CoLA: Model Collaboration for Log-based Anomaly Detection

Xuhang Zhu
Zhejiang University, Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security
zhuxuhang@zju.edu.cn

Xiu Tang*
Zhejiang University, Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security
tangxiu@zju.edu.cn

Sai Wu
Zhejiang University, Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security
wusai@zju.edu.cn

Jichen Li
Zhejiang University, Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security
jichen_li@zju.edu.cn

Haobo Wang
Zhejiang University, Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security
wanghaobo@zju.edu.cn

Chang Yao
Zhejiang University, Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security
changy@zju.edu.cn

Quanqing Xu
OceanBase, Ant Group
xuquanqing.xqq@antgroup.com

Gang Chen
Zhejiang University, Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security
cg@zju.edu.cn

ABSTRACT

Log-based anomaly detection plays a crucial role in ensuring the reliability of systems. While deep learning-based small detection models (SDMs) are efficient, the large language models (LLMs) are accurate and capable of providing explanations. Intuitively, a compelling question arises: Can we seamlessly combine the advantages of both approaches? In this work, we delve into this underexplored research direction and propose CoLA, a novel collaborative log anomaly detection framework. During collaborative inference, an SDM serves as a filter to select potentially anomalous instances, while a downstream LLM acts as an expert to detect anomalies, offer explanations, and refine the SDM. Extensive experiments on three large real-world datasets demonstrate that CoLA significantly outperforms state-of-the-art methods in terms of effectiveness, efficiency, and explainability, while also greatly reducing labor costs.

PVLDB Reference Format:
Xuhang Zhu, Xiu Tang, Sai Wu, Jichen Li, Haobo Wang, Chang Yao, Quanqing Xu, and Gang Chen. CoLA: Model Collaboration for Log-based Anomaly Detection. PVLDB, 18(11): 3979 - 3987, 2025.
doi:10.14778/3749646.3749668

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/skydancer-z/CoLA.

1 INTRODUCTION

With the ever-increasing scale and complexity of systems such as data centers and cloud services, bugs and vulnerabilities have become more prevalent [33, 40], making anomaly detection crucial for system availability and reliability. Logs record system events and states of interest during runtime [2, 8, 17], offering valuable information for engineers to perform troubleshooting. Therefore, log-based anomaly detection (LAD) is promising for system maintenance and has been widely adopted by enterprises [18, 22].

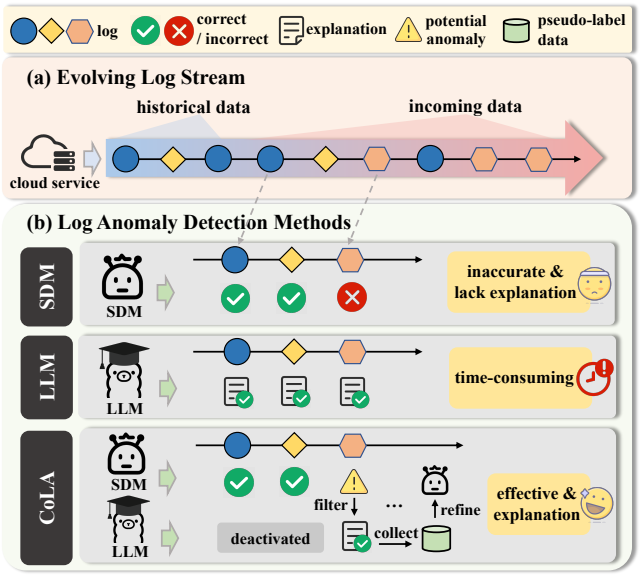

Figure 1: (a) The illustration of evolving log stream in real-world log-based anomaly detection (LAD). (b) Three categories of methods for LAD.

Recently, many deep learning-based small detection models (SDMs) [4, 16, 21, 39] have been proposed to automatically detect system anomalies, obviating the need for manual expert inspection. For instance, DeepLog [4] and LogAnomaly [21] adopt Long Short-Term Memory (LSTM) to learn sequential patterns from normal logs, and LogRobust [39] utilizes semantic vectors to obtain fine-grained log representations. During deployment, SDMs are initially trained on historical data and subsequently employed to perform anomaly detection on incoming data streams. Engineers

*Xiu Tang is the corresponding author.

Proceedings of the VLDB Endowment, Vol. 18, No. 11 ISSN 2150-8097.
doi:10.14778/3749646.3749668

place their hope in statistical coherence, assuming models will keep functioning once deployed, while overlooking the evolving nature of real-world environments. As illustrated in Figure 1(a), the log distribution continuously changes with the system's runtime state, making it challenging for the trained models to remain effective. Moreover, SDMs can only output the probability values of anomalies, without providing corresponding explanations. Consequently, engineers are frequently required to dedicate substantial time to both annotating data for model updates and diagnosing specific underlying issues.

Large language models (LLMs) have demonstrated remarkable capabilities across various tasks [5, 30, 43], with strong generalization abilities and text-based interactions that are easily understandable by users. The success of LLMs brings promising opportunities to log-based anomaly detection, which may enable engineers to obtain accurate and explainable detection results. However, as shown in Figure 1(b), despite their considerable capabilities, the inference of LLMs is costly in terms of time and computational resources. Therefore, given the real-time and large-volume nature of log data, relying entirely on LLMs for timely detection is impractical. To date, it remains questionable how to simultaneously integrate the advantages of LLMs and SDMs for more accurate, explainable, and labor-efficient anomaly detection.

In this work, we propose CoLA, a novel collaborative log anomaly detection framework that harmonizes SDMs and LLMs in a complementary manner. Our intuition is that, while LLMs are inefficient for real-time detection, they are effective detectors and can concurrently offer explanations. On the other hand, SDMs are capable of efficiently providing potentially anomalous instances for further investigation. Additionally, given that the proportion of anomaly instances is notably low [14], it indicates that the majority of instances do not require LLMs for precise detection. As shown in Figure 1(b), to integrate SDMs and LLMs synergistically, we present a collaborative detection paradigm in which the SDM functions as a filter to select potential anomaly instances for further analysis, and the LLM acts as an expert to infer anomalies and refine the SDM. CoLA enables us to achieve an extraordinary trade-off between effectiveness and efficiency, delivering competitive results comparable to those of LLMs, while significantly reducing their inference costs. Moreover, the generated explanations and the automatic updates to the SDM considerably reduce the labor costs.

Specifically, CoLA consists of several key components that support the systematic collaboration mechanism. First, we design an SDM called LogMoE, which incorporates a mixture-of-experts (MoE) approach to learn different aspects of log knowledge, thereby enhancing the model's generalization ability. Second, we adopt a two-stage hybrid training strategy to develop an LLM with domain expertise, termed LAD-LLM. Third, a collaborative inference mechanism is employed, where LogMoE filters suspicious instances, and LAD-LLM provides the judgments along with corresponding explanations. Fourth, by utilizing noisy label learning techniques, LAD-LLM refines LogMoE in a human-free manner.

Overall, our contributions can be summarized as follows:

- We concentrate on a promising yet underexplored research direction, specifically how to effectively combine the strengths of SDM and LLM in practical applications.

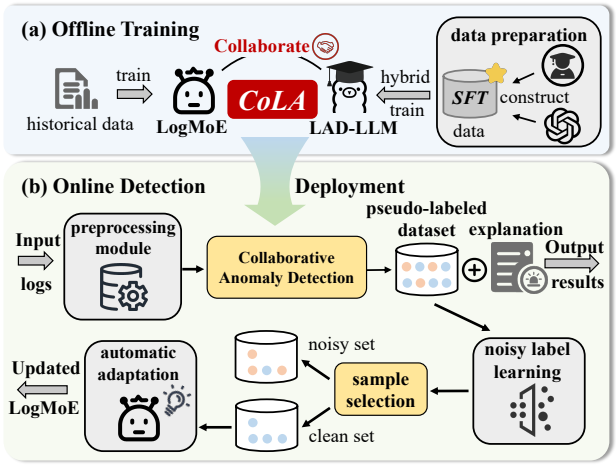

Figure 2: Overview of the CoLA system, consisting of two working stages: offline training and online detection.

- We propose CoLA, a novel collaborative log anomaly detection framework for evolving log streams. It establishes a new paradigm that effectively and efficiently detects anomalies, provides explanations, and alleviates the burden on engineers.
- Extensive experiments on real-world datasets demonstrate the superiority of CoLA in terms of effectiveness, efficiency, output explainability, and labor costs.

2 OVERVIEW

In this section, we first define our problem and then give an overview of our approach.

2.1 Problem Formulation

During system runtime, a log l is generated by the code as a formatted string, recording important information such as timestamps and events. Engineers collect and analyze logs for troubleshooting to ensure system availability.

Problem 1. *Log-based Anomaly Detection (LAD). Given a sequence of system logs $S = [l_1, l_2, ..., l_n]$, where the logs are arranged in chronological order (i.e., each log l_i is generated before l_{i+1}), log-based anomaly detection aims to determine whether the given sequence exhibits anomalous behavior.*

2.2 System Architecture

To achieve an effective and automated log anomaly detection system with output explanations, CoLA is meticulously designed into two working stages: *Offline Training*, and *Online Detection*. Figure 2 provides an architectural overview of CoLA.

Offline Training. The objective of the offline training phase is to obtain an SDM capable of performing initial filtering efficiently and reliably, along with an LLM equipped with domain-specific expertise. For SDM, we propose LogMoE, which incorporates the mixture-of-experts (MoE) framework to utilize specialized experts that capture diverse aspects of the data, thereby enhancing generalization capability to unseen distributions. Additionally, we introduce two regularization strategies: one to ensure balanced expert usage, and the other to encourage diversity in expert capabilities.

For LLM, we introduce LAD-LLM, which excels in log-based anomaly detection while providing user-friendly explanations. Spe-

cifically, the development of LAD-LLM involves two phases: data preparation and hybrid training. Through human-machine collaboration, both real and synthetic data are refined, creating a high-quality and diverse dataset required for model training. Next, we design a two-stage hybrid training strategy to enhance domain-specific capabilities while maintaining the general capabilities.

Online Detection. In this stage, we present how to implement online anomaly detection through data preprocessing, model collaboration and automatic adaptation. When new logs arrive, they are first parsed into event templates and then partitioned into finite sequences. Finally, these sequences are converted into feature vectors for subsequent anomaly detection.

Following log preprocessing, the system conducts anomaly detection via model collaboration. LogMoE efficiently performs initial filtering, classifying logs into potential anomalies and normal instances. Potential anomalies, accounting for a small portion of the data, are sent to the LAD-LLM, where rigorous reasoning is applied to conduct further precise diagnostics and generate explanations in textual form, enabling engineers to swiftly identify underlying errors. To further improve efficiency and reduce generation time, a knowledge reuse strategy is adopted, utilizing a hash table to rapidly index and reuse previously generated results.

During runtime, model adaptation for LogMoE is performed automatically. Specifically, LAD-LLM can be regarded as a specialized annotator for the results produced by LogMoE, enabling the collection of extensive pseudo-labeled data without human labor. Despite the high accuracy of LAD-LLM, a small number of errors still persist. Therefore, we incorporate noisy label learning into the training process to ensure its effectiveness. We employ a parallel training strategy that conducts inference with the latest fine-tuned LogMoE, while automating its offline updates.

3 OFFLINE TRAINING

In this section, we present the design and construction of the efficient filter SDM (LogMoE) and the expert LLM (LAD-LLM), which are key components for the subsequent online detection stage.

3.1 LogMoE: An Efficient and Reliable SDM

As previously discussed, the SDM functions as a filter, enabling rapid and relatively accurate initial detection. Next, we present LogMoE, an SDM meticulously designed for this purpose, trained on historical data, and demonstrates strong generalization capabilities.

3.1.1 Model Architecture Design. The overall design and architecture of LogMoE are shown in Figure 3. Drawing inspiration from the mixture-of-experts (MoE) [27], we leverage multiple experts to capture different aspects of the data's features, thereby improving the model's generalization capability to evolving streams.

First, the preprocessed input log sequence $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ is encoded using a Bi-LSTM, which effectively processes the sequence bidirectionally to produce hidden states $\mathbf{H} \in \mathbb{R}^{n \times d_1}$.

$$\mathbf{H} = \text{Bi-LSTM}(\mathbf{X}), \tag{1}$$

where $\mathbf{H} = \{\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n\}$, n and d_1 are respectively the length of log sequence and the dimension of the hidden state.

Next, a MoE layer is employed, consisting of a gating network and a set of experts. For each encoded log representation $\mathbf{h}_j$, the gating network generates a probability distribution $g(\mathbf{h}_j)$ over the experts, then routes the representation to the top-k experts.

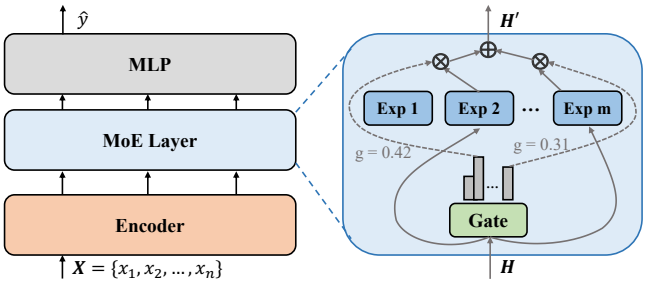

Figure 3: An overview of the LogMoE architecture, where the MoE layer contains m experts and dynamically utilizes the top-k experts for each input.

$$g(\mathbf{h}_j) = \text{Softmax}(W_g \mathbf{h}_j), \tag{2}$$

where $W_g \in \mathbb{R}^{m \times d_1}$ are trainable gating parameters, m is the number of experts. For the i-th expert, $E_i(\mathbf{h}_j)$ consists of a two-layer multilayer perceptron followed by a ReLU activation function, which transforms the input into the output feature space. The MoE-enhanced representation $\mathbf{h}'_j$ is computed as the weighted sum of the selected experts' outputs, based on the gate value $g(\mathbf{h}_j)$.

$$\begin{aligned} E_i(\mathbf{h}_j) &= W^i_{e2} \cdot \text{ReLU}(W^i_{e1} \mathbf{h}_j), \\ \mathbf{h}'_j &= \sum_{i \in \mathcal{T}} g(\mathbf{h}_j)_i E_i(\mathbf{h}_j), \end{aligned} \tag{3}$$

where $W^i_{e1} \in \mathbb{R}^{d_2 \times d_1}$ and $W^i_{e2} \in \mathbb{R}^{d_1 \times d_2}$ are trainable parameters of the i-th expert, d_2 is the hidden layer dimension, and $\mathcal{T}$ denotes the set of selected experts. By applying mean pooling to $\mathbf{H}' = \{\mathbf{h}'_1, \mathbf{h}'_2, ..., \mathbf{h}'_n\}$ and a multilayer perceptron (MLP) classifier, the prediction $\hat{y}$ and cross-entropy loss $\mathcal{L}_{ce}$ are obtained for training:

$$\begin{aligned} \hat{y} &= \text{Softmax}(\text{MLP}(\text{MeanPooling}(\mathbf{H}'))), \\ \mathcal{L}_{ce} &= -[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})], \end{aligned} \tag{4}$$

where y represents the ground truth label for anomaly status.

3.1.2 Auxiliary Losses. To further equip LogMoE, we introduce two auxiliary losses, which are used to ensure balanced expert utilization and enhance the diversity of expert capabilities.

Load Balancing Loss. LogMoE tends to suffer from load imbalance, where the vanilla gating network disproportionately favors a few experts, leaving others underutilized. Such imbalance is reinforced over time [32], as favored experts receive more training, increasing their chance of future selection. To ensure balanced expert utilization, we design a load-balancing loss that penalizes over-reliance on individual experts. Each expert's importance is derived by averaging its gating probabilities across a batch. Given m experts indexed by $i = 1$ to m and a batch $\mathcal{B}$ with T logs, we formulate the load balancing loss $\mathcal{L}_{bal}$ as follows:

$$\begin{aligned} \mathcal{L}_{bal} &= m \sum_{i=1}^{m} f_i G_i, \\ G_i &= \frac{1}{T} \sum_{x \in \mathcal{B}} g_i(x), \\ f_i &= \frac{1}{kT} \sum_{x \in \mathcal{B}} \mathbb{1}(\log x \text{ selects expert } i), \end{aligned} \tag{5}$$

where G_i represents the averaged gating probability for expert i, f_i indicates the proportion of logs assigned to expert i, and $\mathbb{1}(\cdot)$ denotes the indicator function.

Expert Diversity Loss. Experts tend to produce similar representations, limiting the diversity of acquired knowledge and resulting in redundant parameters. To address this redundancy, we introduce an expert diversity loss that maximizes differences among expert representations, thereby promoting richer knowledge acquisition. Specifically, minimum hyperspherical separation (MHS) [19] is employed to maximize the distance among expert weight vectors, and the expert diversity loss $\mathcal{L}_{div}$ is formulated as follows:

$$\max_{\{\hat{\omega}_1,\ldots,\hat{\omega}_m \in \mathbb{S}^{t-1}\}} \{\mathcal{L}_{div}(\hat{W}) := \min_{i \neq j} \rho(\hat{w}_i, \hat{w}_j)\}, \tag{6}$$

where $\hat{W} = \{\hat{\omega}_1, \cdots, \hat{\omega}_m \in \mathbb{S}^{t-1}\}$ denotes the set of expert weight vectors, $\hat{\omega}_i := \frac{\text{vec}(W_i)}{\|\text{vec}(W_i)\|}$ represents the process of vectorizing the i-th expert weight matrix W_i and projecting it onto the unit hypersphere $\mathbb{S}^{t-1} = \{\hat{\omega} \in \mathbb{R}^t \mid \|\hat{\omega}\| = 1\}$, and $\rho(\cdot,\cdot)$ denotes the Euclidean distance between any two vectors on $\mathbb{S}^{t-1}$.

The overall loss function of LogMoE can be formulated as:

$$\mathcal{L}_{LogMoE} = \mathcal{L}_{ce} + \mathcal{L}_{bal} + \mathcal{L}_{div}. \tag{7}$$

With the above design, LogMoE trained on static data can efficiently detect potential anomalies and generalize well.

3.2 LAD-LLM: A Powerful LLM with Domain Expertise

We now elaborate on the development process of LAD-LLM, a specialized LLM tailored specifically for log anomaly detection (LAD). The process comprises two key steps: data preparation and hybrid training, as illustrated in Figure 4.

3.2.1 Data Preparation. We construct training data using both real-world and synthetic data to infuse domain-specific knowledge into the LLM, as shown in Figure 4(a). First, real log sequences are sampled from public LAD training sets (e.g., HDFS), which are labeled by experts. Second, considering that the evolution of logs is driven by system maintenance and running environment [13, 39], we simulate this process using synthetic data to increase the data diversity. Specifically, based on the real-world data, we leverage LLMs and rules to synthesize data (rewrite, duplicate, remove, shuffle), with a volume equal to one-tenth that of the original data. To rewrite the log, LLMs are prompted to make slight modifications to certain expressions in the original log. On the other hand, according to predefined rules, logs in the log sequence are randomly duplicated, removed, or shuffled. These operations do not significantly alter the original semantics, leaving labels unaffected.

After obtaining the initial real and synthetic data, we proceed to investigate explanation generation. Powerful LLMs (such as GPT-4o), are employed to reason and generate preliminary explanations. Next, human experts revise the generated analyses to form accurate explanations. Ultimately, we obtain $D_{log} = \{(Q_i, A_i) \mid i = 1, 2, \ldots, N_d\}$ for LAD-LLM training, where Q_i and A_i denote the question and its corresponding answer, and N_d is the total sample count. To enable LAD-LLM to generate outputs readily callable through interfaces during deployment, we structure A_i as JSON.

3.2.2 Hybrid Training Strategy. After data preparation, we explore how to effectively inject domain knowledge into the LLM. Previous studies [35, 44] indicate that widely used techniques like full fine-tuning and LoRA [12], while improving model performance in new domains, can lead to a degradation of general capabilities, and may even prevent the generation of meaningful text. To address this, we propose a hybrid training strategy, LAD-Mix, which balances log anomaly detection performance with general capabilities.

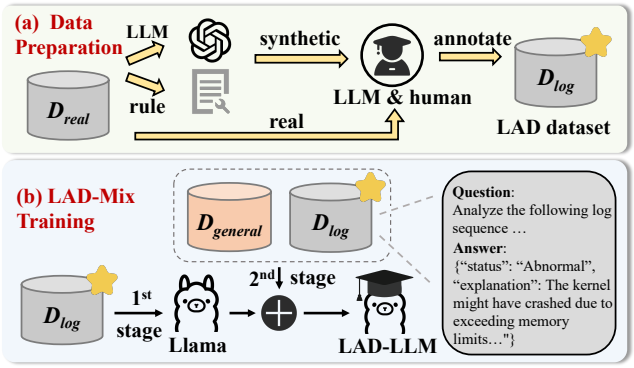

Figure 4: Illustration of LAD-LLM development, consisting of two steps: data preparation and hybrid training (LAD-Mix).

As shown in Figure 4(b), LAD-Mix consists of two stages. The first stage involves training with the curated data, which contains log domain expertise, while the second stage refines the model using a mixture of domain-specific and general data (i.e., Alpaca [31] and LIMA [45]). Extensive experiments validate the effectiveness and ease of use of the LAD-Mix training strategy, which achieves the best balance between detection performance and general capabilities among all methods. The detailed experimental results and analysis will be provided in Section 5.2.3.

After training, LAD-LLM can accurately detect potential anomalous instances in the subsequent online stage, provide explanations, and assist in the automatic updating of LogMoE.

4 ONLINE DETECTION

In the online detection stage, the offline-trained LogMoE and LAD-LLM models are deployed to run collaboratively. Specifically, this involves three phases: data preprocessing, collaborative anomaly detection, and automatic adaptation.

4.1 Preprocessing Module

The preprocessing module consists of three steps: (1) log parsing, (2) log partitioning, and (3) log representation. When new logs arrive, we employ a widely used log parser (i.e., Drain [9]) to convert each log into a specific log template. Next, the logs are partitioned into log sequences for pattern learning, as anomaly detection on a sequence of logs is more effective than on individual logs. In the final step, we convert the logs into semantic feature vectors for subsequent detection models. Specifically, we use pre-trained GloVe embeddings [25] to represent each word in the logs and apply the TF-IDF strategy to weigh their importance. The representation of a log l is computed as $\mathbf{x} = \sum_{i=1}^{N_l} (e_i \times \alpha_i)$, where N_l is the number of words in l, and e_i and α_i respectively represent the word embedding and TF-IDF weight of the i-th word in l. By applying these operations to each log in the log sequence, the preprocessed log sequence $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ is obtained.

Algorithm 1: Collaborative Anomaly Detection

Input: Preprocessed log data $\mathcal{X}$
Output: Anomaly status of $\mathcal{X}$, labeled dataset $\mathcal{D}_l$

1 $\mathcal{D}_l \leftarrow \{\}, T_l \leftarrow \{\}$ // Initialization
2 **for** $\mathbf{X} \in \mathcal{X}$ **do**
3 LogMoE computes the anomaly status s and confidence c for the normal class, with $\mathbf{X}$ as input.
4 **if** s = *abnormal* ***or*** $c < \tau$ **then**
5 explanation e, status s = Retrieve(T_l, $h(\mathbf{X})$)
6 **if** e *is NULL* **then**
7 LAD-LLM generates the anomaly status s and corresponding explanation e, with $\mathbf{X}$ as input.
8 $\mathcal{D}_l \leftarrow \mathcal{D}_l \cup \{(\mathbf{X}, s)\}$, Insert($T_l$, $h(\mathbf{X})$, (s, e))
9 **return** anomaly status s and explanation e
10 **else**
11 **return** anomaly status s
12 **return** $\mathcal{D}_l$

4.2 Collaborative Anomaly Detection

To achieve accurate and efficient anomaly detection, we propose a collaborative detection mechanism combining the strengths of LogMoE and LAD-LLM. LogMoE efficiently filters out a large volume of normal instances, passing the low-proportion potential anomaly instances to LAD-LLM for accurate inference. Since log sequences occasionally exhibit identical patterns, CoLA adopts a knowledge reuse strategy that leverages historical generation results (anomaly status, explanations), thereby further enhancing efficiency and reducing the generation time of LAD-LLM. Given an input log sequence $\mathbf{X}$, hashed by function h, the retrieval of matching results from hash table T_l determines if a match exists. If a match is found, the result is returned directly; otherwise, CoLA defaults to standard LAD-LLM reasoning. A comprehensive description of our collaborative anomaly detection approach is provided in Algorithm 1.

The procedure starts by initializing an empty set $\mathcal{D}_l$ and an empty hash table T_l to store pseudo-labeled data and historical generation results, respectively (line 1). Next, it traverses all the preprocessed log sequences and performs anomaly detection (line 2-11). For each log sequence $\mathbf{X}$, it first utilizes LogMoE for initial detection, obtaining the anomaly status s and the confidence c for the normal class (line 3). If the status is abnormal or the confidence is below a predefined threshold τ, the log sequence is sent to LAD-LLM for precise detection (line 4); otherwise, the log sequence is marked normal (line 10-11). Before LAD-LLM inference, the input is hashed to quickly check for reusable results in T_l (line 5). If a match is found, it is returned directly (line 9); otherwise, standard LAD-LLM detection is applied (line 6). In the LAD-LLM detection step, more accurate anomaly status s and the corresponding explanation e are generated through reasoning and returned as the final result (line 7-9). Each log sequence processed by LAD-LLM is added as a sample to $\mathcal{D}_l$, with its anomaly status s annotated as a pseudo-label. Simultaneously, s and the corresponding explanation e are inserted into T_l (line 8). Ultimately, we obtain anomaly statuses for all log data, with anomalous log sequences accompanied by explanations, along with the dataset $\mathcal{D}_l$ for the automatic adaptation phase (line 12).

4.3 Automatic Adaptation

Model adaptation can effectively mitigate performance degradation caused by changing data distributions. Unlike previous approaches [4, 21, 39] that require time-consuming and expert-involved labeling, our automatic adaptation module updates LogMoE autonomously with LAD-LLM assistance. In the process of anomaly detection with LAD-LLM, a portion of the LogMoE results has been pseudo-labeled, forming a dataset $\mathcal{D}_l$. We then adopt noisy label learning techniques to distill knowledge for model adaptation.

The memorization effect of deep neural networks [29] reveals that models initially fit easy patterns in the early stages of training, whereas noisy samples typically lead to larger loss values. Therefore, after a few warm-up epochs of standard training on noisy labels, we apply the Expectation-Maximization algorithm to fit a two-component Gaussian Mixture Model (GMM) [26] to the loss ℓ of each sample, identifying the clean samples. For each sample, the clean probability is represented by $w_i = p(g|\ell_i)$, where g is the Gaussian component with a smaller mean (indicating lower loss). Then, we divide $\mathcal{D}_l$ into a clean subset $\mathcal{D}_c$ and a noisy subset $\mathcal{D}_n$:

$$\begin{aligned} \mathcal{D}_c &= \{(\mathbf{X}_i, s_i) \mid \mathbf{X}_i \in \mathcal{D}_l, w_i \geq \tau_c\}, \\ \mathcal{D}_n &= \{(\mathbf{X}_i) \mid \mathbf{X}_i \in \mathcal{D}_l, w_i < \tau_c\}, \end{aligned} \tag{8}$$

where τ_c is a preset threshold. Once the time since the last update exceeds the predefined maximum interval, or $\mathcal{D}_c$ reaches a certain size, the model is automatically updated and used for the online detection module. This automated update strategy is not only efficient but also reduces a significant amount of expert labor costs.

Discussion. In future deployments, the system will continuously perform anomaly detection on real-time log streams through model collaboration. Upon identifying an anomaly, it will parse the LLM-generated JSON outputs to extract anomaly information and correlate it with system logs and metadata to generate alerts. These alerts will be dispatched through enterprise incident management platforms (e.g., PagerDuty) to on-call engineers, supporting timely diagnosis and resolution via the operations console.

5 EVALUATION

5.1 Experimental Setup

5.1.1 Datasets. We conduct comprehensive experiments on three real-world public datasets to evaluate the performance of CoLA and baseline methods. (1) **HDFS** [36] was collected during the execution of Hadoop-based MapReduce jobs on the Amazon EC2 platform for 38.7 hours. It consists of 11,175,629 log messages, organized into 575,061 log sequences based on the block_id of each log, with 16,838 identified as anomalous. (2) **BGL** [23] was collected from BlueGene/L supercomputer system at Lawrence Livermore National Labs (LLNL) over a period of 214 days. It comprises 4,747,963 log messages, of which 348,460 are anomalous. (3) **Spirit** [23] was collected from the Spirit supercomputer system at LLNL over a 2.5-year period, totaling 272,298,969 logs. Given its large size and the computational cost of training and inference, we adopt the first 10 million messages as the studied dataset, among which 289,905 are anomalous.

5.1.2 Baseline Approaches. In this work, we implement eight baseline methods, divided into two categories: SDM and LLM. (1) SDM: DeepLog [4] and LogAnomaly [21] are unsupervised approaches

Table 1: Overall anomaly detection performance.

Method	HDFS			BGL			Spirit		
	P	R	F1	P	R	F1	P	R	F1
DeepLog	78.60	93.15	85.26	21.71	77.86	33.95	34.79	98.02	51.35
LogAnomaly	65.39	93.50	76.96	23.65	80.41	36.55	46.57	98.38	63.22
PLELog	76.81	90.85	83.24	66.02	83.98	73.92	39.73	96.94	56.36
NeuralLog	74.34	93.57	82.85	84.89	76.35	80.39	53.60	89.98	67.18
LogRobust	80.91	92.77	86.44	85.74	81.15	83.38	41.94	99.20	58.96
Llama3-8b	5.47	83.19	10.27	41.34	94.83	57.58	20.77	99.22	34.35
Llama3-70b	7.51	99.67	13.97	42.80	96.08	59.22	21.41	99.95	35.27
GPT-4o	9.83	91.28	17.75	49.11	95.82	64.93	24.39	**100**	39.22
LogMoE	85.60	99.99	92.24	87.09	97.78	92.12	77.03	97.89	86.22
LAD-LLM	91.69	**100**	95.66	96.66	**98.68**	97.66	88.73	99.97	94.02
CoLA (off)	91.90	99.99	95.77	**97.08**	97.59	97.33	89.02	97.88	93.24
CoLA	**92.85**	99.75	**96.18**	97.04	98.53	**97.78**	**90.37**	98.33	**94.18**

that predict and detect anomalies using the autoregressive paradigm. PLELog [37] is a semi-supervised method that uses probabilistic label estimation to incorporate historical knowledge. NeuralLog [15] and LogRobust [39] are supervised approaches that incorporate semantic-aware embeddings for anomaly detection. (2) LLM: Llama-3.1-8b and Llama-3.1-70b are open-source models that can be deployed locally, while GPT-4o requires API-based access.

5.1.3 Evaluation Metrics. Following prior research [4, 21, 39], we adopt Precision (P), Recall (R), and F1-score (F1) as the metrics to evaluate the effectiveness of each method.

5.1.4 Implementation Details. For LogMoE, we train the model for 20 epochs with a batch size of 256, a learning rate of 2e-3, and the Adam optimizer. The total number of experts m is set to 8, and the number of selected experts k is set to 2. Following prior work [13, 16], we use 20% of the data for training and set the sliding window size to 100. For LAD-LLM, we adopt Llama3-8b as the base LLM and apply the LAD-Mix strategy, first training for 1 epoch with domain-specific samples, followed by 2 epochs with mixed samples (1:1 ratio of domain-specific to general data). Training is performed with a batch size of 16, a learning rate of 1e-5, and the Adam optimizer. GPT-4o is accessed via API (gpt-4o-2024-11-20). The experiments are conducted on an Ubuntu Linux 20.04 system equipped with four A800 GPUs.

5.2 Experimental Results

5.2.1 Effectiveness. We conduct a comparative analysis of CoLA's anomaly detection performance against various baseline methods, as shown in Table 1. The methods are grouped into three categories: SDM, LLM, and our proposed models (CoLA, LogMoE, LAD-LLM). Among these, CoLA employs an automatic adaptation strategy at runtime, while CoLA (off) does not utilize such a strategy.

The results show that CoLA achieves the best anomaly detection performance across all datasets, with F1-score improvements of 11.27%, 17.27%, and 40.19%, respectively. This verifies the effectiveness of the collaborative detection framework, where the filtering process not only alleviates the workload of LAD-LLM but also improves precision, thereby further enhancing detection capabilities. CoLA (off) exhibits a slight performance decrease compared to CoLA, as the performance of the filter (LogMoE) is affected by changing data distribution. In contrast, automatic adaptation helps maintain superior performance.

Table 2: Ablation study results for LogMoE.

Method	HDFS			BGL			Spirit		
	P	R	F1	P	R	F1	P	R	F1
LogMoE	**85.60**	**99.99**	**92.24**	**87.09**	**97.78**	**92.12**	**77.03**	**97.89**	**86.22**
w/o L_{bal}	83.96	99.24	90.96	87.28	89.09	88.18	77.14	87.81	82.13
w/o L_{div}	84.67	97.38	90.58	86.89	95.32	90.91	73.38	92.03	81.65
w/o L_{both}	80.94	98.95	89.04	85.71	88.44	87.05	70.76	91.77	79.91
w/o MoE	77.29	94.55	85.05	80.14	88.07	83.92	58.31	91.14	71.12

Regarding SDMs, we observe a tiered performance pattern across learning paradigms: supervised approaches (i.e., LogRobust, NeuralLog) demonstrate the strongest performance, followed by the semi-supervised PLELog, while unsupervised approaches (i.e., DeepLog, LogAnomaly) trail behind. This aligns with expectations, as supervised methods utilize semantic log representations and incorporate additional information through labeled training. However, these SDMs show limited generalization and struggle with evolving data distributions. For instance, LogRobust achieves only a 58.96% F1-score on Spirit. By incorporating the mixture-of-experts mechanism to learn different aspects of the data and fuse expert knowledge during inference, LogMoE demonstrates strong generalization capability. Compared to other SDMs, it achieves an F1-score improvement of 6.71%, 10.48%, and 28.34% on HDFS, BGL, and Spirit, respectively.

Regarding LLMs, the performance generally improves with increasing model size and the amount of pretraining data. Nevertheless, even the powerful GPT-4o achieves at most 64.93% F1-score on BGL and performs significantly worse on HDFS, with only 17.75%. This may stem from the specialized nature of log data analysis, which contrasts with the general-purpose knowledge encoded in LLMs, limiting their ability to effectively model domain-specific patterns. For LAD-LLM, we observe that after being trained on curated domain-specific data, it acquires domain expertise and achieves superior performance across all datasets, with F1-scores of 95.66%, 97.66%, and 94.02%, surpassing all baseline methods. It can be deployed locally for inference, mitigating the risk of data leakage. However, its low efficiency prevents it from processing the vast volumes of real-time log data in practical scenarios. Our framework CoLA effectively addresses this critical challenge, with experimental results and analysis detailed in Section 5.2.5.

5.2.2 Ablation Studies. We evaluate the components of both CoLA and LogMoE to understand their contributions.

Ablation Study on CoLA. We investigate the effectiveness of the model collaboration mechanism and the automatic adaptation module. As shown in Table 1, the collaboration mechanism of CoLA effectively improves the precision and F1-score of the original LAD-LLM. By comparing the results of CoLA and CoLA (off), we find that CoLA's performance is further enhanced by the automatic updating of LogMoE.

Ablation Study on LogMoE. We conducted experiments to assess the effectiveness of each LogMoE module: (1) LogMoE w/o L_{bal} denotes the removal of the load balancing loss. (2) LogMoE w/o L_{div} represents the removal of the expert diversity loss. (3) LogMoE w/o L_{both} indicates the removal of both the load balancing loss and the expert diversity loss, using only the cross-entropy loss function. (4) LogMoE w/o MoE refers to the removal of MoE layer.

As shown in Table 2, we found that encouraging a more even distribution of logs across experts, along with regularization to

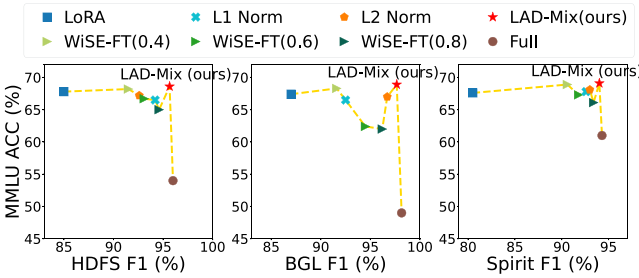

Figure 5: The anomaly detection and general capabilities of Llama3-8b after fine-tuning via different approaches.

enhance the diversity of expert capabilities, can effectively improve the model's anomaly detection performance. The results of LogMoE w/o L_{both} are significantly better than those of LogMoE w/o MoE, indicating that the MoE layer can significantly enhance the model's generalization capability in handling evolving data.

5.2.3 Effectiveness of LAD-Mix Training Strategy. As mentioned in Section 3.2.2, we conduct experiments to compare the performance of LAD-Mix with other common LLM fine-tuning methods, including Full (direct fine-tuning of all parameters), LoRA [12], L1 Norm, L2 Norm, and WiSE-FT [34]. The F1-score is used to measure the anomaly detection capability, while the MMLU benchmark [11] evaluates general capabilities using accuracy.

The experimental results are shown in Figure 5. Since anomaly detection is critical for ensuring system reliability, and general capabilities are essential for generating meaningful explanations, both factors must be jointly considered. Current methods typically improve anomaly detection at the expense of general capabilities, making it challenging to achieve high performance in both simultaneously. For instance, Full achieves the best anomaly detection performance but significantly degrades general capabilities. Lora reasonably preserves general capabilities but struggles to efficiently acquire new domain knowledge. In comparison, LAD-Mix achieves the second-best anomaly detection while best preserving general capabilities, providing a better balance between the two aspects. Thus, LAD-Mix ensures the model can accurately detect anomalies while generating readable and useful explanations for engineers.

5.2.4 Effectiveness of Automatic Adaptation. To assess the effectiveness of our automatic adaptation module, we conducted experiments comparing it with human annotation update method. Here, "w/ human" refers to training using labels annotated manually, while "w/ auto" indicates model updates using our LLM-enhanced automatic adaptation module. We partition the test data into five segments based on time, updating the model with both methods after each segment, using 3000 samples for training in each update.

As shown in Table 3, both manual annotation and our automatic adaptation improve model performance, indicating that updating is an effective approach to handle continuously changing log distributions. Across different datasets and models, automatic adaptation performs comparably to manual annotation, demonstrating the effectiveness of our design. It not only eliminates the need for time-consuming human annotations, but also effectively enhances the detection performance of SDMs.

5.2.5 Efficiency. We assess CoLA's efficiency against other baseline approaches. The results are presented in Figure 6, where we report

Table 3: Comparison of automatic adaptation and traditional human-involved updating.

Method	HDFS			BGL			Spirit		
	P	R	F1	P	R	F1	P	R	F1
DeepLog	78.60	93.15	85.26	21.71	77.86	33.95	34.79	98.02	51.35
w/ human	82.94	89.27	85.98	23.64	78.05	36.28	37.55	94.64	53.77
w/ auto	79.15	92.89	85.47	22.32	76.37	34.54	34.97	97.91	51.53
LogRobust	80.91	92.77	86.44	85.74	81.15	83.38	41.94	99.20	58.96
w/ human	85.67	93.30	89.32	86.28	91.47	88.80	70.56	86.95	77.90
w/ auto	82.51	93.97	87.87	86.59	85.09	85.83	59.83	88.72	71.47
LogMoE	85.60	99.99	92.24	87.09	97.78	92.12	77.03	97.89	86.22
w/ human	88.31	99.53	93.59	90.46	98.43	94.28	86.17	98.83	92.07
w/ auto	87.26	99.68	93.06	89.16	99.07	93.85	83.62	98.34	90.38

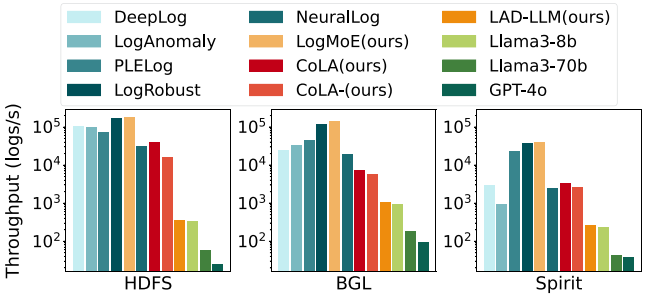

Figure 6: Efficiency comparison among the studied techniques.

the inference throughput, i.e., the number of logs processed per second. In this context, "CoLA-" refers to the variant that does not employ the knowledge reuse strategy.

Compared to the Llama3-8b with the same number of parameters, CoLA achieves a speedup ranging from 6.7× to 115.9× across various datasets, and is also significantly faster than other LLM-based methods. It is because the low throughput of LLMs is the bottleneck in the timely processing of the hundreds or thousands of logs generated in real-time. CoLA's collaborative detection mechanism efficiently filters out normal instances using LogMoE, leaving only a small fraction of potentially anomalous instances to be accurately classified by LAD-LLM. This approach achieves optimal detection performance while significantly improving efficiency. Moreover, the knowledge reuse strategy can avoid redundant reasoning on identical pattern log sequences, leading to an average throughput improvement of 61.26%. Although CoLA's efficiency is marginally lower than that of certain SDMs, which has significantly fewer parameters, its throughput remains within an acceptable range. Furthermore, when log sequences are long and the data volume is large, such as in the case of Spirit, CoLA even outperforms DeepLog, LogAnomaly and NeuralLog.

5.2.6 Explainability. To evaluate the quality of explanations, we randomly selected 150 abnormal log sequences in total from the test sets of HDFS, BGL, and Spirit, with equal sample sizes across datasets. Five annotators with operations and maintenance expertise rated the explanations based on Usefulness (Use.) and Readability (Read.). Usefulness measures an explanation's relevance and practicality, while readability evaluates its clarity and ease of understanding, both scored from 1 to 5. As shown in Table 4, compared to the original Llama3-8b, CoLA achieves an average improvement of

Table 4: Results of human evaluation.

Method	HDFS		BGL		Spirit	
	Use.	Read.	Use.	Read.	Use.	Read.
Llama3-8b	1.88	3.96	2.33	4.15	2.09	4.03
Llama3-70b	2.07	4.04	2.41	4.07	2.12	4.11
GPT-4o	2.31	4.17	2.37	4.23	2.16	4.19
CoLA (ours)	**3.83**	**4.47**	**4.09**	**4.66**	**3.95**	**4.60**

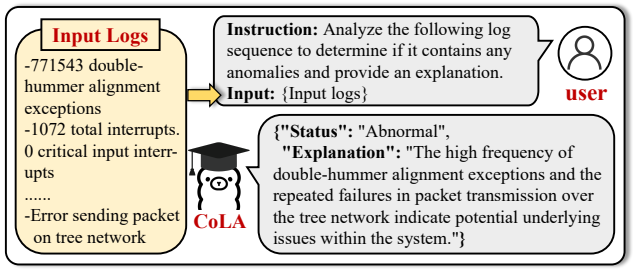

Figure 7: An example of CoLA performing anomaly detection on input logs and providing explanations.

89% in usefulness and 13% in readability. Although the performance of the baseline improves slightly with an increase in parameter size, it still falls significantly short of CoLA. These results suggest that injecting domain knowledge and balancing capabilities across different aspects can deliver more useful and user-friendly information for engineers in decision-making processes.

Figure 7 provides an illustrative example of how CoLA offers explanations to users. When LogMoE detects a potentially anomalous log sequence, the input prompt for LAD-LLM is automatically constructed by concatenating instruction with the input logs. LAD-LLM then performs inference and returns the results (status, explanation) in a JSON format suitable for API calls. The capability of LAD-LLM to generate clear and informative explanations has been acquired through offline instruction tuning on expert-curated datasets. In the current case, CoLA detects an anomaly and attributes the issue to certain exceptions and transmission failures.

5.2.7 Parameter Sensitivity. We evaluate the impacts of two major hyperparameters on LogMoE performance: (1) the number of experts m in the MoE layer and (2) the number of selected experts k.

Figure 8 illustrates the impact of m and k on detection performance. A larger m means more experts in the MoE layer, effectively expanding the model's parameter capacity. For instance, in BGL, with k fixed at 2 and m varying from 4 to 16, the F1-score increases from 91.93% to a peak of 92.12% (at $m = 8$) and then drops slightly. Here, k denotes the the number of experts selected by the gating network. As k increases, more experts are involved in decision-making, the integration of expert knowledge becomes more complex. For instance, in BGL, with m fixed at 8, increasing k from 1 to 8 raises the F1-score from 91.66% to 92.12% (at $k = 2$), then slightly drops to 91.64%. Overall, the anomaly detection performance of LogMoE is not sensitive to hyperparameters.

6 RELATED WORK

Log-based Anomaly Detection (LAD). LAD is critical for system stability and availability. Recent deep learning-based approaches [4, 10, 16, 20, 21] fall into unsupervised, semi-supervised, and supervised categories based on anomaly label availability. For unsupervised models, DeepLog [4] and LogAnomaly [21] utilize LSTM to learn normal log patterns by predicting the next log in a given subsequence. PLELog [37] is a semi-supervised log anomaly detection model that effectively leverages historical anomaly knowledge via probabilistic label estimation. As for supervised models, LogRobust [39] trains a Bi-LSTM that incorporates attention mechanism and semantic embeddings for anomaly detection. NeuralLog [15] extracts semantic vectors from raw logs without log parsing, enabling effective anomaly detection. Despite their efficiency, these methods are either not robust to data evolution, rely on extensive manual labeling, or lack explainability. Our CoLA framework tackles these limitations through model collaboration.

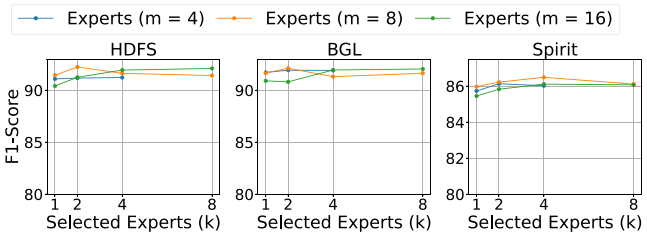

Figure 8: The impact of key hyperparameters (m, k) on anomaly detection performance.

Large Language Models (LLMs). LLMs such as GPT-4 [24], with billions of parameters and pre-trained on vast amounts of data, have demonstrated remarkable performance across various tasks [1, 6, 28, 41–43]. Recent studies have investigated LLMs for anomaly detection, such as in time series [46], video [38], and industrial images [7]. In addition, studies [3, 35] show that although LLMs improve in performance when trained on new domains, their general capabilities decline to the extent that they may fail to generate meaningful text. In this work, we explore the application of LLMs in log-based anomaly detection and achieve promising results by employing a two-stage hybrid training approach to balance domain-specific capabilities with general capabilities.

7 CONCLUSION

In this paper, we propose CoLA, a novel collaborative framework integrating the complementary strengths of SDM and LLM for log-based anomaly detection, in which the SDM serves as an efficient filter to identify suspicious instances, and the downstream LLM functions as an expert to detect anomalies, provide explanations, and refine the SDM. Extensive experiments on three large real-world datasets demonstrate that CoLA outperforms existing state-of-the-art approaches in effectiveness, efficiency, and explainability, simultaneously greatly reducing labor costs.

ACKNOWLEDGMENTS

This paper is supported by the National Regional Innovation and Development Joint Fund (No. U24A20254) and Zhejiang Provincial Natural Science Foundation of China under Grant (No.LQN25F020009). This work was supported by Ant Group through CCF-Ant Research Fund and Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security.

REFERENCES

[1] Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. 2023. Language Models Enable Simple Systems for Generating Structured Views of Heterogeneous Data Lakes. *Proc. VLDB Endow.* 17, 2 (2023), 92–105.

[2] Xiaolei Chen, Peng Wang, Jia Chen, and Wei Wang. 2023. AS-Parser: Log Parsing Based on Adaptive Segmentation. *Proceedings of the ACM on Management of Data* 1, 4 (2023), 1–26.

[3] Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2024. How Abilities in Large Language Models are Affected by Supervised Fine-tuning Data Composition. In *ACL*. Association for Computational Linguistics, 177–198.

[4] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017. DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning. In *CCS*. ACM, 1285–1298.

[5] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, and et al. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783* (2024).

[6] Benjamin Feuer, Yurong Liu, Chinmay Hegde, and Juliana Freire. 2024. ArcheType: A Novel Framework for Open-Source Column Type Annotation using Large Language Models. *Proc. VLDB Endow.* 17, 9 (2024), 2279–2292.

[7] Zhaopeng Gu, Bingke Zhu, Guibo Zhu, Yingying Chen, Ming Tang, and Jinqiao Wang. 2024. AnomalyGPT: Detecting Industrial Anomalies Using Large Vision-Language Models. In *AAAI*. 1932–1940.

[8] Fusheng Han, Hao Liu, Bin Chen, Debin Jia, Jianfeng Zhou, Xuwang Teng, Chuanhui Yang, Huafeng Xi, Wei Tian, Shuning Tao, Sen Wang, Quanqing Xu, and Zhenkun Yang. 2024. PALF: Replicated Write-ahead Logging for Distributed Databases. *Proc. VLDB Endow.* 17, 12 (2024), 3745–3758.

[9] Pinjia He, Jieming Zhu, Zibin Zheng, and Michael R. Lyu. 2017. Drain: An Online Log Parsing Approach with Fixed Depth Tree. In *ICWS*. IEEE, 33–40.

[10] Shilin He, Jieming Zhu, Pinjia He, and Michael R. Lyu. 2016. Experience Report: System Log Analysis for Anomaly Detection. In *ISSRE*. IEEE Computer Society, 207–218.

[11] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring Massive Multitask Language Understanding. In *ICLR*.

[12] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *ICLR*.

[13] Peng Jia, Shaofeng Cai, Beng Chin Ooi, Pinghui Wang, and Yiyuan Xiong. 2023. Robust and Transferable Log-based Anomaly Detection. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–26.

[14] Max Landauer, Florian Skopik, and Markus Wurzenberger. 2024. A Critical Review of Common Log Data Sets Used for Evaluation of Sequence-Based Anomaly Detection Techniques. *Proc. ACM Softw. Eng.* 1 (2024), 1354–1375.

[15] Van-Hoang Le and Hongyu Zhang. 2021. Log-based Anomaly Detection Without Log Parsing. In *ASE*. IEEE, 492–504.

[16] Van-Hoang Le and Hongyu Zhang. 2022. Log-based Anomaly Detection with Deep Learning: How Far Are We?. In *ICSE*. ACM, 1356–1367.

[17] George Lee, Jimmy Lin, Chuang Liu, Andrew Lorek, and Dmitriy V. Ryaboy. 2012. The Unified Logging Infrastructure for Data Analytics at Twitter. *Proc. VLDB Endow.* 5, 12 (2012), 1771–1780.

[18] Rui Li, Zhinan Cheng, Patrick P. C. Lee, Pinghui Wang, Yi Qiang, Lin Lan, Cheng He, Jinlong Lu, Mian Wang, and Xinquan Ding. 2021. Automated Intelligent Healing in Cloud-Scale Data Centers. In *SRDS*. IEEE, 244–253.

[19] Weiyang Liu, Rongmei Lin, Zhen Liu, Li Xiong, Bernhard Schölkopf, and Adrian Weller. 2021. Learning with Hyperspherical Uniformity. In *AISTATS*, Vol. 130. PMLR, 1180–1188.

[20] Lei Ma, Lei Cao, Peter M. VanNostrand, Dennis M. Hofmann, Yao Su, and Elke A. Rundensteiner. 2024. Pluto: Sample Selection for Robust Anomaly Detection on Polluted Log Data. *Proc. ACM Manag. Data* 2, 4 (2024), 203:1–203:25.

[21] Weibin Meng, Ying Liu, Yichen Zhu, Shenglin Zhang, Dan Pei, Yuqing Liu, Yihao Chen, Ruizhi Zhang, Shimin Tao, Pei Sun, and Rong Zhou. 2019. LogAnomaly: Unsupervised Detection of Sequential and Quantitative Anomalies in Unstructured Logs. In *IJCAI*. 4739–4745.

[22] Haibo Mi, Huaimin Wang, Yangfan Zhou, Michael Rung-Tsong Lyu, and Hua Cai. 2013. Toward Fine-Grained, Unsupervised, Scalable Performance Diagnosis for Production Cloud Computing Systems. *IEEE Trans. Parallel Distributed Syst.* 24, 6 (2013), 1245–1255.

[23] Adam J. Oliner and Jon Stearley. 2007. What Supercomputers Say: A Study of Five System Logs. In *DSN*. IEEE Computer Society, 575–584.

[24] OpenAI. 2023. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774* (2023).

[25] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*. 1532–1543.

[26] Haim H. Permuter, Joseph M. Francos, and Ian Jermyn. 2006. A study of Gaussian mixture models of color and texture features for image classification and segmentation. *Pattern Recognit.* 39, 4 (2006), 695–706.

[27] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. 2017. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. In *ICLR*.

[28] Mukul Singh, José Cambronero, Sumit Gulwani, Vu Le, Carina Negreanu, Elnaz Nouri, Mohammad Raza, and Gust Verbruggen. 2023. FormaT5: Abstention and Examples for Conditional Table Formatting with Natural Language. *Proc. VLDB Endow.* 17, 3 (2023), 497–510.

[29] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. 2023. Learning From Noisy Labels With Deep Neural Networks: A Survey. *IEEE Trans. Neural Networks Learn. Syst.* 34, 11 (2023), 8135–8153.

[30] Yushi Sun, Xin Hao, Kai Sun, Yifan Xu, Xiao Yang, Xin Luna Dong, Nan Tang, and Lei Chen. 2024. Are Large Language Models a Good Replacement of Taxonomies? *Proc. VLDB Endow.* 17, 11 (2024), 2919–2932.

[31] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford Alpaca: An Instruction-following LLaMA model. https://github.com/tatsu-lab/stanford_alpaca.

[32] Jianhong Tu, Ju Fan, Nan Tang, Peng Wang, Guoliang Li, Xiaoyong Du, Xiaofeng Jia, and Song Gao. 2023. Unicorn: A Unified Multi-tasking Model for Supporting Matching Tasks in Data Integration. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–26.

[33] Junyu Wei, Guangyan Zhang, Junchao Chen, Yang Wang, Weimin Zheng, Tingtao Sun, Jiesheng Wu, and Jiangwei Jiang. 2023. LogGrep: Fast and Cheap Cloud Log Storage by Exploiting both Static and Runtime Patterns. In *EuroSys*. ACM, 452–468.

[34] Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, and Ludwig Schmidt. 2022. Robust fine-tuning of zero-shot models. In *CVPR*. IEEE, 7949–7961.

[35] Tongtong Wu, Linhao Luo, Yuan-Fang Li, Shirui Pan, Thuy-Trang Vu, and Gholamreza Haffari. 2024. Continual Learning for Large Language Models: A Survey. *arXiv preprint arXiv:2402.01364* (2024).

[36] Wei Xu, Ling Huang, Armando Fox, David A. Patterson, and Michael I. Jordan. 2009. Detecting large-scale system problems by mining console logs. In *SOSP*. ACM, 117–132.

[37] Lin Yang, Junjie Chen, Zan Wang, Weijing Wang, Jiajun Jiang, Xuyuan Dong, and Wenbin Zhang. 2021. Semi-supervised Log-based Anomaly Detection via Probabilistic Label Estimation. In *ICSE*. IEEE, 1448–1460.

[38] Yuchen Yang, Kwonjoon Lee, Behzad Dariush, Yinzhi Cao, and Shao-Yuan Lo. 2024. Follow the Rules: Reasoning for Video Anomaly Detection with Large Language Models. In *ECCV*, Vol. 15139. Springer, 304–322.

[39] Xu Zhang, Yong Xu, Qingwei Lin, Bo Qiao, Hongyu Zhang, Yingnong Dang, Chunyu Xie, Xinsheng Yang, Qian Cheng, Ze Li, Junjie Chen, Xiaoting He, Randolph Yao, Jian-Guang Lou, Murali Chintalapati, Furao Shen, and Dongmei Zhang. 2019. Robust log-based anomaly detection on unstable log data. In *FSE*. ACM, 807–817.

[40] Yupu Zhang, Guanglin Cong, Jihan Qu, Ran Xu, Yuan Fu, Weiqi Li, Feiran Hu, Jing Liu, Wenliang Zhang, and Kai Zheng. 2024. ESTELLE: An Efficient and Cost-effective Cloud Log Engine. In *SIGMOD*. ACM, 201–213.

[41] Yi Zhang, Jan Deriu, George Katsogiannis-Meimarakis, Catherine Kosten, Georgia Koutrika, and Kurt Stockinger. 2023. ScienceBenchmark: A Complex Real-World Benchmark for Evaluating Natural Language to SQL Systems. *Proc. VLDB Endow.* 17, 4 (2023), 685–698.

[42] Yunjia Zhang, Avrilia Floratou, Joyce Cahoon, Subru Krishnan, Andreas C. Müller, Dalitso Banda, Fotis Psallidas, and Jignesh M. Patel. 2023. Schema Matching using Pre-Trained Language Models. In *ICDE*. IEEE, 1558–1571.

[43] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, and et al. 2023. A Survey of Large Language Models. *arXiv preprint arXiv:2303.18223* (2023).

[44] Junhao Zheng, Shengjie Qiu, Chengming Shi, and Qianli Ma. 2025. Towards Lifelong Learning of Large Language Models: A Survey. *ACM Comput. Surv.* (2025).

[45] Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, and et al. 2023. LIMA: Less Is More for Alignment. In *NeurIPS*.

[46] Tian Zhou, Peisong Niu, Xue Wang, Liang Sun, and Rong Jin. 2023. One Fits All: Power General Time Series Analysis by Pretrained LM. In *NeurIPS*.