has emerged as an essential technique for identifying irregularities that may signal critical failures, security breaches, or performance bottlenecks (Dai et al., 2020; Wu et al., 2023; Xu et al., 2024). Accurate and timely detection of anomalies in logs is crucial for maintaining system reliability and ensuring operational efficiency.

Despite its critical role, log anomaly detection poses significant challenges due to the inherent characteristics of log data. A primary challenge is the diversity and variability of logs across different systems (Huang et al., 2025; Sun et al., 2025c). While logs from systems such as the Hadoop Distributed File System (HDFS) and Blue Gene/L (BGL) exhibit common patterns, they also contain system-specific differences in format, structure, and content. For instance, HDFS logs primarily capture file system-related operations, whereas BGL logs are tailored to supercomputing activities. Table 1 provides an overview of the structural differences between logs from various software systems, underscoring the challenges of generalizing anomaly detection models across heterogeneous environments. Effectively capturing the shared patterns across systems while accounting for their unique characteristics is

* Corresponding author.
E-mail addresses: yicsun2-c@my.cityu.edu.hk (Y. Sun), Jacky.Keung@cityu.edu.hk (J.W. Keung), hikuenyu2-c@my.cityu.edu.hk (H.K. Yu), wenqialuo4-c@my.cityu.edu.hk (W. Luo).

https://doi.org/10.1016/j.jss.2026.112781
Received 18 January 2025; Received in revised form 2 January 2026; Accepted 6 January 2026
Available online 8 January 2026

Table 1
Log structural differences between various software systems.

System	Log Type	Description	Key Structural Characteristics
HDFS	Distributed file system	Tracks file system operations, e.g., block creation, deletion	Semi-structured; includes block IDs (e.g., blk_3777400576053320362), timestamps, operation types
BGL	Supercomputer	Records hardware and system-level errors in supercomputers	Unstructured; contains timestamps, node IDs, error codes
Thunderbird	Supercomputer	Logs hardware failures, software errors, and runtime events in a supercomputer environment	Semi-structured; includes timestamps, node IDs, job IDs, and error codes
Spirit	Supercomputer	Aggregates system logs from the Spirit supercomputer, capturing hardware/software anomalies	Semi-structured; includes timestamps, node IDs, event types, and error details
OpenStack	Infrastructure	Monitors cloud orchestration and virtualized resource management	Semi-structured; includes service names, IP addresses, resource states
Apache	Web server	Logs HTTP requests, server activity, and client interactions	Semi-structured; includes IP addresses, HTTP methods, response codes
Windows	Operating system event	Tracks system-level events such as user logins, application errors, and updates	Structured; contains event IDs, timestamps, user IDs
Android	Mobile system	Captures application activities, system messages, and crash reports	Unstructured; includes process IDs, timestamps, stack traces

essential for developing a robust and generalized anomaly detection model (Sun et al., 2025b; Yu et al., 2024). Such a model would enable accurate anomaly detection across diverse environments while reducing reliance on system-specific configurations.

Another critical challenge is the limited availability of labeled log data, particularly for anomalous logs. Labeling logs demands significant manual effort from domain experts, which is both time-intensive and resource-consuming (Jiang et al., 2024a). Moreover, anomalous logs are inherently rare, making it challenging to train conventional supervised models effectively (Wu et al., 2023; Sun et al., 2025c). This scarcity highlights the need for models capable of achieving high performance while learning from minimal labeled data. In addition, models designed for log anomaly detection must address the significant imbalance between normal and anomalous logs, requiring robust and adaptive strategies to accurately identify sparse anomalies under limited supervision.

Existing approaches to log anomaly detection can be categorized into supervised (Zhang et al., 2019; Huang et al., 2020; Le and Zhang, 2021), semi-supervised (Sun et al., 2025b; Lin and Chiang, 2022; Yang et al., 2021), and unsupervised (Du et al., 2017; Zeufack et al., 2021) models. Supervised models achieve high accuracy when ample labeled data is available, but their reliance on large-scale annotations limits their practicality in real-world scenarios. On the other hand, unsupervised models do not require labeled data but often struggle with false positives and lack the ability to leverage domain-specific knowledge (Le and Zhang, 2022). Semi-supervised models, which combine the strengths of supervised and unsupervised learning, provide a promising alternative by utilizing a small amount of labeled data to guide the detection process while leveraging abundant unlabeled data (Sun et al., 2025a). This hybrid approach strikes a balance between performance and practicality, making semi-supervised learning particularly suitable for log anomaly detection.

To address these challenges, we propose **LogMeta**, a novel semi-supervised framework for **log** anomaly detection that integrates a hybrid model and leverages few-shot **meta**-learning techniques to achieve adaptability and efficiency. LogMeta leverages the power of Model-Agnostic Meta-Learning (MAML) to overcome the limitations of traditional approaches. Meta-learning, also known as "learning to learn," focuses on training models that can adapt rapidly to new tasks with limited data. LogMeta can quickly adapt to new systems and detect anomalies even under constrained labeling conditions by learning general-purpose initialisation parameters from diverse log datasets. This adaptability allows LogMeta to generalize effectively across heterogeneous log formats, ensuring high anomaly detection accuracy despite the challenges posed by limited labeled data and diverse system characteristics.

Furthermore, LogMeta integrates the strengths of meta-learning with a hybrid language model, leveraging the contextual understanding of RoBERTa and the sequential feature extraction capabilities of Bi-LSTM. As a semi-supervised model, LogMeta achieves a balance between training efficiency and detection performance, making it well-suited for real-world scenarios with limited labeled data. By utilizing only a small number of labeled examples, the framework demonstrates robust anomaly detection performance, even in highly imbalanced and low-resource datasets. The hybrid language model enables LogMeta to capture both semantic and structural patterns within logs, enhancing its ability to generalize effectively across diverse software systems and heterogeneous log formats. As a result, LogMeta provides a practical, adaptive, and generalizable solution for addressing the complexities of modern log data, consistently outperforming traditional supervised and unsupervised models.

In summary, the contributions of this paper can be summarized in four-fold:

- We propose LogMeta, a novel semi-supervised framework for log anomaly detection that integrates MAML to address the challenges of limited labeled data and the variability in log formats and structures across different systems.
- By combining the advantages of meta-learning and a hybrid language model, LogMeta achieves robust and consistent anomaly detection performance, particularly excelling in low-resource scenarios. It rapidly adapts to new log systems with minimal fine-tuning, demonstrating strong generalization capabilities.
- To the best of our knowledge, this is among the earliest works to integrate MAML with a contextualized hybrid language model for robust cross-system log anomaly detection.
- We conduct extensive experiments on multiple real-world log datasets, showcasing LogMeta's superior anomaly detection performance and exceptional domain transferability across heterogeneous log environments.

The remainder of this paper is organized as follows. Section 2 presents the background and related work. Section 3 details the methodology, including the construction of a stable and diverse training dataset for MAML, log parsing, the integration of Model-Agnostic Meta-Learning, the development of the hybrid language model, fine-tuning process, and the subsequent anomaly detection using the fine-tuned model. Sections 4 and 5 discuss the experimental design and results.

Fig. 1. The example of log messages and log events. **Note:** Dynamic variables are highlighted in **red**. Irrelevant details (i.e., timestamps, APIs, and software systems) at the top of the log will be removed directly. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Section 7 addresses the threats to validity. Finally, Section 8 summarizes the paper.

2. Background and related work

2.1. Log parsing

Logs constitute semi-structured text data, typically generated through log statements in source code (e.g., logging.info(), printf(), and Console.WriteLine()) (Dai et al., 2020). To ensure the reliability of systems, logs are widely used as they capture key states during system operation (Lou et al., 2010; Liang et al., 2007). As shown in Fig. 1, these logs consist of both static messages and dynamic variables. Each log entry records a specific system event and contains dynamic information such as timestamps, numbers, ports, APIs, addresses, block/task IDs, and usernames.

Log parsing is employed to automatically convert each log message into a structured format by identifying an event template (constant part) associated with parameters (dynamic variable parts) (Zhu et al., 2019; Le and Zhang, 2023). Fig. 1 provides examples of raw log messages and the corresponding log events obtained through parsing. For example, in a software log message like "*2023-11-17 00:04:09.377 INFO: 34.139.166.52:0 - 'POST /webhook/order HTTP/1.0' 200 OK delete empty directory: /backend/upload/download/orders/49512/49512*", the timestamp is "*2023-11-17 00:04:09.377*", the severity level is "*INFO:*", and the message content is "*delete empty directory: /backend/upload/download/orders/49512/ 49512*". After appropriate log parsing, this log message can be split into the event template "*'POST <*>' 200 OK delete empty directory: <*>*", and the parameter values [*/webhook/order HTTP/1.0, /backend/upload/download/orders/ 49512/49512*]. Here, "<*>" denotes the position of a parameter.

Prior research (Khan et al., 2022; Zhu et al., 2019) has demonstrated that the effectiveness of log parsing significantly impacts the performance of subsequent anomaly detection models. Consequently, log parsing is a crucial step in log analysis, and its accuracy directly influences the performance of downstream tasks (Li et al., 2023; Ma et al., 2024b). Numerous studies have proposed various log parsing methods (He et al., 2017; Du et al., 2017; Zhu et al., 2019; Du and Li, 2018). These methods can be broadly categorized into approaches based on frequent pattern mining (Nagappan and Vouk, 2010; Vaarandi and Pihelgas, 2015), clustering (Shima, 2016; Tang et al., 2011), and heuristics (He et al., 2017; Jiang et al., 2008). Among these, heuristic methods, which leverage the inherent characteristics of logs, have demonstrated superior performance in terms of accuracy (Zhu et al., 2019) and time efficiency (Le and Zhang, 2023) compared to other techniques. Furthermore, the rise of Large Language Models (LLMs), such as ChatGPT (Roumeliotis and Tselikas, 2023), has led to their successful application in various software engineering tasks (White et al., 2024), achieving promising results (Gao et al., 2023). Several studies (Le and Zhang, 2023; Qi et al., 2023) have highlighted the feasibility of using ChatGPT for log parsing. For instance, LogGPT (Qi et al., 2023) utilized ChatGPT and achieved a maximum F1-score of 0.618 on the BGL dataset, underscoring the need for further refinement of this method. Le and Zhang (2023) evaluated the performance of ChatGPT-3.5 as a log parser for extracting log events and parameters. Given these promising results, we believe that leveraging ChatGPT for log parsing offers high fault tolerance. However, there remains a need for continuous optimization of prompts to improve task completion effectiveness. Thus, while the use of LLMs for log parsing shows great potential, further research is required to fully explore their capabilities in this domain.

2.2. Log anomaly detection models

To ensure system reliability, logs are well suited for tasks because they record key states during system operation. Currently, a substantial number of data-driven methods have been introduced that automatically detect anomalies in systems by analyzing software logs (Sun et al., 2026, 2025a). Traditional machine-learning techniques can mine hidden information from various data sources. He et al. Liang et al. (2007) conducted an evaluation of the performance of six machine learning-based anomaly log detection algorithms; however, these algorithms exhibited issues such as low efficiency and limited flexibility. Limitations encountered in machine learning for log data anomaly detection include the requirement for substantial amounts of labeled data to train models effectively, the challenge of generalizing across various system behaviors, susceptibility to adversarial attacks simulating normal patterns, and complexities in deciphering intricate model decisions. Furthermore, evolving systems can render acquired patterns obsolete, thus demanding ongoing retraining and adaptation.

Deep learning-based algorithms for log anomaly detection have been introduced to address these issues and tackle the common reliance of machine learning-based approaches on manually designed feature representations for log data. Typically, supervised methods generally outperform others due to the vast amount of labeled data (e.g., normal or abnormal) used during training (Sun et al., 2024). However, these methods are not ideal for industrial applications because they require extensive labeled datasets and are highly sensitive to incorrect labels (Le and Zhang, 2022). This procedure often consumes significant human resources and may lead to labeling errors, which can degrade performance (Yang et al., 2021). This makes the effective deployment of unsupervised methods in industry quite difficult. In contrast, semi-supervised methods require only a small amount of labeled data for training, significantly reducing the cost of manual labeling and dependency on labeled data. Unlike unsupervised methods, semi-supervised approaches utilize some training label information (e.g., feature-label correlations), enabling them to optimize the training objective more effectively (Ma et al., 2024a).

LogRobust (Zhang et al., 2019) and HitAnomaly (Huang et al., 2020) are notable deep learning algorithms in the supervised learning category. In detail, Zhang et al. (2019) introduced LogRobust, which represents log events as semantic vectors and uses an attention-based Bi-LSTM model to detect anomalies. Huang et al. (2020) were the first to apply the transformer model to the task of log anomaly detection; they proposed HitAnomaly, which used a hierarchical transformer to model sequences of log templates and parameter values and designed an attention mechanism to merge semantic information with parameter values for classification, demonstrating robustness to volatile log data. Although supervised methods provide accurate diagnoses of system conditions, their main limitation is the need for a substantial volume of normal and abnormal data during the training phase. Therefore, researchers have also explored unsupervised and semi-supervised approaches (Li et al., 2020; McInnes et al., 2017). Du et al. (2017) employed DeepLog to treat log information as natural language sequences, autonomously learn log patterns from normal execution sequences, and identify

anomalies by assessing whether incoming log events deviate from the predictions of a stacked LSTM model. NeuralLog (Le and Zhang, 2021) introduces a novel log-based anomaly detection method based on a Transformer classification model that eliminates the need for log preprocessing. Lin and Chiang (2022), Yang et al. (2021) introduced a semi-supervised learning framework known as PLELog, which utilizes the HDBSCAN (McInnes and Healy, 2017) clustering method for estimating labels probabilistically on unlabeled log sequences, addressing the challenge of limited labeling. LogBERT (Guo et al., 2021) learns patterns within normal log sequences through masked log message prediction and hypersphere volume minimization, yet it cannot specifically identify semantic information in abnormal logs.

Despite their remarkable achievements in log anomaly detection, current deep learning models often exhibit significant performance variability across different datasets. This inconsistency arises from the inherent differences in data distributions, structures, and characteristics among datasets, making it challenging for a single model to generalize effectively across diverse domains. For instance, a model trained on one type of log data may fail to maintain high accuracy when applied to another dataset with distinct features or patterns. This lack of adaptability limits the applicability of traditional deep learning models, especially in real-world scenarios where data heterogeneity is prevalent. Meta-learning offers a promising solution by enabling models to learn generalizable patterns and adapt quickly to new datasets with minimal fine-tuning.

2.3. Meta-learning

Meta-learning (Vanschoren, 2019), also known as "learning to learn," is a widely used few-shot learning paradigm that alters the search strategy in hypothesis space. Its primary goal is to learn good initialization parameters from a diverse set of tasks, enabling models to quickly adapt to new tasks with minimal labeled data. Unlike conventional transfer learning techniques, meta-learning trains models on a variety of heterogeneous sub-tasks, updating their parameters based on query sets (used for validation) within each sub-task to ensure generality across different domains. Several meta-learning methods have been proposed in recent years, with Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017) being one of the most prominent and versatile approaches.

By learning a set of shared parameters across multiple tasks, a meta-learning framework such as MAML can rapidly adjust to new log anomaly detection tasks, thereby enhancing the model's robustness and adaptability. Specifically, meta-learning plays a critical role in addressing the challenges of cross-system variability and limited labeled data, as it enables models to effectively generalize from diverse log systems while adapting to the specific characteristics of a new system. This makes it particularly effective for log anomaly detection, where datasets often exhibit significant heterogeneity in structure and semantics across different systems.

To further strengthen the adaptability and generalization of log anomaly detection models, we introduce meta-learning as a core paradigm in our framework. Despite the promise of meta-learning, existing attempts still face notable limitations in practical log anomaly detection scenarios. MetaLog (Zhang et al., 2024) applies meta-learning on top of shallow event-level representations derived from log parsing and static semantic embeddings, which limits its ability to capture fine-grained contextual semantics and makes it sensitive to template quality and lexical variation across systems. As a result, its generalization capability is constrained when logs exhibit complex or evolving semantic patterns. FreeLog (Zhao et al., 2025), on the other hand, removes the dependency on labeled data by combining unsupervised objectives with meta-learning; however, its reliance on proxy tasks and heuristic anomaly signals often leads to unstable decision boundaries and reduced discriminative power, particularly in highly heterogeneous cross-system settings.

In this paper, we adopt MAML as the few-shot meta-learning method to transfer general-purpose knowledge from diverse log datasets to specific target systems. In contrast to the above approaches, our framework performs meta-learning over a contextualized hybrid language model, enabling joint adaptation of semantic representations and sequential dependencies, which results in more robust and scalable cross-system anomaly detection. We pre-train a hybrid model for log anomaly detection, leveraging MAML to enhance its ability to identify normal and anomalous patterns effectively, even in scenarios with limited labeled data. By equipping the model with a robust initialization, MAML facilitates rapid adaptation to system-specific features, ensuring accurate anomaly detection even with highly imbalanced or sparse datasets. This approach enables the framework to utilize general-purpose knowledge efficiently during pre-training and fine-tune to the nuances of the target system with minimal additional effort. Consequently, our model not only generalizes well across diverse log systems but also demonstrates superior performance in the target domain, achieving reliable anomaly detection under real-world constraints. Furthermore, we emphasize that MAML's domain transfer and context adaptability capabilities are pivotal in maintaining model effectiveness across diverse systems and varying data conditions, addressing critical challenges in log anomaly detection.

2.3.1. Domain transfer

Domain transfer refers to the ability of a model trained on one domain to perform effectively in another, different domain without requiring extensive re-training. In this study, we explore how the MAML approach enhances the cross-domain transferability of our proposed model, LogMeta. For example, we pre-train the model on multiple datasets, such as the 16 public datasets available on the LogHub platform, to extract general anomaly-related patterns. When fine-tuned on a new target domain, such as continuously updated HDFS logs, the model requires only a few-shot sample (20%) to adapt efficiently. Specifically, we evaluate domain transfer by comparing the performance of models with and without MAML on datasets from distinct log systems. This comparison quantifies the impact of MAML in transferring learned knowledge across domains, demonstrating the model's ability to generalize effectively with minimal additional training.

2.3.2. Context adaptability

Context adaptability refers to the model's ability to perform accurately even when trained on incomplete or less relevant data, adapting to new contexts without requiring domain-specific fine-tuning. For example, we test this capability by deliberately excluding highly relevant datasets (e.g., the other 15 datasets on the LogHub platform, excluding the HDFS dataset) during the meta-learning phase. We then evaluate the model's performance on the HDFS dataset, which shares significant similarities with the other 15 datasets, to assess whether our proposed model can adapt to missing contextual information. This approach evaluates the model's ability to accurately detect anomalies even in the absence of optimal contextual data, demonstrating that it can generalize effectively with only partially relevant information.

2.4. Hybrid language model

2.4.1. RoBERTa

RoBERTa (Liu et al., 2019), a robustly optimized BERT (Devlin et al., 2018) pre-training method introduced by Facebook AI in 2019, represents a significant advancement over the BERT model. RoBERTa retains the multi-layer Transformer encoder structure of BERT; however, in this model, key parameters are adjusted, such as an increased number of layers, expanded size of hidden layers, and a larger number of attention heads. These features enable RoBERTa to more effectively capture complex linguistic features and patterns.

In terms of pre-training, RoBERTa differs from BERT by relying solely on MLM as its pre-training task. Additionally, RoBERTa introduces

Dynamic Masking mode, replicating training data and executing a series of masking strategies.

During data processing, RoBERTa employs large-scale and diversified training datasets covering a wider range of text types and larger corpora, significantly enhancing the model's understanding of different text types and its generalization capabilities. Furthermore, RoBERTa employs longer sequence lengths and larger batch sizes during training, further enhancing training efficiency and model performance.

In the context of anomaly detection in software logs, RoBERTa excels at capturing contextual information in software logs and comprehending complex log structures, critical for accurately identifying key information within log sequences. We believe that RoBERTa, with its superior optimization strategies compared to BERT, will undoubtedly exhibit superior performance in anomaly detection in software logs.

2.4.2. Bi-LSTM

In traditional LSTM, information flows unidirectionally, moving solely from the beginning to the end of a sequence. However, when handling complex linguistic structures like log sequences, there exists interdependence between preceding and subsequent information, potentially resulting in the loss of crucial contextual information within log sequences.

Bi-LSTM (Huang et al., 2015) addresses this limitation by introducing a bidirectional structure. It comprises forward and backward LSTMs, responsible for processing the input sequence in both forward (from start to end) and backward (from end to start) directions of information flow (Yildirim, 2018). The bidirectional processing strategy enables Bi-LSTM to capture contextual information from both preceding and subsequent texts. In this setup, the forward hidden layer L_f, the backward hidden layer L_b, and the output sequence X_o are employed to update the network. Iterations are performed from t to 1 for backward updates and from 1 to t for forward updates. The time step t indicates the current position within the input log sequence, where h_t^f and h_t^b are the hidden state vectors at position t during forward and backward propagations, respectively. Ultimately, these two vectors are concatenated as: $h_t = concat(h_t^f, h_t^b)$, capturing information from both forward and backward directions. The update parameters for Bi-LSTM can be represented as:

$$\begin{aligned} L_f &= \sigma(W_1 \cdot x_t + W_2 \cdot L_{f-1} + b_{L_f}) \\ L_b &= \sigma(W_3 \cdot x_t + W_5 \cdot L_{b-1} + b_{L_b}) \\ x_o &= W_4 \cdot L_f + W_6 \cdot L_b + b_{x_o} \end{aligned} \tag{1}$$

where L_f represents the forward pass, L_b represents the backward pass, and X_o denotes the final output layer; W signifies the weight coefficients, while b_{L_f}, b_{L_b}, and b_{x_o} denote the biases. σ is a sigmoid function ranges from 0 to 1. The time step t indicates the current position within the input log sequence, where h_t^f and h_t^b are the hidden state vectors at position t during forward and backward propagations, respectively. Ultimately, these two vectors are concatenated as: $h_t = concat(h_t^f, h_t^b)$, capturing information from both forward and backward directions.

3. Methodology

In this section, we describe the architecture and design of our proposed LogMeta model in detail.

3.1. Overview

In this section, we introduce an innovative log anomaly detection framework named LogMeta, as illustrated in Fig. 2. The framework is designed to enhance model generalization and adaptability across heterogeneous log systems. LogMeta operates through two key phases: the MAML training phase and the anomaly detection phase. In the preprocessing stage, we employ ChatGPT with carefully designed prompts to perform log parsing, effectively extracting structured information from raw log data. This enables the model to handle diverse log formats and prepares the data for subsequent meta-learning tasks.

In the MAML training phase, LogMeta aims to obtain robust initialization parameters that capture shared representations across multiple log datasets. This process involves four main steps: (1) constructing a meta-training dataset composed of diverse log sources in varied formats; (2) partitioning the data into sub-tasks following the n-way k-shot rule for meta-learning; (3) applying the MAML algorithm with a hybrid language model to learn generalized anomaly detection features across tasks; and (4) fine-tuning the hybrid model on a target dataset (e.g., HDFS) using few-shot learning to achieve rapid adaptation to system-specific characteristics. Through this phase, the model learns transferable knowledge that facilitates efficient learning even with limited labeled samples.

In the anomaly detection phase, the fine-tuned hybrid language model is utilized to classify log entries as normal or anomalous. As shown in Fig. 3, the hybrid language model combines the strengths of pre-trained RoBERTa and an attention-based Bi-LSTM architecture. RoBERTa is utilized to encode log entries and generate context-aware representations, which are then passed to an attention-based Bi-LSTM. The attention mechanism assigns different weights to segments of the log context, emphasizing critical information, while the Bi-LSTM captures sequential dependencies by processing both forward and backward contextual information. By integrating these components, LogMeta effectively learns both semantic and structural characteristics of log data, enabling accurate and robust anomaly detection. This phase also evaluates LogMeta's performance on datasets from different log systems, demonstrating its ability to generalize across heterogeneous environments with minimal labeled data.

3.2. Building a stable and diverse training dataset for MAML

To construct a robust training set for MAML pre-training, we collected log data from two primary sources. First, we leveraged 16 publicly available datasets published on the LogHub[1] platform, extracting 2000 logs from each dataset. The decision to use 2000 logs per dataset follows the data sampling strategy reported in the original LogHub studies (Zhu et al., 2023; Jiang et al., 2024b), which indicated that most benchmark datasets contain approximately 1,500–2,000 unique log entries. Although LogHub also released a derived version named LogHub-2k, we found through manual inspection that this version includes a high proportion of duplicate normal logs—some repeated more than ten times—while several rare anomalous logs were omitted. To ensure both representativeness and data quality, we adopted a similar sampling strategy but reconstructed the subsets ourselves to maximize log diversity and coverage.

These datasets encompass a wide variety of log systems commonly used in log analysis research, ensuring a comprehensive representation of log formats and event patterns. However, since these public datasets have been available for several years, they may not fully capture the characteristics of continuously evolving industrial systems. To address this limitation, we additionally collected 2000 logs from each of our four in-house software systems, which reflect the dynamic nature and complexity of modern logging environments. In total, the training set consists of logs from 20 datasets, ensuring a balanced mix of stable and diverse log sources for MAML pre-training.

To construct a labeled training set that is both stable and diverse, we employ the Determinantal Point Process (DPP) (Sun et al., 2026; Kulesza et al., 2012) method. This method is particularly well-suited for log data, where repetitive log sequences and high redundancy are common. By maximizing diversity, DPP ensures that logs from different templates are well-represented, reducing inductive bias and improving

[1] https://github.com/logpai/loghub

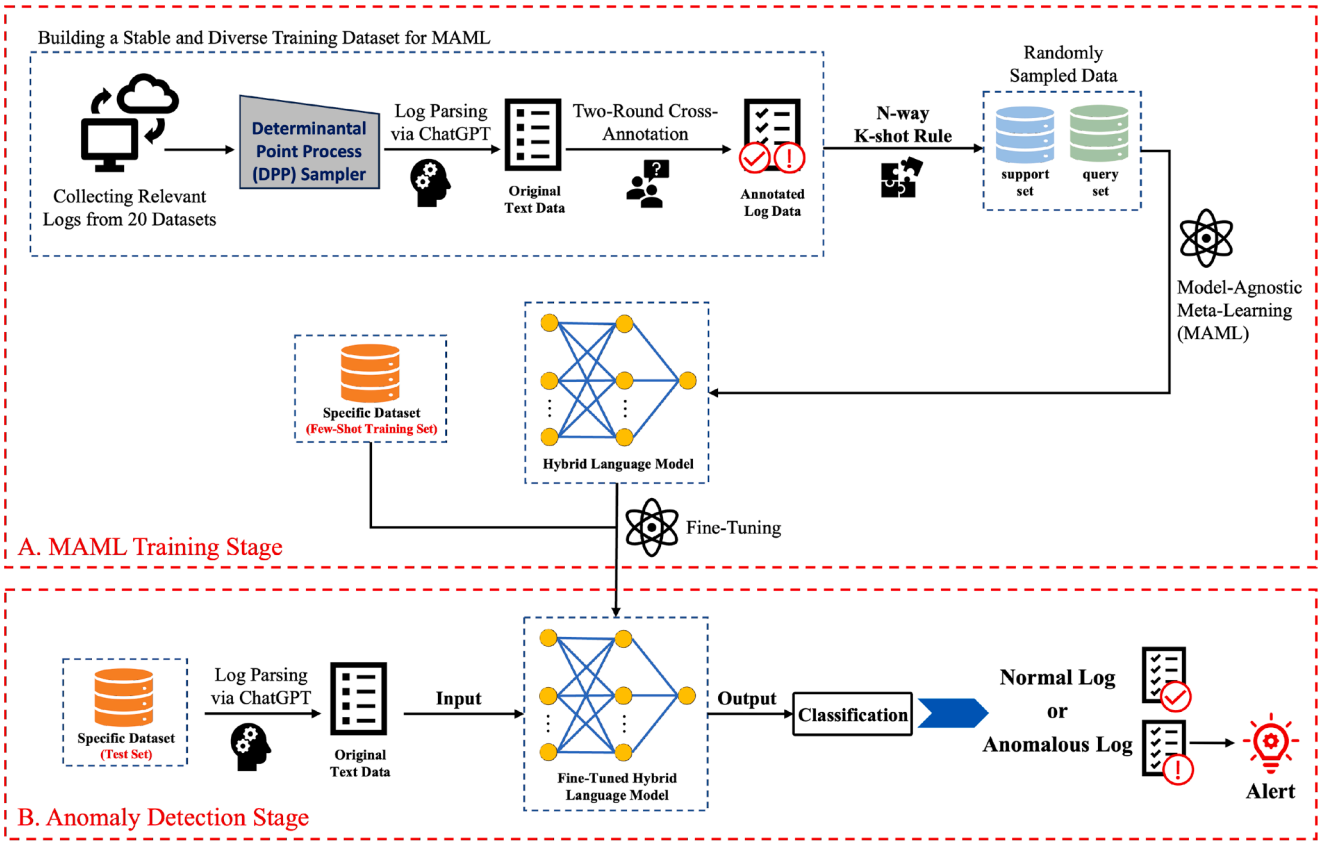

Fig. 2. The overview of LogMeta.

the robustness of the model. Additionally, DPP minimizes the number of logs requiring manual labeling, making the process more efficient.

Algorithm 1 Constructing a stable and diverse labeled training set using the DPP method.

Require: Log sequence dataset $X = \{x_i\}_{i=1}^{N}$, Candidate set size K
Ensure: Stable and diverse labeled set C
1: Initialize embedding matrix $V = \epsilon(X)$, where ϵ is the embedding function
2: Calculate similarity matrix $L = \text{sim}(V, V)$, where sim represents cosine similarity
3: Initialize empty set $C = \emptyset$
4: **for** $i = 1$ to K **do**
5: Select log sequence x_j that maximizes the marginal gain in diversity:

$$x_j = \arg \max_{x_i \in X \setminus C} \frac{\det(L_{C \cup \{x_i\}})}{\det(L_C)}$$

6: Add x_j to candidate set C: $C = C \cup \{x_j\}$
7: **end for**
8: **Return** C

The detailed algorithm is shown in Algorithm 1. Formally, let the log dataset be represented as a set of log sequences $X = \{x_1, x_2, \ldots, x_N\}$, where each sequence x_i is encoded into a vector representation v_i. These vectors are concatenated to form the embedding matrix $V = \epsilon(X) = [v_1, v_2, \ldots, v_N]$ *(Line 1)*. The similarity between two log sequences x_i and x_j is calculated using cosine similarity *(Line 2)*, yielding the similarity matrix L as follows:

$$L_{ij} = \cos(v_i, v_j) = \frac{v_i \cdot v_j}{\|v_i\| \|v_j\|} \tag{2}$$

The DPP method then selects a subset $C \subset X$ of size K, which maximizes the diversity of the candidate set by maximizing the determinant of the submatrix L_C formed by the selected samples *(Lines 4–7)*:

$$\text{argmax}_C \det(L_C) \tag{3}$$

In general, the DPP method ensures that the selected subset C is diverse and stable, effectively covering various log sequences. The DPP sampling process is iterative and continues until the predefined set size K is reached, optimizing the total diversity within the selected training set. By integrating ChatGPT for log parsing and DPP for subset selection, we ensure that the constructed training dataset is both high-quality and representative, providing a solid foundation for MAML pre-training.

3.3. Log parsing

Given the heterogeneity and dynamic nature of log data, constructing a high-quality training set necessitates effective log preprocessing. To address this, we design specialized prompts for ChatGPT to parse raw log data and extract structured information. A key advantage of using ChatGPT for log parsing lies in its ability to generalize across diverse log structures without requiring extensive manual intervention or system-specific configurations. Unlike conventional parsers that rely on predefined rules or patterns, ChatGPT effectively handles dynamic log formats, including timestamps, IP addresses, and event parameters, by leveraging its advanced contextual understanding. Furthermore, ChatGPT facilitates rapid prototyping and updates through simple prompt adjustments, making it particularly suitable for environments with frequently evolving log structures.

These prompts are tailored to address the unique challenges of log parsing, such as handling dynamic fields (e.g., timestamps, IP addresses) and categorizing logs into predefined groups. By providing clear instructions, illustrative examples, and contextual information, the prompts

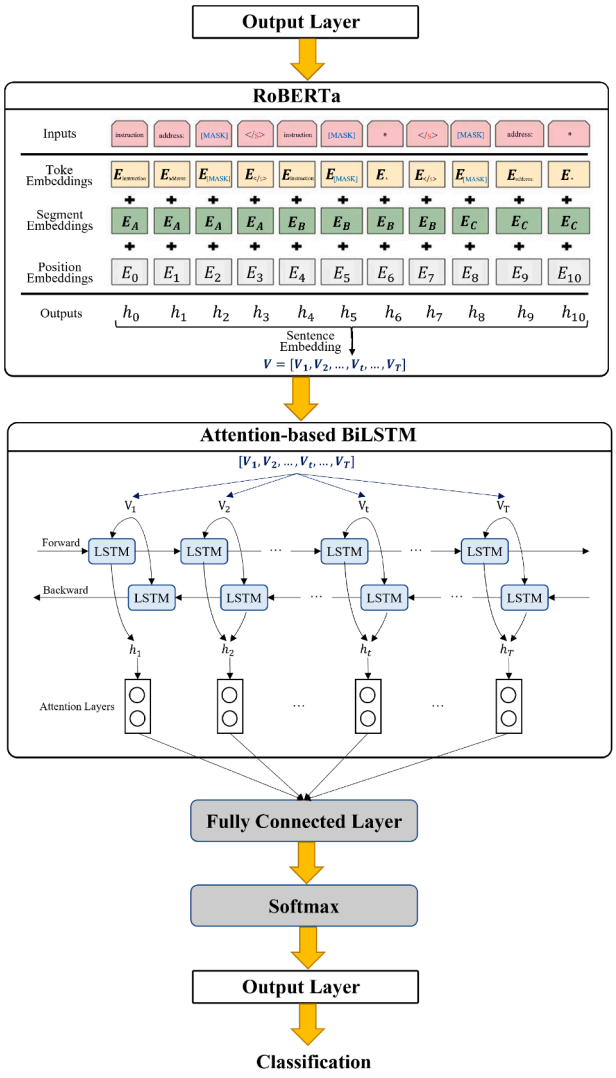

Fig. 3. The architecture of proposed hybrid language model.

enable ChatGPT to generate accurate and consistent parsing outputs. An example of the designed prompt template is provided below:

Task Description: *You need to parse the content of the logs and replace the dynamic variables with templates. Please output the results directly without explanation.*

Replacement Rules: *1) Replace all timestamps with "TIME"; 2) Replace all IP addresses with "IP"; 3) Replace user IDs with "USER"*

…(more rules)

Situated Learning: *Here are some examples:*

Original Log: "[2023-12-01 10:00:00] User 123 logged in from IP192.168.0.1."

Parsed Log: "[TIME] User [USER] logged in from IP [IP]."

…(more examples)

Target Log: *Please parse the log template from this log message: [LOG]*

After ChatGPT processes the raw log data, we employ a manual annotation method to assign labels to the parsed logs. This step is necessary because only 5 out of the 16 LogHub datasets provide ground-truth labels that distinguish normal and anomalous logs, while the remaining datasets are unlabeled. Consequently, we manually annotated logs for 15 datasets in total, including 11 from LogHub and 4 from our in-house software systems. The annotation team consisted of four members with prior experience in software development, log analysis, and related technical expertise. To ensure the reliability and consistency of the labeled data, we adopted a two-round cross-annotation process, where each log entry was independently reviewed by at least two annotators and discrepancies were resolved through group discussion. Overall, approximately 30,000 log entries were manually inspected and labeled during this process.

In the first round, two annotators independently verified the accuracy of ChatGPT's log parsing and assigned labels (normal or anomalous) to each log entry based on their domain knowledge. If both annotators agreed on the label, it was accepted; otherwise, the log entry was flagged for further review. In the second round, a third annotator examined all flagged logs to determine their final labels. For cases where conflicts remained unresolved after the second round, a fourth expert conducted a final review, making the definitive labeling decision or discarding the log if it was deemed unsuitable for training. Across all annotated datasets, approximately 1.4% of the total logs were initially flagged for disagreement between annotators. After the second-round review, only 0.2% of the logs remained unresolved and required expert adjudication. The overall inter-annotator agreement reached a Cohen's Kappa coefficient of 0.97, indicating a high level of consistency and reliability in the manual labeling process.

To ensure data quality, we removed logs exhibiting high redundancy—retaining at most ten instances per unique log message—and those with significant labeling discrepancies that remained unresolved after two rounds of cross-annotation, accounting for less than 0.01% of the total dataset. At the same time, we prioritized retaining low-frequency anomaly logs to ensure accurate and diverse labeling. This rigorous process resulted in a high-quality dataset comprising 36,741 labeled log entries (from an initial total of 40,000 entries), which is used for MAML pre-training.

3.4. Model-agnostic meta-learning

In this section, we adapt Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017), a meta-learning method, into our LogMeta framework to learn general-purpose knowledge from the log datasets constructed in the previous section. The goal of MAML is to equip the model with an initialization that enables rapid adaptation to new log anomaly detection tasks with minimal data. To achieve this, we configure the n-way k-shot rule for random sampling, which is used to construct tasks. For our experiments, we adopt a 2-way 5-shot setup, where logs are divided into two categories: normal and anomalous. Each task contains at least 5 samples from each category. These samples are further split into a **support set** (used for training) and a **query set** (used for validation). The support set contains 5 samples per category, while the query set includes a larger number of samples to improve the model's generalization ability.

After constructing tasks, we integrate the hybrid anomaly detection model into MAML. Specifically, we utilize MAML to initialize the parameters of our hybrid model, which combines a Transformer-based encoder (e.g., RoBERTa) with an attention-based Bi-LSTM for log parsing and anomaly classification. This hybrid model is trained to detect normal and anomalous logs efficiently across diverse datasets. As shown in Algorithm 2, the MAML algorithm optimizes the hybrid model's parameters in two stages: the **inner loop** and the **outer loop**, enabling the model to quickly adapt to new tasks and generalize effectively across log systems.

The meta-learning process begins with the initialization of the hybrid model parameters θ. During each iteration of meta-training, a batch of tasks is sampled from the task distribution $p(\mathcal{T})$, where each task represents a log anomaly detection scenario derived from a specific log system. The **inner loop** of MAML fine-tunes the model for each task $\mathcal{T}_i$ (Lines 5–9). This involves calculating the task-specific loss $\mathcal{L}_{\mathcal{T}_i}(\theta)$ using a small support set $\mathcal{D}_i^{train}$ (Line 7). The model parameters are then updated via gradient descent, producing task-specific parameters θ_i' (Line 8). This step simulates how the model would adapt to a new anomaly detection task by focusing on the unique characteristics of logs within the task.

Algorithm 2 MAML for initializing the hybrid language model parameters.

1: **Input:** Task distribution $p(\mathcal{T})$, learning rate α, meta-learning rate β
2: Initialize the hybrid language model parameters θ (for RoBERTa and attention-based Bi-LSTM model)
3: **for** each iteration **do**
4: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
5: **for** each task $\mathcal{T}_i$ **do**
6: Sample K training examples $\mathcal{D}_i^{train}$ from $\mathcal{T}_i$
7: Compute task-specific loss $\mathcal{L}_{\mathcal{T}_i}(\theta)$ on $\mathcal{D}_i^{train}$
8: Compute adapted parameters with gradient descent:

$$\theta_i' \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(\theta)$$

9: **end for**
10: Sample N validation examples $\mathcal{D}_i^{val}$ from each task $\mathcal{T}_i$
11: Compute meta-loss across all tasks:

$$\mathcal{L}_{meta}(\theta) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(\theta_i')$$

12: Update model parameters θ using meta-loss:

$$\theta \leftarrow \theta - \beta \nabla_\theta \mathcal{L}_{meta}(\theta)$$

13: **end for**
14: **Output:** Initialized hybrid language model parameters θ

The **outer loop** evaluates how well the model's task-specific parameters θ_i' generalize to new data, focusing on general-purpose knowledge learning (Line 3–13). This is done by testing the adapted model on a separate validation set $\mathcal{D}_i^{val}$ for each task $\mathcal{T}_i$ (Line 10). The meta-loss $\mathcal{L}_{meta}(\theta)$ is calculated as the sum of validation losses across all tasks (Line 11), using the task-adapted parameters θ_i'. Finally, the model's original parameters θ are updated using this meta-loss (Line 12), which ensures that the learned initialization is robust and capable of quickly adapting to various anomaly detection tasks in future iterations.

Through this iterative meta-training process, the hybrid language model is equipped with initialization parameters that enable rapid adaptation and improved generalization. This results in a more robust and efficient anomaly detection framework, capable of identifying anomalies across diverse log systems with minimal labeled data.

3.5. Hybrid language model

In this section, we present the hybrid language model (HLM) employed in LogMeta, which integrates the contextual understanding capabilities of RoBERTa with the sequential feature extraction strengths of Bi-LSTM. This combination enables precise anomaly detection, even with limited training data. The HLM begins by utilizing RoBERTa to extract semantic and contextual information from log data. RoBERTa employs dynamic masking, generating a new masking pattern each time a log sequence is fed into the model, thereby enhancing its ability to capture diverse contextual representations. As depicted in Fig. 3, the symbol "</s>" denotes the separator for each log sequence, indicating the end of an individual sequence. Log entries are tokenized and treated as input units to be processed by RoBERTa. Each token is converted into a word embedding vector W by looking up its corresponding representation in the vocabulary table. These word embeddings are learned during pre-training and ensure that each token has a unique vector representation. Additionally, positional encoding is applied to maintain the sequential order of tokens within each log sequence.

Word embedding is generated through the combination of token embedding, segment embedding, and position embedding. Token embedding converts each token into a 768-dimensional vector denoted as T, segment embedding generates vector S, and position embedding produces vector P. The fusion of these three embeddings yields the embedded vector X for the log sequence:

$$X = W = T + S + P \tag{4}$$

RoBERTa produces a vector representation h for each token, and these h vectors are integrated into a sequence-level semantic vector V_S. Specifically, the CLS token is used in RoBERTa to encapsulate the semantic information of the entire input sequence. The output vector corresponding to the CLS token is treated as the semantic vector V_S for the log sequence:

$$V_S = [V_1, V_2, \ldots, V_t, \ldots, V_T] \tag{5}$$

The semantic vector sequence V_S is used as the input to an attention-based Bi-LSTM network for anomaly classification. The Bi-LSTM network captures both forward and backward contextual information from log sequences, leveraging its capability to handle diverse and complex patterns in log data. To further enhance the detection of anomalies, particularly in complex log structures, we introduce a "multihead" attention mechanism.

The "multihead" attention mechanism comprises eight attention heads, each calculating attention scores between log tokens. To capture semantic relationships effectively, three matrices are utilized: query matrix Q, key matrix K, and value matrix V. These matrices are derived by multiplying the semantic vector V_S with weight matrices W_Q, W_K, and W_V. For each attention head, the self-attention mechanism computes a new vector using these matrices, and their weights are normalized using the Softmax function. The multihead attention mechanism is defined as:

$$\begin{aligned} \text{multihead} &= \text{concat}(head_1, head_2, \ldots, head_n) \cdot W^O \\ head_n &= \text{Softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \cdot V \end{aligned} \tag{6}$$

During the training of the HLM, we minimize the cross-entropy loss function by comparing the predicted outputs with the ground-truth labels. The Adam optimization algorithm is employed to update the model parameters, including the weights of both the Bi-LSTM and Attention layers.

Following the attention mechanism layer, we incorporate log features via a fully connected layer. This layer synthesizes the outputs from the Bi-LSTM and Attention mechanism layers. Finally, a Softmax layer is applied to convert the outputs of the fully connected layer into a probability distribution, enabling precise classification of log anomalies. By combining these components, the HLM effectively identifies anomalous patterns in log data.

3.6. Fine-tuning

In the fine-tuning phase, the hybrid language model is initialized with the meta-parameters θ learned during the meta-learning phase. Fine-tuning is then performed on a specific target dataset, such as the HDFS dataset, which consists of system logs with unique structures and patterns. The meta-learning phase utilized a pre-training dataset comprising 20 datasets collected from diverse log systems. These datasets exhibit high-level textual similarities with the HDFS dataset, including common log formats, dynamic fields such as timestamps and IP addresses, and structured patterns in normal log events. However, notable differences also exist, such as system-specific keywords (e.g., block blk_<*>, username), distinct log semantics, and variations in anomaly characteristics. These shared features enable the model to effectively transfer the general knowledge acquired during the meta-learning phase, while the differences allow it to adapt to the unique aspects of the HDFS dataset. This transfer significantly reduces the amount of labeled data required for fine-tuning, ensuring efficient and targeted adaptation.

To further enhance efficiency, we employ a few-shot learning strategy, using only 1% of the labeled data for fine-tuning. Despite the limited amount of labeled data, the robust initialization provided by the meta-learning phase allows the model to effectively leverage general knowledge and adapt to the unique features of the target dataset. This minimal data requirement significantly reduces the need for extensive manual labeling, which is often time-consuming and labor-intensive. By achieving robust adaptation with such a small dataset, our approach not only saves valuable annotation resources but also accelerates the fine-tuning process, making it highly practical and efficient for real-world applications where labeled data is scarce.

It is noteworthy that MAML is not applied during the fine-tuning phase. MAML's primary role is to provide a robust initialization of parameters θ, optimized across multiple related tasks during the meta-learning phase. This initialization ensures that the model requires minimal fine-tuning to achieve high performance on new tasks. During the fine-tuning phase, the objective is to refine the pre-trained hybrid language model to adapt to the specific characteristics of the target dataset without requiring extensive retraining.

This process demonstrates the adaptability and efficiency of the LogMeta framework. By leveraging meta-learning for parameter initialization and employing few-shot fine-tuning, the model achieves superior performance even on domain-specific datasets, such as HDFS, with significantly reduced training resources and time requirements.

3.7. Anomaly detection using the fine-tuned hybrid language model

In the anomaly detection stage, we first utilize ChatGPT for log parsing to extract structured information from raw log data. The parsed log sequences from the test set are then input into the fine-tuned hybrid language model, which classifies each sequence as either normal or anomalous based on the model's predictions. By leveraging the fine-tuned hybrid language model, the system effectively captures both semantic and contextual information from the log data. This capability enables accurate anomaly classification, even in scenarios with limited labeled training data, highlighting the robustness and adaptability of the proposed approach.

Once the anomaly detection model constructed using LogMeta is fine-tuned, it is deployed for real-time system monitoring. Upon receiving a new log sequence, LogMeta processes the logs and leverages the trained hybrid language model to classify the sequence as either normal or anomalous. In the event of an anomaly, the system promptly issues an alert to notify engineers. This proactive detection and notification mechanism reduces system downtime, facilitates early diagnosis and resolution of issues, and significantly enhances overall operational efficiency. These capabilities make LogMeta an invaluable tool for ensuring the reliability and stability of modern software systems.

4. Experimental setup

To evaluate the performance of LogMeta, we utilized four widely recognized public datasets: HDFS, BGL, Thunderbird, and Spirit. Initially, we compared LogMeta against existing software log anomaly detection models to assess its overall effectiveness. Subsequently, we conducted experiments on datasets of varying sizes to examine its performance, particularly in smaller-scale data scenarios. Finally, we performed detailed comparative experiments to highlight the improvements brought by MAML to LogMeta's anomaly detection capabilities. We also conducted detailed comparative experiments to showcase the contribution of each module.

4.1. Datasets

To evaluate the performance of our models for log anomaly detection, we utilize four publicly available datasets: HDFS (Xu et al., 2009), BGL (Oliner and Stearley, 2007), Thunderbird (Oliner and Stearley, 2007), and Spirit (Oliner and Stearley, 2007). These datasets are labeled with log types and exhibit substantial differences in log formats, enabling us to assess the model's generalization performance. The specific distribution of logs is outlined in Table 2.

Table 2
Number of log messages in each dataset.

Dataset	Category	Numbers	Anomalies
HDFS	Distributed system	11,175,629 (575,061 blocks)	16,838 blocks (2.93%)
BGL	Supercomputer	4,747,963	348,469 (7.34%)
Thunderbird	Supercomputer	10,000,000	4934 (0.49%)
Spirit	Supercomputer	5,000,000	764,500 (15.29%)

HDFS dataset was generated from over 200 Amazon EC2 nodes, consisting of a total of 11,175,629 log messages. Based on unique block_ids, all log messages were segmented into 575,061 blocks, forming log message sequences, each of which was labeled as normal or abnormal. Among these, 16,838 (2.93%) log blocks are considered to indicate system anomalies.

BGL (Blue Gene/L) dataset is a supercomputing system log dataset collected by Lawrence Livermore National Labs (LLNL). It contains 4,747,963 log messages, each of which is manually labeled as normal or abnormal. Among these, 348,460 (7.34%) log messages are labeled as abnormal.

Thunderbird dataset is an open log dataset collected from the Thunderbird supercomputer at Sandia National Labs (SNL). The dataset contains manually labeled normal and abnormal log messages. For computational feasibility, we use a subset of 10,000,000 continuous log messages from the original dataset of over 200 million messages. Among these, 4934 (0.49%) are labeled as abnormal.

Spirit dataset is a supercomputing system log dataset collected from the Spirit supercomputer at Sandia National Labs (SNL). The original dataset consists of over 172 million log messages, each labeled as either normal or abnormal. For this study, we use a subset of 5,000,000 log messages, of which 764,500 (15.29%) are labeled as abnormal.

4.2. Model parameters

In the following experiments, LogMeta was instantiated with specific parameter configurations. The hidden size for RoBERTa and input layers were both designated as 768. A learning rate of 2e-5 was selected for training. Regarding batch sizes, we employed 32 for the HDFS dataset and 64 for the BGL dataset. The vocabulary size was defined as 30522. Subsequently, the Bi-LSTM's hidden size was set to 256 with 4 hidden layers and 8 attention heads. Notably, the maximum sequence length was specified as 128. During the training process, the CrossEntropyLoss function was utilized in conjunction with the Adam optimizer.

To ensure robustness and mitigate randomness, we conducted each experiment five times and reported the average results. All experiments were performed on a system running Windows 11 with an Intel(R) Core(TM) i7-12700 CPU, 64GB RAM, and an NVIDIA RTX A4000 GPU.

4.3. Research questions

This paper mainly focuses on the following four research questions and designs our experiments accordingly.

- **RQ1: Can the proposed LogMeta model outperform existing models in log anomaly detection?**

  In this experiment, we evaluated the log anomaly detection models on four benchmark datasets, namely HDFS, BGL, Thunderbird, and Spirit, following the dataset selection strategies adopted in prior research (Le and Zhang, 2022). HDFS represents logs from a distributed file system and features a relatively simple and consistent structure. In contrast, BGL, Thunderbird, and Spirit contain more complex and heterogeneous log sequences, often generated by large-scale systems such as supercomputers or embedded systems. These

datasets differ significantly in terms of log volume, template diversity, and structural complexity, thus providing a comprehensive basis for assessing the generalization and robustness of anomaly detection models.

We compared the LogMeta model with six benchmark models. The selected comparative models encompass various mainstream architectures such as BERT, LSTM, and Transformer, among others. These benchmark models encompass unsupervised learning-based models (DeepLog Du et al., 2017), supervised learning-based models (HitAnomaly Huang et al., 2020, NeuralLog Le and Zhang, 2021, LogRobust Zhang et al., 2019), and semi-supervised learning-based models (PLELog Lin and Chiang, 2022; Yang et al., 2021, LogBERT Guo et al., 2021). Detailed descriptions of these benchmark models can be found in Section 4.4. Notably, the experimental outcomes of DeepLog are sourced from the LogBERT paper, while the remaining results are from their respective original research.

- **RQ2: What is the contribution of each individual module within the hybrid language model?**

  In this study, we disassembled each module within LogMeta to evaluate their individual contributions. Specifically, RoBERTa is designated as the first module, Bi-LSTM as the second module, and the attention mechanism as the third module. Given the complexity of the BGL dataset compared to other datasets, the differences in performance among modules are expected to be more pronounced. Therefore, we selected BGL datasets of varying scales for a detailed comparison and analysis.

- **RQ3: How does LogMeta perform in low-resource scenarios against existing models?**

  To systematically evaluate performance in low-resource scenarios, we experimented with varying scales of the HDFS, BGL, and Thunderbird datasets, ranging from 0.1% to 15% of the original size. Specifically, samples of 0.1%, 1%, 5%, 10%, and 15% were used. These datasets were chosen for their diversity in log formats and complexity, making them ideal benchmarks for anomaly detection evaluation. To ensure comprehensive comparisons, we included representative models from different learning paradigms, providing a robust basis for contrasting their performance with LogMeta.

  In addition, we developed an intelligent laboratory management system designed for managing experimental workflows in chemical and materials research. This system, built in Java and utilizing Apache Log4j for logging, logs events in a structured JSON format, enhancing both readability and automated processing. Each log entry contains metadata such as timestamps, log levels, and custom context variables (e.g., user IDs and session details), following the Log4j User's Guide.[2] During a 50 h period, system logs were collected, resulting in a dataset comprising 54,637 entries. Although relatively small compared to datasets from interconnected systems, this dataset represents a low-resource scenario due to the limited number of logs. It also includes more complex and lengthy log sequences, with 371 distinct formats. The following are examples of log contexts at various levels within our software system.

  Note: Certain parts containing user information are replaced with asterisks (*).

  (1) **"level": "Info"**, "context": {"userId": "********", "ipAddress": "**.***.***.***", "sessionId": "*******"}

  (2) **"level": "Warning"**, "context": {"error": "Connection timeout", "attemptNumber": 3, "lastAttemptTime": "2023-05-19T12:30:00Z"}

  (3) **"level": "Error"**, "context": {"class": "OrderProcessor", "method": "processOrder", "exceptionDetails": "java.lang.NullPointerException: Invalid null value"}

  In this dataset, "Info" and "Warning" messages indicate normal logs, while "Error" messages denote anomalies. Of these, 3129 (5.73%) logs were identified as abnormal. The model parameters were configured as described in Section 4.2.

- **RQ4: What Improvements Does MAML Bring to the Anomaly Detection Performance of LogMeta?**

  The generalization capabilities and flexibility of MAML significantly enhance the cross-domain transferability and context adaptability of LogMeta. In this section, we systematically investigate the advantages of integrating MAML into LogMeta for anomaly detection. The experiment is structured into two main parts to comprehensively evaluate the impact of MAML on improving model performance.

  Part 1: Performance Comparison With and Without MAML. The first part of the experiment evaluates the performance of LogMeta with and without the MAML approach, focusing primarily on the benefits of MAML in enhancing domain transferability. By leveraging meta-learning, MAML facilitates the learning of a robust initialization of parameters, enabling LogMeta to adapt efficiently to new log systems with minimal fine-tuning. To assess this, we compare LogMeta's anomaly detection accuracy across diverse log datasets under both configurations—with and without MAML—quantifying the improvements in generalization and adaptability achieved through the use of MAML.

  Part 2: Evaluating Context Adaptability in MAML. The second part of the experiment focuses on the meta-learning pre-training stage, examining how MAML equips LogMeta to adapt to incomplete or suboptimal training data. Specifically, we simulate a scenario where highly relevant datasets are excluded during the meta-learning phase to evaluate the impact on model performance. For instance, during MAML training, we exclude logs from datasets that share structural or semantic similarities with the target dataset (e.g., HDFS dataset). We then evaluate LogMeta's anomaly detection performance on the HDFS dataset, assessing whether the model maintains accuracy when trained on partially related information. This experiment underscores MAML's importance in enabling LogMeta to generalize effectively, even when faced with limited or less relevant contextual data.

  These experiments demonstrate the contributions of MAML in improving LogMeta's performance for anomaly detection by enhancing both domain transfer and context adaptability, critical for handling diverse and evolving log datasets in real-world applications.

- **RQ5: How does LogMeta perform in terms of efficiency?**

  To evaluate the efficiency of LogMeta, we compared its performance with that of four benchmark models—DeepLog, LogRobust, PLELog, and LogBERT—across four widely used benchmark datasets. The efficiency analysis focused on the total time required for each model to complete both the training and testing phases. We specifically chose to analyze the total time for running all four datasets because of the characteristics of MAML, which only requires a single training session on our well-constructed, stable, and diverse training dataset. Specifically, for LogMeta, we specifically measured three time components: (1) **MAML meta-training time:** The time required to perform the meta-learning phase on our diverse training dataset, as described in Section 3.2; (2) **Task-specific fine-tuning time:** The time taken to fine-tune the model on the target dataset after the MAML meta-training (in Section 3.6); and (3) **Testing time:** The time required to evaluate the model on the test set, including the inference time for predicting anomalies in log entries.

  For the other models, we recorded the total time spent on both the training and testing phases. The training time for these models includes the time taken to train each model on their respective datasets, while the testing time reflects the time taken for evaluating the models on the test set. This comprehensive time analysis allows for a fair comparison of the efficiency of each model, considering both the time spent during training and testing stages.

[2] https://logging.apache.org/log4j/2.x/manual/index.html

4.4. Benchmark models

In this section, we present benchmark models for comparison with LogMeta in terms of performance among various log anomaly detection models. The benchmark models were selected based on the use of a log parser and the learning method.

DeepLog (Du et al., 2017) stands as an innovative framework that employs deep learning techniques, notably Long Short-Term Memory (LSTM) networks, to analyze system log data for anomaly detection. It learns the patterns of typical operations from sequential log entries and detects deviations as potential threats or system malfunctions. By emphasizing the sequential aspect of log data, DeepLog accurately predicts and distinguishes between normal and anomalous system behaviors, making it highly effective for security and maintenance tasks in IT systems.

HitAnomaly (Huang et al., 2020) is a supervised model based on a Transformer architecture. It utilizes a specialized log parser for encoding log information as parameters. Employing a standard log parser, it transforms raw log data into a structured format, which is then analyzed using Transformer encoders. HitAnomaly adopts a classification-based approach, separately encoding both normal and abnormal data types to improve its ability to differentiate between them. This method aims to accurately detect anomalies in system logs by focusing on their unique patterns, making it an invaluable tool for maintaining system integrity and security.

NeuralLog (Le and Zhang, 2021) is a supervised classifier model built on the Transformer architecture and incorporates a tokenizer. This strategy empowers the model to learn unique features from raw log data directly without the intermediate step of log parsing, which frequently results in information loss. By integrating a mixture of normal data from the target system and abnormal data from another system, NeuralLog enhances its anomaly detection capabilities, rendering it versatile and efficient for real-world scenarios.

LogRobust (Zhang et al., 2019) is a supervised model built on an attention-based Bi-LSTM architecture. It employs a specialized log parser to preprocess logs, generating TF-IDF and word semantic vectors to extract log features. Both normal and abnormal logs contribute to the training process. The incorporation of attention mechanisms bolsters the model's capacity to pinpoint significant anomalies in log data, thereby enhancing detection accuracy and reliability across various operational contexts.

PLELog (Lin and Chiang, 2022; Yang et al., 2021) introduces a semi-supervised log-based anomaly detection technique that employs probabilistic label estimation to efficiently handle unlabeled log data. This innovative approach integrates semantic information from log events into fixed-length vectors, employs HDBSCAN for clustering these vectors, and utilizes an attention-based GRU network to refine the detection process. Empirical evaluations conducted on datasets such as HDFS and BGL showcase PLELog's robust anomaly detection capabilities, thereby diminishing the necessity for extensive manual labeling.

LogBERT (Guo et al., 2021) is a BERT-based model for anomaly detection, employing MLM and DeepSVDD losses during training. It processes log sequences refined by a Drain parser to learn normal log patterns through tasks such as masked log key prediction and hypersphere minimization. This strategy enables LogBERT to grasp profound contextual relationships within log data, establishing a robust framework for anomaly detection based on deviations from learned log patterns.

4.5. Evaluation metrics

Log anomaly detection is treated as a classification problem, and to assess the effectiveness of models in detecting anomalies, we rely on Precision, Recall, and F1-Score metrics to evaluate model performance. The accuracy metric, which calculates the proportion of all predictions correctly made by the anomaly detection model out of the total number of samples, is not used. This is because, in most datasets, normal logs make up the majority, while anomalous logs constitute only a small portion, which is characteristic of well-qualified software. Consequently, a model could attain a high accuracy score even if it predicts all logs as normal. The definitions of each metric are as follows:

Precision can be regarded as a measure of quality, particularly in binary classification tasks. It quantifies the proportion of correctly identified anomalous log sequences out of all the anomalous log sequences detected by the model. A high precision score signifies that the model is reliable and avoids triggering unnecessary alerts.

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

Recall evaluates the model's capability to identify all instances of the positive class, representing the percentage of log sequences correctly identified as anomalies out of all anomalies. Recall primarily assesses whether the model can capture all positive samples, and a model with a high recall value indicates that it does not overlook many exceptional cases.

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

The F1-score represents the harmonic mean of precision and recall. It serves as a metric for assessing the overall effectiveness of a model, particularly when dealing with datasets characterized by imbalanced positive and negative classes. The term "F1" reflects the equal weighting given to precision and recall, making it a commonly used single criterion for evaluating models in the context of anomaly detection.

$$F1\text{-}score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{9}$$

Specifically, TP (True Positive) is the count of anomalous log sequences correctly detected by the model. FP (False Positive) is the count of normal log sequences incorrectly identified as anomalies. TN (True Negative) is the count of normal logs that are correctly classified. FN (False Negative) is the count of anomalous log sequences that the model failed to detect. The above four states (TP, FP, TN and FN) can be more effectively visualized and represented using a confusion matrix.

5. Results and analysis

5.1. Effectiveness of LogMeta (RQ1)

Setup. In this section, we evaluate the anomaly detection performance of LogMeta by comparing it against six benchmark models across four datasets: HDFS, BGL, Thunderbird, and Spirit, as shown in Table 3. The results demonstrate that both semi-supervised and fully supervised models significantly outperform the unsupervised model DeepLog, emphasizing the critical role of leveraging historically labeled data for effective anomaly detection. Notably, LogMeta consistently achieves superior performance across all datasets, highlighting its robustness, adaptability, and effectiveness in handling diverse log formats and characteristics.

Result. Certain models, such as HitAnomaly and LogRobust, achieve higher performance on the HDFS dataset, while others, like DeepLog, PLELog, and LogBERT, perform better on the BGL dataset. Similar trends are observed on the Thunderbird and Spirit datasets. This discrepancy highlights the inherent challenge of achieving consistent performance across datasets with diverse log formats and structures. Although supervised approaches, such as NeuralLog, which leverage tokenization and Transformer architectures, deliver strong results, they still fall short of the overall performance achieved by LogMeta.

LogMeta, in contrast, demonstrates exceptional F1-scores across the four datasets, consistently outperforming all competing models. For instance, while models like LogBERT and LogRobust exhibit significant variations in performance across the datasets, LogMeta maintains consistently high results, achieving F1-scores of 0.995 on HDFS, 0.996 on BGL, 0.992 on Thunderbird, and 0.993 on Spirit. This stability is largely attributed to the robust parameter initialization provided by MAML,

Table 3
The performance of different models on four datasets.

Model	HDFS			BGL			Thunderbird			Spirit		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
Unsupervised learning-based												
DeepLog	0.884	0.695	0.773	0.897	0.828	0.861	0.484	0.895	0.628	0.438	**1**	0.609
Supervised learning-based												
HitAnomaly	**1**	0.970	0.980	0.950	0.900	0.920	0.950	0.810	0.880	0.920	0.750	0.840
NeuralLog	0.960	**1**	0.980	0.980	0.980	0.980	0.920	**0.990**	0.950	0.919	0.908	0.914
LogRobust	0.980	0.960	0.970	0.910	0.770	0.830	0.610	0.780	0.680	0.985	0.915	0.949
Semi-supervised learning-based												
PLELog	0.950	0.963	0.957	0.965	**0.999**	0.982	0.864	0.941	0.901	0.931	0.767	0.841
LogBERT	0.870	0.781	0.823	0.894	0.923	0.908	0.962	0.965	0.963	0.862	0.807	0.834
LogMeta	0.994	0.996	**0.995**	**0.993**	0.992	**0.993**	**0.993**	**0.990**	**0.992**	**0.994**	0.993	**0.993**

Table 4
F1-scores of LogMeta and baseline models on the same training sets.

Model	HDFS	BGL	Thunderbird	Spirit
DeepLog	0.742 (↓ 3.1%)	0.327 (↓ 53.4%)	0.292 (↓ 33.6%)	0.536 (↓ 7.3%)
HitAnomaly	0.916 (↓ 6.4%)	0.841 (↓ 7.9%)	0.658 (↓ 22.2%)	0.737 (↓ 10.3%)
NeuralLog	0.862 (↓ 11.8%)	0.903 (↓ 7.7%)	0.887 (↓ 6.3%)	0.825 (↓ 8.9%)
LogRobust	0.903 (↓ 6.7%)	0.762 (↓ 6.8%)	0.503 (↓ 17.7%)	0.827 (↓ 12.2%)
PLELog	0.847 (↓ 11.0%)	0.664 (↓ 31.8%)	0.569 (↓ 33.2%)	0.792 (↓ 4.9%)
LogBERT	0.748 (↓ 7.5%)	0.826 (↓ 8.2%)	0.905 (↓ 5.8%)	0.735 (↓ 9.9%)
LogMeta	0.995	0.993	0.992	0.993

which allows LogMeta to effectively adapt to the structural and semantic variations between datasets.

Compared with MetaLog (Zhang et al., 2024) and FreeLog (Zhao et al., 2025), our approach achieves more stable performance across cross-system transfer settings. While MetaLog demonstrates strong results under few-shot supervision, its performance is sensitive to representation quality and system pairs. FreeLog further explores unsupervised domain adaptation with meta-learning; however, its reported cross-domain results still exhibit noticeable performance variation, with F1-scores ranging from approximately 77.55 to 86.01 across different transfers, indicating that robustness under distribution shift remains challenging even with UDA and meta-learning.

Furthermore, LogMeta demonstrates exceptional performance while requiring significantly fewer labeled samples for training. During the fine-tuning phase, it utilizes only 1% of labeled data from each dataset, yet consistently outperforms three supervised models (LogRobust, NeuralLog, and HitAnomaly) across all four datasets. In contrast, models like PLELog and LogBERT, despite their strong performance, heavily rely on extensive labeled data—PLELog requires all labeled normal logs for training, while LogBERT extensively leverages normal logs during self-supervised learning. This highlights LogMeta's remarkable efficiency in achieving superior detection performance with minimal reliance on labeled data, significantly reducing the burden of manual annotation and computational overhead. These characteristics position LogMeta as a benchmark for label-efficient log anomaly detection, making it particularly suitable for deployment in environments with scarce labeling resources or heterogeneous log formats.

To provide a more fair comparison between LogMeta and other baseline models, we standardized the training datasets. We used the same dataset for training as LogMeta, which consists of a stable and diverse training dataset for MAML, as well as 1% of each target dataset used for task-specific fine-tuning. It is important to note that the size of the dataset used in this experiment is considerably smaller than the original training datasets used by the baseline models, particularly for supervised learning-based models.

The results presented in Table 4 demonstrate that, when trained on the smaller dataset used by LogMeta, the performance of baseline models exhibits varying degrees of degradation, particularly for datasets where they initially performed slightly worse. For instance, LogRobust, which originally achieved an F1-score of 0.680 on the Thunderbird dataset—its worst-performing dataset among the four—suffered a 17.7% decrease in F1-score when trained on our smaller dataset. This performance drop was more significant on datasets where LogRobust's original performance was already lower. In contrast, LogMeta maintained an exceptional F1-score of above 0.99 across all four datasets, further demonstrating the superior generalization ability of our approach, even with fewer labeled samples for training.

Answer to RQ1: LogMeta consistently outperforms state-of-the-art models across all four datasets, showcasing exceptional generalization capabilities across diverse log formats. Its ability to achieve high performance while efficiently utilizing minimal labeled data highlights its practicality for real-world anomaly detection tasks, particularly in scenarios characterized by heterogeneous log structures and constrained availability of anomaly logs.

5.2. Contribution of each module within hybrid language model (RQ2)

Setup. In this section, we disassembled each module within LogMeta to evaluate their contribution. As detailed in Table 5, Line 1 is the full model of LogMeta, Lines 4 and 5 are designated to explore the enhancement effects of the RoBERTa module, Line 2 is used to evaluate the improvement of the Bi-LSTM module, and Line 3 is used to investigate the enhancement effects of the attention mechanism. Given the complexity of the BGL dataset compared to other datasets, the differences in performance among modules are expected to be more pronounced. Therefore, we selected BGL datasets on various scales 1%, 10%, 20%, 50%, and 100% to carry out more extensive experiments.

Result. Overall, each module of LogMeta plays a critical role in enhancing the model's overall performance. Through systematic evaluation, we observed that as the dataset size increased, F1-scores consistently improved, highlighting the importance of having sufficient data to support robust anomaly detection. Additionally, the performance degradation of Line 5, when compared to RoBERTa and BERT-based configurations, underscores the superior ability of large-scale pre-trained language models to capture intricate contextual information and semantic relationships within log data. These findings reinforce the effectiveness of incorporating pre-trained models in LogMeta, enabling it to generalize effectively across diverse datasets and achieve state-of-the-art anomaly detection performance.

In terms of horizontal comparison, it is worth emphasizing that when the dataset size is reduced to 1%, the limited number of labeled training logs does not encompass all log variations. Despite this limitation, LogMeta, supported by MAML, demonstrates remarkable resilience, resulting in only a 1.3% decrease in F1-score compared to the 10% dataset

Table 5
The performance of three modules on the BGL dataset.

Module		DataSet Size				
		1%	**10%**	**20%**	**50%**	**100%**
RoBERTa						
(1) + Bi-LSTM + Attention	F1	0.974	0.987	0.991	0.993	0.993
(2) + Attention	F1	0.871 (↓ 10.3%)	0.899 (↓ 8.8%)	0.912 (↓ 7.9%)	0.922 (↓ 7.1%)	0.925 (↓ 6.8%)
(3) + Bi-LSTM	F1	0.963 (↓ 1.1%)	0.973 (↓ 1.4%)	0.981 (↓ 1%)	0.982 (↓ 1.1%)	0.985 (↓ 0.8%)
BERT						
(4) + Bi-LSTM + Attention	F1	0.949 (↓ 2.5%)	0.970 (↓ 1.7%)	0.977 (↓ 1.4%)	0.981 (↓ 1.2%)	0.982 (↓ 1.1%)
(5) Bi-LSTM + Attention	F1	0.762 (↓ 21.2%)	0.825 (↓ 16.2%)	0.848 (↓ 14.3%)	0.847 (↓ 14.6%)	0.832 (↓ 16.1%)

scale. In contrast, Lines 2 to 5 exhibit significant performance drops, highlighting the limitations of simpler or less robust architectures when handling diverse log patterns and limited training data.

For vertical comparison, replacing or removing the first module with BERT, instead of leveraging RoBERTa, results in average F1-score improvements of 1.58% and 16.48%, respectively, when using RoBERTa. This underscores the superior contextual understanding and semantic representation capabilities of RoBERTa compared to standard BERT. Furthermore, introducing the Bi-LSTM module enhances the average F1-score by 8.18%, while adding an attention layer provides an additional 1.08% improvement. At the 1% dataset scale, LogMeta achieves F1-scores approximately 1.1% and 2.5% higher than Lines 3 and 4, and demonstrates substantial improvements of 10.3% and 21.2% over Lines 2 and 5, respectively. These results underscore the importance of integrating large-scale pre-trained language models and highlight LogMeta's ability to deliver robust performance even with limited labeled data.

At the 10% dataset scale, the F1-scores of Lines 2 and 5 exhibit only marginal increases of 1.5% and 0.8%, respectively, compared to the 1% dataset size, indicating limited improvements. In contrast, LogMeta's hybrid architecture consistently achieves high performance across dataset scales, demonstrating its adaptability and robustness. As the dataset size increases to 20%, all models tend to stabilize in their F1-scores. This trend further highlights LogMeta's superior generalization capabilities across varying data volumes, enabling it to deliver exceptional anomaly detection performance even in small-scale and challenging scenarios.

Answer to RQ2: Each component of LogMeta plays a vital role in enhancing log anomaly detection performance, with the RoBERTa module (Module 1) making the most substantial contribution by effectively extracting meaningful log features and maintaining robust performance, even on small-scale datasets.

5.3. Exploration in low-resource scenarios (RQ3)

Setup. To systematically evaluate the performance of our proposed LogMeta model in low-resource scenarios, we conducted experiments on varying scales of the HDFS, BGL, Thunderbird, and our system-generated dataset (as outlined in Section 4.3), ranging from 0.1% to 15%. For the HDFS, BGL, and Thunderbird datasets, experiments were performed at 0.1%, 1%, 5%, 10%, and 15% scales. Given that our system-generated dataset comprises only 54,637 entries, which inherently qualifies as a low-resource dataset, we conducted experiments at scales of 10%, 50%, and 100%.

To ensure a comprehensive comparison, we selected representative models from different learning paradigms: DeepLog (unsupervised), PLELog (semi-supervised), and LogRobust (supervised). The experimental results, highlighting the performance of LogMeta alongside these models under various low-resource conditions, are presented in Fig. 4.

Result. In summary, due to the integration of MAML, LogMeta demonstrates stable and exceptional performance across four datasets with differing log formats, effectively addressing the challenges of data heterogeneity and limited labeled resources. As shown in Fig. 4, reducing the dataset size to scales between 15% and 0.1% leads to significant F1-score drops for models such as DeepLog, PLELog, and LogRobust. Specifically, DeepLog and PLELog exhibit average F1-score decreases of 19.7%–28.4% and 3.5%–20.5%, respectively, across the HDFS and BGL datasets. Similarly, the fully supervised LogRobust model experiences a maximum performance drop of 21.6%. Notably, the performance differences on the other two datasets are even more pronounced, further underscoring the challenges faced by these models in generalizing across diverse datasets with varying log structures. In contrast, LogMeta maintains remarkable robustness, with an average F1-score reduction of only 6.2% across all four datasets, even when the dataset size is at its smallest scale. This highlights LogMeta's ability to deliver strong anomaly detection performance in low-resource scenarios. At the 10% dataset scale, LogMeta surpasses other models, achieving average F1-score improvements of 46.7% over DeepLog, 21.4% over PLELog, and 16.8% over LogRobust across all datasets.

In detail, in low-resource scenarios, most models exhibit instability in detecting anomalies. For instance, as illustrated in Fig. 4(a), at a dataset size of 0.1%, LogMeta achieves F1-score improvements of 68.4%, 33.4%, and 30.9% over DeepLog, PLELog, and LogRobust, respectively. This demonstrates LogMeta's superior ability to generalize log features and maintain high detection performance, even with extremely limited data. While all models show noticeable F1-score improvements as the dataset size increases, LogMeta exhibits the least variability. At the 5% dataset scale, DeepLog, PLELog, and LogRobust improve their F1-scores by 29.5%, 17.8%, and 44.9%, respectively, compared to the 0.1% scale. LogMeta, in contrast, achieves a smaller but steady improvement of 1.4%, highlighting its resilience and consistency across smaller datasets. At the 10% scale, the F1-scores of all models stabilize, but LogMeta maintains its leading performance with minimal fluctuations.

On the BGL and Thunderbird datasets, LogMeta demonstrates a distinct advantage, particularly in low-resource scenarios. As shown in Fig. 4(b), at the 0.1% dataset scale, LogMeta achieves an F1-score of 0.962, outperforming PLELog, the second-best model, by 33%. In contrast, LogRobust struggles to generalize effectively, achieving an F1-score of only 0.627, highlighting the limitations of fully supervised models when dealing with complex log structures and limited labeled data. As the dataset size increases, LogMeta continues to show stable performance improvements. At the 15% scale, it achieves an F1-score of 0.990, surpassing PLELog by 13.5%. Meanwhile, LogRobust exhibits instability, with its F1-score dropping 3.4% at the 15% scale compared to the 10% scale. A similar trend is observed on the Thunderbird dataset, as illustrated in Fig. 4(c). At the 0.1% dataset scale, LogMeta achieves an F1-score approximately 50.1% higher than the other three models on average, demonstrating its exceptional predictive and generalization capabilities in low-resource settings.

Fig. 4(d) presents the performance of various models on our system-generated log dataset, designed for managing experiment workflows in the chemical and materials fields. Due to the dataset's limited size (54,637 entries), training data is inherently constrained, resulting in lower F1-scores compared to larger datasets. At the 10% scale, all models experience significant performance degradation, with the

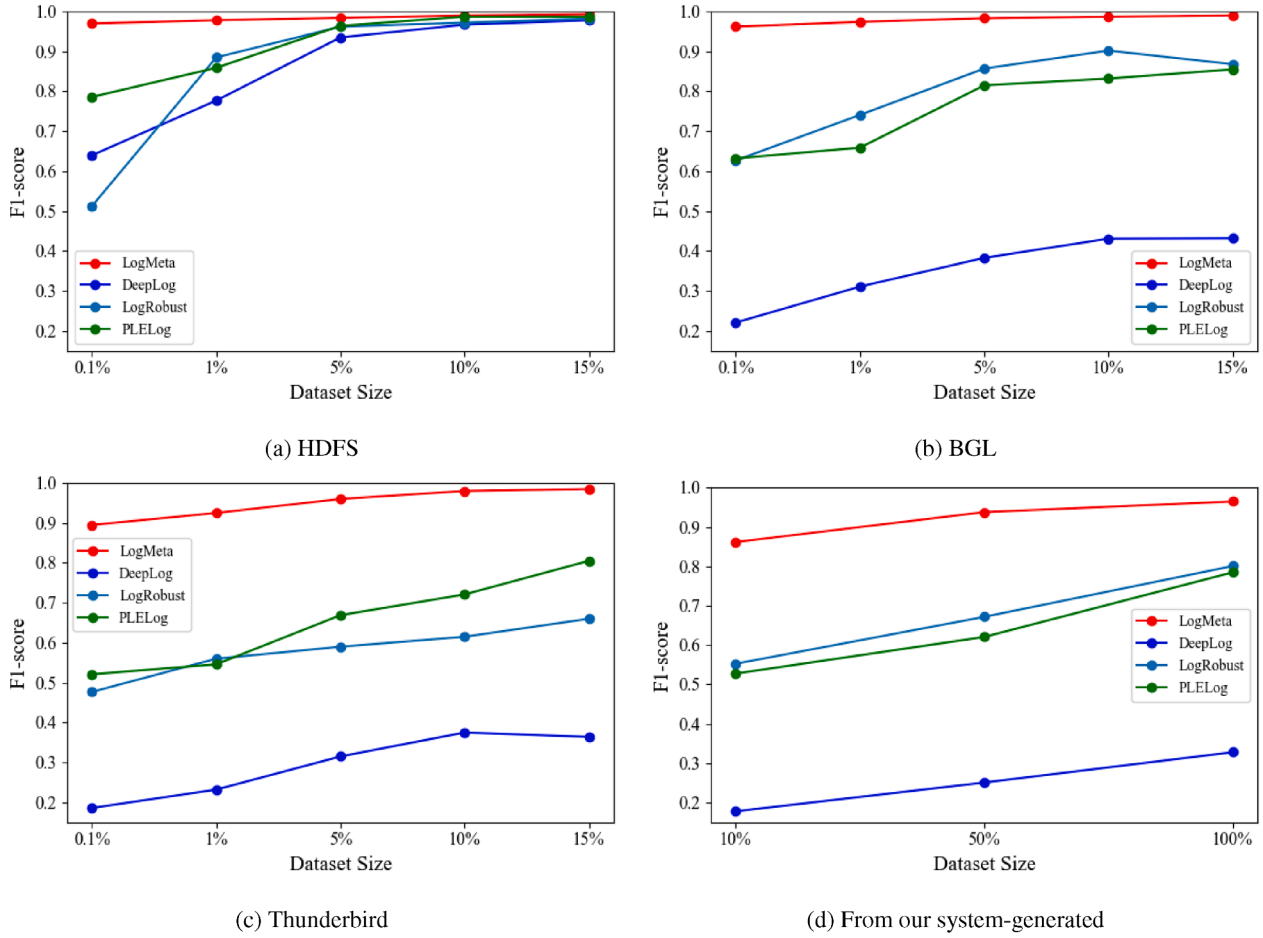

Fig. 4. The performance of models on three low-resource datasets.

unsupervised DeepLog performing the worst. LogMeta, however, significantly outperforms all other models, achieving an average F1-score improvement of 44.2% over competing models. Even as the dataset size increases to 50% and 100%, LogMeta maintains its leading position, with average F1-scores 42.3% and 32.7% higher, respectively, than the next-best models. These results demonstrate LogMeta's robustness, adaptability, and scalability across diverse software systems, even in low-resource conditions.

Answer to RQ3: LogMeta consistently outperforms other benchmark models across various learning paradigms, showcasing exceptional robustness and adaptability across diverse data scales. Its ability to deliver superior and consistent performance across datasets with differing log formats highlights its advanced generalization capabilities and effectiveness in handling heterogeneous log structures.

5.4. Contribution of MAML approach (RQ4)

5.4.1. Evaluating MAML's domain transfer effectiveness

In this section, we evaluate the impact of the MAML approach on LogMeta's performance, focusing on its domain transfer capabilities and the benefits of meta-learning in enhancing model adaptability across diverse tasks and datasets. To achieve this, we compare LogMeta's anomaly detection performance with and without MAML across four datasets—HDFS, BGL, Thunderbird, and our system-generated dataset—each exhibiting distinct log formats and characteristics. It is important to note that all other experimental settings remain consistent, including the use of 1% of labeled logs from each dataset for fine-tuning or training. The results, summarized in Fig. 5, highlight the performance differences and demonstrate the significant advantages of integrating MAML into LogMeta, enabling robust and adaptive anomaly detection even in heterogeneous log environments.

To summarize, integrating MAML into LogMeta significantly enhances its performance across diverse datasets, particularly by improving generalization capabilities and effectively addressing data imbalance. Notably, LogMeta demonstrates exceptional robustness, achieving an average F1-score reduction of only 6.2% across all four datasets, even at the smallest dataset scales.

In detail, when reducing the dataset size to scales between 15% and 0.1%, significant F1-score drops are observed in models without MAML. For instance, on the HDFS dataset, LogMeta with MAML experiences only a 2.3% decline in F1-score, compared to a 5.3% decline in its non-MAML counterpart, reflecting an improvement of 3.0%. The advantage of MAML is particularly evident on the BGL and Thunderbird datasets, where LogMeta with MAML achieves F1-score improvements of 6.3% and 4.9%, respectively, compared to its non-MAML counterpart. Similarly, on our system-generated dataset, LogMeta with MAML maintains its robustness with a 3.4% performance improvement.

LogMeta with MAML demonstrates exceptional performance on low-resource datasets, highlighting its strong generalization ability and adaptability under conditions of limited labeled data. For instance, on the HDFS dataset, at a data scale of 0.1%, LogMeta achieves an F1-score of 0.970, which is 3.8% higher than its counterpart without MAML. Similarly, on the BGL and Thunderbird datasets, LogMeta outperforms the version without MAML by 7.1% and 7.8%, respectively, demonstrating a significant improvement. Furthermore, on our system-generated dataset, where the total dataset size is inherently small (containing 54,637 log entries), LogMeta achieves a remarkable 5.8% improvement in F1-score at the 10% scale compared to the non-MAML version, using

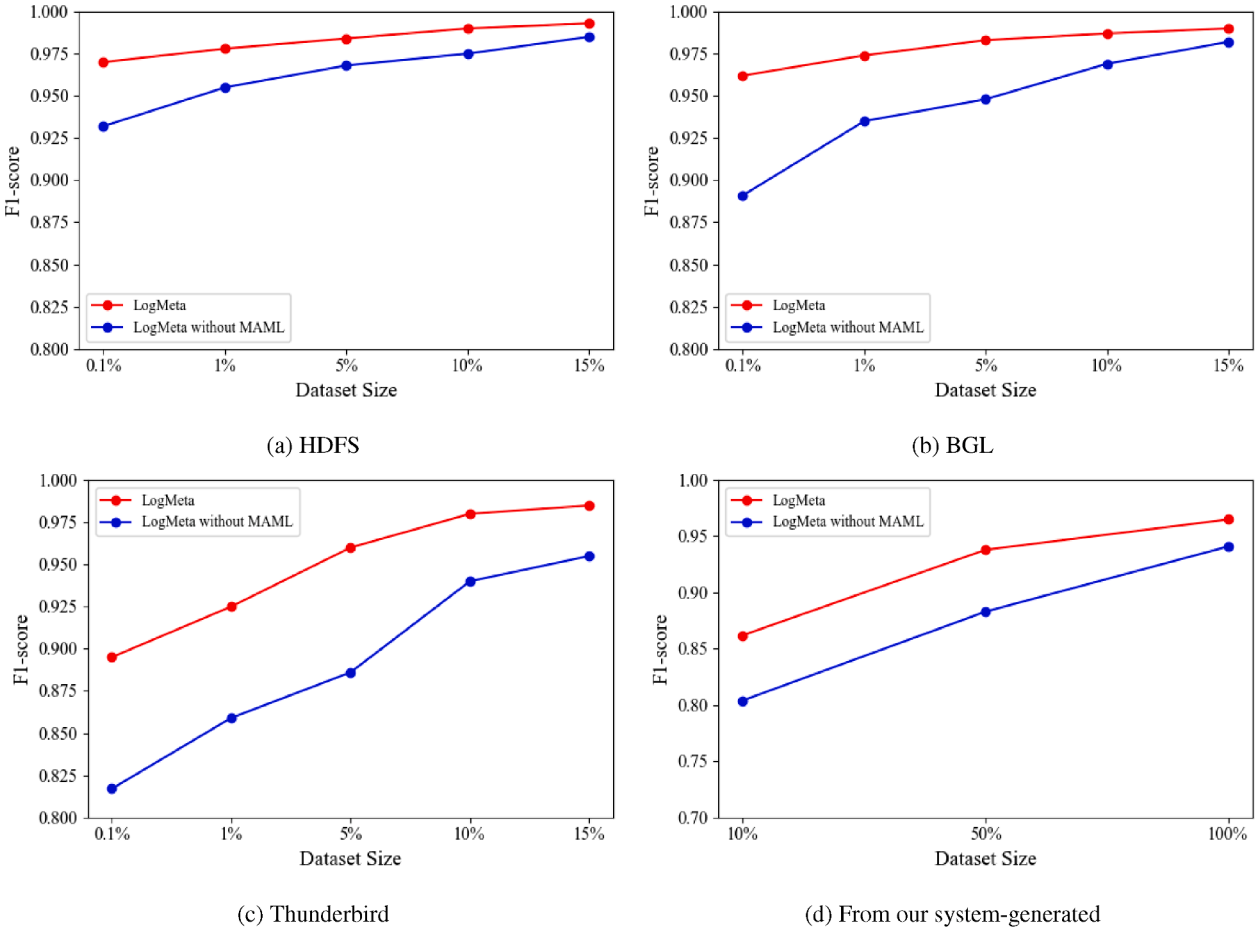

Fig. 5. The performance of models with and without MAML on four datasets.

only 5000 logs for training. These results underscore the effectiveness of integrating MAML for handling low-resource datasets, as it equips LogMeta to quickly adapt and extract meaningful log features with minimal labeled samples. This versatility enables LogMeta to handle both large-scale log data from complex systems and low-resource scenarios, positioning it as a robust and practical solution for diverse log anomaly detection tasks.

Answer to RQ4 (1): By leveraging MAML, LogMeta significantly enhances generalization, adaptability, and robustness, achieving superior performance across diverse datasets and demonstrating exceptional effectiveness in low-resource and imbalanced data scenarios.

5.4.2. *Assessing MAML's context adaptation capabilities*

In this section, we evaluate the impact of MAML on LogMeta's ability to adapt to incomplete or suboptimal training data, highlighting its role in enhancing context adaptability. To simulate challenging scenarios, we deliberately exclude datasets with structural or semantic similarities to the target dataset during the meta-learning phase. For instance, logs from datasets similar to HDFS are omitted during MAML pre-training, and the model's anomaly detection performance is then evaluated on the HDFS dataset. A similar experimental setup is applied to the BGL, Thunderbird, and our system-generated datasets. This approach allows us to assess whether LogMeta can maintain high detection accuracy even when trained on partially related information. The results, as illustrated in Fig. 6, demonstrate the robustness of LogMeta in effectively generalizing to unseen log data, further validating the benefits of incorporating MAML into the framework.

To summarize, LogMeta with MAML consistently achieves strong anomaly detection performance, even when structurally or semantically similar datasets are excluded during the pre-training phase. At the smallest dataset scales, we observe a slight drop in F1-scores across all four datasets when similar data is excluded during pre-training. Specifically, the average F1-scores decrease by approximately 2.16%, emphasizing the importance of including similar datasets in the MAML pre-training phase for optimal performance. Despite this difference, LogMeta with MAML maintains consistently high F1-scores, showcasing its robustness and adaptability even under suboptimal conditions with limited contextual information.

As the dataset size increases to 5% (or 50% for our system-generated dataset), the performance gap narrows significantly, with the average F1-scores decreasing by only 0.63%. This result suggests that LogMeta, even with less relevant pre-training data, can effectively compensate for missing contextual information through subsequent fine-tuning, further emphasizing MAML's role in providing a strong parameter initialization.

At larger dataset scales, the F1-scores of LogMeta with and without similar pre-training data converge, showing virtually no performance difference. This demonstrates that once sufficient data coverage is available, LogMeta can achieve stable and consistent anomaly detection performance, regardless of the absence of similar datasets during the pre-training phase.

Answer to RQ4 (2): While including similar datasets during MAML pre-training provides a slight performance boost on extremely small datasets, LogMeta's robust design and effective fine-tuning enable it to achieve exceptional performance and strong generalization across all dataset scales.

5.5. *Efficiency of logmeta (RQ5)*

Setup: To assess the efficiency of LogMeta, we measured the total duration for each model to complete the entire training and testing process

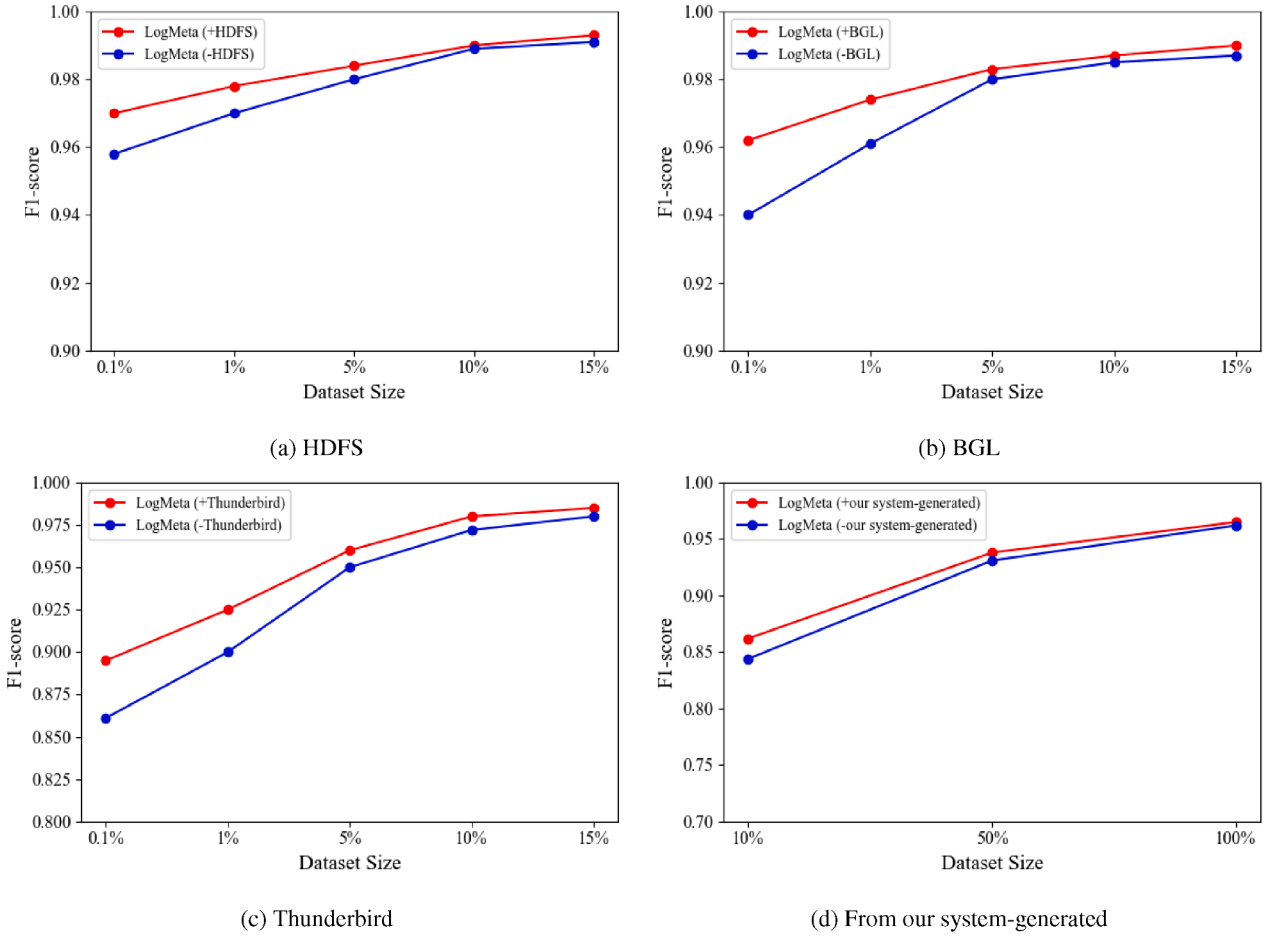

Fig. 6. The impact of MAML pre-training with and without highly relevant dataset on model performance.

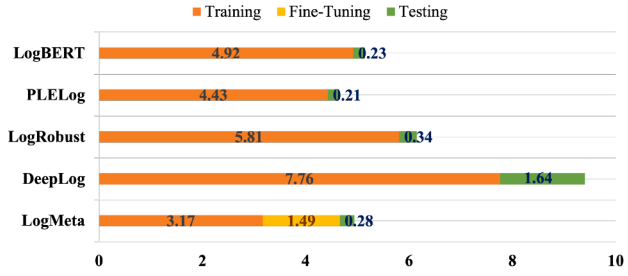

Fig. 7. Overall total duration (in hours) for anomaly detection across all four datasets.

on four benchmark datasets. Specifically, we recorded the time taken for LogMeta to complete the MAML meta-training, task-specific fine-tuning, and testing phases. For comparison, we also measured the total time taken by four benchmark models: DeepLog, LogRobust, PLELog, and LogBERT. As shown in Fig. 7, the efficiency analysis presents the total time required for each model, highlighting the differences in time consumption across the models.

Results: The experimental results demonstrate the efficiency advantages of the semi-supervised model, LogMeta, particularly when compared to fully-supervised and unsupervised models. The total duration for LogMeta is on par with PLELog and LogBERT, both of which require similar time expenditures during training and testing. Specifically, LogMeta's MAML meta-training time is 3.17 h, which corresponds to an average of 0.29 s per log entry.

During the task-specific fine-tuning phase, LogMeta spent more time due to the choice of using 1% of the data from each dataset for fine-tuning. Since the dataset size directly impacts time consumption, the meta-training and task-specific fine-tuning times of LogMeta's MAML are comparable to the training times of PLELog and LogBERT, both of which exhibit similar time expenditures. However, as shown in the results of RQ1, LogMeta still demonstrates a significant advantage in terms of generalization performance. For example, PLELog achieves an F1-score of 0.841 on the Spirit dataset, while LogBERT performs at 0.823 on the HDFS dataset. In contrast, LogMeta consistently outperforms these models, achieving an F1-score of greater than 0.99 across all four datasets.

In real-world applications, particularly in systems with high-frequency log generation, both training and inference efficiency are crucial. LogMeta's ability to achieve high accuracy while maintaining fast inference speeds makes it a strong candidate for real-time deployment. Additionally, although the training process includes a one-time meta-training phase, it can be efficiently managed in a controlled batch setting, making it suitable for industrial applications. This suggests that while LogMeta's time cost is comparable to other models, its superior generalization capability makes it a more effective choice for practical log anomaly detection. The model's high performance across diverse datasets further emphasizes the effectiveness of its semi-supervised learning approach, which balances efficiency with accuracy.

Answer to RQ5: Although LogMeta's total time expenditure is comparable to that of PLELog and LogBERT, its superior generalization performance across all four datasets highlights its clear advantage in both efficiency and accuracy for log anomaly detection.

Table 6
Categorization of FN and FP in LogMeta's anomaly detection.

Pattern (FN/FP)	Example
Parser Incorrectly Parsed Keyword (FN)	ciod: LOGIN chdir(/p/gb2/glosli/8M_5000K/t800) failed: No such file or directory (Parser error, "failed" keyword was incorrectly parsed out)
Data Source Connection Errors (FP)	data_thread() got no answer from any [Thunderbird_A4] datasource
Physical Dimension Mismatches (FP)	ciod: Z coordinate 32 exceeds physical dimension 32
Node Configuration Mismatches (FP)	Node card VPD check: U11 node in processor card slot J18 do not match
Assertion Failures in Code (FN)	idoproxydb hit ASSERT condition: ASSERT expression = 0
Instruction Cache Errors (FP)	instruction cache parity error corrected
Unresolved Data Sources (FP)	Warning: we failed to resolve data source name cn782 cn783 cn784 cn785

6. Discussion

Across all experimental settings, including cross-system transfer and limited-label scenarios, LogMeta consistently demonstrates strong generalization performance and reduced sensitivity to domain shifts. The results indicate that learning transferable representations and adaptation strategies is critical for achieving stable log anomaly detection across heterogeneous systems.

Recent meta-learning–based approaches, such as MetaLog (Zhang et al., 2024) and FreeLog (Zhao et al., 2025), have demonstrated the potential of leveraging cross-task knowledge for log anomaly detection; however, their effectiveness is still influenced by task-specific assumptions. MetaLog applies meta-learning on top of event-level representations that rely on log parsing and static semantic embeddings, which may limit robustness when logs exhibit complex semantic variations or evolving formats across systems. FreeLog further combines unsupervised domain adaptation with meta-learning to reduce dependence on labeled data; nevertheless, its reported results show noticeable performance variability across domain pairs, with F1-scores ranging from approximately 77.55 to 86.01, suggesting that distribution shift remains a challenge even under UDA and meta-learning settings. In contrast, our framework performs meta-learning over a contextualized hybrid language model, enabling joint adaptation of semantic representations and sequential dependencies. This design better aligns with the practical characteristics of log anomaly detection—such as structural heterogeneity, semantic evolution, and limited supervision—resulting in more stable and consistent cross-system performance.

Although LogMeta employs a hybrid language model combined with meta-learning, its design remains compatible with practical large-scale log anomaly detection scenarios. The computationally intensive meta-training stage is a one-time, offline process that can be conducted in a controlled batch environment using historical logs from multiple systems. Once meta-training is completed, task-specific adaptation requires only a small amount of data and limited fine-tuning, which significantly reduces the computational burden compared to training models from scratch. Furthermore, our experimental results show that the overall end-to-end time consumption of LogMeta is comparable to existing deep learning–based baselines that rely on larger volumes of training data, indicating that architectural complexity does not necessarily translate into prohibitive runtime costs. By decoupling offline meta-training from online anomaly detection, LogMeta supports scalable deployment in high-throughput production environments, where robustness to cross-system variability and limited supervision is often more critical than minimizing per-sequence inference latency alone.

Our experiments reveal that while LogMeta achieves high accuracy in anomaly detection, some edge cases result in undetected anomalies (false negatives) and misclassifications of normal logs as anomalies (false positives). The categorization of these missed and falsely detected anomalies provides critical insights into the model's limitations. In this section, we extract the misclassified logs from the experimental results and conduct an additional analysis. We provide a detailed examination of the false negatives (FN) and false positives (FP) generated by LogMeta, as shown in Table 6. We also discuss the underlying reasons for these failures and offer insights into the limitations of the proposed approach. This failure analysis not only aids in understanding the current challenges faced by LogMeta but also informs future improvements to enhance its robustness and detection capabilities.

In the case of false negatives, we identified several patterns where LogMeta failed to detect certain anomalies. One such issue involves incorrect parsing of keywords, particularly when critical failure indicators such as "failed" are either omitted or misinterpreted. For instance, in the example "*LOGIN chdir(/p/gb2/glosli/8M_5000K/t800) failed…*", the parser incorrectly parsed out the "failed" keyword. This is an example of how noise in the data can lead to the omission of important indicators, rendering the anomaly undetected. Although this happens infrequently, it poses a challenge for LogMeta, particularly when dealing with complex log structures where failure conditions may not always follow predictable patterns.

Furthermore, some anomalies are missed due to their similarity to normal logs. For instance, logs that contain normal operational information but also include keywords resembling failure conditions may be falsely ignored. This is the case with certain physical dimension mismatches and node configuration mismatches, which can sometimes appear as normal system behavior but are flagged as anomalies in other cases. Such overlap can lead to missed detections, particularly when similar patterns appear across different datasets.

On the other hand, false positives arise when normal logs are incorrectly classified as anomalies. The most common cause of false positives in LogMeta is the misidentification of normal logs with patterns resembling anomalies. For example, a regular system warning might be misclassified as an anomaly due to the presence of certain keywords like "error", "failed", or "warning", even though the log does not indicate any actual system failure. An example of this can be seen in the "*instruction cache parity error corrected*" log, where a system-level cache error is flagged as an anomaly despite being a routine correction.

Additionally, data source connection issues such as "*data_thread() got no answer from any [Thunderbird_A4] datasource*" can also trigger false positives. These types of logs, although often critical in practice, do not always correspond to system anomalies but are flagged as such due to the model's over-sensitivity to certain error phrases in the logs. The detection of these false positives is a challenge, especially when dealing with distributed systems where connection timeouts or minor failures are frequent but not necessarily indicative of critical anomalies.

Another challenge for LogMeta is the model's overfitting to certain keywords that may appear in logs. For example, "*Node card VPD check: U11 node in processor card slot J18 do not match*" could be flagged as an anomaly, even though it may reflect normal behavior in specific systems, such as when nodes are being upgraded or replaced. Similarly, warnings about system configuration mismatches, which are routine in some systems, might be misclassified as critical anomalies if the model interprets them as deviations from normal behavior. These issues stem from the model's reliance on keyword matching, which can sometimes lead to overfitting to certain patterns and misclassification.

The ability of LogMeta to generalize across diverse log formats is another factor contributing to the challenges of both false positives and false negatives. In our experiments, we observed that logs from

heterogeneous systems, such as unresolved data sources, which refer to missing connections or unresolved queries in a distributed system, can confuse the model if the logs have different formatting or reporting conventions. The model's generalization ability is influenced by the training data's diversity, and certain log formats or terminologies may not be well-represented in the training phase, leading to misclassifications.

In conclusion, while LogMeta performs well overall, the issues discussed above demonstrate that there are still challenges to overcome in terms of both false positives and false negatives. Future work will focus on improving the model's robustness by enhancing its ability to handle noisy data, better distinguishing between normal system behavior and true anomalies, and refining its handling of edge cases. Additionally, more diverse training data, including logs from various systems with different reporting formats, will be incorporated to improve LogMeta's generalization capabilities.

7. Threats to validity

This section clarifies the threats to the dependency on preprocessing tools, privacy issue, randomness in DPP construction, inconsistency in manual annotations, and computational complexity and scalability, respectively.

7.1. Dependency on preprocessing tools

From a practical perspective, the reliability of log anomaly detection is closely tied to the quality of log parsing. In this study, we adopt an LLM-based parser as one instantiation due to its reported advantages in handling complex and evolving log formats, as well as its superior accuracy compared with traditional rule-based parsers (e.g., AEL, Drain, and UniParser) in recent studies (Jiang et al., 2024a; Huang et al., 2025). However, we emphasize that the proposed framework is not inherently dependent on a specific parsing technique. In operational settings where deterministic behavior, strict real-time constraints, or minimal resource consumption are required, lightweight domain-specific or rule-based parsers remain a practical and viable alternative. This modular design allows the framework to accommodate different parsing strategies according to deployment requirements, ensuring that the core anomaly detection and adaptation mechanisms remain effective across diverse practical scenarios.

7.2. Privacy issue

From an enterprise perspective, system logs often contain sensitive information, including configuration details, access patterns, and network identifiers, which raises legitimate concerns regarding data privacy and confidentiality. To address these issues, LogMeta is designed as a flexible and deployment-agnostic framework that can be trained and operated entirely in local or on-premise environments. This design ensures that proprietary log data does not need to be transmitted to external services during either training or inference, thereby mitigating potential privacy and security risks.

Moreover, while our experimental evaluation includes an LLM-based parsing component for research purposes, this component is not a mandatory dependency of LogMeta. In privacy-sensitive or regulated settings, organizations can replace external or cloud-based models with self-hosted or open-source language models, or adopt deterministic rule-based parsers, without altering the core anomaly detection and meta-learning mechanisms. By decoupling privacy-sensitive preprocessing from the learning framework, LogMeta aligns with common enterprise security practices and regulatory requirements, enabling safe and compliant deployment in real-world environments.

7.3. Randomness in DPP construction

The use of the DPP method to construct the training dataset introduces an element of randomness that may impact the consistency of LogMeta's performance. While DPP is designed to generate diverse and representative subsets of log data, the inherent variability in its sampling process could result in fluctuations in anomaly detection performance metrics. To mitigate this threat, we repeated the DPP sampling process multiple times and averaged the performance results across these runs. Additionally, we ensured that each constructed dataset maintained a consistent proportion of normal and anomalous logs across different sampling iterations, thereby reducing the variability caused by random selection.

7.4. Inconsistency in manual annotations

In log anomaly detection, annotation inconsistency is a common challenge due to the subjective interpretation of log messages and the absence of clear anomaly definitions in many systems. To address this issue while keeping manual effort minimal, we adopt a lightweight two-round cross-annotation process on a small subset of logs. This process is designed not to construct large-scale labeled datasets, but to ensure the reliability of the limited supervision required by our framework.

Specifically, initial discrepancies between annotators are reviewed and resolved through collaborative discussion, followed by a final verification conducted by domain experts. This step serves as a quality control mechanism to reduce labeling noise and improve consistency, rather than an extensive manual labeling procedure. By restricting human involvement to a small and controlled annotation subset, LogMeta balances the need for annotation reliability with the practical goal of minimizing human effort, providing a stable foundation for meta-learning–based cross-system adaptation.

7.5. Computational complexity and scalability

LogMeta leverages both meta-learning and a hybrid language model, which may require significant computational resources, particularly when applied to large-scale log datasets or real-time system monitoring. These computational demands can be alleviated through the use of advanced hardware, such as GPUs or TPUs, and by implementing optimized parallelization techniques. Additionally, as computing technologies continue to advance, newer hardware will make it increasingly feasible to deploy LogMeta efficiently in operational environments, minimizing the impact of computational constraints.

8. Conclusion

In this paper, we proposed LogMeta, a novel semi-supervised log anomaly detection framework that integrates Model-Agnostic Meta-Learning (MAML) and a hybrid language model. Extensive experiments on multiple datasets demonstrated that LogMeta consistently outperforms state-of-the-art benchmark models across varying data scales. The framework exhibits exceptional robustness, cross-dataset generalization, and adaptability to heterogeneous log formats. Notably, LogMeta maintains high anomaly detection performance even in low-resource scenarios, effectively addressing challenges such as limited labeled data and diverse log structures. The integration of MAML enables LogMeta to learn strong parameter initialization, facilitating rapid adaptation to new log systems with minimal labeled data and fine-tuning overhead.

In the future, we aim to further enhance LogMeta by evaluating its performance on broader datasets encompassing more diverse log formats and structures. While the current framework has been successfully deployed in real-world software systems, we plan to assess its scalability and efficiency in large-scale, industry-grade environments. Furthermore, we will explore advanced techniques, such as adaptive prompt engineering for log parsing and self-supervised learning strategies, to

further minimize dependence on labeled data and improve model adaptability.

CRediT authorship contribution statement

Yicheng Sun: Writing – original draft, Software, Methodology, Formal analysis, Data curation, Conceptualization, Validation; **Jacky Wai Keung:** Writing – review & editing, Supervision, Methodology, Funding acquisition; **Hi Kuen Yu:** Writing – review & editing, Validation, Formal analysis; **Wenqiang Luo:** Resources, Project administration.

Data availability

Data will be made available on request.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work is supported in part by the General Research Fund of the Research Grants Council of Hong Kong and the Research Funds of the City University of Hong Kong (6000796, 9229109, 9229098, 9220103, 9229029).

References

Dai, H., Li, H., Chen, C.-S., Shang, W., Chen, T.-H., 2020. Logram: efficient log parsing using *n* n-gram dictionaries. IEEE Trans. Software Eng. 48 (3), 879–892.

Devlin, J., Chang, M.-W., Lee, K., Toutanova, K., 2018. BERT: pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805.

Du, M., Li, F., 2018. Spell: online streaming parsing of large unstructured system logs. IEEE Trans. Knowl. Data Eng. 31 (11), 2213–2227.

Du, M., Li, F., Zheng, G., Srikumar, V., 2017. DeepLog: anomaly detection and diagnosis from system logs through deep learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1285–1298.

Finn, C., Abbeel, P., Levine, S., 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In: International Conference on Machine Learning. PMLR, pp. 1126–1135.

Gao, S., Wen, X.-C., Gao, C., Wang, W., Zhang, H., Lyu, M.R., 2023. What makes good in-context demonstrations for code intelligence tasks with LLMs? In: 2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, pp. 761–773.

Guo, H., Yuan, S., Wu, X., 2021. LogBERT: log anomaly detection via bert. In: 2021 International Joint Conference on Neural Networks (IJCNN). IEEE, pp. 1–8.

He, P., Zhu, J., Zheng, Z., Lyu, M.R., 2017. Drain: an online log parsing approach with fixed depth tree. In: 2017 IEEE International Conference on Web Services (ICWS). IEEE, pp. 33–40.

He, S., Zhu, J., He, P., Lyu, M.R., 2016. Experience report: system log analysis for anomaly detection. In: 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE). IEEE, pp. 207–218.

Huang, J., Jiang, Z., Chen, Z., Lyu, M., 2025. No more labelled examples? An unsupervised log parser with LLMs. Proc. ACM Software Eng. 2 (FSE), 2406–2429.

Huang, S., Liu, Y., Fung, C., He, R., Zhao, Y., Yang, H., Luan, Z., 2020. HitAnomaly: hierarchical transformers for anomaly detection in system log. IEEE Trans. Netw. Serv. Manage. 17 (4), 2064–2076.

Huang, Z., Xu, W., Yu, K., 2015. Bidirectional LSTM-CRF models for sequence tagging. arXiv:1508.01991.

Jiang, Z., Liu, J., Chen, Z., Li, Y., Huang, J., Huo, Y., He, P., Gu, J., Lyu, M.R., 2024a. LILAC: log parsing using LLMs with adaptive parsing cache. Proc. ACM on Software Eng. 1 (FSE), 137–160.

Jiang, Z., Liu, J., Huang, J., Li, Y., Huo, Y., Gu, J., Chen, Z., Zhu, J., Lyu, M.R., 2024b. A large-scale evaluation for log parsing techniques: how far are we? In: Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis, pp. 223–234.

Jiang, Z.M., Hassan, A.E., Flora, P., Hamann, G., 2008. Abstracting execution logs to execution events for enterprise applications (short paper). In: 2008 The Eighth International Conference on Quality Software. IEEE, pp. 181–186.

Khan, Z.A., Shin, D., Bianculli, D., Briand, L., 2022. Guidelines for assessing the accuracy of log message template identification techniques. In: Proceedings of the 44th International Conference on Software Engineering, pp. 1095–1106.

Kulesza, A., Taskar, B., et al., 2012. Determinantal point processes for machine learning. Found. Trends® in Mach. Learn. 5 (2–3), 123–286.

Le, V.-H., Zhang, H., 2021. Log-based anomaly detection without log parsing. In: 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, pp. 492–504.

Le, V.-H., Zhang, H., 2022. Log-based anomaly detection with deep learning: how far are we? In: Proceedings of the 44th International Conference on Software Engineering, pp. 1356–1367.

Le, V.-H., Zhang, H., 2023. Log parsing with prompt-based few-shot learning. arXiv:2302.07435.

Li, X., Chen, P., Jing, L., He, Z., Yu, G., 2020. SwissLog: robust and unified deep learning based log anomaly detection for diverse faults. In: 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE). IEEE, pp. 92–103.

Li, Z., Luo, C., Chen, T.-H., Shang, W., He, S., Lin, Q., Zhang, D., 2023. Did we miss something important? Studying and exploring variable-aware log abstraction. In: 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE). IEEE, pp. 830–842.

Liang, Y., Zhang, Y., Xiong, H., Sahoo, R., 2007. Failure prediction in ibm bluegene/l event logs. In: Seventh IEEE International Conference on Data Mining (ICDM 2007). IEEE, pp. 583–588.

Lin, Y., Chiang, Y.-Y., 2022. A semi-supervised learning approach for abnormal event prediction on large network operation time-series data. In: 2022 IEEE International Conference on Big Data (Big Data). IEEE, pp. 1024–1033.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V., 2019. RoBERTa: a robustly optimized bert pretraining approach. arXiv:1907.11692.

Lou, J.-G., Fu, Q., Yang, S., Xu, Y., Li, J., 2010. Mining invariants from console logs for system problem detection. In: 2010 Usenix Annual Technical Conference (USENIX ATC 10).

Ma, X., Keung, J., He, P., Xiao, Y., Yu, X., Li, Y., 2024a. A semisupervised approach for industrial anomaly detection via self-adaptive clustering. IEEE Trans. Ind. Inf. 20 (2), 1687–1697.

Ma, Z., Chen, A.R., Kim, D.J., Chen, T.-H.P., Wang, S., 2024b. LLMParser: an exploratory study on using large language models for log parsing. In: Proceedings of the 46th IEEE/ACM International Conference on Software Engineering, pp. 1–13.

McInnes, L., Healy, J., 2017. Accelerated hierarchical density based clustering. In: 2017 IEEE International Conference on Data Mining Workshops (ICDMW). IEEE, pp. 33–42.

McInnes, L., Healy, J., Astels, S., 2017. HDBSCAN: hierarchical density based clustering. J. Open Source Softw. 2 (11), 205.

Nagappan, M., Vouk, M.A., 2010. Abstracting log lines to log event types for mining software system logs. In: 2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010). IEEE, pp. 114–117.

Oliner, A., Stearley, J., 2007. What supercomputers say: a study of five system logs. In: 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07). IEEE, pp. 575–584.

Pirelli, S., 2024. Scalable teaching of software engineering theory and practice: an experience report. In: Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training, pp. 286–296.

Qi, J., Huang, S., Luan, Z., Yang, S., Fung, C., Yang, H., Qian, D., Shang, J., Xiao, Z., Wu, Z., 2023. LogGPT: exploring chatgpt for log-based anomaly detection. In: 2023 IEEE International Conference on High Performance Computing & Communications, Data Science & Systems, Smart City & Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys). IEEE, pp. 273–280.

Roumeliotis, K.I., Tselikas, N.D., 2023. ChatGPT and open-AI models: a preliminary review. Future Internet 15 (6), 192.

Shima, K., 2016. Length matters: clustering system log messages using length of words. arXiv:1611.03213.

Sun, Y., Keung, J., Yang, Z., Liu, S., Yu, H.K., 2026. Improving anomaly detection in software logs through hybrid language modeling and reduced reliance on parser. Autom. Software Eng. 33 (1), 12.

Sun, Y., Keung, J., Yu, H.K., Liu, S., Liao, Y., Zhang, J., 2025a. Beyond log parsers: a scalable AI-driven framework for efficient log anomaly detection in software engineering. In: 2025 IEEE 49th Annual Computers, Software, and Applications Conference (COMPSAC). IEEE, pp. 1382–1387.

Sun, Y., Keung, J., Zhang, J., Yu, H.K., Luo, W., Liu, S., 2024. Unveiling hidden anomalies: leveraging SMAC-LSTM for enhanced software log analysis. In: 2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC). IEEE, pp. 1178–1183.

Sun, Y., Keung, J.W., Yang, Z., Liu, S., Liao, Y., 2025b. SemiSMAC: a semi-supervised framework for log anomaly detection with automated hyperparameter tuning. Inf. Softw. Technol. , 107869.

Sun, Y., Keung, J.W., Yang, Z., Liu, S., Yu, H.K., 2025c. SemiRALD: a semi-supervised hybrid language model for robust anomalous log detection. Inf. Softw. Technol. , 107743.

Tang, L., Li, T., Perng, C.-S., 2011. LogSig: generating system events from raw textual logs. In: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, pp. 785–794.

Vaarandi, R., Pihelgas, M., 2015. Logcluster-a data clustering and pattern mining algorithm for event logs. In: 2015 11th International Conference on Network and Service Management (CNSM). IEEE, pp. 1–7.

Vanschoren, J., 2019. Meta-learning. Autom. Mach. Learn. Methods Syst. Challenges , 35–61.

White, J., Hays, S., Fu, Q., Spencer-Smith, J., Schmidt, D.C., 2024. Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design. In: Generative AI for Effective Software Development. Springer, pp. 71–108.

Wu, X., Li, H., Khomh, F., 2023. On the effectiveness of log representation for log-based anomaly detection. Empirical Software Eng. 28 (6), 137.

Xu, J., Yang, R., Huo, Y., Zhang, C., He, P., 2024. DivLog: log parsing with prompt enhanced in-context learning. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering, pp. 1–12.

Xu, W., Huang, L., Fox, A., Patterson, D., Jordan, M.I., 2009. Detecting large-scale system problems by mining console logs. In: Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles, pp. 117–132.

Yang, L., Chen, J., Wang, Z., Wang, W., Jiang, J., Dong, X., Zhang, W., 2021. Semi-supervised log-based anomaly detection via probabilistic label estimation. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). IEEE, pp. 1448–1460.

Yildirim, Ö., 2018. A novel wavelet sequence based on deep bidirectional LSTM network model for ECG signal classification. Comput. Biol. Med. 96, 189–202.

Yu, B., Yao, J., Fu, Q., Zhong, Z., Xie, H., Wu, Y., Ma, Y., He, P., 2024. Deep learning or classical machine learning? an empirical study on log-based anomaly detection. In: Proceedings of the 46th IEEE/ACM International Conference on Software Engineering, pp. 1–13.

Zeufack, V., Kim, D., Seo, D., Lee, A., 2021. An unsupervised anomaly detection framework for detecting anomalies in real time through network system's log files analysis. High-Confid. Comput. 1 (2), 100030.

Zhang, C., Jia, T., Shen, G., Zhu, P., Li, Y., 2024. MetaLog: generalizable cross-system anomaly detection from logs with meta-learning. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering, pp. 1–12.

Zhang, X., Xu, Y., Lin, Q., Qiao, B., Zhang, H., Dang, Y., Xie, C., Yang, X., Cheng, Q., Li, Z., et al., 2019. 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, pp. 807–817.

Zhao, X., Jia, T., He, M., Wu, Y., Li, Y., Huang, G., 2025. From few-label to zero-label: an approach for cross-system log-based anomaly detection with meta-learning. In: Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering, pp. 661–665.

Zhu, J., He, S., He, P., Liu, J., Lyu, M.R., 2023. Loghub: a large collection of system log datasets for ai-driven log analytics. In: 2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE). IEEE, pp. 355–366.

Zhu, J., He, S., Liu, J., He, P., Xie, Q., Zheng, Z., Lyu, M.R., 2019. Tools and benchmarks for automated log parsing. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP). IEEE, pp. 121–130.